Instituto Universitário de Lisboa

*DEPARTAMENTO DE CIÊNCIAS E TECNOLOGIAS DA INFORMAÇÃO*

# IM-SGI

# AN INTERFACE MODEL FOR SHAPE GRAMMAR IMPLEMENTATIONS

**Joana Maria Oliveira Tching Nascimento de Abreu Andrade**

Tese especialmente elaborada para obtenção do grau de

**Doutor em Ciências e Tecnologias da Informação**

Orientador:

Prof. Doutor Joaquim António Marques dos Reis

Co-Orientadora:

Prof.ª Doutora Alexandra Cláudia Rebelo Paio

[*Julho*, 2018]

Escola de Tecnologias e Arquitetura

Departamento de Ciências e Tecnologias da Informação

# IM-sgi – An Interface Model for Shape Grammar Implementations

Joana Maria Oliveira Tching Nascimento de Abreu Andrade

Tese especialmente elaborada para obtenção do grau de

Doutor em Ciências e Tecnologias da Informação

Júri:

Doutoramento, Carlos Costa, Professor Associado, ISEG, Universidade de Lisboa

Doutoramento, Luís António dos Santos Romão, Professor Associado, Faculdade de Arquitetura, Universidade de Lisboa

Doutoramento, Carlos Costa Fernandes, Professor Auxiliar, Instituto Superior Técnico, Universidade de Lisboa

Doutoramento, Joaquim António Marques dos Reis, Professor Auxiliar, Orientador, ISCTE-IUL

Doutoramento, Alexandra Cláudia Rebelo Paio, Professora Auxiliar, Co-orientadora, ISCTE-IUL

[Julho, 2018]

# SUMMARY

This research arises from the interest in computing as offering new paradigms in the design practice. Information technologies are the driving force for progress in the processes of design, enabling new forms of creativity. The increasing sophistication of computer applications, their easier access, and lower cost have had a significant impact on design practices and can be regarded as a paradigm shift. The invention and creativity are thus seen as knowledge processing activities and can, at least partially, be carried out with the support of computer applications. In this context Shape Grammars (SG) as production systems of designs through rules have the potential to create designs with variable user input and the ability to evaluate a large number of alternatives that may lead to innovative designs. Most architects and designers use computers on their daily practice as a representation tool for their projects, but not as a facilitator or increaser of the creative process. SG computational implementations have the potential to enhance creativity with the test of a wide range of design options, helping the appearance of new solutions, either through the emergence of new shapes or by stimulating the designer’s creativity with the possibilities presented.

As Architects and Designers haven't adopted existing SG computational implementations, that take advantage of computation to facilitate and enhance their work, could the problem be on the communication between the applications and the user? If the interface of the SG implementation does not allow the user to understand how to use it or how to control and make use of its results, it can’t be successfully used.

With interest in SG implementations as creative partners in the creative process, our research starts with the analysis of existing SG implementations, trying to understand if they had the potential to be adopted by architects and designers in their practice and, if not, what could be done to lead to that objective.

User Interface Inspection Methods were used to perform this analysis and allowed us to understand that there are interactions and communication issues that need to be addressed for SG implementations to be adopted by designers.

Taking this direction, we understood that models of interaction between the user and SG implementations have already been developed. The present research proposal started from the analysis of the interaction model of Scott Chase, where he defines the different levels of interaction between the user and SG implementation, with more or less input from the user, establishing different ways to combine synergies to obtain new creative solutions.

Taking this interaction model as a starting point, next, we must assure the correct communication between user and implementation occurs. The means of communication between these two agents is the computational interface. Understanding the importance of the interface to allow the user to know how to use the computational implementation and be able to produce results, our research presents the development of an interface model for SG implementations to help to take a step towards the adoption of SG for creative projects.

For this, we used methods from Human-Computer Interaction discipline, and we also took Bastien & Scapin's "List of Ergonomic Criteria Guidelines" as guiding lines to define the Criteria of our interface model, called IM-sgi, Interface Model for Shape Grammar Implementations.

Thus, IM-sgi, an interface model for SG Implementations, has the purpose of helping SG implementations developers to address the interface on the right path to a correct communication with the particular type of user that architects and designers are.

Interface prototypes following IM-sgi criteria are finally developed and presented to test the suitability of the IM-sgi Model to SG implementations and validate the objectives we propose.

Keywords: Human-Computer Interaction; Computational Creativity; Computer Interface; Shape Grammars

# RESUMO

A presente pesquisa surge do interesse na computação por oferecer novos paradigmas na prática do design. As tecnologias da informação são a força motriz para o progresso nos processos de design, permitindo novas formas de criatividade. A crescente sofisticação das aplicações computacionais, o acesso mais fácil às mesmas e menor custo associado tiveram um grande impacto nas práticas de projeto e podemos considerar estar perante uma mudança de paradigma. A invenção e a criatividade são, portanto, vistas como atividades de processamento de conhecimento e podem, pelo menos parcialmente, ser realizadas com o suporte do computador. Nesse contexto, As Gramáticas de Forma, como sistemas de produção de designs através de regras, têm o potencial de criar projetos com entradas com níveis variáveis de intervenção do utilizador e apresentam a capacidade de avaliar um grande número de alternativas que podem levar a designs inovadores. A maioria dos arquitetos e designers usa o computador na sua prática diária como uma ferramenta de representação para seus projetos, mas não como um facilitador ou potenciador do processo criativo. As implementações computacionais de Gramáticas de Forma têm o potencial de aumentar a criatividade com o teste de uma ampla gama de opções de design, ajudando no surgimento de novas soluções, seja pela emergência de novas formas ou estimulando a criatividade do designer com as opções desenvolvidas.

Uma vez que as implementações computacionais de Gramáticas de Forma existentes não foram adotadas por Arquitetos e Designers, que claramente tiram proveito de aplicações computacionais para facilitar e aperfeiçoar o seu trabalho, poderá o problema estar na comunicação entre as implementações e o utilizador? Se a interface da implementação não permitir que o utilizador entenda como usá-la ou como controlar e utilizar os seus resultados, ela não poderá ser utilizada com sucesso.

Com interesse nas implementações de Gramáticas de Forma como parceiros criativos no processo criativo, a nossa investigação começa com a análise das implementações de Gramáticas de Forma existentes, tentando entender se estas têm potencial para ser adotadas pelos criativos na sua prática e, se não, o que poderia ser feito para chegar a este objetivo.

Foram usados métodos de inspeção para realizar esta análise para nos permitir compreender que há interações e problemas de comunicação que precisam de ser resolvidos para que as implementações de Gramáticas de Forma sejam adotadas pelos projetistas.

Seguindo essa direção de investigação, percebemos que foram já desenvolvidos modelos de interação entre o utilizador e as implementações de Gramáticas de Forma. A presente proposta de pesquisa surgiu a partir da análise do modelo de interação de Scott Chase, onde este define os diferentes níveis de interação entre utilizador e implementação de Gramáticas de Forma, com maior ou menor input do utilizador, estabelecendo diferentes formas de combinar sinergias para obter novas soluções criativas.

Tomando esse modelo de interação como ponto de partida, devemos assegurar que a comunicação correta entre o utilizador e a implementação ocorra. O meio de comunicação desses dois agentes é o interface computacional. Entendendo a importância do interface para permitir que o utilizador entenda como usar a implementação computacional e seja capaz de produzir resultados, a nossa pesquisa apresenta o desenvolvimento de um modelo de interface para implementações de Gramáticas de Forma para ajudar a dar um passo na direção da adoção das Gramáticas de Forma para projetos criativos.

Deste modo, aplicámos métodos da disciplina de HCI e também adotámos a Lista de Diretrizes e Critérios Ergonómicos de Bastien & Scapin como linhas de orientação para definir os Critérios do nosso modelo de interface, denominado IM-sgi.

O IM-sgi, um modelo de interface para Implementações de Gramática de Forma, tem a finalidade de ajudar programadores de implementações de Gramáticas de Forma a endereçar a interface no sentido de atingir uma comunicação correta com o tipo particular de utilizador que são os arquitetos e designers.

Protótipos de interface seguindo os critérios IM-sgi são finalmente desenvolvidos e apresentados para testar a adequação do modelo IM-sgi e validar os objetivos que propomos.

Palavras-chave: *Human-Computer Interaction*; Criatividade Computacional; Interface Computacional; Gramáticas de Forma

# ACKNOWLEDGMENTS

I want to thank my Ph.D. advisors, Professors Joaquim Reis, and Alexandra Paio, for supporting me during these past seven years. I firmly believe I could not finish this research without their endless patience and guidance.

I thank all the architectural students from ISCTE-IUL that participated in the survey, who supported my work and helped me get results of better quality.

I will forever be thankful to my family and close friends that helped me to keep strong through all the adversities.

I thank specially to my husband Hugo Nascimento and daughter Sofia Nascimento, for always helping me take the time needed to complete this work.

I dedicate this thesis to you, my father, Rui Andrade e Tching, my light, my star in the sky…

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| Artificial Intelligence | AI |
| Cognitive Walkthrough | CW |
| Computational Creativity | CC |
| Computer-Aided Design | CAD |
| Ergonomic Criteria | EC |
| Graphic User Interface | GUI |
| Human-Computer Interaction | HCI |
| Inspection Methods | IM |
| Interaction Design | ID |
| Interface Model for Shape Grammar Implementations | IM-sgi |
| Shape Grammar (s) | SG |
| System Usability Scale | SUS |
| User Interface | UI |

# 1 INTRODUCTION

The present study arises from the interest in computation as a partner for creative projects. With this purpose in mind, our focus is on the use of Shape Grammars (SG) and how SG computational implementations have the potential to become tools on the architects and designers' daily practice.

As communication between the user and computational implementation is crucial to its success, our thesis addresses the usability aspects related to the interface, to find ways to ease the use of SG computational implementation by architects and designers in their creative process.

Following Lewis and Rieman (1994) process to create a new interface that focuses on the success of the task performance, we adapted it and used it, not for the creation of a specific interface, but to create an interface model for SG implementations. This model is to be used by programmers on SG implementations, helping to assure that the communication with architects and designers is accomplished.

Intending to study the aspects of creating an interface for SG computational implementations, our research starts with the analysis of existing SG implementation, using a Human-Computer Interaction (HCI) inspection method, the Cognitive Walkthrough (CW).

This analysis allowed the understanding of the need to address interface issues to assure the correct communication between the user and SG implementation. Therefore, we studied existing interaction models and ergonomic criteria for User Interface design.

We finally propose the structure of an interface model and types of user interaction, defining IM-sgi – Interface Model for SG Implementations, composed by a set of ergonomic criteria applied to three different types of users, according to their expertise level and use of SG.

To validate IM-sgi, we used two different HCI methods: (1) the CW; and (2) the test of interface prototype with users. To validate IM-sgi through a CW, we defined system actions and applied IM-sgi ergonomic criteria to the previously analyzed SG implementations. This CW allowed the analysis of what criteria they follow and if that helps to understand their lack of use. To further validate IM-sgi, a prototype is presented and tested by students of architecture, to help understand if an interface following IM-sgi criteria is well received and understood. The results of the test lead us to a reformulation of the prototype, also presented.

We then simulate in a step-by-step what would be the use of a SG application with our prototype as the interface. We show it with the basic shapes illustrated in the prototype and then simulating the use of an architect in a real project question-solving.

Finally, we suggest this work contributes to surpass the barriers of communication between users and SG computational implementations, helping take a step forward on the adoption of SG in the early stages of the design process. Consequently, we seek to contribute to providing architects and designers with new computation tools that allow them to explore computational creativity.

## 1.1 MOTIVATION

As an Architect, it has been interesting to see that architects are responsible for the definition of an individual style of architecture, their own style (Mayer & Turkienicz, 1998), connected with their intentions and purposes, which might have aesthetic concerns or others. From this point of view, the style is related to the idea of the purposes of the design and to the way the artist solves a specific problem.

Highlighting the connection between style and problem-solving, we realize that there is a link between creativity and hypothesis exploration, which leads to the possibility of a broader range of results through the use of SG. Gero (2006) points out that creativity resides not only in a new artifact evaluated by society but also in the processes that have the potential to generate objects that might be evaluated as creative. The designer deals with a particular context, with unique features, and the perception of the purposes, constraints, and contexts that enable him to explore relationships and solutions in an individual and creative way.

According to Reffat (2003), creativity in the field of architecture and design can be seen as an emergence of new shapes and combinations between them. Design concepts arise from sets of relationships that could be translated by SG rules.

After using the computer as a drafting tool, computation started being used in architecture with scripts, sets of instructions, with the purpose of problem-solving, as architects realized the potential of the use of the computer to address complex design problems (Rocker, 2006).

Digital Creativity and new technologies, helping the architect to explore new possibilities, can lead to innovation and changes in the design process and style (Symeonidou, 2018).

SG were one of the first attempts to introduce the computational approach to art and architecture (Tepavcevic & Stojakovic, 2012) when created by Stiny and Gips (1972) in the 1970s. When a designer defines an SG, defining a shape vocabulary and a set of rules, which can be translated into solutions to a specific problem, he generates the principles that lead the generative process. This process has a trial and error component as it goes through several hypotheses until an acceptable solution is found. This possible formal solution may not correspond exactly either to the intention or to the planned purpose at the beginning of the process. There can be some lack of control of the outcome, as the generative process might lead to unsuspected results. Mayer and Turkienicz (1998) pointed out that this lack of control, as a constraint to the expansion of SG in practical applications, could actually be seen as an advantage and an opportunity in the development of new solutions to the project issues in architecture and design.

Architectural projects, in contrast to other artistic areas, develop in different phases that can be described not only as a need to solve a big group of settings, but also the fulfillment of legal, environmental, economic or formal rules and constraints. It's from the solution of all these issues that the final project arises, and the architect shows his creativity, combining all these items in an aesthetic and functional product. If the architect must respond to challenges, he looks for the ideal solution and follows the recipes that he has already tested and evaluated in his past practice. Therefore, his creativity might be conditioned by his experience or to the lack of time/tools to explore and test the new hypothesis without assurance of success (J. Carroll & Rosson, 2003).

The above-mentioned lack of control of the results that opens the way to new solutions can mean the break of the creativity limits that the architect and designer can feel on their daily practice, opening the horizon to new exploration paths and solutions finding.

To make use of SG, the architect must consider the rules he has or intends to fulfill for each problem or sub-problem and which shape vocabulary is involved. This way, he can define, control, and modify the SG that gives new answers to his question.

An overwhelming majority of architectural projects have situations that are common and also project-specific that can be dealt with using SG. Architecture, in its essence, is made of shapes, spaces, and relationships. There is a large number of themes that can be explored by the architect to create his work (Consiglieri, 1994), even if they are carefully planned or created intuitively. Themes such as relationships between full and empty spaces, light and shadow, conjugations between similar shapes and/or dissimilar shapes can be recognized in every architectural work. On the other hand, architectural work is deeply connected with social and legal rules. Every project must respond to objective standards and to urbanistic laws that sometimes confine the development of the solution.

Understanding that the architectural work means a series of decisions and that the decision process is made by choosing among alternatives, using computational intelligence for decision support means that we can use intelligent systems to act as decision advisors (Chen, 1999). SG, as intelligent production systems, can become an exciting decision partner in the creative practice.

The architect has been changing the way of doing his job since the advent of the computer. He can now use computational applications to reproduce and generate the designs he defined, for example, in tridimensional models. It is now possible to explore multiple hypotheses, reducing project errors, and spending less time in searching for more complex solutions to achieve more ambitious, innovative, and creative projects. With the development of AI, and within this area, the growth of applications to emulate human creativity, architectural projects can use computational applications that boost not only the technical aspects but also the creative and aesthetic ones. It is now common practice to use computational applications, which reproduce the architect's manual design, among other technical aspects of the project (such as automatic measurements, thermal simulations, three-dimensional visualization, etc.). Therefore, it is crucial to show how architects may use SG and its applications as a common practice and a way of optimizing their ideas, using not only Computational Design but also Computational Creativity (Grasl & Economou, 2013).

The development of computer tools that support the creative stages, primarily through the exploration of alternatives, can take architects and designers to discover new solutions and develop new styles (Reffat, 2003).

Figure 1 illustrates the several stages involved in the design process and where Computational applications give more and less support.

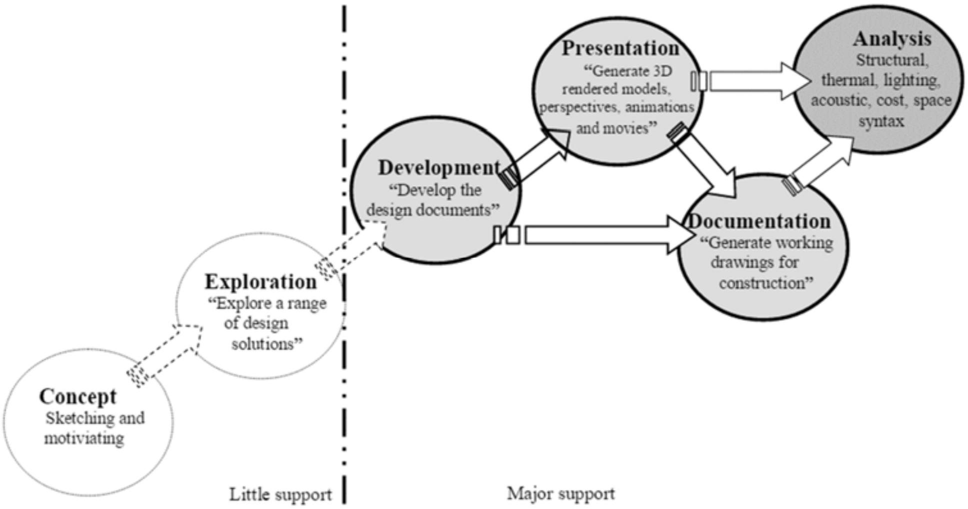

Presentation
"Generate 3D rendered models, perspectives, animations and movies"
Analysis
Structural, thermal, lighting, acoustic, cost, space syntax
Development
"Develop the design documents"
Documentation
"Generate working drawings for construction"
Exploration
"Explore a range of design solutions"
Concept
Sketching and motiviating
Little support
Major support

Figure 1 – Computational support at the various stages of the design process (Reffat, 2003)

Consequently, this research focuses on SG computational implementations to help on the concept and exploration stages of design (Figure 1), where there is little support from computational applications.

As an architect, passionate about exploring creativity and about new technologies, the primary motivation is to find ways to help SG computational implementations to successfully enter the project practice, believing that SG can open ways to bigger creativity exploration and new forms of design assisted by the computer.

While computers are now crucial to materialize the architect and designer ideas faster and more appealing, either through graphics software, 3D simulation or even 3D printing, they only help to materialize and to enhance existing ideas. Our vision is to have the same type of engagement and use of technologies, not on the stages of substantiating the project ideas, but on the creative stages of defining new ideas. We believe SG are an example of intelligent systems that, through the exploration of shapes, can help potentiate the conceptual stages of design and assist the architect and designer exploring new ways of creativity.

## 1.2 Scope and Objectives

This research focuses on the interaction issues between SG user and SG implementations. Studying the existing SG implementations, we suggest that the interface needs to be addressed appropriately to guarantee the correct communication between the user and the computational tool. To answer this issue, our scope is to create an interface model for SG computational implementations that addresses specific types of users (Architects, Designers and Artists), that clearly defines how SG implementations must communicate with the users to allow them to perform the actions needed to create, apply and modify SG and use their results.

Hence, our work was conducted seeking two main objectives:

The first objective is to help the existing group of researchers that are interested in creating SG implementations to have guiding lines that assure the proper communication of their computational applications with the end user. It is essential to make the bridge between the user needs and the SG computational implementation.

Focusing on this objective, we understood that the interface is a very time-consuming task, and many times the software engineers do not have enough time to adequately address the best way to communicate with specific types of users. Architects and designers have peculiar ways of thinking and communicating, so we suggest that creating the right guidelines for the interface to address these particular users means a significant impact on the computational implementation success.

The second objective, with a broader vision, is to contribute to the architectural and design practice. This objective goes in line with the field of the Design Science (DS) introduced by Herbert Simon (1969), in an effort to make sure that the studies around design creation and design theory were correctly addressed, along with the possibilities of using computer applications to help the design process. He was the leader in the development of this science of complex information processing that gathers visions on the human cognitive system and computer simulation. DS searches for mechanisms on problem-solving and concept formation that are connected with design tasks and how computers can help. Herbert Simon shows how important it is to make design theory explicit so that we can introduce computers into the process. In a later edition of the book, *The Sciences of the Artificial* (Michalos & Simon, 2006), the authors refer as "Artificial" the systems that adapt according to defined goals. They show an information-processing theory of man's thinking process and combine it with the developments in computer science, believing this can lead to emerging sciences on the engineering of design.

Sharing this vision, with the potential that SG computational implementations have to stimulate new ways of creativity and to explore more hypothesis than possible without computer processing, we think that helping the synergy between SG implementation and artist leads to new original designs, with increased quality and complexity.

These objectives led us to study what could be a missing part to make sure designers and architects successfully adopted SG. Looking carefully to the existing SG implementations, the interface design quality appeared to be a missing target in all of the analyzed ones. Being the interface such an important and complex task and being so decisive to the success in the use of a computational application, our Scope was defined. It would be to help SG implementations to communicate correctly with the user through a well-designed interface, correctly adjusted to the user characteristics and following the concepts of good usability.

## 1.3 METHODOLOGY

According to our scope and objectives, the discipline within computer science that gathers the knowledge we are searching for is HCI. As we become more and more tech-dependent, HCI research and design become more relevant to today's world (Richard Harper, Tom Rodden, 2008) (Petrovic, 2012). Computer systems and the demands of the users are changing, and HCI must explore new ways of understanding the users, new ways of designing, and how to evaluate the technology. The present paradigm in HCI is to find new ways to engage with society as a whole.

Nevertheless, our scope is intimately related to creative design and the main artifacts and actions related to design projects are shapes, their drawing, and their relations. This means that, although new digital supports and techniques are emerging, we still find in Graphical User Interfaces (GUI) good support for the emulation of the drawing sheet and the intuitive exploration activities of drawing. Even with the use of multi-modal interfaces, for example with the use of speech, CAD applications are intrinsically related with graphic interfaces (Khan & Tuncer, 2019). In this same line, we also found in major developments of HCI in earlier stages excellent support to guide our investigation on an interface model for a GUI environment.

Using the process to be followed when creating a new interface (Lewis & Rieman, 1994), first, we analyze the target users and the tasks that they would perform using an SG implementation. Using inspection methods from HCI to test existing SG implementations, we address the interface issues detected, developing IM-sgi, an interface model for SG implementations. After defining IM-sgi, new HCI methods are used, to test it, with the predominance of methodologies that allow evaluating without users. These methods include prototyping and analyzing the results. A user test is applied on a first proposed prototype, and a second prototype is defined addressing the conclusions taken from the user test. We then simulate the prototyped interface showing a step-by-step use of the computational application related to the interface prototype.

According to Lewis and Rieman (1994), the process to be followed when creating a new interface that focuses on the success of the task performance has the following steps:

1. Analysis of the target users and the tasks they perform with the application;
2. Selection of the main tasks the application must perform
3. Analysis of existing interfaces of similar applications
4. The initial definition of the interface
5. Evaluation of the interface proposed without users (by the designer of the interface or experts in the area)
6. Prototyping
7. User test
8. Iteration of prototype corrections
9. The final development of the interface

This research follows these steps, not for creating a final interface, but for a general model of interfaces for SG Implementations, called IM-sgi. Stages 1 to 3 are described in Chapter 3 and published in (Tching, Reis, & Paio, 2016). Stage 4 focused on the definition of the interface model, through a set of criteria that should be followed to create a successful interface, presented on Chapter 4 and published in (Tching, Reis, & Paio, 2018).

The guidelines for these criteria are based on the Bastien and Scapin (1993) list of Ergonomic Criteria (EC). We adopt the ergonomic criteria developed by the authors to groups of criteria for the IM-sgi, each group containing sets of criteria for the tasks that are performed to use SG.

The stages 5 to 9 of the process to create a new interface model for SG implementations are described in Chapter 4.4. This work was submitted for publication and is waiting for a decision.

## 1.4 STRUCTURE OF THE THESIS

This thesis is comprised of five chapters as follows: (1) Introduction; (2) State of the Art; (3) Analysis of Shape Grammar Systems and Users – Cognitive Walkthrough; (4) IM-sgi; and (5) Conclusion and Future Work.

Chapter 1 – Introduction presents what is the purpose of this research, addressing the objectives and the methodologies used to achieve them.

Chapter 2 – State of the Art explores concepts of the two main fields of knowledge related to this study: Computational Design (CD) and Human-Computer Interaction (HCI). On the area of the CD, we address SG exploring chronologically the implementations that have been developed. Still, in this chapter, we explore existing models of interaction for SG implementations, seeking to understand SG systems and users. On the field HCI, the concepts related to Computer Interfaces and HCI are exposed.

Chapter 3 – Analysis of Shape Grammar Systems and Users – Cognitive Walkthrough: HCI method Cognitive walkthrough is applied to analyze existing SG implementations to gather conclusions on the interface issues that must be addressed. A group of five existing SG implementations that were available online was elected as the corpus for the analysis.

Chapter 4 – IM-sgi – Interface Model for Shape Grammar Implementations describes a proposal of an Interface Model composed by a set of criteria that can be followed to create interfaces that correctly communicate with the end user and allow users to do the tasks needed to create, explore, control and make use of SG. Finally, it is described the model validation through HCI methods again (Cognitive Walkthrough and Prototyping), seeking to understand if IM-sgi obtains the desired results.

Chapter 5 – Simulation and Discussion – we present a step-by-step functioning simulation of an SG implementation with the interface prototype showing how it could be used for architectural projects. We also compare the usability with the existing tools we studied in Chapter 3 to understand the possible improvements led by the good interface usability

Chapter 6 – Conclusion and Future Work discusses the research results achieved, and the work thought about possible future work.

# 2 STATE OF THE ART

This research starts with the study of the fields related to our scope and objectives: Computational Design (CD) and Human-Computer Interaction (HCI). In CD, Artificial Intelligence (AI) and Computational Creativity (CC) are explored and then focus on Shape SG implementations. These are studied chronologically to understand its evolution through time and the point they are presently. After we approach HCI to understand better computational interfaces and how the concepts of HCI can lead to better communication between users and computer applications.

## 2.1 COMPUTATIONAL DESIGN

*"Computational formal explorations do not eradicate human imagination but rather extend its potential limitations. Computation is not a substitute for human creativity and therefore, cannot be antagonistic. Rather it provides the means for exploration, experimentation, and investigation in an alternative realm."* (Terzidis, 2003:6)

Computation has been responsible for new paradigms in the design practice, revealing itself as the driving force for progress in the design process and creative use of shapes. The increasing sophistication of computer applications, greater access, and lower costs have had a significant impact on design practices. Information technologies are mostly responsible for progress in design, mainly in computer-assisted design. The increasing ability of the applications and their availability and speed of computation have markedly changed the work modes and goals reached by architects, designers or artists (Kicinger, Arciszewski, & De Jong, 2005).

Current senior architects and designers assisted to a shift from hand drawings to the use of computational tools during their education or practice, while new architecture and design students are already digital natives (Symeonidou, 2018). This fact shows that computer applications are following the tendency to enter deeply in creative practice. The existing applications have mainly been used for analytical purposes and phases of design detail. However, with the progress of information technology, possibilities are open for applications targeted to all stages of design: the generation of concepts, preliminary specifications, formal specifications, and details.

Design engineering is currently in a shift of paradigm (Kicinger et al., 2005). The previous analytical paradigm is being replaced by the paradigm of knowledge and understanding. In the first paradigm, the important thing was the creation of analytical models to recognize the operation of objects, while the second combines system representation, the design process and the relationship between the attributes and the representation space (Arciszewski, Michalski, & Wnek, 1995).

Currently, the design focus is on how to use existing knowledge through information technologies to develop new designs. Information technology brings together disciplines relating to the acquisition, processing, distribution, and generation of information or knowledge through computer technology (Michalos & Simon, 2006). In the current paradigm, conceptualization and creativity are seen as knowledge processing activities that can, at least partially, be carried out on the computer. Through AI, for example, it is possible to achieve the development of design concepts through computation and the construction of new classes of applications for the design (Turkay & Baykasoglu, 2010).

### 2.1.1 ARTIFICIAL INTELLIGENCE

Alan Turing (1950) stated that the idea behind digital computers was to carry out any operations that could be done by a human, including processes that we define as intelligence. Thus, AI is the area of choice for the development of tools that, through the incorporation of explicit representations of knowledge and intelligent capabilities, can optimize the work of the architect, designer or artist (Russell & Norvig, 2010). These tools can help, not only in the reproduction of technical drawings but also raising the level of creativity and innovation – working as an extension of human activities (Mitchell, 1975).

AI is the study of computations that enable perception, reasoning, and action (Winston, 1993). It differs from most of computer science by emphasizing these three components. Looking at its goal, AI can be seen as having a mix of engineering and science. The purpose of engineering in AI is to solve real-world problems using ideas about representation and use of knowledge and operation of systems. The purpose of science is to determine ideas on knowledge representation, knowledge utilization, and to explain various types of intelligence. AI is, therefore, a computational science with systems that exhibit characteristics associated with the intelligence in human behavior, such as learning, reasoning, and problem-solving, amongst others (Turkay & Baykasoglu, 2010).

In design and industry, AI techniques related to representation, organization, and use of knowledge through computers have not only had a significant impact in reducing the time of product creation but also on improving their quality. AI has several techniques with an application in industry, such as expert systems, and rule-based computer systems that use knowledge and reasoning techniques to solve problems with a degree of complexity that would usually require an expert in the field. Consequently, an expert system can simulate the behavior of a specialist and may imitate human mental processes as closely as possible, having the ability to justify the reason behind every solution presented.

In this context, SG are an example of expert systems that work with shapes and rules that emulate the process of design, applying shape algebras to create new shapes, explore the hypothesis, and define new designs.

The use of rules is an intuitive and attractive method for knowledge representation and intelligent resolution of problems. SG, as rule systems, can certainly represent the advantage of the use of AI in the design process. In this sense, SG have great potential when coupled with intelligent systems, contributing to the creation of tools that can support creativity (Reffat, 2003).

### 2.1.2 COMPUTATIONAL CREATIVITY

*A machine is not a neutral element, it has got its history, logic, an organizing view of phenomena* (Bottazzi, 2018)

CC seeks to create computer applications that can simulate creativity in art and science (Dartnall, 1994). Combining knowledge of the area with others, such as cognitive psychology and philosophy, CC aims to reproduce creative actions, being an example of AI in reproducing a human action (Minsky, 1967) using computer programs. CC tools produce behaviors that would be recognized as creative and broaden the possibilities of response to a determined artistic project (McCormack & D'Inverno, 2012).

Creativity can have multiple interpretations. According to Margaret Boden (2004), creativity is the ability to create something new, surprising, and valuable through the exploration of conceptual spaces. CC emulates this concept. It contributes to artistic work through new ways of interaction with digital information, producing effects that simulate human drawbacks and simulate the creativity of the artist. On the other hand, algorithmic models, with the potential to produce creative results, allow the artist to acquire new solutions (Maher, 2008). In this sense, it is an area that offers the potential to assist the work of the designer, a creator, and innovator, through creative systems that allow different approaches and results. In the scope of creativity, it turns out that there are aspects of this where computer applications can intervene. From the perspective of Gero (2006), creativity resides not only in a new artifact valued by society but also in the processes that have the potential to create artifacts that can be evaluated as creative. The designer operates in a particular context and own interpretation of goals, contexts, and constraints that lead to his/her exploration of relationships and personal creative solutions.

While a broad interpretation of creativity can be given as the ability to create new, innovative and valuable ideas or artifacts, Boden (2004) argues that the question is not whether something is creative, but rather how creative it is and in what sense. New solutions can quickly receive creative endorsement when they elicit surprise, and when they can be seen as a new idea that can be inserted into an existing style, turning to a revealing surprising idea never before conceived.

Boden argues that there are three forms of creativity: (1) first form of creativity arises from creating new combinations of familiar ideas that can be generated intentionally or unconsciously. For this type of action, the mind must have a wide range of knowledge and ability to use different ways of combining this knowledge; (2) the second form of creativity is the exploration of conceptual spaces of thought. In this case, the individual explores areas of knowledge and creates new demonstrations, following the pattern of thinking of those same areas, as happens in creating the choreography of a dance; (3) third and the last form of creativity is more complex. In this case, creativity is when a process transforms conceptual spaces of thought, an idea never before thought of – supposedly impossible within the existing doctrine. In this perspective, computational creativity can exceed the boundaries of representation and discover designs that could not have been defined by existing representations. According to the author, a style transformation occurs, and new conceptual spaces are opened (Boden, 2004). Under this last perspective of creativity, conceptual spaces of thought use mental processes to exploit new creations or modify existing ones. This is where it is possible to use AI through the construction of concepts that define spaces and knowledge by testing hypotheses with structures and processes that emulate human thinking. The categories defined by Boden for creativity may also relate to the creative potential of SG addressed in section 2.1.4 of this research.

### 2.1.3 Computational Design

In the design field, computers have often been considered mere designer assistants (Akin & Anadol, 1993), providing just a set of tools that primarily facilitate representation. However, over time, the importance of the use of computers in the area of design has gained prominence (Do & Gross, 2004), (Leach & Yuan, 2018), (Menges & Ahlquist, 2011). The optimization of its use led to computers being not simple facilitators of some drawing tasks, organization, and storage of data. Instead, they gained a new status where computer and human mind complement each other. CD sets a different perspective and usefulness about computational "drawing". This is nothing more than the use of computer applications that reproduce what would be drawn by the hand of the artist. CD is, in essence, a computational algorithm, which allows the synthesis and analysis of designs (Mcgill, 2001a). This points to three key issues for the design optimization through computer applications, taking as an example, the perspective presented by (Kicinger et al., 2005) for evolutionary computation. They are: (1) The selection of appropriate representations of the drawings of the area (architecture, engineering, design, etc.); (2) The definition of efficient operators for this purpose; (3) The definition of evaluation functions, making it possible to qualify the solutions generated.

Representation in the field of design incorporates the representation of the object as well as a design process (Arciszewski et al., 1995). The evolution of computational science has allowed designers to use symbolic representations to describe objects, attributes, relationships, and concepts, and to capture abstract and conceptual knowledge of design. According to Arciszewski et al. (1995), there are three representations that computational tools for design must allow: (1) A representation of the design engineering – computational description of the object system; (2) A spatial representation of multidimensional space design – in whose design attributes can be described; and (3) Design concept – description of an object, system or abstract, in respect to the combination of its attributes and value.

Presently computer programming plays a significant role in the digital arts, extending the creative process to the execution of algorithms (McLean & Wiggins, 2012). SG are an example of production systems that, through the use of algorithms, create representations of designs, with the potential to participate in the creative process in the arts field.

Our research focal point is the use of SG computational implementations on the architectural and design practice, believing the use of AI in the arts can bring exciting new possibilities and results.

### 2.1.4 Shape Grammars

The design process is the result of a search for a structure that satisfies objectives and constraints (Arciszewski et al., 1995). SG are one of these processes. The formalism of SG was introduced in the early 1970s by George Stiny and James Gips (1972) that presented a system for rule-based generation of shapes able to describe paintings and sculptures. The basic idea of this formalism has its origins in the production system of the mathematician Emil Post (1943), in the generative grammar of the linguist Noam Chomsky (1955) and the similarity between the structure of sentences and structure patterns identified by Russell Kirsch (1964).

SG represent to geometric composition what a grammar represents for written sentences of a language. However, while the letter uses an alphabet of symbols and one-dimensional chains of those symbols (words), SG uses an alphabet of shapes with rules of composition originating visual compositions (Stiny, 1980). Stiny and Gips explain the essential principles of SG in their early articles on SG (Stiny, 1976), (Gips & Stiny, 1978), (Gips & Stiny, 1980). Synthetically, SG consist of production systems composed of a set of basic shapes considered the alphabet of shapes, and a condition-action rule set that specifies the combination of possible relations between the shapes. SG are defined through algebras of objects to which they apply addition and subtraction operations and a set of transformations. More specifically, SG are made of an alphabet of shapes that define the range of situations that need to be handled. These are a condition-action ruleset that specifies the ways of combining shapes according to spatial relationships between them, called the SG rules, and a particular shape defined as an initial shape, which must originally be present in the composition to start the SG application. Each rule is used to produce changes in the composition. A rule is applicable when the associated condition is true, and its application results in the execution of the action associated with the rule. The condition of a rule normally triggers if a given shape contained in the condition of the rule is present in the composition. The action of a rule might consist in the addition of a new shape or the removal of an existing shape in the composition.

In its simplest way, the rules are defined by the shapes on the left, as condition, or the antecedent of the rule, and the shapes on the right, which are the action or consequence of the rule. The shapes on the right, are to be replaced for the shapes on the left when these last shapes exist in the composition (Gips & Stiny, 1980). The creators of SG showed that by using geometric shapes as elements to apply rules, originating new geometric shapes combinations, it was possible to both analyze and recreate sophisticated styles of a design style and even synthesize new styles. These aspects can be and have been not only used with educational goals but also for discovery and a better understanding of the structure and principles behind particular designs (Knight, 1999), as well to create original design compositions (Stiny, 1977), (Gips, e Stiny, 1975).

Two classes of SG can be recognized and distinguished according to their function: (1) Analytical SG; and (2) Original SG. The Analytical SG are developed for the description and analysis of existing styles. Original SG, as the name implies, are developed for the creation of new styles. In essence, SG work as systems for the production of shapes that may transpose to computer production systems in the field of AI, allowing the amplification of a variety of hypotheses of the SG composition(Tomas Trescak, Esteva, & Rodriguez, 2009).

SG arose when AI already had great prominence in a variety of areas and when the first computational graphical editor was developed (Sutherland, Blackwell, & Rodden, 1980), the origin of Computer-Aided Design (CAD) applications, that gained ground as systems for design with computer support, instead of manual systems. SG have the capacity for synthesis and analysis of design/architecture/art styles and allow the creation of new design shapes integrated into a particular language or the definition of new languages (Stiny, 1980). Their use in the design areas, through computer applications, allows the role of the designer to incorporate the unique capabilities of the human mind and the ones from AI.

### 2.1.5 Computational Implementations of Shape Grammars

The maximum potential of SG is only reachable using tools based on AI, or the universe of design options that an SG can generate cannot be fully explored (Gips, 1999). The implementation requires an SG interpreter, able to perform the match of sub-shapes within complex geometries, to determine which rules should or can be applied. This way, the computer handles the tasks for computation of shapes, rules, and design solutions, while the designer defines, explores, and selects alternatives (Gips, 1999). The computational matching of shapes with the SG computational implementation is also important so that emergence – the appearance of new solutions that are created without being expected - may exist, reflecting the creative potential of SG (McCormack & Cagan, 2006). The use of an SG in a manual approach is a more intuitive, conscious, and structured way. Its use manually does not eliminate the intuitive component because the slightest possibility of hypothesis testing leads to intuitive choices that are made over the generative process (Chase, 1989). Therefore, computational implementations allow designers to study more solutions than the ones the designer would select intuitively, extending the creative process and the possible responses to the design problem at hand.

SG arose from the development of specific algorithms for the evaluation and creation of artistic objects, in the sense of creating aesthetic systems based on algorithms. Stiny and Gips (1980) pointed to the possibility of their application to analyze existing designs or create new designs since these are major areas of aesthetics and theory of art. The authors defined an aesthetic system formed by four algorithms: (1) What determines the set of interpretations; (2) what defines the references to decisions; (3) What evaluates; and (4) What compares. In an algorithm for criticism, the algorithm of analysis is used in conjunction with the one of interpretation and evaluation to evaluate if a work is artistic. An algorithm to design is used as an algorithm synthesis that builds the description of an artistic work, describing solutions that meet the conditions proposed.

The definition of SG, that is, of the algorithms referred to by Stiny and Gips, presupposes high capacities of formalization of perception and cognitive elements. The use of AI, more specifically rule-based systems, is exciting for the development of knowledge and results in the area of SG, which would otherwise be extremely difficult to obtain. It is thus natural that Gips has since 1974 presented computational interpreters for the study of shapes and their relations, even before the formal implementation of SG (Gips, 1974).

Since digital computers are symbolic-based, symbolic computations have been far more developed than visual computations, which would be appropriate for working with SG (Gips, 1999). The computational implementation of SG has to involve an equivalence between visual computations and traditional symbolic computation. The difficulty that occurs in the computational implementation of SG is between their visual nature and the symbolic nature of representations and computational processing.

Gips (1999) identifies four categories of implementations for SG. (1) The first is the most immediate implementation as it helps in the formal generation of SG. A program that responds to this task is entitled Form Shape Interpreter. This program should allow SG to be introduced in the computer, and the program should generate shapes of the language directly or by the selection of rules to be applied by the user. (2) Another type of implementation is an analysis program that allows the determination of whether a design belongs to the style of a particular SG. If the design belongs, then it should be able to demonstrate the rules and in what sequence they arise. (3) A third type of program is an inference program. In this case, the program must adhere to a set of shapes and build the SG that generates the design. The program should be able to receive a body of design elements and be able to set the SG that generate the style of these designs, producing them and new ones. (4) Finally, we can use computer-aided design programs for SG that helps the user to design them. The development of SG is identical to what the user would do manually, but with the help of the computational drawing.

The study in the field of SG implementations has evolved towards the design of conceptual tools that support the work modes of the designer, allowing results and alternatives that would not exist without the use of such tools (Chase, 2010). The designer tends to seek all solutions to a given design problem (Gips & Stiny, 1980). Project activity in the arts is a problem-solving activity where there is a search system of possible solutions and whose maximum objective is to find the ideal solution. Since it is impractical to scan the universe of solutions to a design problem manually, the logical step is to use computer programs to generate these same solutions.

Fundamentally, for the computational implementation of SG, it is necessary to define shape algorithms. SG are an example of generational design, which is determined by the method of generation of drawings by applying algorithmic sets of rules, allowing the exploration of new concepts and designs. For the definition of the algorithms, the correct representation of shapes needed to work is essential, as this defines how the rules are applied to generate designs.

The researches led by Krishnamurti and Chase were very significant for the evolution of the study of SG associated with computational implementations. Several systems of SG, both two-dimensional and three-dimensional, developed in the 1980s, served as the basis for the computational models proposed below, taking advantage of the algorithms and mechanisms developed previously (Krishnamurti & Giraud, 1986).

The representation of spatial shapes and relations in first-order predicate logic that facilitates the development of systems for reasoning in the design area have been developed in that time.

The computational systems for the realization of designs must encompass the questions related to the representation of shapes and relations between them. This is the most natural and intuitive method for the reasoning with shapes and systems based on logic as the most suitable for computational implementations of SG (Chase, 1996b).

Scott Chase presents a model of approximation to the question of relations between shapes with representations of algebraic shapes, introduced by Stiny (1992), using principles of geometry, topology, and logic to obtain more precise and generalized definitions of spatial shapes and relations. These relations can be used to describe designs in a more diversified form than simple geometric compositions. They allow the representation of conceptual aspects and the properties of algebraic shapes to overcome some limitations that simple representations based on the logic show, especially concerning the emergence of new shapes. The vector lines of the computer graphics systems define shapes that cannot be decomposed since parts of lines are not recognized. In the definition of shapes by maximal lines, the recognition of emergent shapes is possible. An emergent shape is one that contains at least one maximal line that is not the maximal line of the original shape. This allows unpredicted modifications to occur throughout the design process. It is with algebraic shapes that Scott Chase presents a model for application in the field of architecture that allows spatial relations and operations to be used for the definition of architectural plans, in which spaces emerge from the definition of the central axes of walls previously constructed by SG. The author focuses on the study of logical formulations, that is, a high level of knowledge, rather than the low level of data structures and concrete implementations. These high-level studies were essential for the development of implementations, not only because they provide the logical principles that are applied computationally but also enable us to identify the issues that affect the implementation.

From the 1990s onwards, a series of computational implementations with SG began to be developed, parallel to the in-depth study of the problems related to the representation of shapes, their transformations, and spatial relations. These implementations followed the two branches of the SG, that is, implementations of Analytical Grammars and Original Grammars.

Scott Chase kept developing very relevant work on the SG area, (Chase, 1996a), (Chase, 2005), (Lim, Prats, Chase, & Garner, 2003), (Scott & Sumbul, 2005), (Mckay et al., 2012), including the interaction model that we approach in detail in Chapter 2.2.1.

On Figure 2 below, a chronological summary of the SG development and computational implementations that have been created until 2005 is presented. This was the time frame when most of the existing work on SG was developed.

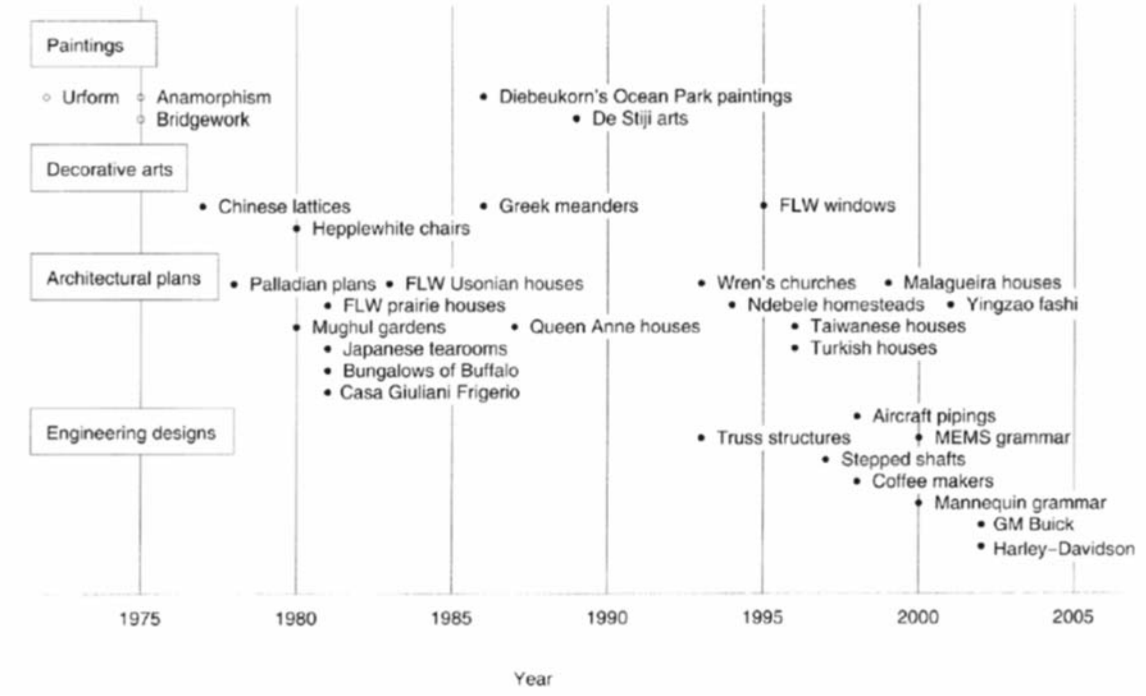


Figure 2 – Chronology of Application of Shape Grammars (Chau et al., 2004: 358)

In 2002, verifying that the development of tools for the use of SG was progressing slowly, Scott Chase (2002) refers to the lack of proper interaction between the user and the generational design systems produced. The author then proposes there is the need for a generic model of how this interaction between user and the form-based tool should happen, believing that addressing this issue better SG implementations could be created.

Scott Chase distinguishes three usage paradigms for SG implementations: (1) One in which the user has complete control, where the user chooses each rule to be applied. This resembles a non-computational system; (2) One in which the user has partial control and can select some aspects of the application of the rules; (3) One in which the SG is predefined, and the system automatically generates the designs without user intervention.

According to the author, the interaction model of an SG computational implementation must fit one or several of these paradigms and must also consider the possibility of SG being used only as part of the design process, in which the designer decides when to use it. The interaction model must present the options of user control, the various entities that interact with the SG, and the different phases of the design process in which SG is involved. It is also essential that a functional interaction model can withstand emergence. Scott Chase then pursues an analysis of these aspects in existing implementations, where he searches to identify the control levels defined on each.

It is important to highlight that Scott Chase's model presented above is one of the primary references for the proposal of this thesis. Our proposal arises from the detailed analysis of this model and the creation of an interface model based on its assumptions. This model is presented in detail in Chapter 2.

### 2.1.6 Chronology of Shape Grammars Implementations

Next, in Table 1, a chronologic list is presented, based on (Chau et al., 2004) list and completed until the year 2013, according to the existing bibliography on SG implementations.

From the thirty-four listed implementations, we present in detail fourteen of them and briefly mention eleven others. The fourteen studied more deeply are the ones that were more connected with the two fields that we are most interested in: (1) the use of SG in architecture and design: and (2) development of SG computational implementations. The eleven referred briefly report the most recent work developed, showing the interest in the field by several authors.

| | Name | Reference | Tool(s) used | Shape Emergence | 2D/3D |
|---|---|---|---|---|---|
| 1 | Simple interpreter | Gips 1975 | SAIL | No | 2D |
| 2 | Shepard-Metzler analysis | Gips 1974 | SAIL | No | 2D/3D |
| 3 | Shape grammar interpreter | Krishnamurti 1982 | Conventional language | Yes | 2D |
| 4 | Shape generation system | Krishnamurti and Giraud 1986 | PROLOG | Yes | 2D |
| 5 | Queen Anne Houses | Flemming 1987 | PROLOG | Yes | 2D |
| 6 | Miro Shapes | Kirsch and Kirsch 1988 | Conventional Language | | |

Table 1 – Comparison of the implementations of Shape Grammars (Chau et al., 2004: 360) completed with references until the year 2013

| | | | | | |
|---|---|---|---|---|---|
| 7 | Shape grammar system | Chase 1989 | PROLOG | Yes | 2D |
| 8 | Genesis (CMU) | Heisserman 1991 | C/CLP® | No | 3D |
| 9 | GRAIL | Krishnamurti 1992 | | Yes | 2S |
| 10 | Grammatica | Carlson 1993 | | No | |
| 12 | | Stouffs 1994 | | Yes | 2D/3D |
| 12 | Genesis (Boeing) | Heisserman 1994 | C++/CLP® | No | 2D/3D |
| 13 | GEdit | Tapia 1996 | LISP | Yes | 2D |
| 14 | Shape grammar editor | Shelden 1996 | AutoLISP | Yes | 2D |
| 15 | Implementation of basic grammar | Simondetti 1997 | AutoLISP | No | 3D |
| 16 | Shape grammar interpreter | Piazzalunga and Fitzhorn 1998 | ACIS Scheme | No | 3D |
| 17 | SG-Clips | Chien et al 1998 | CLIPS | No | 2D/3D |
| 18 | 3d Shaper | Wang 1998 | Java/Open Inventor | No | 3D |
| 19 | Coffee Maker grammar | Agarwall 1998 | Java | No | 2D/3D |
| 20 | MEMS grammar | Agarwall et al, 2000 | LISP | | 2D |
| 21 | Shaper 2D | McGill 2001 | Java | No | 2D |
| 22 | Chinese Houses | Li 2001 | Conventional language | No | 2D |
| 23 | U13 shape grammar implementation | Chau 2002 | Perl | Yes | 3D |
| 24 | Grammar for Buick Cars | McCormack and Vogel 2002 | C++ | No | 2D |
| 25 | MALAG | Duarte 20050 | Java | Yes | 3D |
| 26 | SG for CNC fabrication | Ertelt & Shea 2009 | C++ | No | 3D |
| 27 | MHAS | El-Zanfaly, 2009 | Autodesk Maya | Yes | 2D |
| 28 | SGDE | Li et al., 2009 | | Yes | 2D/3D |
| 29 | Shape grammar system | Jowers , et al. 2010 | | Yes | 2D |
| 30 | Virtual World Grammar (VWG) | Trescak et al. 2010 | Java | Yes | 3D |
| 31 | Grape (GRaphs and shAPES) | Grasl & Economou 2011 | C# | Yes | 2D |
| 32 | GRAMATICA | R. Correia et al., 2012 | Rhino | Yes | 3D |
| 33 | DESIGNA | Correia, 2013 | CAD | Yes | 3D |
| 34 | SG based on Graphs | Strobbe, Meyer, & Campenhout, 2013 | | | |

Table 1 (cont) – Comparison of the implementations of Shape Grammars (Chau et al., 2004: 360) completed with references until the year 2013

**Shape Grammar Interpreter** (Krishnamurti, 1982)

Ramesh Krishnamurti, professor, Carnegie Mellon University was one of the first pioneers of the study of shapes (recognized in 1980 by Stiny and Gips) (Krishnamurti, 1980). He created algorithms for Boolean operations and computational data structures for the implementation of shape algorithms (Ramesh Krishnamurti, 1982). He and Giraud (1986) presented the possibility of using shape rules as a mechanism for affecting changes on partial shapes in drawings. They introduced this system in PROLOG. C. F. Earl (Centre for Configurational Studies, Faculty of

Technology, Open University, Milton Keynes MK7 6AA, Bucks, England) and Krishnamurti (1992) presented a model of shape recognition in three dimensions. Shang-Chia Chiou (1995) (School of Design National Yunlin University of Science and Technology, Douliu City Taiwan, ROC) and Krishnamurti presented Subshape that presents a grammar that allows the recognition and definition of traditional Taiwanese houses.

Krishnamurti defines so-called rational shapes, that is, shapes defined by points whose coordinates can be expressed as a ratio of two integers as an ordered pair (a, b). Given the definition of maximal lines ( Stiny, 1980) in which each shape is composed of a set of maximal lines, a shape is rational when its maximal lines are rational, as well as its limit points, that is, the coordinates of the points must be pairs of integers. The algorithms presented for SG have as fundamental information the maximal lines and the labeled points of the labeled shapes to which they apply. The author defines the shape algorithm where computational mechanisms are required. These mechanisms are a mechanism for determining if two lines are collinear and a mechanism for determining the coincidence of points with lines. The representation of maximal lines of a shape is applied in these two mechanisms, allowing a correct representation of the shapes in the implementation. After the definition of algorithms for the representation of shapes, based on the maximal lines, algorithms are presented for the Boolean operations in the shapes (union, difference, and intersection) and relations between shapes (definition of equivalent shapes and subshapes). Finally, he defines the data structure necessary to implement the presented algorithms, based on the programming language ALGOL.

**Shape Grammar System** (Chase, 1989)

Scott Curland Chase, an independent researcher, followed the method proposed by George Stiny to define a universal computational system for SG and has published many articles in computer-aided design, generative design and building information modeling.

Following the method proposed by Stiny for the representation of shapes through their maximal lines, Scott Chase (1989) presents a generic computational system for SG with the representation by sets of lines, starting from the algorithms for Boolean operations and data structures of Krishnamurti referred above. After the presentation of the algorithms for the representation of shapes and operations, like Krishnamurti, Scott Chase has an implementation of the generic system for SG, defining three designer modes of interaction with the system to serve these purposes, mentioned briefly in 2.1.5. These three modes are called deterministic, non-deterministic (Figure 3), and autopilot.

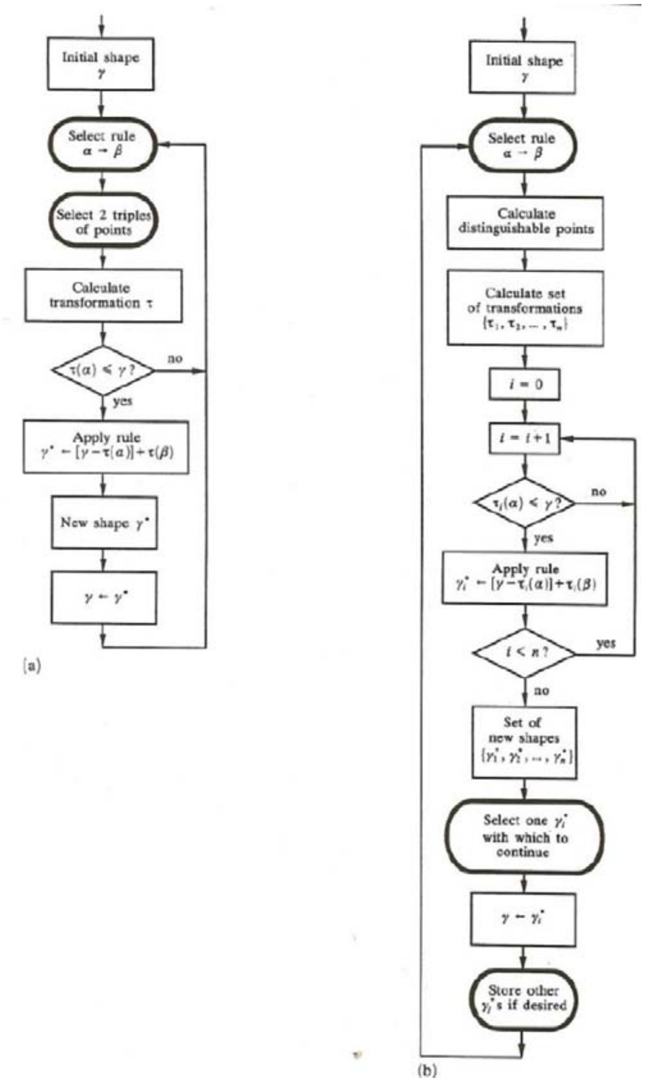
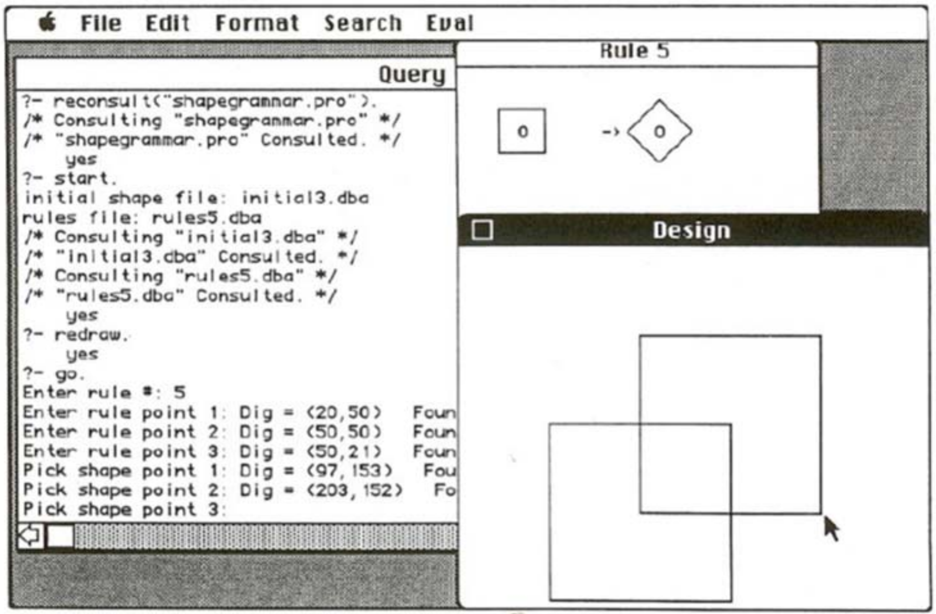

Initial shape $\gamma$
Select rule $\alpha \to \beta$
Select 2 triples of points
Calculate transformation $\tau$
$\tau(\alpha) \leq \gamma$?
no
yes
Apply rule $\gamma^* \leftarrow [\gamma - \tau(\alpha)] + \tau(\beta)$
New shape $\gamma^*$
$\gamma \leftarrow \gamma^*$
(a)
Initial shape $\gamma$
Select rule $\alpha \to \beta$
Calculate distinguishable points
Calculate set of transformations $\{\tau_1, \tau_2, \dots, \tau_n\}$
$i = 0$
$i = i+1$
$\tau_i(\alpha) \leq \gamma$?
no
yes
Apply rule $\gamma_i^* \leftarrow [\gamma - \tau_i(\alpha)] + \tau_i(\beta)$
$i < n$?
yes
no
Set of new shapes $\{\gamma_1^*, \gamma_2^*, \dots, \gamma_n^*\}$
Select one $\gamma_i^*$ with which to continue
$\gamma \leftarrow \gamma_i^*$
Store other $\gamma_i^*$'s if desired
(b)

Figure 3 − Deterministic and Non-Deterministic Mode for Application of Shape Grammars (Chase, 1996:277)

In the deterministic mode, the designer can select a rule and determine a single transformation applied to a labeled shape, being this the traditional way of using a SG, in which the designer controls the entire system without computing. In a non-deterministic way, the designer can select the rule to apply to the current shape, after which the system is responsible for calculating the set of all possible transformations and results of applying the rule. In this way, the designer can select a new shape from the resulting one to continue to generate and store the chances not selected. This mode has the advantage of eliminating the need to specify the desired transformation explicitly and opens the hypotheses of presenting emerging shapes.

However, the exploitation of all the possibilities of applying a rule has disadvantages because it can be tangentially infinite. This leads to the production of incomplete sets of options. In the method of "automatic pilot", the designer lets the system select a rule in the SG, which then calculates the possible transformations and selects the application of the rule. In this case, the designer is involved in setting the rules and not its application. With the definition of system-operating modes and their relationship with the designer, Scott Chase demonstrates the implementation of the system in the programming language PROLOG (Figure 4). He shows the advantage for the system builder of changing the user role between selecting the rules or producing them. The limitations that the system presents are mainly related to the power of computation, especially in the area of shape recognition in the graphics. PROLOG, as a declarative language, facilitates the development of the SG system, but the author acknowledges that it may not be very efficient in data structures and algorithms because it has problems regarding memory management.



```
?- reconsult("shapegrammar.pro").
/* Consulting "shapegrammar.pro" */
/* "shapegrammar.pro" Consulted. */
    yes
?- start.
initial shape file: initial3.dba
rules file: rules5.dba
/* Consulting "initial3.dba" */
/* "initial3.dba" Consulted. */
/* Consulting "rules5.dba" */
/* "rules5.dba" Consulted. */
    yes
?- redraw.
    yes
?- go.
Enter rule #: 5
Enter rule point 1: Dig = (20,50)   Foun
Enter rule point 2: Dig = (50,50)   Foun
Enter rule point 3: Dig = (50,21)   Foun
Pick shape point 1: Dig = (97,153)   Fou
Pick shape point 2: Dig = (203,152)   Fo
Pick shape point 3:
```

Figure 4 – Implementation in PROLOG: an example of assignment in the use of Form Grammar (Chase, 1989: 234)

**Genesis (CMU)** (Heisserman, 1991)

Jeff Alan Heisserman (1991), Associate Technical Fellow at Boeing introduced what he called the Boundary Solid Grammars, which he later embodied in a PROLOG program for the  Queen Anne-style houses studied by Flemming (1986), that is, an implementation for an Analytical

Genesis Grammar (Heisserman, 1994). He aimed to bypass the limitations of CAD systems when using computational graphics support for the design of models of a particular style with specific constructive and spatial components. He also bypassed limitations of the geometric and mathematical representation used in the computational implementations of SG to date. He proposed Mechanisms for implementations that can work with solids and provide SG that generate solid languages. Thus, he presents the Genesis prototype, a formalism with a three-dimensional SG for the houses of the Queen Anne style. In this implementation, instead of formalizing the representation of two-dimensional shapes, the representation of solids is formalized. This representation of solids has 4 essential notations: (1) a topological graph that exemplifies how the various elements of solids relate; and (2) coordinates in a three-dimensional space associated with each vertex; and (3) a set of labels to identify the nodes of the topological graph; and (4) A state associated with each instance of the representation. The topological graph below consists of nodes representing the elements (such as vertices, edges, faces, solid and solid inverse) and arcs representing the adjacencies (Figure 5).

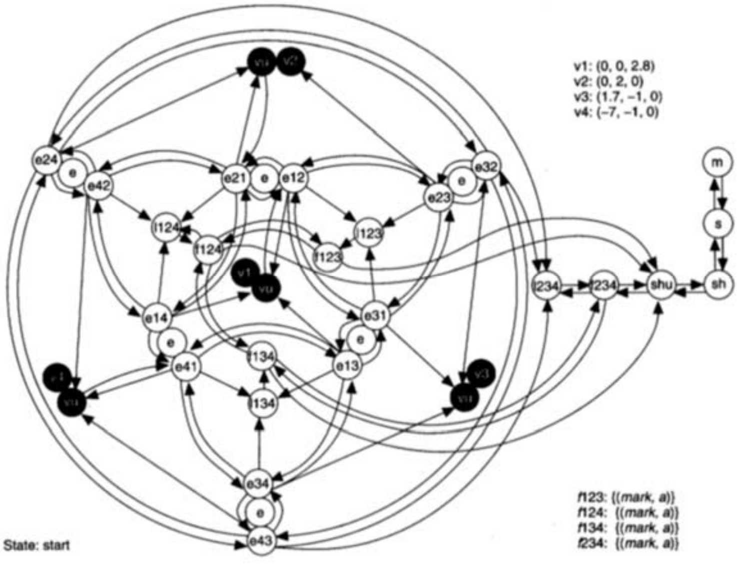


Figure 5 – Topological graph for the envelope of a tetrahedron (Heisserman, 1994:39)

The state associated with each labeled element allows an efficient computational mechanism to determine whether or not a given generated design is a member of the grammar language, also to indicate or restrict the rules that can be applied in a particular state of the design. Thus, this implementation, although demonstrated for Analytical Grammar, is important because it allows reasoning with solids, operations on solids and rules for solids, giving rise to solid models that can serve several design domains (Heisserman & Callahan, 1996), (Borkowski, Gross, Heisserman, Knight, & Nakakoji, 1998).

The Genesis implementation was later harnessed to Boeing's design of hydraulic aircraft systems, demonstrating its diversified operation and applied to particular industrial design needs.

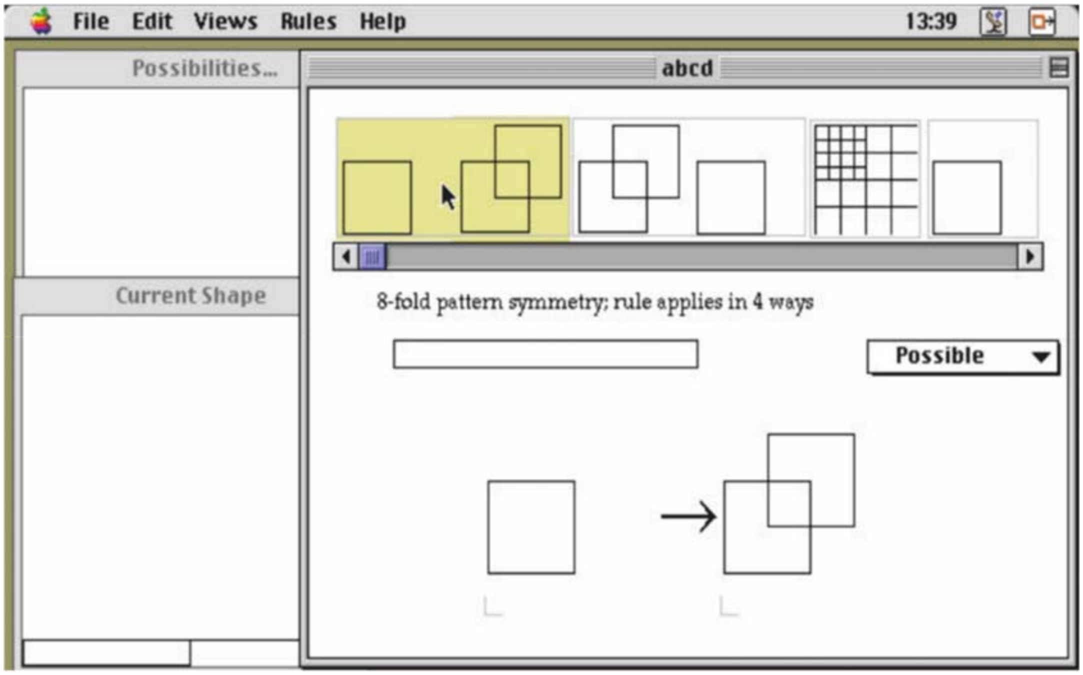
**Miro shapes** (Kirsch & Kirsch, 1988)

In 1998, Joan Kirsch and Russel Kirsch present, in the area of painting, a SG for Miro paintings implemented in Macintosh Common Lisp (1988). In the interpretation of artistic works, it is always relevant to understand not only the work but the process for its realization, which is not always achieved. When it is possible to recognize a homogeneous collection of works of art, that is, a style, its communication becomes more precise and more evident in a series of common characteristics, such as shape, color, composition, and texture, among others. The use of computational tools for algorithms that allow us to describe patterns and shapes, also allows us to approach the art object in the same way as scientific results, that is, the object's understanding can be tested, validated and applied. The descriptive process, which allows the in-depth understanding of the works, loses its interest if it cannot be used in future works. However, the best computational applications are the ones that hardly allow the analysis of new occurrences. In this way, the authors introduce as future research the topic of creating "intermediate power" algorithms, which are not fully descriptive, allowing the analysis to be performed (Kirsch, 1998).

**GEdit** (M. A. Tapia, 1996)

M.A. Tapia, MIT, published an article in 1992 on a mechanism for performing recursive shape computations. He established a general paradigm for computer implementation. Tapia (1996) introduces GEdit, a system for the realization of the Chinese Ice-Ray designs presented by Stiny. This first implementation of SG from Tapia arises in the programming language LISP. GEdit is a visual system implementation for SG in which the main focus of the computational implementation was the interface and communication with the designer and end user. In computational implementations, the computer performs the tasks of representation and computation with shapes, rules, and grammars for the correct presentation of design alternatives, leaving the role of the designer reserved for the development of the design languages and selection of the alternatives that are returned to him. However, computational implementations to date have focused on the development of algorithms for SG but have not been dedicated to the interface and visualization of the SG system as a whole. Tapia defines three phases for the development and use of SG: 1) SG creation, in which the designer defines the rules, initial shapes and verifies the logical and spatial constraints; 2) Compilation of SG, in which the system checks the constraints of the rules and desired shapes in which they can be applied; 3) Exploration of the design language defined by the SG, in which the designer can then impose additional constraints in order to manipulate the generational system.

Thus, in a visual implementation, the designer specifies the SG, the system presents its hypotheses, and the designer explores the results. In this way, the computation means the design of the artistic object of SG. For better interaction between the computational design and the designer, the interface must be graphic and simple, allowing the designer to directly manipulate the objects and concentrate exclusively on them, not on hidden menus and commands that force the designer to divert his attention. The Tapia interface has multiple windows for simultaneous presentation of the various hypotheses created by the application; it has menus that appear automatically in the window, allowing the designer to make his choices without changing the environment. After the first definition, he continued to work on the SG field (Tapia, 1999). In Figure 6, the interface of GEdit is presented.



8-fold pattern symmetry; rule applies in 4 ways

Figure 6 – Interface GEdit (Tapia, 1999: 14)

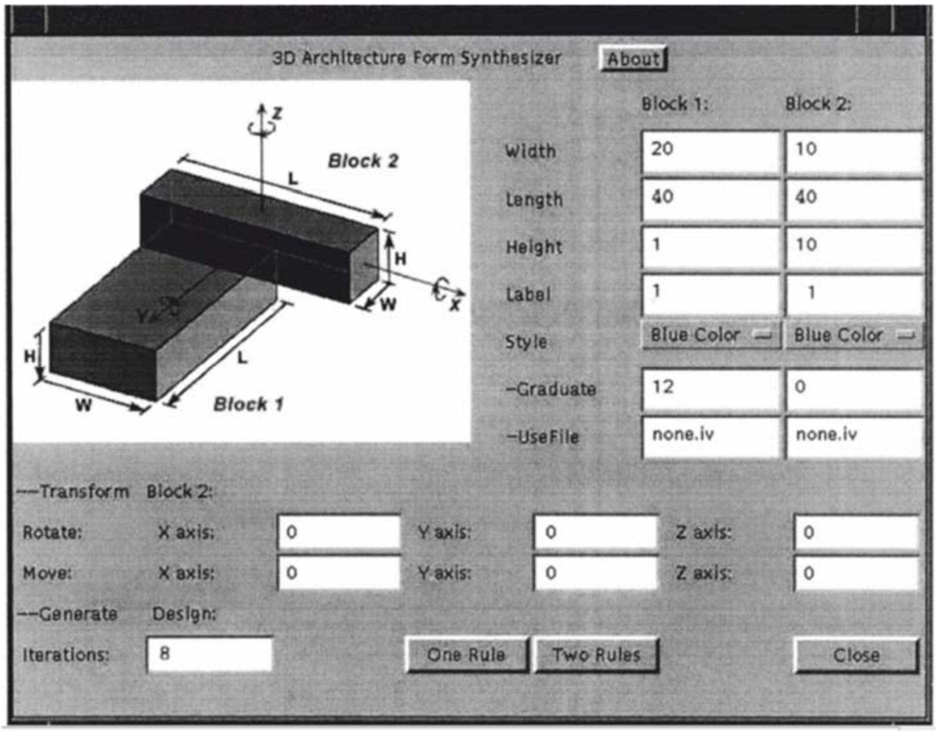
**Shape Grammar Interpreter** (Piazzalunga & Fitzhorn, 1998)

Ugo Piazzalunga, Department of Mechanical Engineering, Colorado State University, Fort Collins, CO, USA, and his colleague Patrick A. Fitzhorn, also at Colorado State University, published an article on 3D implementation based on a commercial solid modeling kernel and an associated functional language.

In 1998 Piazzalunga and Fitzhorn (1998) addressed four issues related to implementation difficulties, pointing four areas of complexity for the interpreters of implementations of SG: (1) Data structures for the representation of line sets, mostly maximal lines, set of planes or sets of lines and planes. These are declared and maintained during the execution to include a hierarchy of sets of shapes, for example by means of labeled points or labels for its own form; 2) Algorithms for form operations that work with data structures and allow spatial transformations, shape equalities, and subshape detection, among others that support most implementations in integer arithmetic; 3) In the detection of subshapes, the determination of all possible linear transformations when one form becomes a subshape of another as this case is difficult in three dimensions; and 4) General algorithms for the development of a simple grammar that encompasses the difficulties of the three previous areas, and its implementation is not linear. Following these principles, Piazzalunga and Fitzhorn developed an interpreter of SG in ACIS Scheme that can be applied in three-dimensional SG, but without emergence support.

**SG-Clips** (Chien, 1998 )

Also, in 1998, a group of students presented two interesting implementations in the area of SG. One of these implementations was the SG-Clips (Chien, Donia, Snyder, & Tsai, 1998), derived from the tool that supports it, CLIPS. This is a tool for rule-based systems created by the Software Technology Branch (STB). SG-Clips intends to be a tool that allows the exploration of designs generated by SG with a good interface. The system consists of a programming environment, an inference engine and a graphical interface. The programming environment provides primitive shapes, points, lines, and shapes for the representation of design objects. Rules are applied to Primitive shapes, producing new representations. The CLIPS tool is used as the inference engine, determining the rules that apply to shapes, executing them, and creating new designs. The graphical interface allows the user to choose an SG and explore the designs it can produce. The possible shapes in this SG implementation are only two-dimensional and do not allow for emergent shapes. The authors recognize the limitations of CLIPS, because they consume more memory than, for example, C ++-based programs, and the graphical interface is also limited concerning user interaction and update (Figure 7).



|  | Block 1: | Block 2: |
|---|---|---|
| Width | 20 | 10 |
| Length | 40 | 40 |
| Height | 1 | 10 |
| Label | 1 | 1 |
| Style | Blue Color | Blue Color |
| -Graduate | 12 | 0 |
| -UseFile | none.iv | none.iv |

---Transform Block 2:

Rotate: X axis: 0 Y axis: 0 Z axis: 0

Move: X axis: 0 Y axis: 0 Z axis: 0

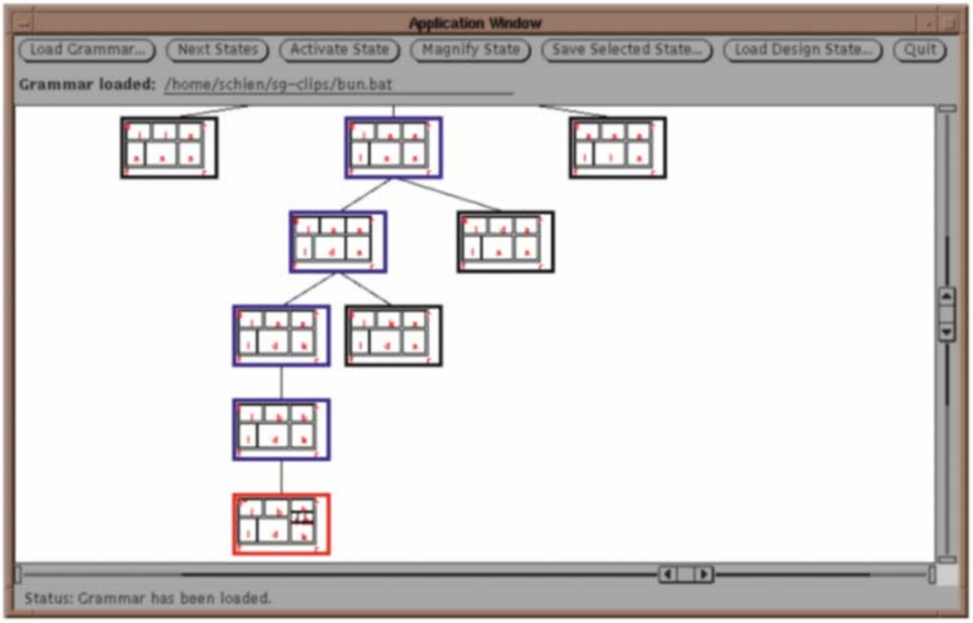
---Generate Design:

Iterations: 8

Figure 7 – Interface do SG-Clips (Chien, 1998: 4)

**3D Shaper** (Wang, 1998)

Yufei Wang, Department of Architecture, MIT, published in 1998 his article was describing how architects would be able to work in a virtual 3D environment, generating designs with the aid of a creative computer. Wang presented the 3D Shaper (1998) a three-dimensional system with a focus on allowing the designers to be able to play an active role in the design process. However, it is a system that does not support emergence, based on Java / Open Inventor. The programming language used allows the tool to be run via the internet, which allows reaching a greater number of people. Spatial relationships were programmed according to Stiny's three-dimensional definitions (Figure 8).



Grammar loaded: /home/schien/sg-clips/bun.bat

Status: Grammar has been loaded.

Figure 8 – Interface for 3D Shaper (Wang, 1998: 28)

**Coffee Maker Grammar** (Agrawal, 1998)

Noteworthy in the area of design was the SG of coffee machines presented in 1998 (Agarwal & Cagan, 1998) (Figure 9). Jonathan Cagan, Professor of Mechanical Engineering, Carnegie Mellon University, and Manish Agarwal, Department of Mechanical Engineering, Carnegie Mellon University made a significant contribution to SG implementation in shapes of product design (1997).

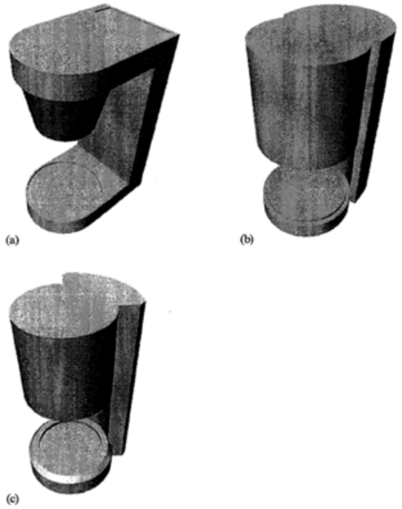


Figure 9 – New coffee machine designs obtained by shape grammar (Agarwal, et al., 1998: 224)

The coffee machines SG generates a series of designs for coffee machines that existed in the market at the time. This was possible because coffee machines, before any aesthetic design, have an identical mechanism, which implies an equal sequence of parts, to ensure the correct functioning. This type of SG adapts to other groups of industrial objects, such as toasters, telephones, ovens, mixers, among others, which vary in their aesthetic form but follow the same mechanical and functional principles. SG for product design allows conceptual design support as well as a powerful exploration of variants that can be obtained by manipulating one or more parameters or one or more rules. The design supported in the SG also allows a particular company to define rules for the image of its products so that it remains recognizable in various hypotheses or evolutions of the products.

On the other hand, companies can also translate the product of their competitors into a grammar, allowing a more in-depth study of the image with which they compete in the market. Although the authors recognize that SG allows greater control and reliability in the design of standard products, this can help in the development of new design concepts more quickly than the designer could achieve manually. In particular, the SG for Coffee Machines is divided into functional design and formal design. The function precedes the shape, and SG follows that sequence. SG is parametric and two-dimensional, but allows the creation of the top, front and side views of an object, reproducing a three-dimensional shape. The coffee machines have three main units, the filter, the water tank, and the base. The set of these three elements defines the initial shape. The first set of rules separates two classes of machines, those that have a heating element and those that have two, and then applies the rules to the design of the three regions of the initial shape. After the rules for drawing the three functional parts, rules are applied to join them, forming the machine entirety. The implementation of this SG was performed in Java, not allowing emerging shapes.

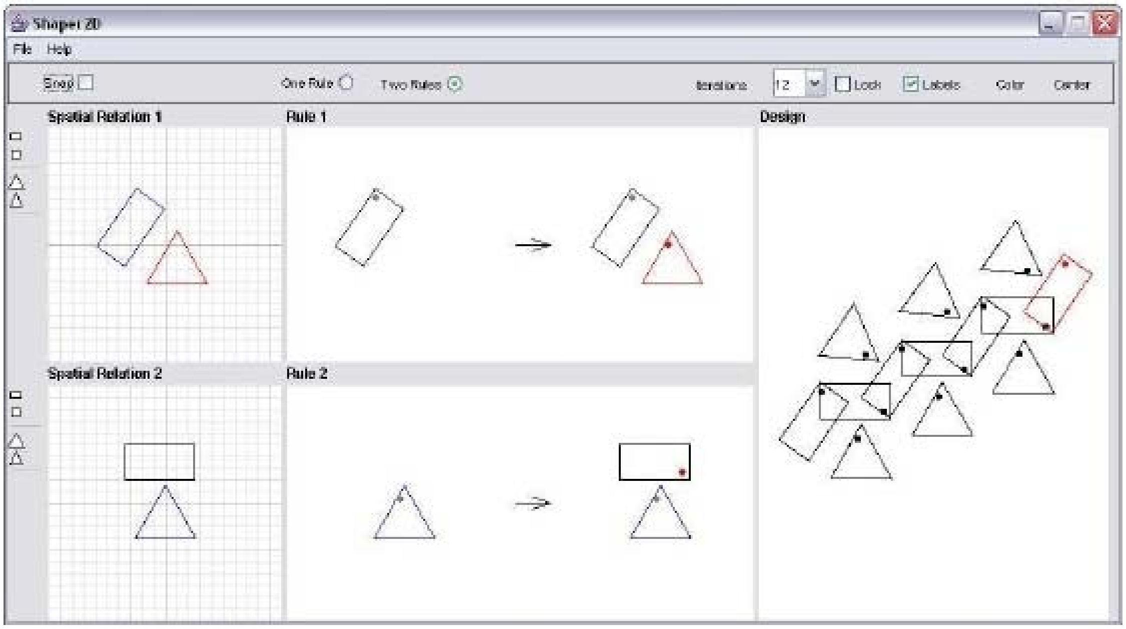
Further work was developed in (Agarwal, Cagan, & Constantine, 1999), (Agarwal, Cagan, & Stiny, 2000) and (Agarwal et al., 2000).

**Shaper 2D** (McGill, 2001)

M.C. McGill, MIT, published two important papers on using a visual approach for exploring computational design and Shaper2D, a software for use in learning SG.

In 2001, Miranda McGill presented Shaper 2D, a dynamic visual tool for the use of SG (2001a). To explore the power of computational design and how it can be an inspiration for creativity, in a way different from computer-assisted design, the author proposed an implementation based on an SG interpreter with specific tools for its objectives; she was not interested in universal tools. Thus, the purpose of Shaper 2D is the exploration of basic two-dimensional SG in a perspective of learning and understanding of the potentialities of computational design based on SG. This application has been written in a way that is not limited to any platform, operating on Windows, Mac OS, and Linux systems and any web browser running Java.

The Shaper 2D was built to have an appealing interface for designers, whose interactivity allowed the use of grammars to be as transparent as possible. The rules are hidden from the user, but the result of their application is graphically demonstrated immediately, allowing the designer to perceive the generation process directly and to experiment with different results by applying different rules and iterations, comparing the results. The shape rules applied by the program are linear transformations that use common algebraic techniques. Transformations such as rotation and translation are represented by matrices that act on augmented vectors, allowing combinations of transformations to be made at once (Figure 10).



Snap □ One Rule ○ Two Rules ● Iterations 12 □ Lock ☑ Labels

Figure 10 -Shaper 2D Interface (McGill, 2001)

The Shaper 2D interface consists of a menu bar, five windows, and two toolbars for selection of different shapes. The five windows are divided into pairs of two that allow the choice of spatial relationships and rule applications with a window that presents the final result. Whenever the user manipulates a shape in the spatial relations window, the rules application window and the final result window are dynamically updated in real time. The number of iterations can also be manipulated with a response in the final design window, allowing the designer to visualize the possible results. This application is not a plug-in for the commonly used vector application for architects and designers - AutoCAD - and cannot be connected to it. But as AutoCAD is in widespread use by the designers, Shaper 2D allows the drawings obtained to be exported to DXF.

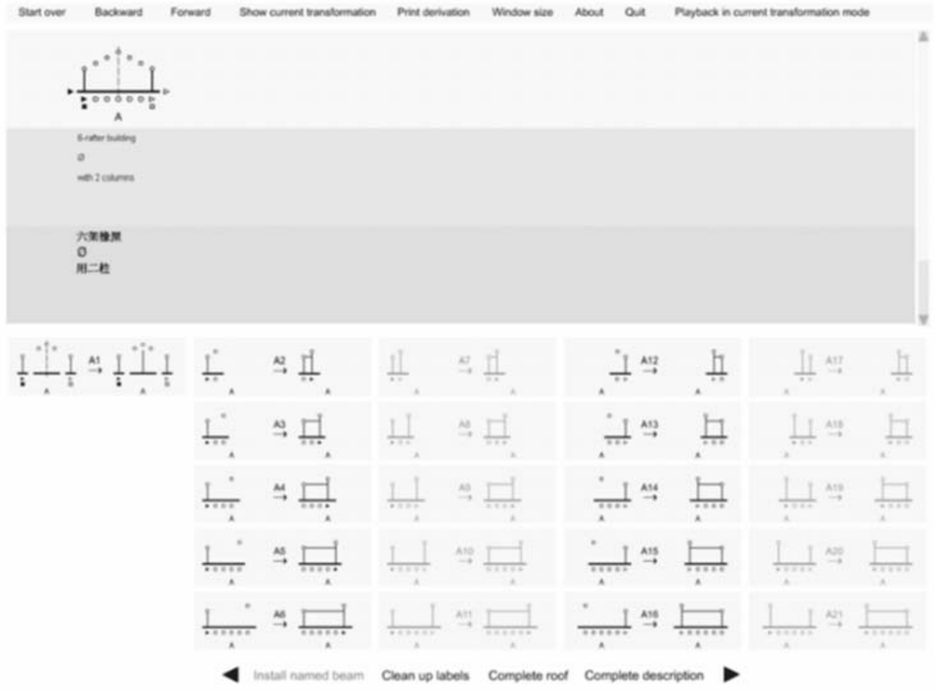
**Chinese Houses** (Li, 2001)

Andrew I-Kang Li, Associate Professor in design, management, and technology at the Kyoto Institute of Technology (KIT), Japan, continues to teach and publish articles on computational design. He began these publications in 2001 with a thesis at MIT, which illustrates the use of SG for teaching the architectural style of the Yingzao Fashi (2001).

In 2004, Li developed an interpreter that was implemented as an Analytical Grammar in Multimedia Flash for the presentation of sections of the Chinese wooden houses of the XII century. The interpreter was implemented in Flash for its relatively easy use and for allowing maximum control of the interface. The interpreter was directed to the teaching of the architectural style in question. This was important for the interaction between the user and the interpreter.

The implemented SG is parametric and has descriptions ( Li, Lau, Kuen, & Kuen, 2004). Each design starts from an initial shape that is a section with descriptions of what it has, such as columns, base, vertical axis, labels, and symbols.

The grammar is developed in four stages. The first defines the main building elements, such as columns and beams, and these are inserted in the section and in the descriptions. In the second step, the necessary labels in the previous phase are eliminated. In the third stage, the required constructive elements are placed for the conclusion of the design and, finally, in the fourth stage, the descriptions are reduced to the generic ones of the style (Figure 11).



6-rafter building
Ø
with 2 columns

六架椽屋
Ø
用二柱

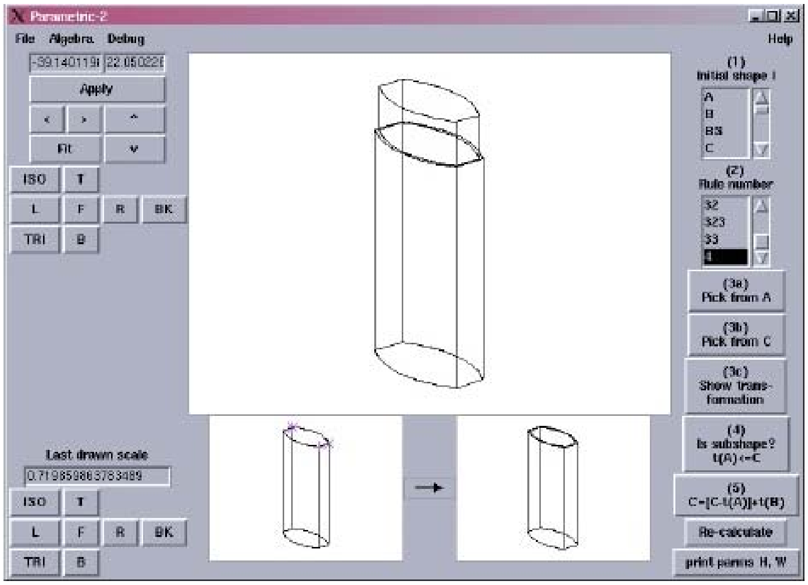
Figure 11 – Initial panel of the interpreter for sections of traditional Chinese houses of the XII century (Li, et al., 2004: 4)

This implementation, which is accessed via the Web, presents the user with a sequence of questions identical to those that the architect would ask the client in a first meeting, to be able to perceive the housing needs, such as the characterization of the family members, their habits, the number of necessary rooms, what budget is available, among others. With this data, the program defines a script that also considers the restrictions and laws that apply to the dwelling in question. The user then can validate the script or introduce changes to the requirements indicated initially, and the program updates the script automatically. With the script approved, the program generates a housing solution that meets the objectives.

In this implementation, a mathematical model was used that has three computational methods to solve three essential questions, which the author calls Problem Simulation, Problem Generation, and Problem Optimization. In Problem Simulation, it is necessary to translate the requirements given by the customer and verify that the design meets these requirements. In the Generation phase, the application produces design configurations that comply with the previously defined rules. Finally, in the Optimization phase, the previous configurations are used to create a housing solution whose design obeys the overall goal. The number of solutions is potentially wide.

**U13 Shape** Grammar Implementation (Chau, 2002)

Also, in 2002, Chau (2002) developed an implementation in the programming language Pearl that supports SG with rectilinear and curvilinear entities in a three-dimensional space, called U13 (Figure 12).



-39.140119 22.050226

Last drawn scale 0.719859863783489

Figure 12 – U13 implementation (Chau, 2002: 368)

This implementation uses mathematical representations for the basic elements allowing a simpler representation of maximal lines and computation with them. To validate the use of the implementation and usefulness of SG, the author presents two case studies of widespread commercial products (Coca-Cola bottles and Head&Shoulders shampoo bottles), showing how SG can be used for Product Design.

**Grammar for Buick Cars** (McCormack and Vogel, 2002)

In the same year of the study, 2002, McCormack and Vogel present a grammar interpreter that allows the recognition of subshapes, implemented in the programming language C ++ (McCormack, Cagan, & Vogel, 2004). In contrast to existing interpreters, limited to the recognition of parametric (non-emergence) shapes, or interpreters that perform recognition of non-parametric subshapes, this interpreter allows the identification of parametric subshapes, allowing emergence and reunion for the first time. The identification of parametric shapes and subshapes resides in the same implementation.

The method used for the recognition of parametric subshapes is based on the hierarchical decomposition of two-dimensional shapes. The hierarchy is defined by a set of spatial relationships that exist in the framework levels of hierarchy or decomposition. The hierarchy is based on the assumption that the designers specify the rules so that they can manipulate the relationships between line segments present in the shapes. Groups of line segments are then recognized, which align the hierarchy. The first group, s1 (Figure 13), is composed of the lines subject to greater restrictions, which are those that intersect perpendicularly and have the same length and lines symmetrical to two or more lines that are not parallel. The group s2 (Figure 14) is composed of lines that intersect perpendicularly and lines that are symmetrical to a single line or several parallel lines.

The group s3 (

Figure 15) is composed of lines that only intersect, and the group s4 (Figure 16) is composed of the lines without spatial relations, requiring additional information to be paired. Hierarchical decomposition is performed in such a way that the rules-defining shapes reflect the intentions of the author of the rules, which should then dominate how the hierarchy works.

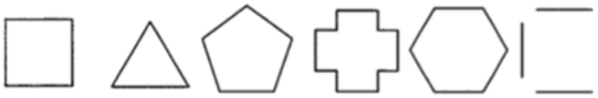


Figure 13 – Group Lines S1 (McCormack, et al., 2002: 917)

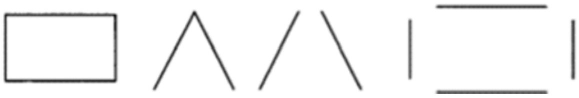


Figure 14 – Group Lines S2 (McCormack, et al., 2002: 917)

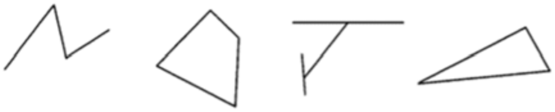


Figure 15 – Group Lines S3 (McCormack, et al., 2002: 918)

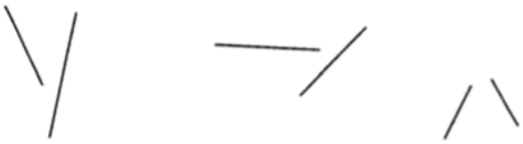


Figure 16 – Group Lines S4 (McCormack, et al., 2002: 918)

Thus, the McCormack and Vogel interpreter can be used to implement any grammar in a way that is based on pairing to determine the application of the rules and the fact that it allows the use of curved shapes. In this way, one can explore shapes that are not represented only by straight lines, adapting to the true needs of the design. After presenting SG for the front design of the Buick cars, (McCormack et al., 2004) demonstrated the usefulness of SG for the translation of the rules that define the language of a brand. The fact that it allows the use of curved shapes allows one to explore shapes that are not only represented by straight lines, thus adapting to the true needs of the design.

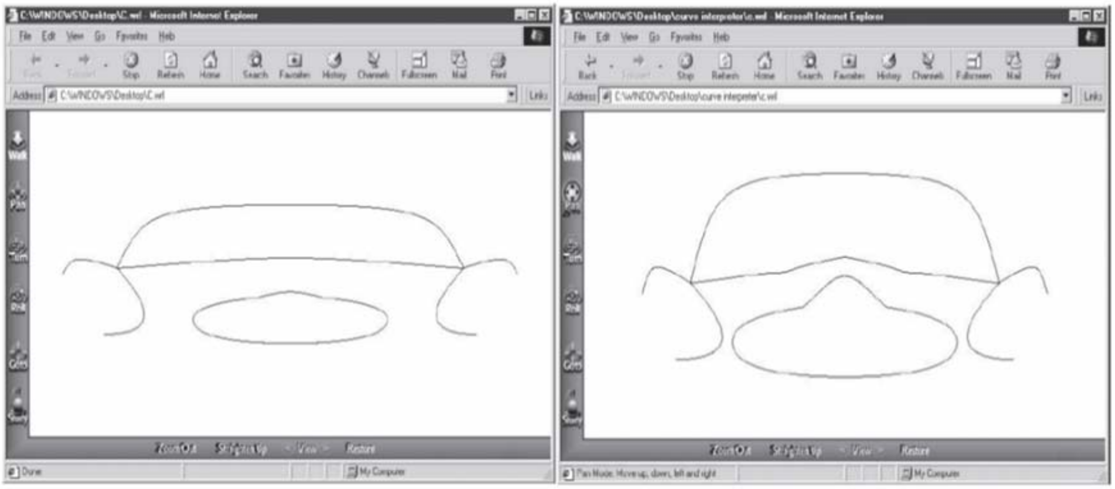

C:\WINDOWS\Desktop\C.wrl - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address C:\WINDOWS\Desktop\C.wrl
Links
Walk
Pan
Turn
Roll
Goto
Study
Zoom Out
Straighten Up
View
Restore
Done
My Computer
C:\WINDOWS\Desktop\curve interpreter\c.wrl - Microsoft Internet Explorer
Address C:\WINDOWS\Desktop\curve interpreter\c.wrl
Pan Mode: Move up, down, left and right

Figure 17 – Results of Grammar implementation for Buick cars (McCormak, et al., 2006: 537)

After this grammar, other works were developed seeking the definition of SG for product design and architecture. He Sungwoo Lim, Lecturer, Department of Engineering and Innovation, Open University, UK, and Miguel Prats, Argentinian architect, and Steve W Garner developed important work using SG in architecture (2003). Sumbul Ahmad and Chase (2005) used composite grammars to elucidate hybridity in design.

**MALAG** (Duarte, 2005)

José Pinto Duarte, Department of Architecture, Penn State Landscape Architecture Stuckeman School, is the Chair in Design Innovation and Director of the Stuckeman Center for Design Computing. He continued to be a scholar and researcher of the highest status and began this distinguished career in 2005 with the publication of two papers illustrating the use of discursive grammar for customizing mass housing using Siza's houses at Malagueira.

In 2005 Duarte introduces the concept of Discursive Grammar and Descriptive Grammar (Duarte, 2005a). Descriptive Grammars consist of symbolic descriptions, whereas in SG, there are descriptions of shapes. Discursive Grammars consist of SG with a Descriptive Grammar and a set of heuristics. Descriptive Grammars are used to choose the rules to apply to each step of the design generation or to restrict the choice to a limited number of rules. Discursive Grammars select which generated designs best meet the requirements. In this way, Discursive Grammar refers to the mathematical model that leads to the formal and semantic generation of correct designs. In another point of view, the Discursive Grammar is formed by a Grammar for Programming and a Grammar for Drawing. The Programming Grammar allows the creation of the script for housing through the processing of the requirements to be fulfilled. This grammar has only one descriptive part. The Grammar for Drawing uses the application for the general design solution, having a descriptive component and a drawing component (Figure 18).

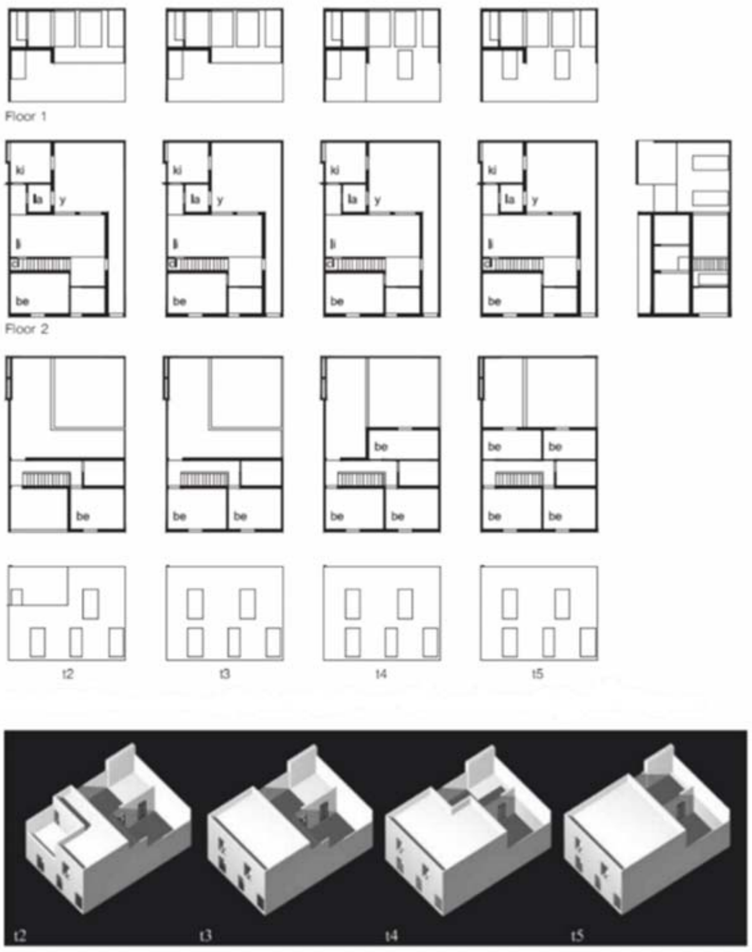


Figure 18 – Plants, cuts, elevations, and axonometries of four variations of a Casa Pátio da Malagueira (Duarte, 2005: 377)

This aspect leads to the possibility that, in theory, it is possible, for the same Grammar of Design, to conjugate different Programming Grammars, giving rise to varied Discursive Grammars. Following this study, Duarte developed an SG based on Alvaro Siza Vieira's Malagueira housing projects, using for his research the corpus of 35 houses designed and constructed in Évora between 1977 and 1996 (Duarte, 2005b). House generation of Malagueira' style is accomplished through a Design Grammar called PAHPA-Malagueira that generates housing solutions of the defined style. The generation of houses in Malagueira is based on the manipulation of rectangles and the use of the rules that function in them that assign them functions. The generation begins by dividing the lot into four functional zones (patio, living area, service area, and sleeping zone) and adding a ladder, which defines the type of house. The zones are then divided into compartments to obtain the layout. The drawing rules have a formal part and a descriptive part, as explained above. The formal part uses a series of algebras for the representation of several points of view (plants, elevations, cuts, axonometries). The descriptive part includes a program description and a design description, the elements of which are directly linked. In summary, MALAG is the computational implementation of the PAHPA-Malagueira grammar, which, in addition to automating the design of Malagueira's home designs, also serves as a design assistant, allowing the user to explore design hypotheses.

**OIDS** (Celani, 2005)

Gabriela Celani is an Associate Professor at the School of Civil Engineering, Architecture and Urban Design at the University of Campinas, Brazil. She is the founder of LAPAC, the Laboratory for Automation and Prototyping for Architecture and Construction. Her paper with R. Pupo, G. Mendes, and E. Pinheiro in 2005 illustrated generative design systems for housing called OIDS (2005). In OIDS, the use of SG as a generational tool is adapted to the new needs of architects. Due to the importance of the construction constraints, OIDS intends to consider the conditions of the surroundings from the natural characteristics of the terrain, such as the influences of the constructions already existing in the place. The proposal for OIDS was an application capable of addressing important architectural issues, such as the design process, environmental comfort, and design framed with its context and external surroundings to achieve more organic and integrated housing.

**SGTOOLS** (Romão, 2005b)

Also in 2005 Luís Romão, architect and teacher that holds a Ph. D. in architecture: design and computation, from the School of Architecture and Planning, MIT, developed an SG implementation called SGtools (Romão, 2005b). The main goal of this tool was to facilitate the use of SG by those who didn't have a programming background and the vision was also to allow architects and designers to make use of the computational abilities in their practice. The author further explores SG to study illegal housing in Lisbon from 1974 to 1984, defining a SG that describes the morphology of their architecture composition (Romão, 2005a).

**TRIGGER** (Biswas, 2006)

K. Biswas is a professor at the School of Engineering and Applied Sciences in the Computer Science Engineering at Bennet University, India. He continues to publish and research video processing, machine learning, and fuzzy modeling. His MIT paper on a computational model of virtual interpretation in 2006 was yet another step in the use of SG for use in computational design (2006). Biswas presents a computational model of visual interpretation of shapes, to obtain a tool capable of interpreting the sketches of the designer and matching particular designs of a given theme. This model was conceived in the programming language LISP and dubbed TRIGGER, a model that uses visual routines to create shapes from initial sketches, that is, it functions as an interpreter of drawings. Since TRIGGER allows the analysis of drawings and their correspondence to structures of predefined shapes, it proposes the implementation of SG. With TRIGGER, rules can be defined by sketches which, through pattern and pattern recognition, find common concepts and match sketches with an alphabet of initial shapes and rules to apply to achieve formal goals.

**SG application for fabrication** (Ertelt, 2009)

Christoph Heinrich Walter Ertelt and the sponsor for his dissertation, University Prof. Kristine Shea's, 2009 demonstrated generative design-to-fabrication automation using spatial grammars and heuristic search (2009), presenting a SG application for Computerized Numerical Control (CNC) fabrication. The manufacturing process with CNC Machines involves translating the design information into machine operations. Ertelt and Shea propose an SG application that helps the needed systematic formulation of volumes needing to be removed during the machine operation, setting rules for removal volume calculation and CNC code generation.

**MHAS** (El-Zanfaly, 2006)

Also in 2009, MHAS was introduced, an interface in Autodesk Maya made by El-Zanfaly (2009) to produce several alternatives to housing complexes, exploring the possibility of introducing parameters and rules by the designer and obtaining three-dimensional objects that respond to concrete and real architectural context.

**VWG** (Trescak, 2010)

Tomas Trescak (Institute for the Investigation of Artificial Intelligence at Western Sydney University), along with his colleagues, continued to publish articles on general SG interpreters of intelligent design generation. He and his colleagues published a paper in 2010 using Virtual World Grammar (VWG) for the generation of virtual worlds ( 2010).

The VWG was created with the purpose of using SG to support the creation of Virtual Worlds. This system, among sets of heuristics and multiagent system specifications, uses a SG to introduce design elements.

**GRAPE** (Grasl and Economou, 2011)

Thomas Grasl (Institute of Architecture and Design, Vienna University of Technology, Vienna, Austria) and Athanasios Economou (Professor, Georgia Institute of Technology) have published articles on grammar interpreters, SG, graph grammars, subshape detection, and subsymmetry since 2011. The SG library has been implemented in C#, and its code-named GRAPE to playfully recall the parallel representations of 'GRAphs' and 'shAPEs' (2011).

GRAPE was presented as a Rhino plugin implementation of a SG interpreter, using Graphs to represent the structure of the shapes, as presented by Heisserman and Keles before (Jeff Heisserman, 1994) (Keles, O Ë Zkar, & Tari, 2010). Graphs are multidimensional data structures that can formally structure and visually represent shapes. The author continued his work in the development of SG with Graphs, presenting GRAPPA (Grasl & Economou, 2013), an implementation for the Palladian grammar defined by Stiny and Mitchell (George Stiny & Mitchell, 1978).

**SGI** (Trescak, Esteva, & Rodriguez, 2012)

In 2012 an SG Interpreter called SGI was presented (Tomas Trescak, Esteva, & Rodriguez, 2012), developed in the programming language LISP, with the purpose of allowing the user to comfortably define shapes and rules and have full control of the grammar generation process, allowing the direct drawing on the canvas using the mouse. This tool is further described, as it was analyzed more in-depth during our research.

**GRAMATICA and DESIGNA** (Correia, 2013)

Rodrigo Correia, Technical University of Lisbon, along with J.P. Duarte and A. Leitao, presented a general 3D shaper grammar interpreter targeting the mass customization of housing at Proceedings of the 30th ECAADe Conference, 2012, and then in 2013, published an article on a SG interpreter (2012). In  2012 GRAMATICA was created, a 3D SG interpreter that was later used in the design module DESIGNA (R. C. Correia & Coutinho, 2013). GRAMATICA follows a graph-based boundary representation for solids, again similar to Heisserman work, but unlike Heisserman, it uses multiple representations for numbers and geometric calculation, allowing different levels of precision.

DESIGNA used GRAMATICA as the 3D modeler and labeling mechanism and used CAD tools for visualization of the generated shapes.

**Graph-based implementation** (Strobbe et al., 2013)

Tiemen Strobbe ( 2013), along with Ronald De Meyer, Jan Van Campenhout, Rudi Stouffs, Sevil Sariyildiz (all from the University of Ghent) presented an important paper at a conference Computation and Performance – Proceedings of the 31st eCAADe Conference at the Department of Architecture and Urban Planning at Delft University of Technology in the development of performance-based design, evolutionary algorithm, SG, generative design and their implementation.

This implementation is a rule-based system also using a graph-based representation of SG, with the motivation of allowing the quick exploration of design alternatives. This graph-based implementation allows subshape detection, important for emergence, parametric rules, and attributed shapes.

After analyzing the SG implementations referred above, we searched similar existing analysis, to gather the best conclusions possible on what could be the path to help SG implementations enter the design practice.

In 2002, verifying that the development of tools for the use of SG was progressing slowly due to the lack of proper interaction between the user and the generational design system, Scott Chase refers to the need for a generic model of how this interaction between the user and the form-based tool should happen (Chase, 2002). We can distinguish three usage paradigms from implementations for SG: (1) One in which there is complete control, in which the user chooses each rule to be applied. This resembles a non-computational system; (2) One in which the user has partial control and can select some aspects of the rules' application; and (3) SG is predefined, and the system automatically generates the designs without user intervention. In this way, the interaction model must fit any of these paradigms for the use of the implementations and must also consider the possibility of SG being used as only part of the design process, in which the designer decides when to use it. The interaction model must represent the possibilities of user control, must represent the several entities that define the SG, and allow the SG manipulation during the different phases of the design process in which SG is involved.

It is also essential that a good interaction model can withstand emergence. Scott Chase then pursues an analysis of these aspects in some of the implementations mentioned above, all of which are related to the control levels defined by the author. A detailed presentation of Scott Chase's interaction model is presented in Chapter 2.2, as it revealed of extreme importance to our research. The creation of our interface model is based on its assumptions about the different interaction ways different types of users have with the computational implementation

## 2.2 INTERFACES AND INTERACTION

This chapter introduces the areas of User Interface (UI) and HCI. These concepts emerged automatically with the appearance of the computer (Karray, Alemzadeh, Saleh, & Arab, 2008).

The use of computers brought to light the aspects of the interface, that is, in what ways the user interacts with the machine and the computer application.

These two areas are very relevant to our research, as we are seeking to solve usability and interaction questions in SG implementations' interface. Many times, the opinion the user gets from a computer system is developed solely on his experience of the user interface, meaning the UI becomes the "product to be sold". The user seeks a system that works but will choose the one that is easiest to use (Petrovic, 2012).

So, to address the interface means to decisively help an application to be adopted.

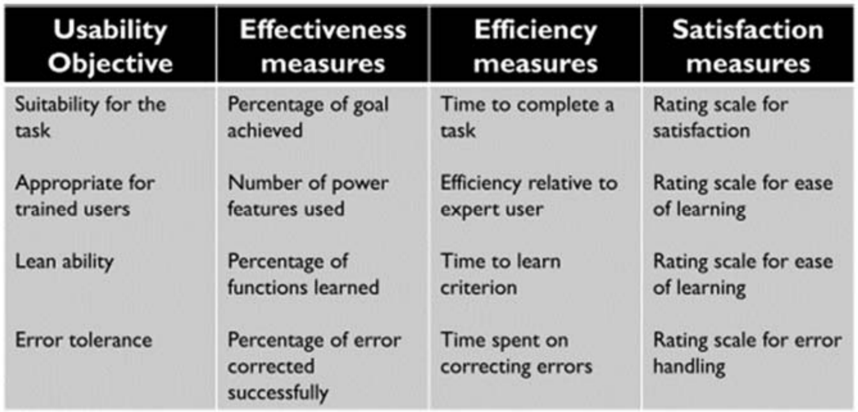
### 2.2.1 User Interface

The UI is all that the user finds in a computer application (Shneiderman & Plaisant, 2007), namely: functionality, content, labels, presentation, layout, navigation, the speed of response, documentation, and help, among others. The computational applications in which the UI is decisive for its success range from text editing programs, spreadsheets, CAD, and drawing software for video games. The importance and complexity of the UI, that is, the direct connection between user and application, is so significant that the use of toolkits - code libraries written to support the creation of interfaces - is now widespread. The use of Interface Builders - interactive tools composed of buttons, menus, among others - and even component architectures allow the creation of interfaces by connecting components written separately (B. A. Myers, 1992).

There are several difficulties in setting the UI of an application. The design of this is a creative process, and often, designers have a hard time thinking as the end users. UI design involves standards, graphic design, written documentation, performance, adaptation to multiple platforms, suitability to external social factors, and legal factors (Shneiderman & Plaisant, 2007). Ultimately, a computational system is created for the function it intends to perform, and the functionality of a system is defined by the set of tasks it offers to its user (Karray et al., 2008). The value of the functionality is visible when the system is used efficiently by the user, that is when the usability of the system in a certain function allows the user to fulfill his objectives.

Thus, to assess usability, inspections methods, and heuristic evaluations were defined. Noticing its importance, at the same time, these methods were defined, sets of usability assessment metrics and criteria were created, and International standards were also prepared (Némery & Brangier, 2014).

Due to UI verified importance, the International Standardization Organization (ISO) defined a list of rules for ergonomics of human-computer interaction in 1996, with criteria for the objectives summarized in Figure 19.



| Usability Objective | Effectiveness measures | Efficiency measures | Satisfaction measures |
|---|---|---|---|
| Suitability for the task | Percentage of goal achieved | Time to complete a task | Rating scale for satisfaction |
| Appropriate for trained users | Number of power features used | Efficiency relative to expert user | Rating scale for ease of learning |
| Lean ability | Percentage of functions learned | Time to learn criterion | Rating scale for ease of learning |
| Error tolerance | Percentage of error corrected successfully | Time spent on correcting errors | Rating scale for error handling |

Figure 19 – ISO 9241 - Ergonomic of Human System Interaction, refers to usability categories and measures of efficiency evaluation (ISO 9241-10, 1996)

According to ISO 9241, usability is about the efficiency and satisfaction that users obtain in achieving specific objectives. Usability determines the success of any computational application, being directly related to the computational interface. Usability is linked to learning, efficiency, productivity, ease of storage, the absence of errors, and satisfaction. Good usability is important because it reflects the notion of user system quality. Good usability enables novices to become more efficient faster, to make experts more efficient, to reduce errors, to identify real needs, and to succeed in the market (Jørgensen & Myers, 2008). According to Jorgensen and Myers (2008), there are three essential points to achieve good usability. (1) to know the user and his tasks well, through the analysis of these and contextualized inquiries; (2) to guarantee the suitability of the design, through prototypes, tested by the users, with participative and iterative design; and (3) to make the final product usable and effective by analyzing the interface through various methods, heuristic analysis and others.

With this definition of Usability and ergonomics on Interaction, we enter the field of HCI, presented next.

### 2.2.2 Human-Computer Interaction

*"Human-computer interaction is a discipline concerned with the design, evaluation, and implementation of interactive computer systems for human use and with the study of major phenomena surrounding them."* Hewett, 1999: 5

HCI is the discipline related to the design, evaluation, and implementation of interactive computer systems, studying usability issues. HCI had significant developing in the 1980s and 1990s when the aspects of usability and interaction of computational systems started being analyzed more throughout (Carroll, 1997). HCI crosses the areas of social sciences, psychology, and information technologies (J. Carroll & Rosson, 2003). In short, HCI is the study of usability (Figure 20). According to Carroll (1997), there are four technical developments of the 1960s and 1970s that triggered the formation of HCI as a discipline: (1) The use of prototypes and iterative development in software engineering; (2) The human factors of the software, with the need to create systems that lowered the rate of human error and stress factors during World War II; (3) The mental models analyzed by the cognitive sciences; and (4) The emergence of graphic interfaces for software.

HCI has as main objective to improve the interaction between users and computers. It is specifically directed to: (1) the study of methodologies and processes of interface design; (2) methods for the implementation of these interfaces; (3) techniques of evaluation and comparison of interfaces; and (4) descriptive development of models and theories of interaction that allow the creation of better interfaces and better techniques of interaction.

HCI has several approaches to interface design, not only to guide the design of new interfaces but also to evaluate their usability. Four different approaches can be distinguished: (1) Anthropomorphic - Analysis / Design of interfaces that have their own human qualities; (2) Cognitive - Analysis / Design of interfaces that consider the capacities of human thought and sensory perception; (3) Empirical - Interface analysis by comparison of the usability of conceptual designs; and (4) Predictive models - Analysis of individual usability components in order to analyze the time that may be required to efficiently complete a task.

For any of these approaches, there are several techniques, defined by several authors (D. Norman, 1988), (Molich & Nielsen, 1990), (Shneiderman & Plaisant, 2007), presented in Table 2.

| | |
|---|---|
| **Donald Norman (1988) 7 principles:** | |
| 1. | Use personal and universal knowledge |
| 2. | Simplify the structure of tasks |
| 3. | Make elements visible |
| 4. | Create correct mappings |
| 5. | Explore natural and artificial constraints |
| 6. | Draw foreseeing mistakes |
| 7. | When all else fails, standardize |
| **Molech and Nielsen (1990) 10 heuristics for usability:** | |
| 1. | Visibility of system status |
| 2. | Correspondence between the system and the real world |
| 3. | User control and freedom |
| 4. | Consistency and standardization |
| 5. | Help the user to recognize, diagnose and recover errors |
| 6. | Preventing error |
| 7. | Recognition instead of recall |
| 8. | Flexibility and efficiency of use |
| 9. | Aesthetics and minimalist design |
| 10. | Help documentation |
| **Schneiderman and Plaisant (1998) 8 essential rules for HCI design:** | |
| 1. | Consistency |
| 2. | Use shortcuts for frequent users |
| 3. | Informational feedback |
| 4. | Dialog boxes for completing tasks |
| 5. | Error prevention |
| 6. | Easy stock reversal |
| 7. | Internal control support |
| 8. | Reduced short-term memory requirements |

Table 2 – HCI design rules by different authors

A set of techniques called Inspection Methods (IM) are pointed out as relevant (C Wharton, Rieman, Lewis, & Polson, 1994), (J. Nielsen & Mack, 1994). The importance of usability and methods to ensure it gained great prominence from the '90s. The main argument for the new methods introduced was the search for ways that involved fewer costs since the usability tests are efficient, but in general, quite expensive.

The IM are analytical techniques that allow usability evaluation, that is, they refer to the inspection of the facility users have in learning a particular application, how efficiently they use it after learning and whether the use of the system is enjoyable.

Before the development of the HCI discipline, the vast majority of interfaces were analyzed through techniques that require expertise in the UI area. However, these techniques have several limitations on their use since it is not always possible to call on specialists with the appropriate experience to carry out the analyzes. These techniques are also difficult to apply before the interface is completed, forcing the conclusions to be reached at an advanced stage of development, where significant changes may already be impossible (Jeffries, Miller, Wharton, & Uyeda, 1991) appropriate to the proposed interface model.

Thus, HCI distinguishes two methods of evaluation: (1) the user tests; and (2) the methods of inspection. User testing is based on observations and usability tests on controlled experiments.

IM dispense users and encompass a set of methods that are all based on inspecting an interface using sets of criteria. Inspection is intended to address design usability issues, although some methods also address issues such as the severity of usability issues and the overall usability of an entire system. Many inspection methods evaluate interface specifications that have not yet been implemented, which means that inspection can be performed early in the system engineering life cycle (J. Nielsen & Mack, 1994).

The use of IM brought the promise of cost control, allowing results to be obtained more quickly and economically, than empirical techniques involving user usability testing, supported by the review and analysis performed by experts in the field.

The analytical techniques of IM do not involve users. They are based on the observation and analysis of the actions that users can perform and the intentions they may have. For a preliminary analysis of the usability of existing SG tools, these were considered to be the most appropriate methods. The use of user testing provides an excellent opportunity to observe how well the interface is adapted to the user's work environment, but for preliminary analysis of existing tools, it would imply a temporary and human investment that does not appear to be the most suitable for the type of conclusions that are intended. In short, IM does not require users and uses heuristic evaluations, walkthroughs, and others, which are variants or conjugations of the two above.

Heuristic evaluation is a method that involves gathering a small group of usability experts to test an application through a set of criteria, noting the severity of each problem encountered. This is a very positive method when used in the early design phases of the interface, allowing to understand changes needed before finalizing the entire process.

In this way, HCI has the primary goal to improve the interaction between users and computers, being specifically directed to: (1) the study of methodologies and processes of interface design; (2) methods for the implementation of these interfaces; (3) techniques of evaluation and comparison of interfaces; (4) and descriptive development of models and interaction theories that allow the creation of better interfaces and better techniques of interaction. With this HCI objective, it is easy to see that HCI-related professionals are generally designers concerned with the application of design methodologies applied to real-world problems (Sodiya, 2009).

It is also interesting to look at HCI as problem-solving the relation between the user and computer application (Oulasvirta & Hornbæk, 2016).

Most recent advances in HCI are now combining former methods of interaction with the technology advances connected to networking and animation. As wearable, wireless, and virtual devices are developed, HCI needs to address the new needs of interaction adequately. This evolution made appear two categories within HCI: Intelligent HCI and Adaptive HCI (Karray et al., 2008), involving different levels of user activity (physical, cognitive, and affection). Intelligent HCI interfaces incorporate AI in its perception and response to the users (e.g., speech-enabled interfaces and visual track of movements). On the other hand, Adaptive HCI designs use intelligence, not in the creation of the interface, but to adapt it to the users cognitive and affective activity (e.g., a regular GUI of an online store that shows new content according to previous searches and known tastes of the user).

Looking at the inputs and outputs, the channels of communication between user and computer, we now have not only Sensor-Based (from the primitive mouse and keyboard to sophisticated sensors), but also Visual, Audio based, or Multimodal, combining more than one type of communication channel. Virtual Reality is also an advancing field in HCI.

On another scope, in the last decade, HCI has also been focusing on persuasive interfaces, as persuasive technological systems have been developed, to assure certain user behaviors and actions (Némery & Brangier, 2014). For persuasive interfaces, we can find sets of Dynamic Criteria that focus on user behavior, trying to influence his relationship with the system (an example are interfaces for online stores).

Of the several aspects of HCI that we can find, we are interested in emphasizing the section denominated Interaction Design (ID). The ID is an approach to HCI from a user-centered perspective to create interactive, easy-to-learn, effective-use products that provide a user-friendly experience, focusing on the user's perspective (Rogers, Sharp, & Preece, 2011). Thus, the main focus is on the user's experience of using a computational application, addressing usability issues early in the interface design process, with an emphasis on cognitive and experiential factors Figure 20. In short, the main goal is to minimize the barrier between what the user wants to achieve and the task the computer can perform.

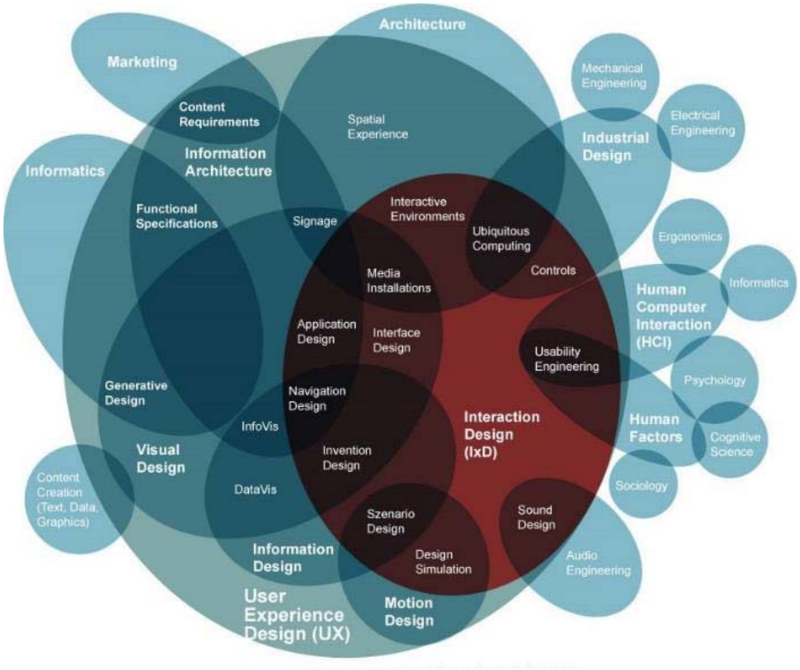


Figure 20 - HCI and ID as the basis of Usability (Saffer, 2008)

Currently, the advances in technology added to ID the notion of User Experience (UX). The UX on the general interaction experience, beyond the computer use (Roth, 2017), as new digital objects appeared (e.g., smartphones and embedded computational systems in daily life objects).

It is still with ID and UI perspective in mind that the present thesis is conducted. As mentioned in Chapter 1.3, the creative project is still very connected to drafting ideas, to the illustration of a design concept before its full physical/technical definition. This means that SG implementations are still connected to the use of the computer and Sensor-Based inputs, that shall allow the user, through its interface, to simulate the drawing exploration of ideas. A GUI seems to be still useful to apply to these systems and its interface defined through UI principles.

Our focus is on the usability issues of SG computational implementations, believing that addressing those we accomplish the main objective of this study: the use of SG in the creative process of design. The ID use is discussed in more depth in Chapter 4, which describes our proposal and how it is based on these principles.

## 2.3 Models of Interaction for Shape Grammars Implementations

After developing the State of the Art on the areas of study of the present thesis, with emphasis on SG implementations, we start analyzing the existing models of interactions for SG implementations, searching ways to approach the communication issues between user and implementation.

Evaluation and decision making through iterative interactions between project design and analysis is common practice in architectural and design project teams. The speed of representation provided by digital tools, allowing rapid analysis of design options, offers unique opportunities to formalize and add rigor to the interactive decision-making process. As such, efforts to research and develop formal decision support models have expanded along with design automation and the availability of reliable analysis tools in the form of simulation software. Digital decision support tools, from search engines, applications for integrated simulation, applications for collaborative design or applications for data management, have gained great prominence in research in the area of CAD. The process of exploring designs that meet specific criteria to decide on the most appropriate action has helped decision making in both architecture and design to get final products with better performance (Choudhary &

Michalek, 2005). The vision of this work is that computer applications can help explore designs not only for performance but also for creative and innovative solutions.

The use of computer applications in the field of design and architecture, or in other words, the role of the computer in these areas has undergone several changes. Starting with the computer by mimicking the manual work of the designer, evolution has gone in the direction of becoming a contributor to the design process (Akin & Anadol, 1993). According to Akin and Anadol (1993), the fact that computers primarily serve only as sophisticated drawing tools may be linked to factors such as lack of technological development or specific algorithms, as well as the fact that it is intimidating for creative designers to use computational applications. The authors consider that the essential problem here lies in the fact that systems are developed to perform what the designer, in one way or another, slower or less rigorously, could accomplish without using the computer.

Computer applications in the field of computer-aided design tend not to consider the aspects related to human interface, the implications of the shapes of input and output of the information, and also the requirements of the tendencies and cognitive capacities of the target users.

At a time when computers are no longer just helpers in the performance of tasks, but instead partners in the way of thinking and solving problems, it is of great importance to mitigate any barrier to machine/user communication. In this sense, it is essential to realize that users tend to follow certain behaviors that influence this same communication and their ability to use computational tools (Carroll & Rosson, 1986). It is also interesting to note that there are not only general issues, but also parallel issues characteristic of new users, and other specific issues felt by experienced users.

Overall, users have some resistance to learning how to use applications. They often have little appetite for exploring new functions or seeking information if they know other methods to achieve the same goals. They tend to solve situations with procedures they already know, even if these are less effective. It is also important to note that, cognitively, people apply what they already know to interpret new situations. However, there is a cognitive paradox: although similarities help to recognize how to act, these same similarities between new and old information are also a problem, precisely because they lead to comparisons and conclusions that do not help to recognize the potentialities of a new function.

According to Carroll and Rosson (1986), questions relating to the learning of computer applications by new users arise primarily because they have habits already, not being "flat" for new ways of using the applications. Considering that user guides are important especially for new users, the users tend not to be available for reading, not to practice the proposed exercises and, above all, do not intend to develop activities to refine their domain of the tool. New users tend to rush to the use of applications, wanting to experience the functions immediately, and some impatience and haste are common. Activities involving practice tend to offer resistance, and the step-by-step structures of the manuals are belittled by this need to get results quickly. The best way for the interface designer to think about the new user is to be an expert in their non-computational domain. This view is positive for the tools to support the user's activities much like work in the "real world". In this way, this initial restlessness is diminished by the fulfillment of tasks and routines similar to the usual work without the application.

In the case of experienced users, the main issue in the use of a new computational application is the learning time versus the performance that it allows. What is important is the perception of productivity gain or promptly efficiency, considering the effort of familiarization, because if these gains do not occur, preference is given to already known methods, even inefficient ones. On the other hand, it is unwise to consider that if an application presents functions from basic to advanced that savvy users naturally acquire the skills to use the tools that most efficiently answer their needs. Even though they are experienced, users tend not to discover new features, and the question becomes even more evident, the more steps they need to take to get an extra function.

Also, Carroll and Rosson (1986) argue that there are two aspects that lead users to greater comfort when using a computational tool that must be considered: (1) controlling the consequences of any action taken by the user;  and (2) controlling the actions available to the user. These two aspects make adapting to a tool easier by reducing fears of error, and relevant information is easier to find. The adaptation of the application to motivational questions should also be made less dependent since these tend not to match what was expected.

### 2.3.1 THE INTERACTION MODEL OF SCOTT CHASE

SG systems can be considered as active tools as opposed to CAD tools. These are pioneering in their interfaces. They are quite sophisticated thanks to the many years of development and use. There has been little success in the computational implementation of SG due to the complexity of object representation, which makes the operations restrictive and gives the user little chance of interaction and control of the systems. A UI model and the description of the actions the SG must perform are the theoretical support for an SG system, serving as a definition of the behavior of the system, as well as a template for interface prototypes, leading to better interfaces and better use of SG tools.

The work carried out by Scott Chase, researcher, and professor in the area of CAD is the origin of the idea proposed in the present thesis. The relevance of a model that explores the different ways users can relate to SG computational implementations has led to the idea that, between this definition and obtaining a SG application that responds to users' needs, there is an essential element to ensure the correct human/machine communication, the interface.

In Scott Chase's Interaction Model (Chase, 2002), the author discusses in detail how a computer application for the use of SG in the area of design should be developed. Featuring different modes of user interaction and possible control of the SG systems, this work is an essential guide for implementations of SG with the potential to become a creative partner of the designer. An implementation of SG only succeeds if it speaks the language of the user. This can be translated into the ability of the interface to be accessible, clear, and efficient. With this consideration, we believe there is the need to extend the premises of the Scott Chase model for the characterization of the interface of the SG implementations so that they can behave to ensure effective communication with the user. Thus, the present research searches the creation of an interface model with its basis in the Scott Chase interaction model.

In this chapter, the Scott Chase model is described in detail so that the aspects that the IM-sgi has to adapt are clearly defined.

Considering that the slow development of SG applications is due in part to the lack of good user-system interaction, Chase presents a model that defines different modes of user interaction and possible control of SG systems, that we describe in the following section:

#### 2.3.1.1 Model Description

Scott Chase's model starts by introducing the three ways of use of SG on the design field: (1) analysis of design styles, with the development of grammars for the generation of designs of a style, existing or new; (2) Generation of new stylistic languages; and (3) Transforming existing styles into new design styles.

Scott Chase defines the paradigms of interaction that can be found on existing systems, and that should also be addressed by his proposed model of interaction. These paradigms are three, varying from full control of the user on the application on the rules and choice of the shapes, a compromise between user control and computational implementation control, to full control by the computational implementation on the SG generation. To respect these paradigms, the model represents the aspects of the interaction related to the different phases of the design process in which SG is involved, the various entities interacting with SG and the User control possibilities.

The model then defines levels and entities. The complete design-generation process by the grammar involves two main levels: (1) SG creation by building vocabulary, rules, initial shape; and (2) the application of the SG. In both levels, the manipulation of the SG can be done by 3 different entities: (1) the creator of SG, usually someone who has mastered the technical aspects of the SG and the computational environment in which it is implemented; (2) the end user, such as the designer, who generates the design language by invoking the rules; and (3) the computer system, which can handle any aspect of SG development and application.

Each level is associated with an interaction paradigm involving one of these entities, and the combination of these is the foundation of an interaction model. Thus, the application of the SG and a design of the SG language can be obtained by the user or by the computer.

Scott Chase clearly explains how the process of using the SG occurs and how the user or computer acts in each stage of the process. To explain the use of the SG, Scott Chase explains that for each invocation of the rule, three actions occur: the rule to apply is selected from the rule set, a part of the drawing state (for example, a subshape) is selected so that the action is taken and a match condition (transformation from rule to shape) is determined. Often there are interdependencies between each action. Thus, the possibilities of different forms of interaction can be varied. By selecting a rule and a match condition, the drawing zone to which the rule applies automatically limits the result of applying the rule to a possibility.

When defining the SG control, the appropriate control scenario for a SG application is constructed by mapping the entities into levels. First, we determine who (programmer, user, computer) should have control over creating the internal representation of objects, creation of control mechanisms, and creation of rules.

Then, when invoking the SG, it is necessary to consider how the user operates the system, what choices are offered to the user (parameters, rule selection, rule application), how the system presents alternatives to the user, if the control level can be changed by the user and if the use of SG can be combined with other techniques.

With the answers to these questions, one obtains the entity-level mapping and can build an interaction model according to it. Some of the possible scenarios are:

- Full Control - the user is responsible for each invocation of each rule, similar to a "manual" mode.
- Partial Control - the user selects the rule or object on which the rule applies.
- No Control - SG is preset, and the system generates designs automatically without user intervention (Figure 21 and Figure 22).

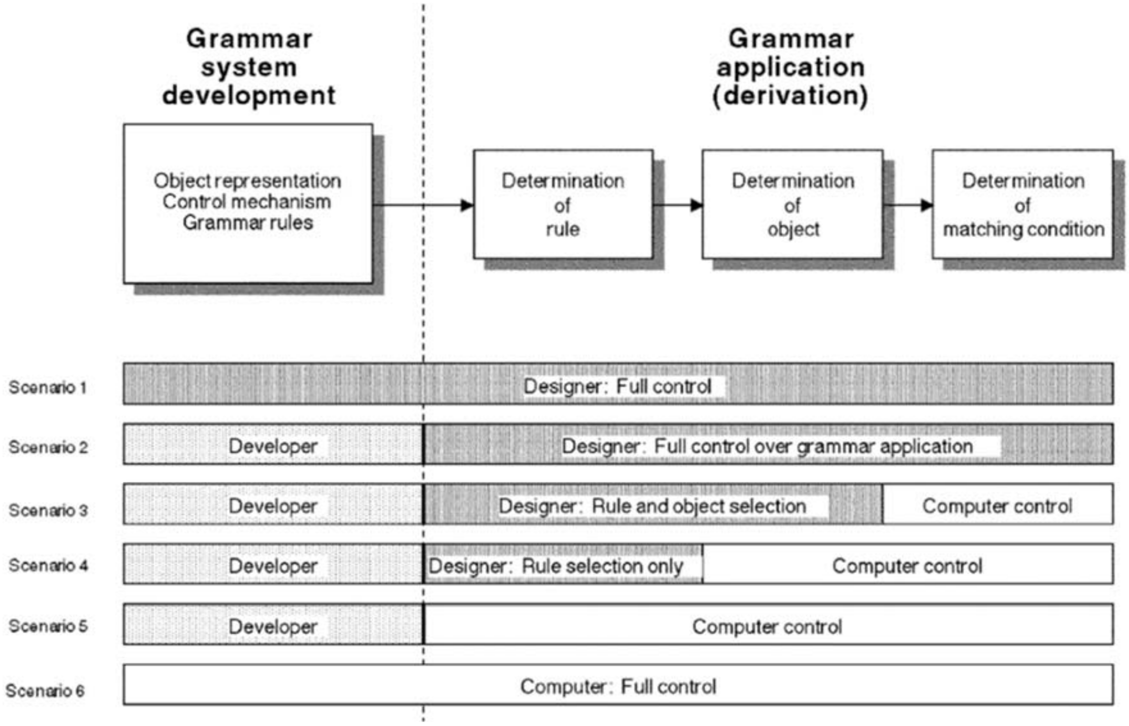


Figure 21 - Some of the possible user control scenarios for the development and application of a grammar. (Chase, 2002: 164

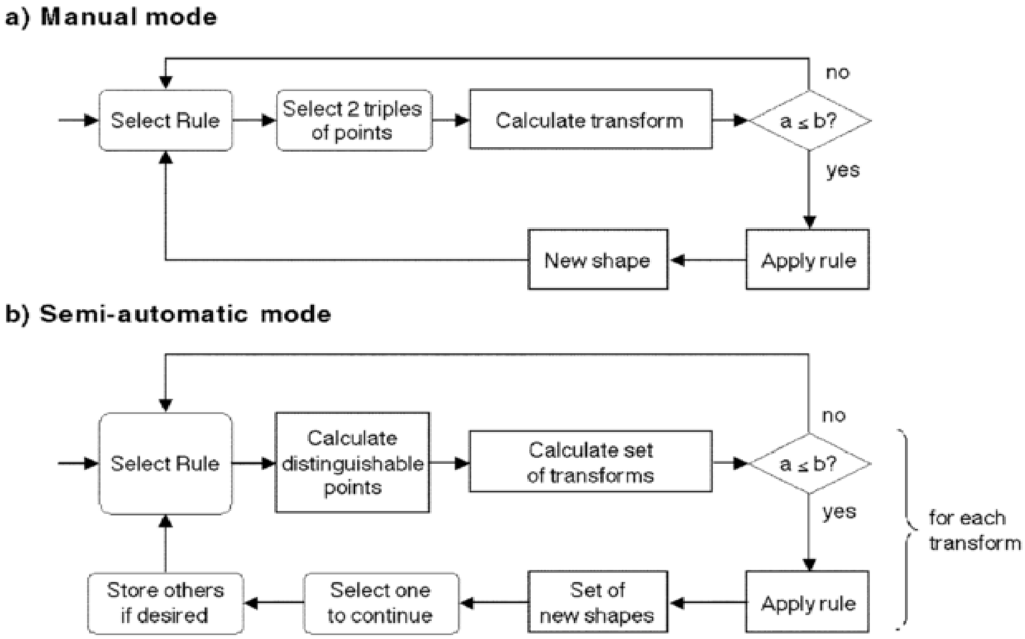


Figure 22 – Two modes of user interaction (Chase, 2002: 167)

After studying Scott Chase's interaction model, we prepared the structure to analyze the existing SG implementations and their model of interaction, trying to understand patterns in how the implementations address the user and communicate with him. This is presented in the next chapter.

# 3 ANALYSIS OF SHAPE GRAMMAR SYSTEMS AND USERS – *COGNITIVE WALKTHROUGH*

In this chapter are presented the first stages of development of our proposal, the SG Interface Model called IM-sgi. These first stages correspond to steps 1 to 3 of Lewis and Rieman (1994) guide for task-centered user interface design.

To contribute to the effective adoption of computational SG tools from users that can use them for creative projects, we propose that an interface model for SG is a response to the communication failure between person and tool. Recognizing the difficulty inherent in creating computational implementations for SG, the user interface issue, which has a great responsibility for the success of the application, may not yet have been given due attention. The development of interfaces is a time-consuming and expensive task. Typically a graphical interface of an interactive system represents 48% of the code, occupies 45% of the development time, 50% of the implementation time and still 37% of the maintenance time (B. Myers & Ko, 2009).

To address the interface, there is a need for understanding how the users apply and manipulate SG and their results. The model of interaction between SG and the user developed by Scott Chase (2002) is an excellent example of the studies on this matter. This model addresses the computational implementation of SG and how they must respond to the users' needs and objectives when using them. But analyzing this and other models, there seems to be a lack of guiding lines for an unambiguous interface for SG implementations that fulfill the objectives of these models of interaction, already addressing a large number of issues about the needs of the SG users. The interface is the primary means of communication between the user and the SG implementation. When this communication is not well addressed, the efficient use of the SG implementation is compromised, and the problem solutions that could be generated are unlikely to happen.

The interface of a SG implementation, with the design practice in sight as we intend to, must take into account that architects are trained and feel comfortable using CAD software, which interface is well adopted and stabilized. The complexity of CAD systems and the type of tasks associated impose a high-quality interface. In this sense, the best way to ensure that a SG implementation is well accepted and understood by an architect is to apply the basics that the final user is adapted to. Therefore, our proposal is the creation of an interface model, that we named IM-sgi, that intends to respond to computing ergonomics and suitability for architects needs that SG implementations should respond. This model also takes into account other types of users of SG, like designers, artists, and students.

IM-sgi is our proposal to make the bridge between the user and SG implementation, leading SG to the project practice and leaving the analytical and educational fields where they have mainly been used. For this to take place, there is a gap that needs to be addressed: make the SG implementations clear and simple, through a user directed interface. The answer to this problem is a model of the interface that reports the user needs. Following this need, it is developed an IM-sgi interface model for SG implementations.

For a systematization of the fundamentals for IM-sgi, we started with the analysis of the existing SG implementations. In this way, we intend to understand how the interface questions for these implementations have been addressed and what considerations can be made of them, in terms of suitability for the user and task to be performed. This analysis, to gather conclusions that allow the proposal of an interface model, is based on HCI methods, selected according to its objectives. Since this discipline is dedicated to the study of interface design, it is potentially the one that best guides this process.

When we look at usability of the implementations for existing SG, excluding the capacity of response of the implementations to real objectives of the creative projects (motivation for the present thesis), we verify that there is no standard of correspondence to the parameters of HCI. A type of standard interface or a similar communication logic, even when the results are identical does not exist.

There are several HCI techniques for the evaluation of interfaces that can be applied in the desired analysis, and it is essential to understand which help us reach the desired conclusions most expeditiously. Therefore, we use IM techniques (Novick, Hollingsed, & Martin, 2007) to understand how easily the users learn to work with SG computational implementations, how efficiently they use them after learning and if the use of the application is pleasant.

The vast majority of interfaces are criticized through techniques that require expertise in the UI area. Examples of these techniques are usability testing and heuristic evaluation. However, these techniques have several limitations to their use since it is not always possible to call on specialists with the appropriate experience to carry out the analyses. These techniques are also challenging to apply before the interface is completed, forcing conclusions to come at an advanced stage of development, where it may already be impracticable to implement significant changes (Jeffries et al., 1991).

It was intended, therefore, to apply an IM that neither requires a UI specialist, nor tests with users. Within these parameters, the Cognitive Walkthrough (CW) method proposed by Olson, Lewis, Riemen, and Wharton (1991) which is a formalization of a user's possible thoughts and actions while interacting with an interface, was employed. This approach stems from the adaptation of Design Walkthrough techniques used in software engineering that involved manual testing of code sections to test specific functionality with cognitive models of exploration learning. In this case, the objective is to manually test the user's cognitive activities to analyze how the user can learn to perform the tasks that the system must support, based on the theory of learning by exploration. The CW method has evolved, and several versions and extensions have emerged by various authors (Mahatody, Sagar, & Kolski, 2010). Given its success, the basic principle is to simulate the user's cognitive behavior by answering questions related to learning, manipulation, and adoption of the analyzed computational application.

Since the purpose of the preliminary analysis of existing SG applications is their usability, considering that this is directly related to the adoption of SG by designers, this technique allows us to understand the user's relationship with the application in the aspects relevant to the desired conclusions, that is, in the ease of handling and use of SG.

## 3.1 First Steps towards an interface model for Shape Grammar Implementations

IM-sgi main goal is the development of a friendly interface model for SG that gives a response to the existing communication failure between the user and the tool. Hence, to address this problem, we used the following methodology divided into three main phases.

Phase 1 is the preliminary analysis of the existing implementations of SG, with focus on their interface. This analysis allows us to understand the types of interaction that have been already used and tested and understand their strengths and weak points.

Phase 2 is the definition of IM-sgi, structuring the model of an interface according to types of users and their interaction with the system, according to their different goals. This model has its roots in the Interaction Model of Scott Chase (2002), as the author identifies the ways that designers can interact with SG and how they relate with an SG implementation. IM-sgi intends to connect these interaction definitions and the application functioning, as the interface is the means of communication between user and machine. This is presented in Chapter 4.

Phase 3 is the creation of a prototype that follows the IM-sgi guidelines so that we can test its usefulness and response to the issues addressed. This chapter presents the Phase 1 of the IM-sgi definition. As stated above, for systematization of the IM-sgi fundamentals, the first step to be taken is a thorough analysis of the existing SG implementations. Thereby, it is intended to understand how these implementations have approached the interface issues and which considerations may be made from them, in terms of suitability to the user and performed tasks. This is presented in Chapter 4.4.

This analysis, to gather conclusions that enable the proposal of an interface model settles on HCI methods, selected according to the objectives of this thesis. Since this discipline studies interface designs, it is the one that can guide this process by the adequate principles.

When addressing the usability of existing SG implementations, excluding the analysis of their capability of application in the creative project (which is the motivation for the present study), we realize that there is no pattern in the fulfillment of HCI parameters. There is not a common type of interface or a common communication logic, even when the results are similar.

We intended, thus, to apply an IM, which did not demand a UI specialist or tests with users. Within these parameters, noteworthy is the method of CW proposed by (Polson et al., 1991), which is a formalization of the possible thoughts and actions of a user while interacting with the interface. This method emerges from the adaption of Design Walkthrough techniques used in Software Engineering, which involved manually testing code sections to proof determined functionalities, with cognitive models of learning by exploration. With the CW, the intention is to examine the user’s cognitive activities manually, to analyze how they can learn to perform the tasks that the system supports, based on the learning by exploration theory.

The CW method has evolved over the time, and various versions have been put together by several authors (Mahatody et al., 2010), resulting in a successful and fundamental principle to simulate the cognitive behavior of the user through the responses to learning-related questions, manipulation, and adoption of the analyzed application.

In short, the CW enables the evaluation of the ease with which the user completes a task with very little system knowledge and the ease of learning and exploring the interface.

To make this rating possible, it is necessary to set up an action script that reflects the manipulation and the application by the user to achieve a specific goal. This process shall be put to work when the application is still to be developed, aiming the correction of determined aspects, or it may be utilized after the application is already established, to determine the difficulties in the use of the system to run determined scenarios.

The purpose of the preliminary analysis of existing SG is its usability, considering that this aspect directly relates to the possible adoption of the SG by the designers. The CW technique allows the perception of the relationship between user and application in the features that are relevant to the intended results, that is, in the ease of handling and use of SG.

### 3.1.1 COGNITIVE WALKTHROUGH PROCESS DEFINITION

For an observation of the existing SG implementations, that enables critical analysis of the interfaces that these possess, this chapter presents a comparative survey on existing SG implementations. The corpus of SG implementations was defined according to their public availability to be used and tested. Since the critical point to the analysis is the communication between the user and the computational tool, the simple theoretical review of the functioning of the application or the image of the interface has not been considered adequate for this study. This premise is also essential for the correct use of the CW, as usage can only be considered when the handling of the application is possible.

Following the SG implementations studied here, the present analysis was performed by comparatively testing of six implementations (Table 3).

| Author | Year | Application Name | Reference |
|---|---|---|---|
| **Miranda McGill** | 2001 | Shaper 2D | (Mcgill, 2001c) |
| **Jowers et al** | 2008 | Subshape | (I Jowers et al., 2008) |
| **Trescak et al** | 2009 | SGI | (Tomas Trescak et al., 2009) |
| **Li et al** | 2009 | SGDE | (A. I. Li, Chen, Wang, & Chau, 2009) |
| **Jowers et al** | 2010 | SD2 | (Iestyn Jowers, Hogg, McKay, Chau, & de Pennington, 2010) |
| **Hoisl** | 2010 | Spapper | (Hoisl, 2012) |

Table 3– List of Applications Used for Preliminary Analysis of SG Implementations (Tching et al., 2016)

These six SG implementations were selected because they were the ones that were available and fully functional, allowing the creation of SG from start to finish. Also, they had a graphical user interface that could be analyzed by our thesis objectives and its authors freely provided them.

Once the implementations available for utilization had been gathered, the process of the CW development was followed to evaluate the interface in the context of specific user-performed tasks. The description of the interface design shapes the CW session; it is considered an action scenario, a user profile and, at last, a sequence of actions that the user must successfully perform to complete the desired task (Cathleen Wharton, Rieman, Lewis, & Polson, 1993). The CW process performed is made by a series of definitions, preliminary analyses and, finally, application of the method itself, that is, the practical testing of SG applications. The description of the process is presented next, with a summary of the information and results. The analysis of the selected SG implementations was made accordingly to four steps: (1) Considered users; (2) Tasks for evaluation; (3) Preliminary Analysis; and (4) Development of Cognitive Walkthrough.

The first step is the definition of the characteristic users that can give us a wider range of aspects to analyze. Studying the Scott Chase control scenarios and with the perspective of SG use in creative projects, we identified three main groups of users of SG implementations. As we have different purposes, our groups differ from Scott Chase's groups in the sense that, instead of focusing on control, we focus on SG expertise. Instead of Developer, Designer and Computer, our three groups are: (1) Students - users that use SG in an exploratory manner with the purpose of learning or pure experimentation, which corresponds to Chase' scenario 4 (the user only selects pre-existing elements); (2) Designers/Artists - users that apply SG for creative projects, more simply or elaborately as needed or according to their SG knowledge, corresponding to Chase' scenarios 2 and 3, where the user can create and manipulate shapes and rules; and (3) Experts in SG - users with the ability to explore all areas of the SG and their computer implementations, which corresponds to Chase' scenario 1, where the user can control the entire implementation, including its code.

From the groups of users defined above (Students; Designers/Artists and Experts in SG), we established that the ones to be considered for the CW analysis are the Designers/Artists. This choice allows us to consider the users that use computational tools of vector and non-vector computational drawing in their professional practice.

The second step is the definitions of the main tasks the users can perform, defining a hierarchy of tasks for evaluation. These are the tasks that guide de CW, meaning they are the ones to be used in step 4. We defined five tasks that sum the steps the users take to create and use an SG:

1. Creation of shapes
2. Creation of rules
3. SG Application
4. Manipulation of SG-obtained solutions
5. SG Modifications

The third step is the Preliminary Analysis. This step happens when it is gathered, and registered the information considered relevant to the analysis, through a list of essential aspects to scope:

1. General analysis (Table 4)
   - 1.1. Type of Interface – if graphic or not
   - 1.2. Dimensions – if SG are two-dimensional or tridimensional, or both
   - 1.3. First impression – analysis of the impact in the first visualization of the general interface of the application
   - 1.4. Learning – analysis of the ease of the tool manipulation, if fast or slow, if difficult or easy without script (this analysis was performed according to the work processes of the designers, that is, verifying the resemblance of the work with the computer-assisted drawing applications, which are commonly utilized by this group of users)

1.5. Graphics – graphical resemblance analysis with the generality of the computer-assisted drawing applications presently used

2. Usability analysis in accordance to the (ISO 9241-210, 2010) which defines the ergonomic principles of the dialog between humans and information systems, allowing to qualify the experience of the user when performing the tasks described above, these being:
   2.1. Suitability to the task
   2.2. Ease of learning
   2.3. Suitability to individualization
   2.4. Accordance with user's expectations
   2.5. Self-descriptive
   2.6. Controllability
   2.7. Error tolerance

Following the (ISO 9241-210, 2010), which defines the scope of efficiency, satisfaction, and effectiveness, this analysis was performed using a rating scale of 5 points according to the level of efficiency. According to this standard, efficiency refers to the resources used by the user to ensure the successful completion of a task. In the performed CW, we measure the time taken to complete a task, reflecting the ease of interpretation of the interface and whether it was necessary to use the instructions manual after failure to solve the task. Thus, the outline evaluation scale used is: (1) without success in meeting the objective; (2) with great difficulty; (3) with some difficulty; (4) easily; and (5) very easily.

This analysis was made joining complementary information found in referred bibliography, to obtain a broader perspective of the selected implementations (Mckay, et al., 2010).

Step four is the Development of the Cognitive Walkthrough, performing the analysis of the selected implementations according to the usability test with predefined tasks.

Both step 3 and 4 will be applied in the next chapters, as we describe our Cognitive Walkthrough and the conclusions, we gathered from the testing of the SG computational applications available for public use.

### 3.1.2 Preliminary Analysis OF SG Implementations – Cognitive Walkthrough

Completed the definitions from step 1 and 2 described above, step 3 is the preliminary test to the SG implementations selected, to gather information about the implementations that are to be used in the CW (Table 3 and Table 4).

After this preliminary analysis, a Usability Analysis was performed, according to Dialogue Principles of ISO 9241-210, 2010 (Table 5). Finally, the CW analysis was performed on the six selected SG implementations, enumerated in Chapter 3.1.3. and resumed in Table 6.

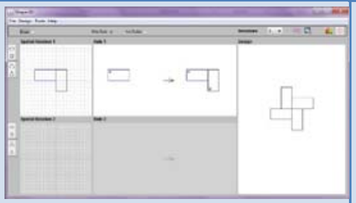
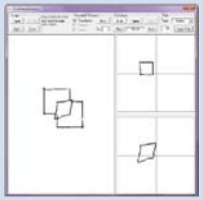
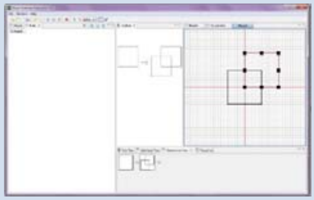
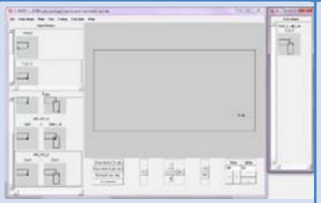
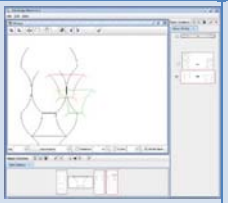
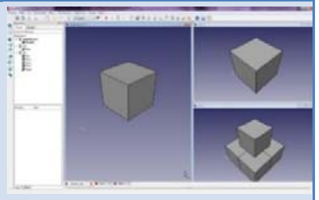
| Implementation / Authors | Shaper 2D **(Mcgill, 2001b)** | Subshape **(I Jowers et al., 2008)** | SGI **(T. Trescak, Rodriguez, & Esteva, 2009)** | SGDE **(A. I. Li et al., 2009)** | SD2 **(Iestyn Jowers et al., 2010)** | Spapper **(Hoisl, 2012)** |
|---|---|---|---|---|---|---|
| **Interface** |  |  |  |  |  |  |
| **PRELIMINARY ANALYSIS** | | | | | | |
| **Interface Type** | OOGUI | OOGUI | GUI | GUI | GUI | GUI |
| **Dimensions** | 2D | 2D | 2D | 2D with visualization 3D | 2D | 3D |
| **1st printing** | Fixed / All-Visible / Basic-Limited Window | Visible / Old-fashioned | Information of the same element dispersed by windows | Independent windows | Windows | FreeCAD Environment |
| **Learning** | Very Fast and Easy | Difficult without help; Easy with the help of the guide | Difficult; Unrelated dependencies | Easy but with little noticeable buttons | Easy with predecessor and guide knowledge | Difficult without help; Easy with manual use |
| **Graphics** | Basic | Old-fashioned | Basic | Old-fashioned | Basic | Supported by CAD |
| **Self descriptive** | 5 | 3 | 3 | 4 | 3 | 4 |
| **Controllability** | 5 | 2 | 4 | 4 | 2 | 4 |
| **Fault tolerance** | 5 | 1 | 2 | 2 | 2 | 3 |

Table 4 - Preliminary Analysis of the selected SG implementations

| Implementation / Authors | Shaper 2D (Mcgill, 2001b) | Subshape (I Jowers et al., 2008) | SGI (T. Trescak, Rodriguez, & Esteva, 2009) | SGDE (A. I. Li et al., 2009) | SD2 (Iestyn Jowers et al., 2010) | Spapper (Hoisl, 2012) |
|---|---|---|---|---|---|---|
| **ISO 9241-10:1996** | | | | | | |
| **Suitability to task** | 5 | 3 | 4 | 4 | 3 | 4 |
| **Learning facility** | 5 | 3 | 4 | 4 | 3 | 4 |
| **Suitability to Individualization** | 2 | 3 | 3 | 4 | 3 | 4 |
| **Compliance with user expectations** | 4 | 3 | 3 | 4 | 3 | 3 |
| **Self descriptive** | 5 | 3 | 3 | 4 | 3 | 4 |
| **Controllability** | 5 | 2 | 4 | 4 | 2 | 4 |
| **Fault tolerance** | 5 | 1 | 2 | 2 | 2 | 3 |

Table 5 - Usability Analysis

### 3.1.3 Development of the Cognitive Walkthrough

The CW is here presented for each SG implementation showing the five tasks to be performed, together with a description and an illustration picture for each one. For each SG computational implementation the five task tested are as follow: (1) creation of shapes; (2) creation of rules; (3) SG application; (4) manipulation of SG obtained solutions; and (5) SG modifications.

**Shaper 2D_Miranda Mcgill (2001)**

In 2001, Miranda McGill presented Shaper 2D, a dynamic visual tool for the use of SG (Mcgill, 2001c). The purpose of Shaper 2D is the exploration of two-dimensional basic SG in a learning perspective and aiming the understanding of the potential use of computer-based SG in the design area.

Task 1 - Shape creation

The pre-existing shapes are only selected and are displayed as buttons on the left side of the display. We can select the initial shape, the shape to which the rule is applied to (Figure 23).

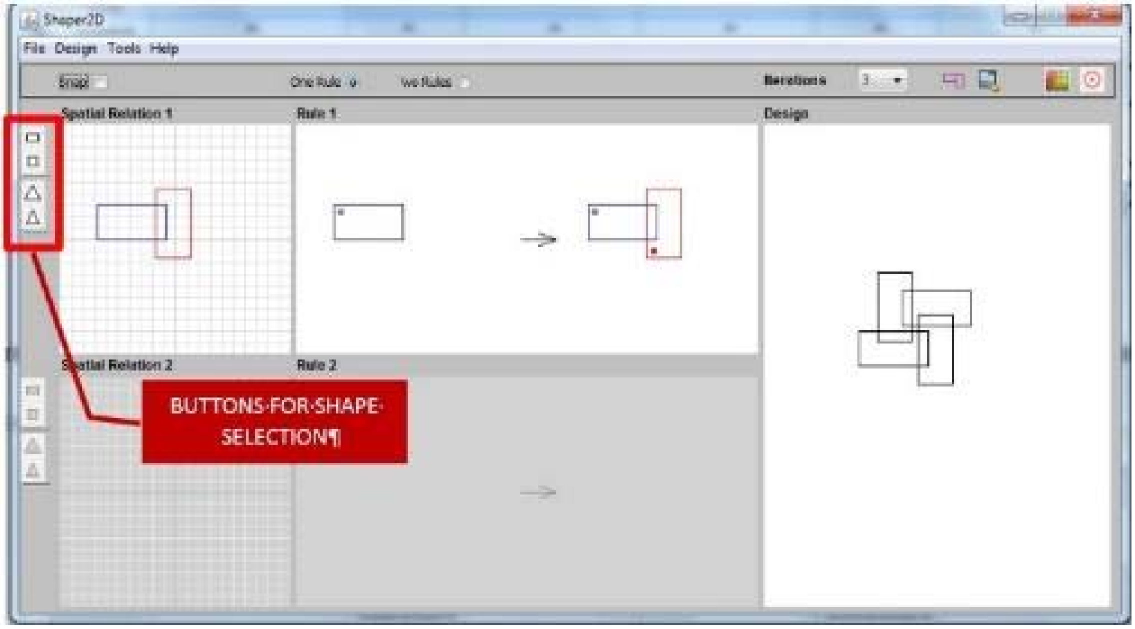


Figure 23 Shape Drawing (from Shaper 2D available at (http://www.designmasala.com/miri/shaper2d/) accessed September 2014)

Task 2 - Rule creation and Task 3 - SG application

Rules are created "invisibly" to the user, which only arranges groups of shapes in order to illustrate a rule. Immediately and automatically, the result of both the rule and the final shape created are shown.

The user can select the number of rule iterations and can also choose to apply between one or two rules at a time (Figure 24).

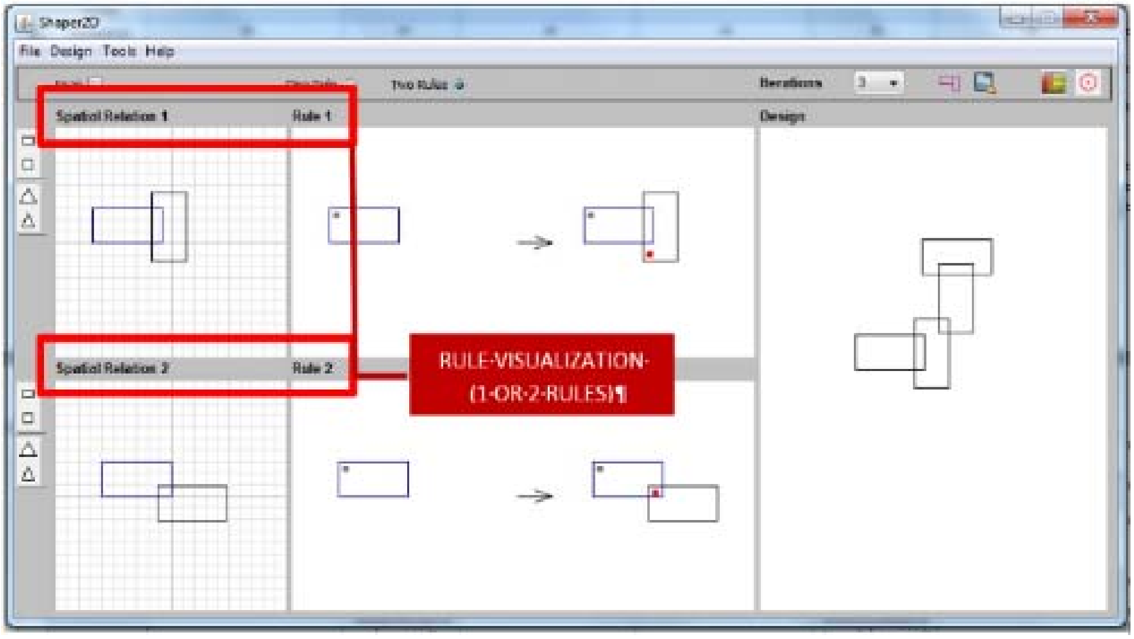
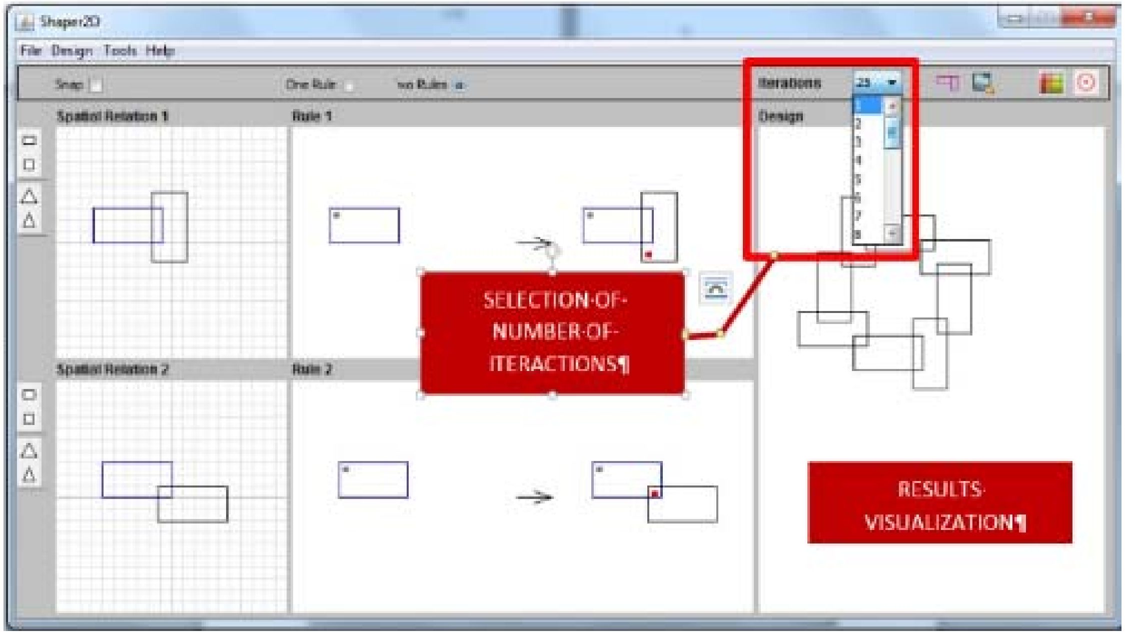


Figure 24 - Rule creation and results in visualization (from Shaper 2D available at (http://www.designmasala.com/miri/shaper2d/) accessed September 2014)

Task 4 - Solutions manipulation

The manipulation of the solutions is performed by changing the number of iterations through a dropdown menu that allows choosing between 1 and 25 iterations (Figure 25).



Iterations 25

Figure 25 - Choice of rule iterations (from Shaper 2D available at (http://www.designmasala.com/miri/shaper2d/) accessed September 2014)

Task 5 - SG modification

Changes in the SG are performed simply and immediately, by manipulating the position of the shapes, or by changing the initial shape, the shape to which the rule is applied to.

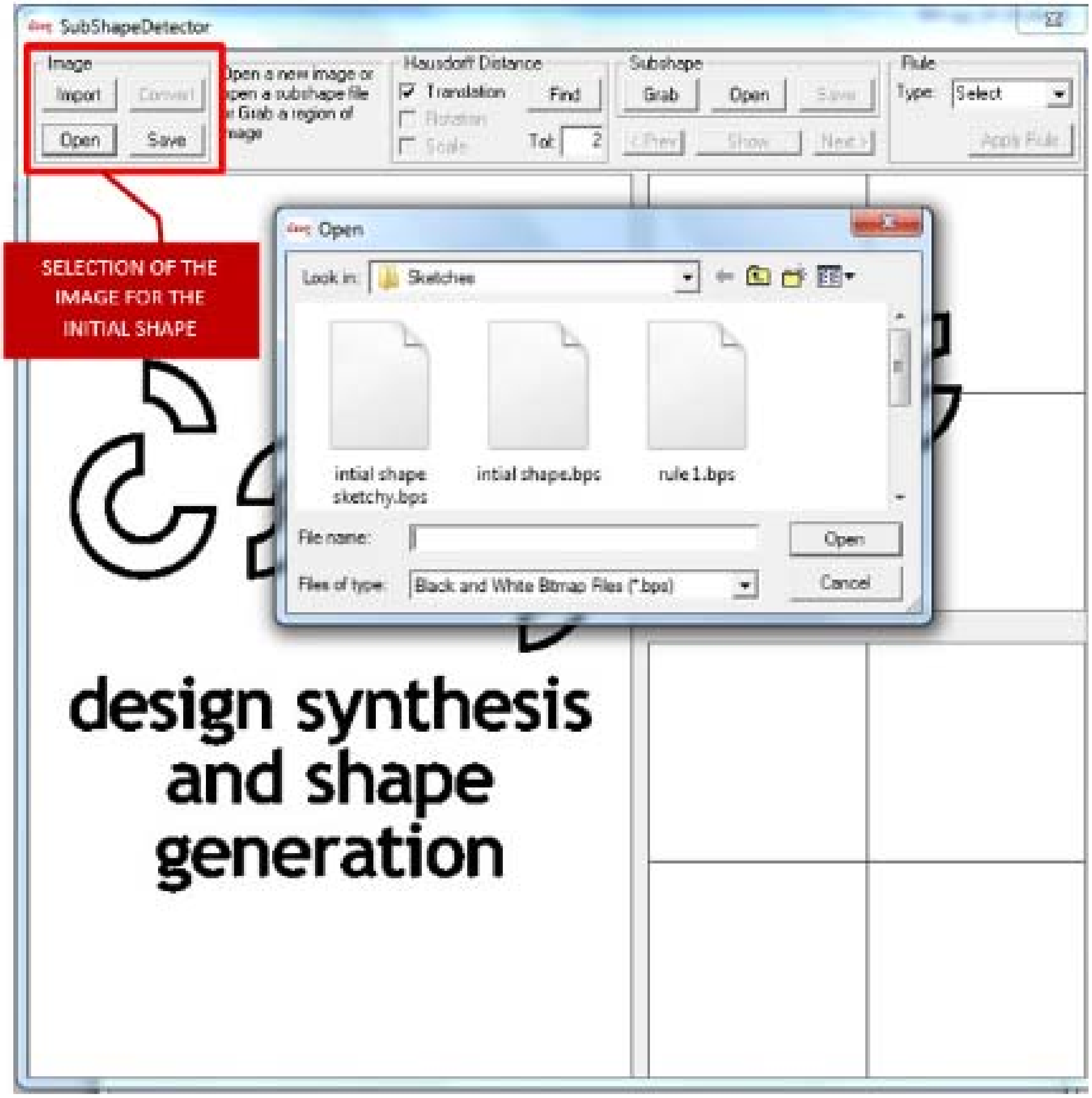
**Subshape_Jowers, Prats, Lim, McKay, Garner & Chase (2008)**

Subshape (Jowers et al., 2008) was developed with the purpose of creating a SG implementation based in visual correspondence. The author used the Hausdorff distance, which is a measure of the maximum distance from a set of points to the nearest point in a second set. Unlike any other SG implementation, this one works with bitmap images.

Task 1 - Shape creation

Initial shapes are obtained by importing a BMP format image file, or directly opening an image file in black and white (bps). It is not possible to draw directly in the application. When importing BMP images, we need to convert into bps, which is done directly in the application (Figure 26).

]

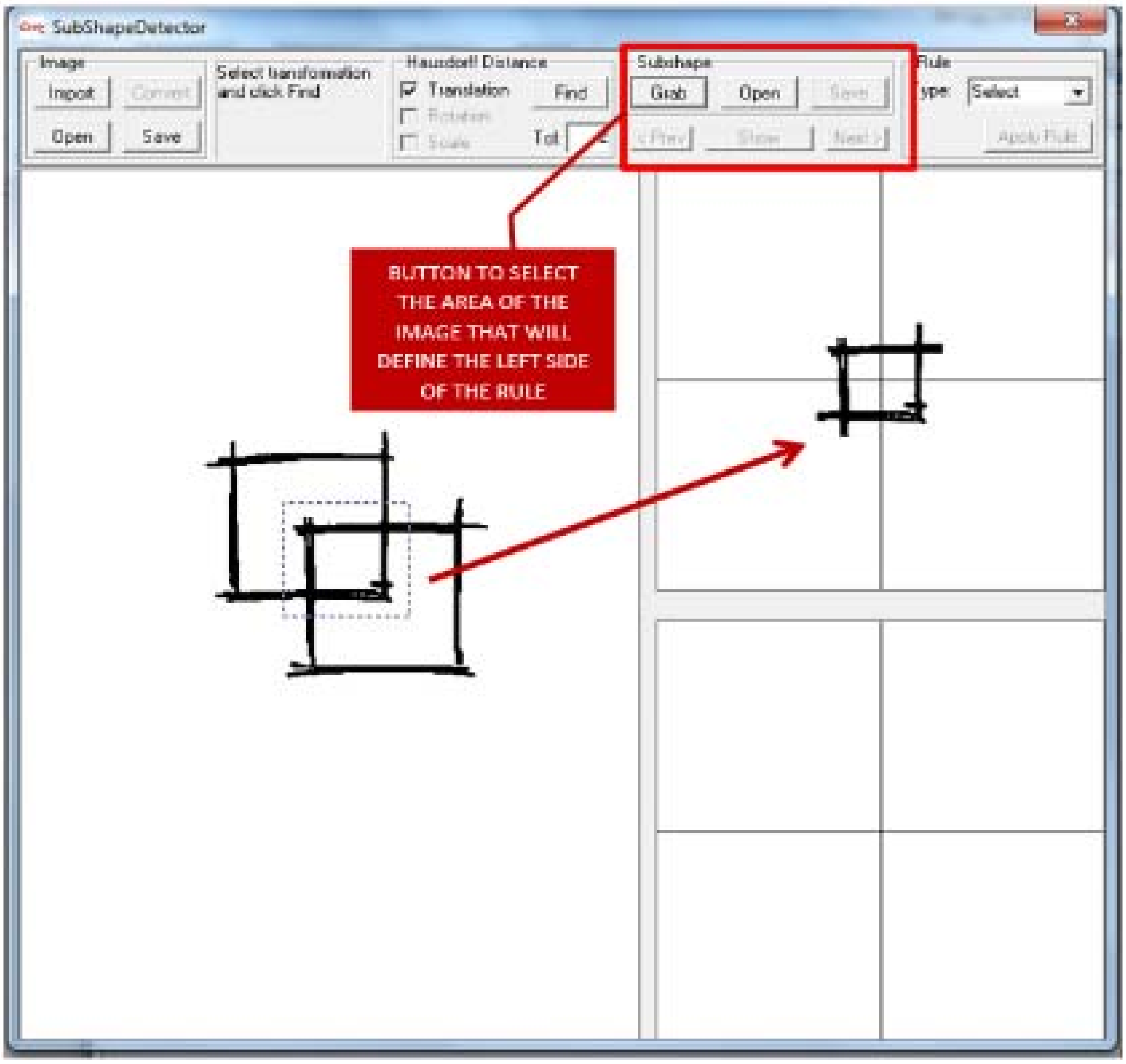


SELECTION OF THE IMAGE FOR THE INITIAL SHAPE

Figure 26 - Bitmaps as initial shapes (from Subshape available at (http://www.engineering.leeds.ac.uk/dssg/downloads/requestForm.php) accessed September 2014)

### Task 2 - Rule creation

The application only allows the use of a substitution rule and for it to work, it is necessary to follow exactly the steps, or it crashes. We accept this could be a problem that only happens in the published free version of the application, not giving relevance to this issue. To define the left side of the rule, we can open a bpm file through the Subshape menu, or the left side can be obtained by selecting an area of the initial shape, using the Grab command in the same menu (Figure 27).



Select transformation and click Find

☑ Translation
□ Rotation
□ Scale

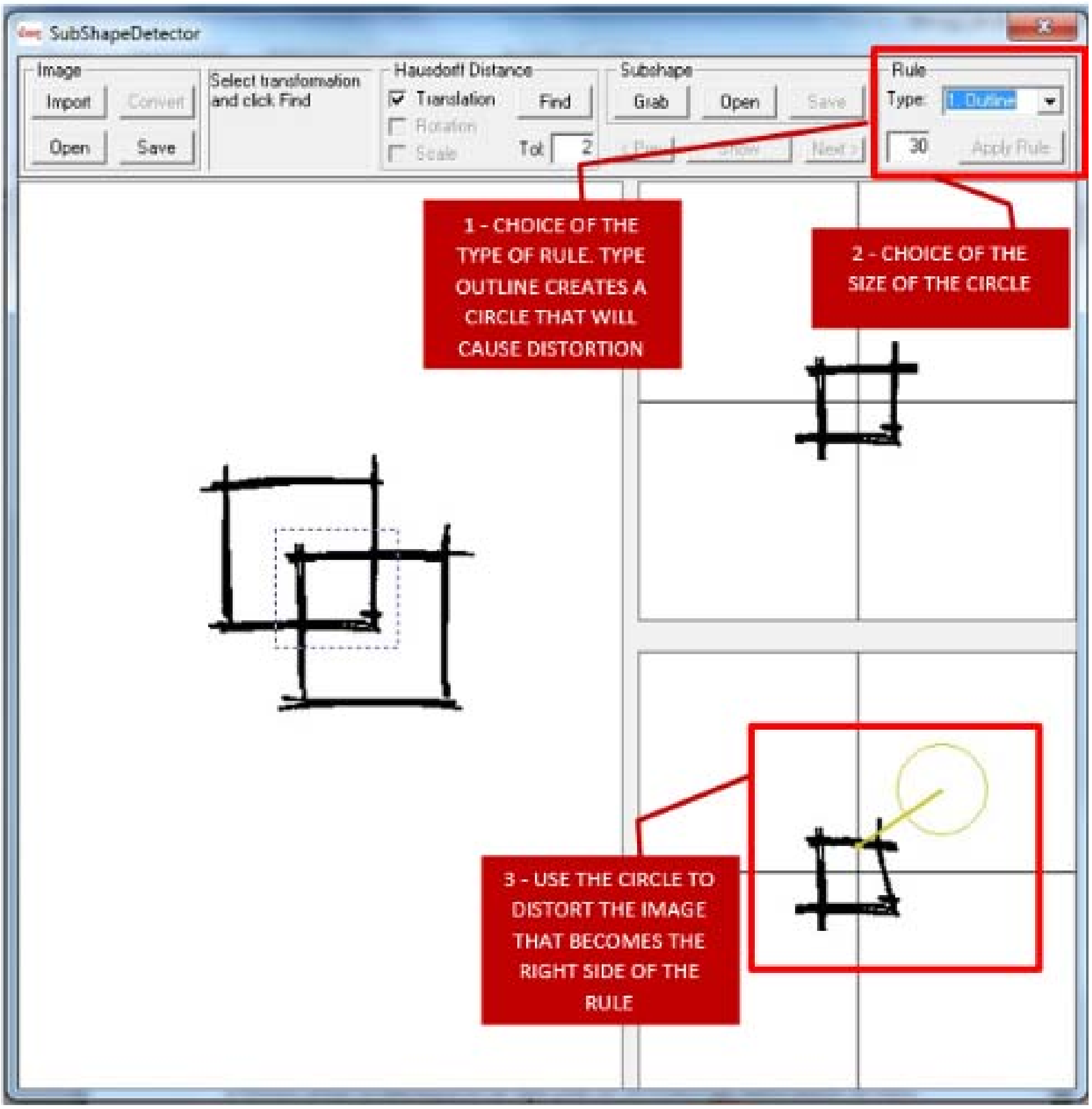
Type: Select

BUTTON TO SELECT THE AREA OF THE IMAGE THAT WILL DEFINE THE LEFT SIDE OF THE RULE

Figure 27 - Left side shape creation through a selection of an area (from Subshape available at (http://www.engineering.leeds.ac.uk/dssg/downloads/requestForm.php) accessed September 2014)

After selecting the shape of the left side of the rule, we can choose in the dropdown list how to create the shape of the right side. There are two options for this.

The first option is the Outline. This option copies the shape of the left side of the rule, and it can be changed manually by distortion rays applied graphically, meaning we can use invisible circles that distort the image (Figure 28).



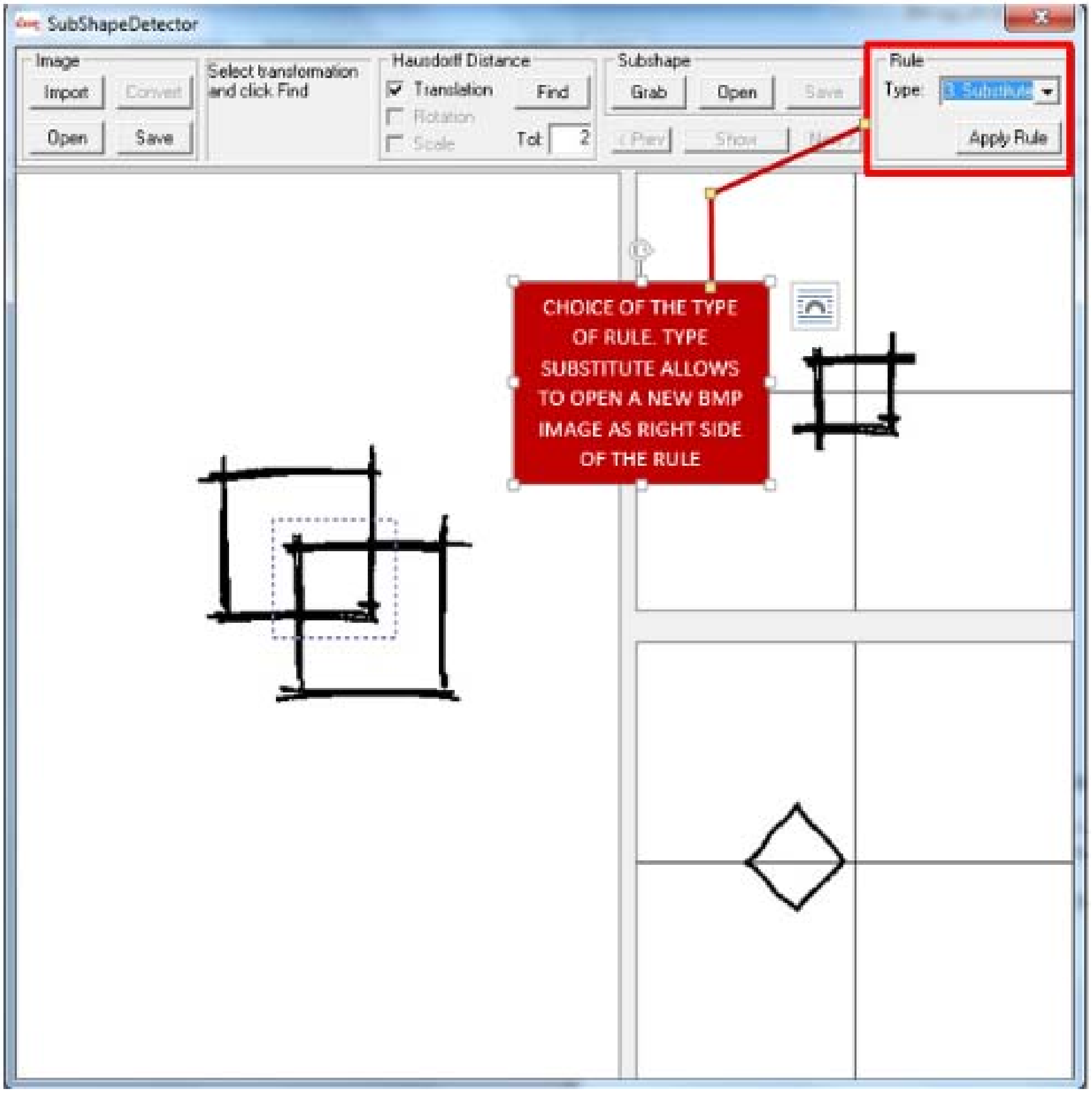
1 - CHOICE OF THE TYPE OF RULE. TYPE OUTLINE CREATES A CIRCLE THAT WILL CAUSE DISTORTION

2 - CHOICE OF THE SIZE OF THE CIRCLE

3 - USE THE CIRCLE TO DISTORT THE IMAGE THAT BECOMES THE RIGHT SIDE OF THE RULE

Figure 28 - Visual transformation of the shape copied to define the right side shape (from Subshape available at (http://www.engineering.leeds.ac.uk/dssg/downloads/requestForm.php) accessed September 2014)

The second option is the command Substitute, which opens a new image file that is used as the right side of the rule (Figure 29).



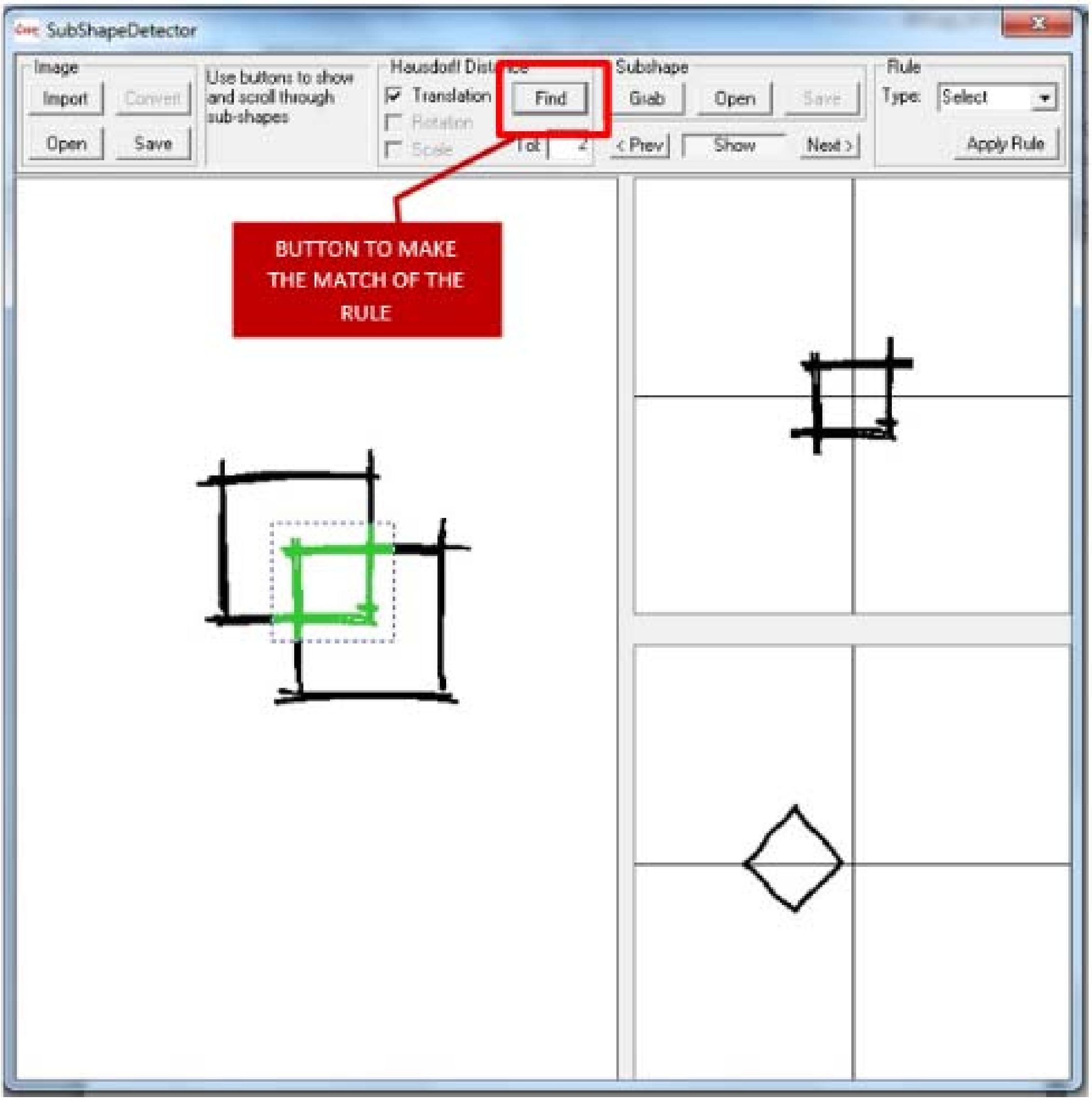
☑ Translation
□ Rotation
□ Scale
Tol: 2
Type: 3. Substitute

CHOICE OF THE TYPE OF RULE. TYPE SUBSTITUTE ALLOWS TO OPEN A NEW BMP IMAGE AS RIGHT SIDE OF THE RULE

Figure 29 - Bitmap file for definition of the right side of the rule (from Subshape available at (http://www.engineering.leeds.ac.uk/dssg/downloads/requestForm.php) accessed September 2014)

Task 3 – SG application:

To apply the SG rule, after defining the left and right shapes of the substitution rule, we must use the Find button, which makes the match. If more than one match can be found, we can navigate between them to select the desired one. As seen in the image below, the menu name is not clear for this function (Figure 30).



Use buttons to show and scroll through sub-shapes

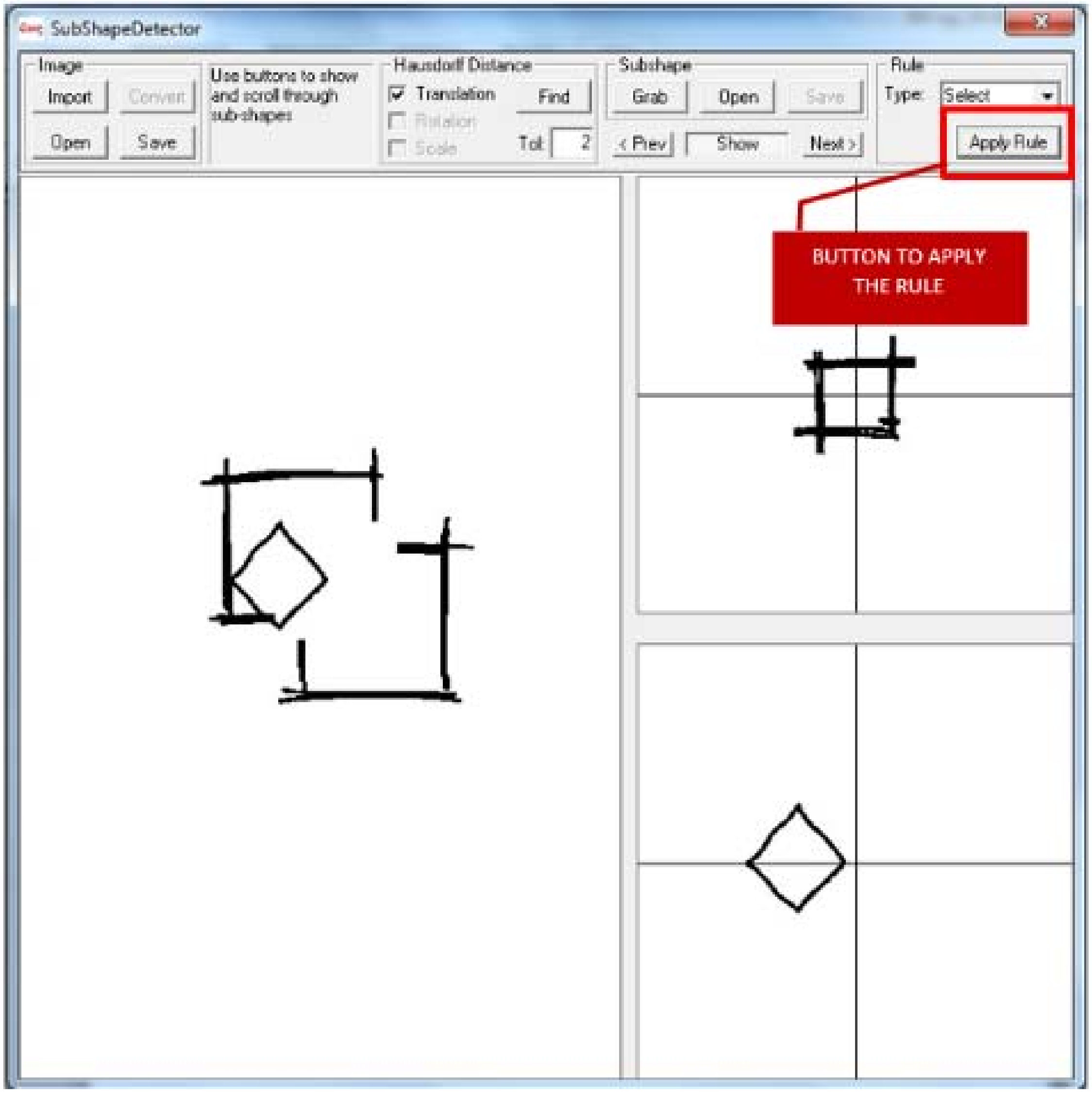
☑ Translation
☐ Rotation
☐ Scale
Tol: 2

Type: Select

BUTTON TO MAKE THE MATCH OF THE RULE

Figure 30 - Matching of the rule (from Subshape available at (http://www.engineering.leeds.ac.uk/dssg/downloads/requestForm.php) accessed September 2014)

After selecting the desired match, we must use *Apply Rule*. We do not have a single menu to make the match and apply the rule. We must jump from one side to the other of the interface (Figure 31).



Use buttons to show and scroll through sub-shapes

☑ Translation

□ Rotation

□ Scale

Tol: 2

Type: Select

BUTTON TO APPLY THE RULE

Figure 31 - Substitution Rule application (from Subshape available at (http://www.engineering.leeds.ac.uk/dssg/downloads/requestForm.php) accessed September 2014)

Task 4 - Solutions manipulation and 5 - SG modification

After this process, there is no possibility of handling the obtained drawing. To change the created SG, we must repeat the process from the beginning.

The menus are not organized according to the sequence of actions to be taken, or set of actions related to the same task. This CW was only possible following the user's guide, as it was not understandable how to perform the correct order of the commands without reading it. Also, if the actions are not performed in the correct sequence, the application crashes, as stated above.

**SGI_Trescak, Esteva & Rodriguez (2009)**

SGI (Tomas Trescak et al., 2009) is an interpreter for two dimensional SG that supports real-time subshape detection, and the author defined its Graphical user interface to have an easy and visual manipulation of the shapes and rules.

Task 1 - Shape creation

This application displays multiple windows with information on the shapes, rules, and grammars. The logic of the creation of all elements is simple and straightforward; the presentation of information is graphical and direct.

The creation of the shapes is performed using freehand lines and curves. We can create multiple shapes, name them, and organize them. The left panel is used to organize and create new shapes, and the lower panel is used to check and classify the shapes created (Figure 32).

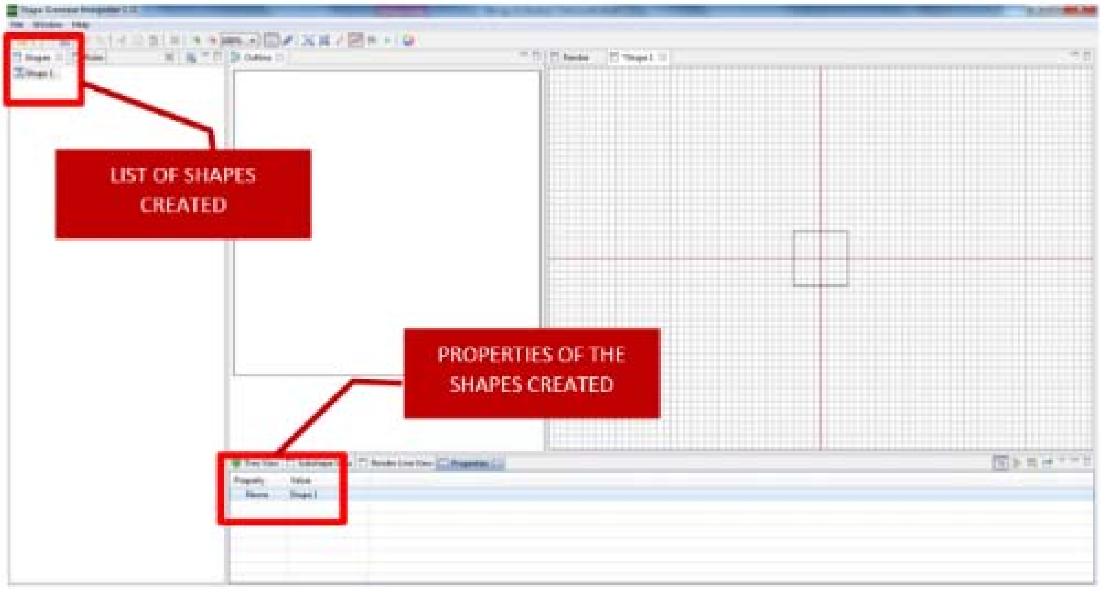


Figure 32 - Shape creation and name configuration (from SGI available at (https://sourceforge.net/projects/sginterpreter/) accessed September 2014)

It is possible to create multiple shapes and use them for rule creation (Figure 33).

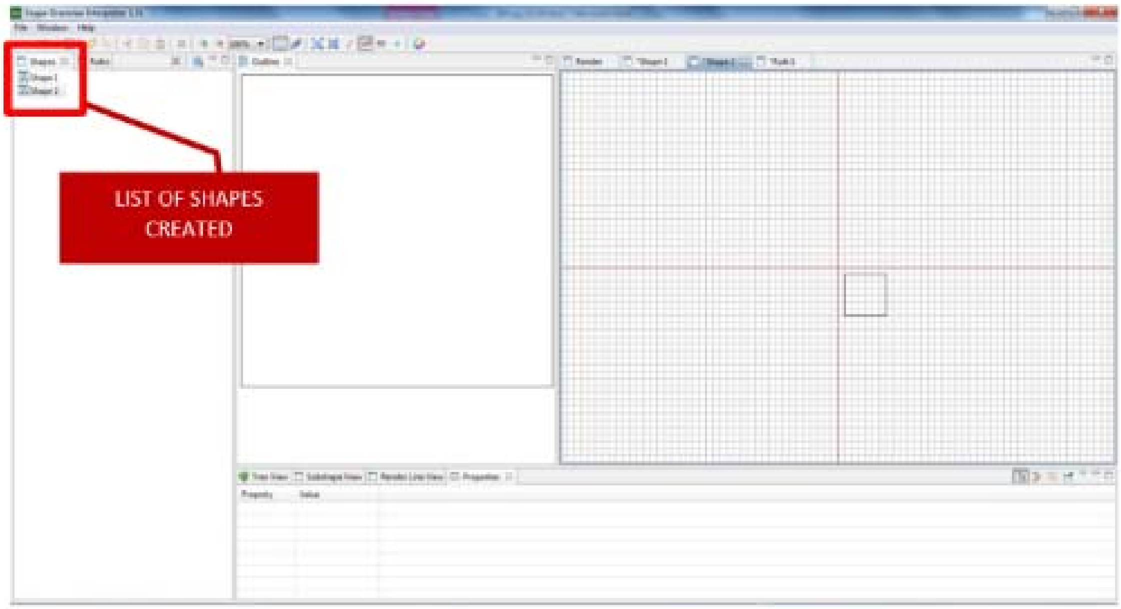


Figure 33 - Shape definition - more than one allowed (from SGI available at (https://sourceforge.net/projects/sginterpreter/) accessed September 2014)

Task 2 - Rule creation

The creation of rules is effortless, and it is shown graphically. In the Rule tab, we can select rules of Substitution, Modification, and Addition (Figure 34).

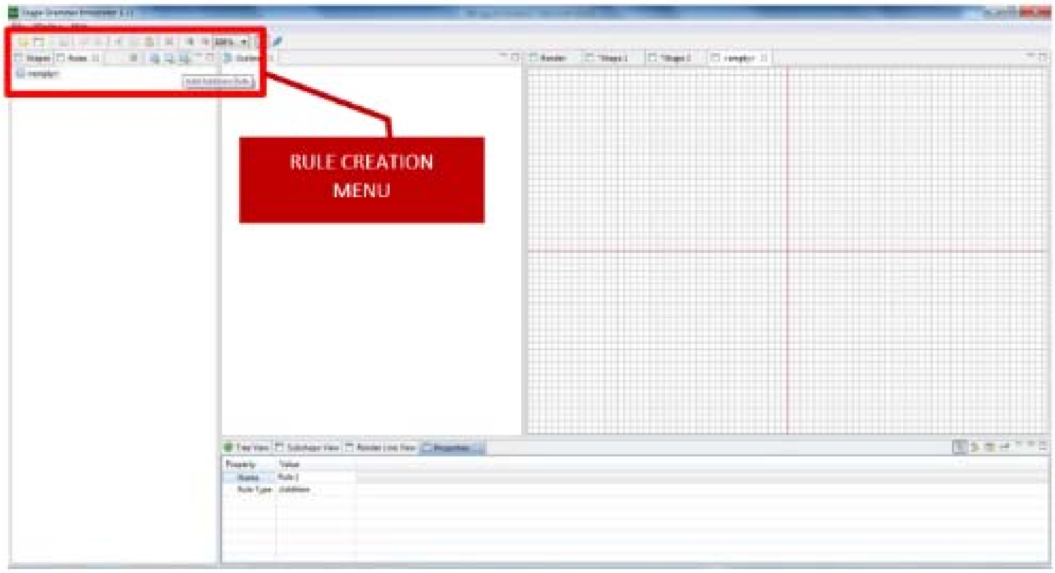


Figure 34 - Rule type selection (from SGI available at (https://sourceforge.net/projects/sginterpreter/) accessed September 2014)

After selecting the rule type, the shapes for the left and right side of the rule are chosen from the shapes created previously. The rule is graphically displayed (Figure 35).

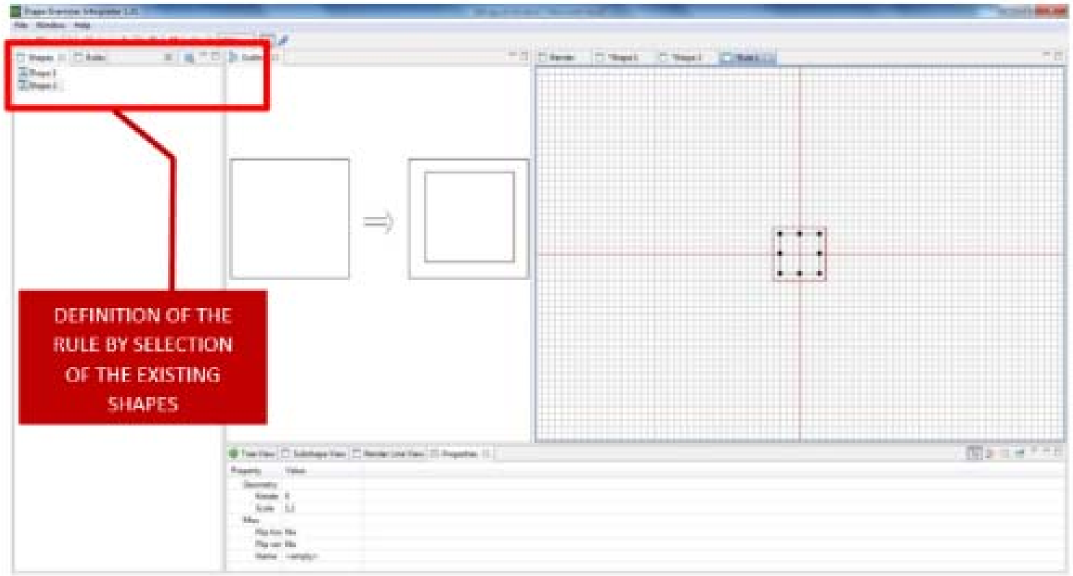


Figure 35 - Choice of shapes for the left and right side of the rule (from SGI available at (https://sourceforge.net/projects/sginterpreter/) accessed September 2014)

The positioning of the drawn shapes is not considered, but the leftmost window can be used to manipulate the shapes, both in position and in size. The changes made in the rule are immediately displayed (Figure 36).

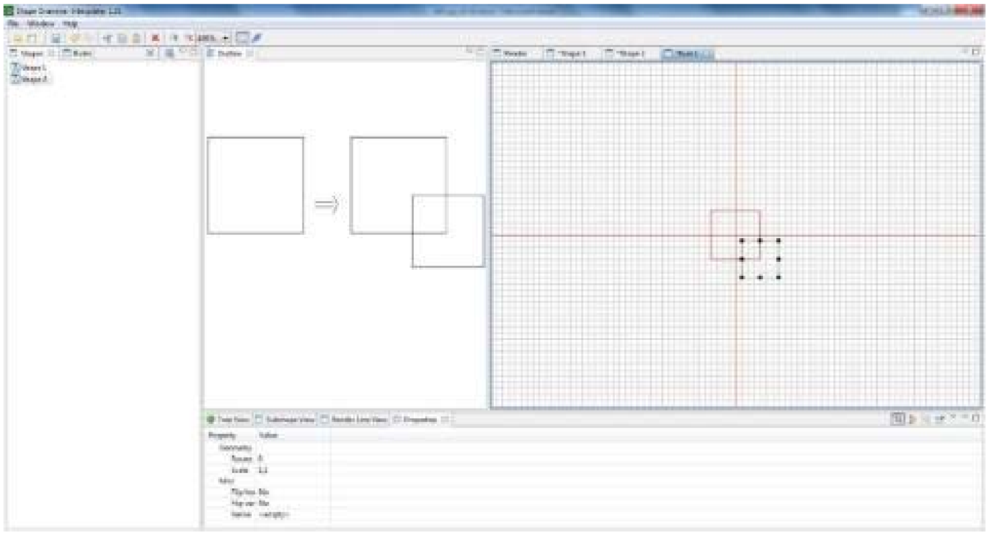


Figure 36 - Direct manipulation of the objects for rule creation (from SGI)

Task 3 – SG rule application:

The SG application is made in the window *Render* with the choice of the number of iterations (Figure 37).

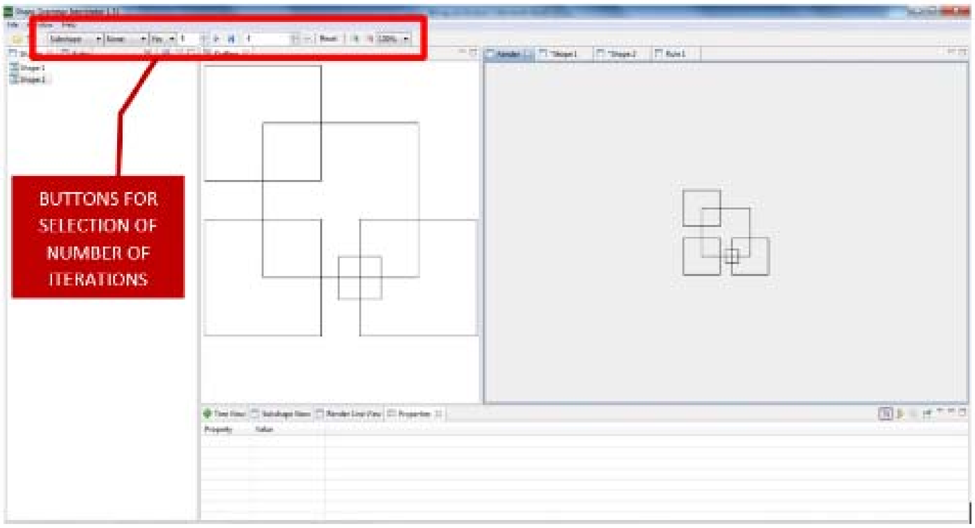


Figure 37 - SG rule application through a selection of iterations number (from SGI available at (https://sourceforge.net/projects/sginterpreter/) accessed September 2014)

Task 4 - Solutions manipulation and 5 - SG modification

The manipulation of solutions is possible only in the change of the number of iterations of the rules created. If we want to change the left or the right side of the rule, we must create a new rule.

**SGDE_Li, Chau, Chen & Wang (2009)**

SGDE (A. I. Li et al., 2009) was created by the authors to implement a system that allowed the user to edit and testing SG, switching easily between the two types of activity, emphasizing this with the graphical manipulation of the shapes.

Task 1 – Shape creation:

The CW to this application was only possible using the manual since there are menus that are not easily seen (they are presented with the names of the windows, looking invisible). It has also become impossible to explore the application without the use of the manual because the inaccurate sequence of the necessary steps, leads to malfunctioning of the software.

The application allows creating shapes by drawing rectangles and lines in a limited drawing area. The procedure is the same as the one for the creation of initial shapes as for the shapes for the left and right of the rule (Figure 38).

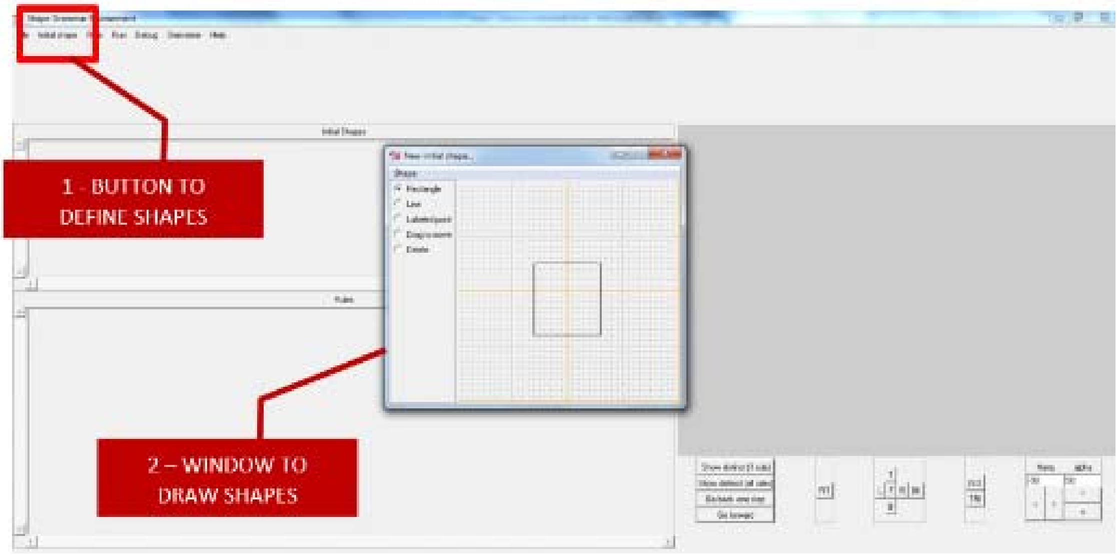


Figure 38 - Creation of polygonal shapes (from SGDE. Available at (http://andrew.li/ downloads/sgdepackage.zip) accessed September 2014)

Task 2– Rule creation:

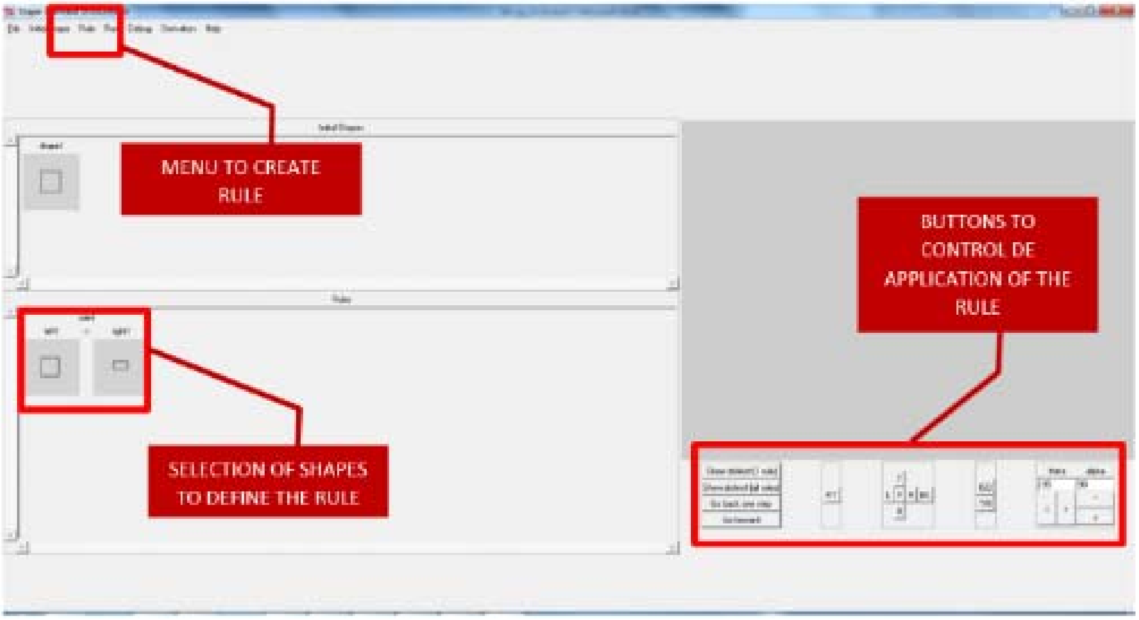

Initial Shapes
MENU TO CREATE RULE
BUTTONS TO CONTROL DE APPLICATION OF THE RULE
Rules
SELECTION OF SHAPES TO DEFINE THE RULE

Figure 39 - Shape creation - drawn as in Fig. 41 (from SGDE. Available at (http://andrew.li/ downloads/sgdepackage.zip) accessed September 2014)

Creating rules is accomplished through the design of the left and right side shapes, after a new rule creation command in the Rule menu (Figure 39).

Task 3 – SG application:

The application of a rule is performed on the Run menu or using the buttons on the right. All possible results are shown in a new window. There are buttons to manipulate the display of the initial shape (from the top, side views or backward), but as the shapes are only bi-dimensional, this option does not give great advantages (Figure 40).

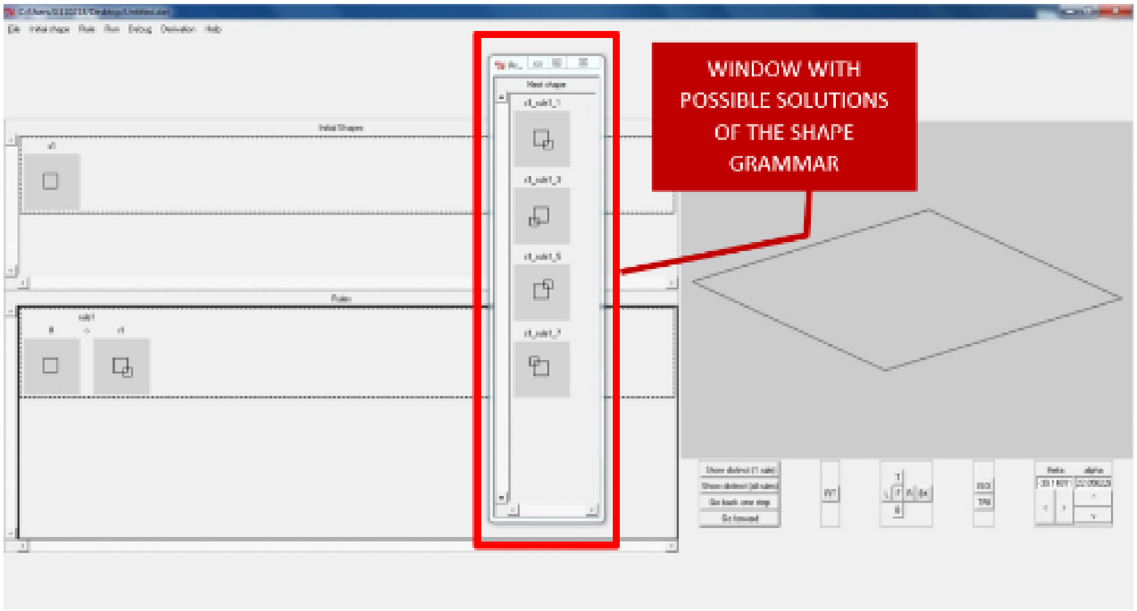


Figure 40 - Solutions visualization (from: SGDE. Available at (http://andrew.li/ downloads/sgdepackage.zip) accessed September 2014)

Task 4 - Solutions manipulation and 5 - SG modification

There is no possibility of manipulating the obtained solutions. We can edit the initial shape and left and right shapes of the rule, creating a new SG rule.

**SD2_Jowers et all (2010)**

SD2 (Iestyn Jowers et al., 2010) comes as an evolution of the Subshape, done by the same authors. In this newer implementation, the user can freely draw shapes, instead of just importing them.

Task 1 – Shape creation:

In this application, we can freely draw shapes. The design is performed using only free lines, which hinders the creation of geometric shapes, but allows great artistic freedom (Figure 41).

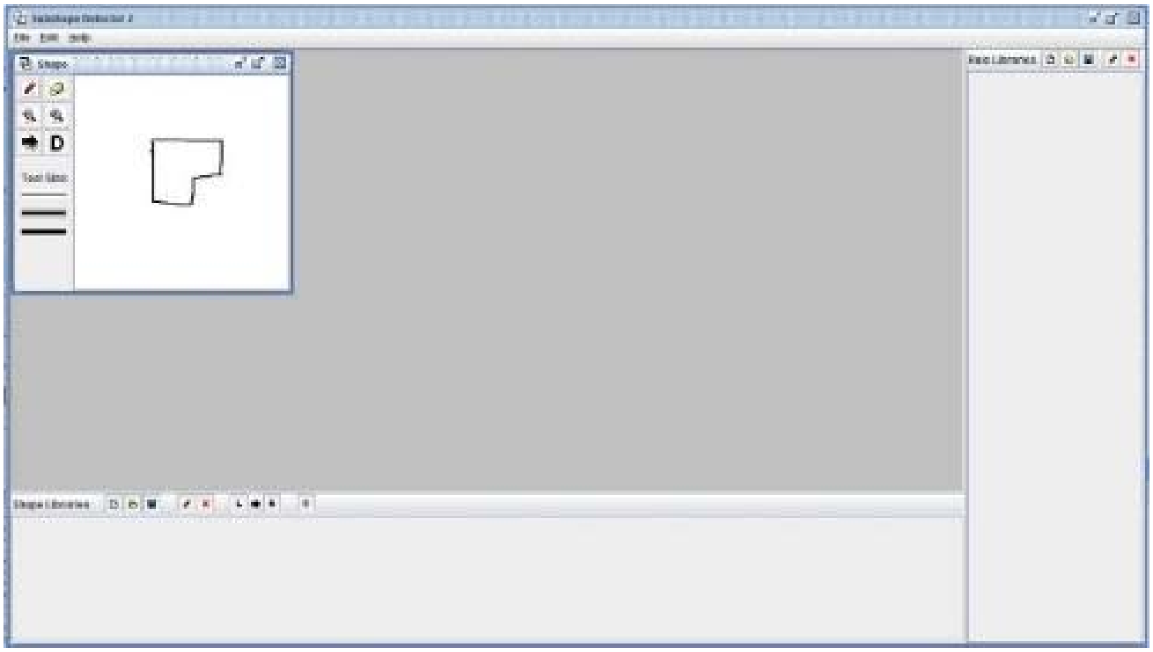


Figure 41 - Free shape drawing (from SD2 available at

(http://www.engineering.leeds.ac.uk/dssg/downloads/requestForm.php) accessed in September 2014)

Task 2 – Rule creation:

The creation of rules is performed by adding drawings to the left and right side of the rule (drawn directly or by opening saved images). In the bottom pane are collected the opened or designed shapes, while in the right pane the rules are created (Figure 42).

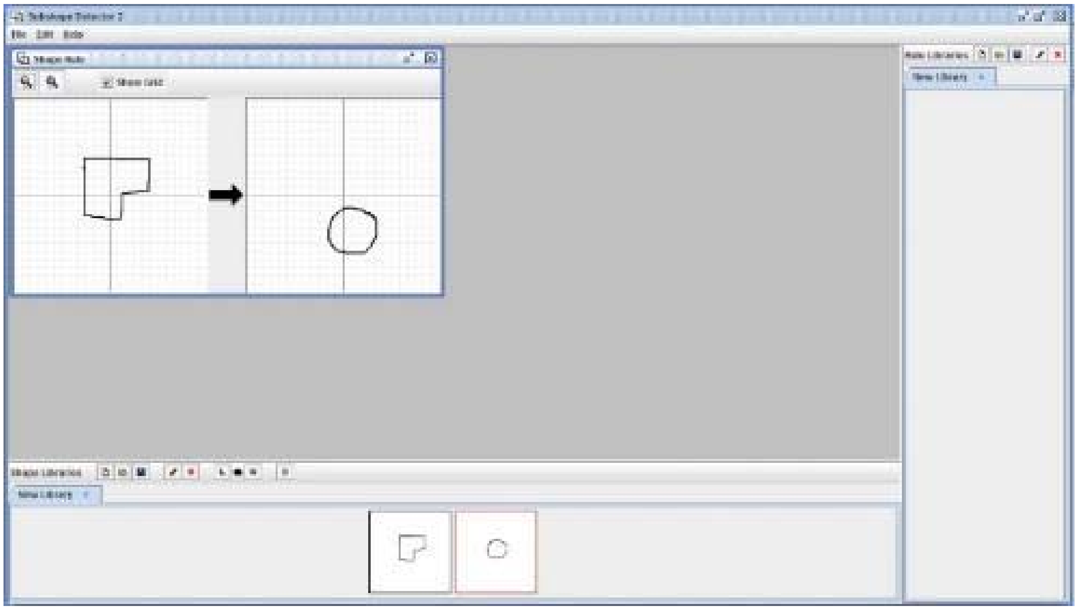
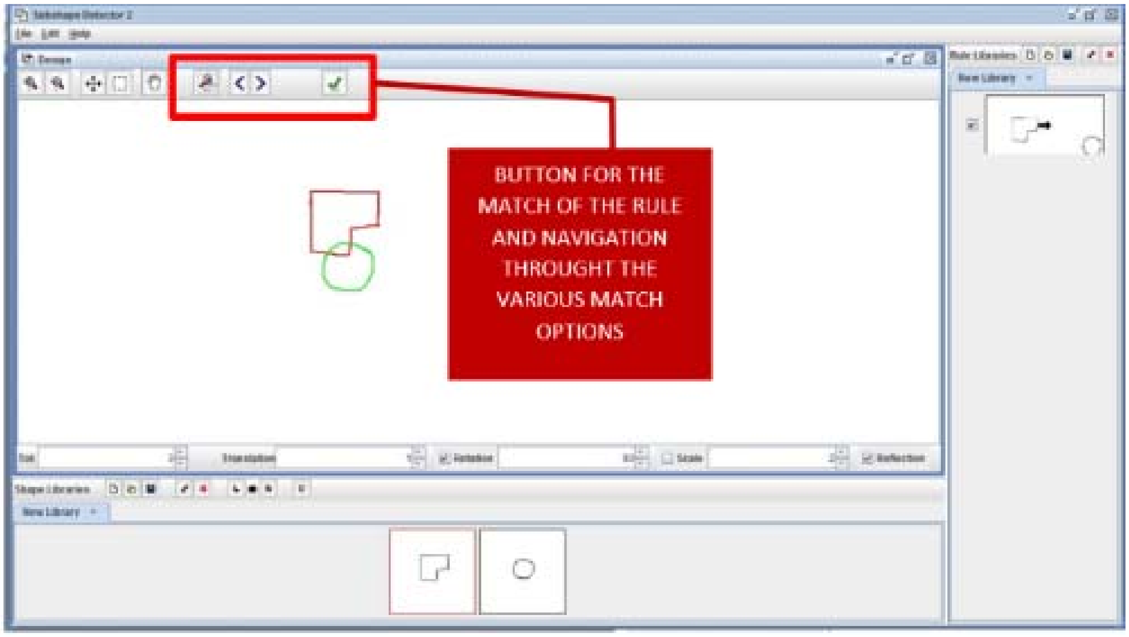


Figure 42 - Shape selection and rule definition (from SD2 available at (http://www.engineering.leeds.ac.uk/dssg/downloads/requestForm.php) accessed in September 2014)

Task 3 – SG application:

The application of the SG is performed by selecting the initial shape, the rule to apply and using of the "D" button (design) where you can search for the rule match and apply the rule. All the rules are replacement ones (Figure 43).



BUTTON FOR THE MATCH OF THE RULE AND NAVIGATION THROUGHT THE VARIOUS MATCH OPTIONS

Figure 43 - Match of the rule and SG application (from SD2 available at (http://www.engineering.leeds.ac.uk/dssg/downloads/requestForm.php) accessed in September 2014)

Task 4 - Solutions manipulation and 5 - SG modification

There is no possibility of manipulating the shapes previously defined or changing what the SG created. Only the creation of new rules and solutions is allowed.

**Spapper_Hoisl (2012)**

Spapper application (Hoisl, 2012) is the most sophisticated of the group, in the sense that it is a plug-in for use in CAD applications. It allows the creation of three-dimensional SG, with no restriction on the orientation of the objects.

Task 1 – Shape creation:

As this implementation is a plug-in for use in CAD applications, we followed the user guide prescription of using a free CAD software called FreeCAD.

To accomplish the tasks, the instructions manual was essential, despite the software's resemblance with other CAD applications. FreeCAD allows the creation of three-dimensional solids, and the tools of this application are used to create the shapes needed for the SG (Figure 44).

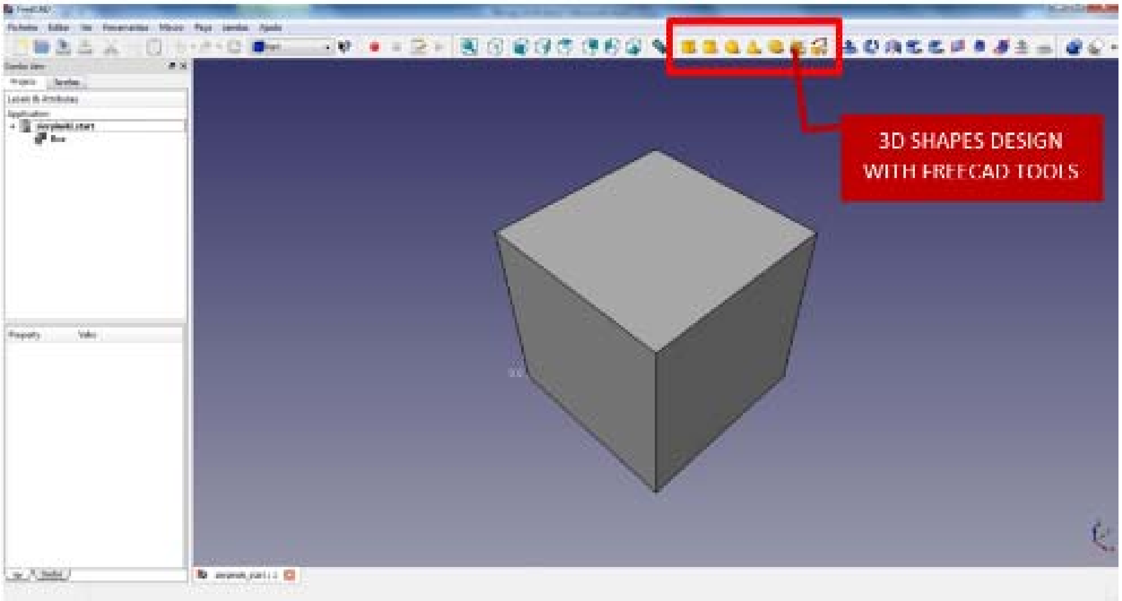


Figure 44 - Solids creation (from Spapper available at (http://sourceforge.net/projects/spapper/) accessed September 2014)

Task 2 – Rule creation:

The Spapper plug-in allows the drawing of shapes in the left and right side of the rule, using the FreeCAD commands to create three-dimensional solids. The rules created use FreeCAD objects visualization, and the handling is made with the modification of the size parameters of the solids. However, there are a back and forward steps that do not make the use of the tool very logic or user-friendly. We need to load the plug-in to open the windows that allows the creation of the left and right shapes, but this action makes the drawing tools disappear. Thus, we need to reload the drawing tools, which make the plug-in disappear and, after drawing the shapes needed, re-load the plug-in to be able to create the rules and SG. This means we cannot have opened all the tools needed during the entire process (Figure 45).

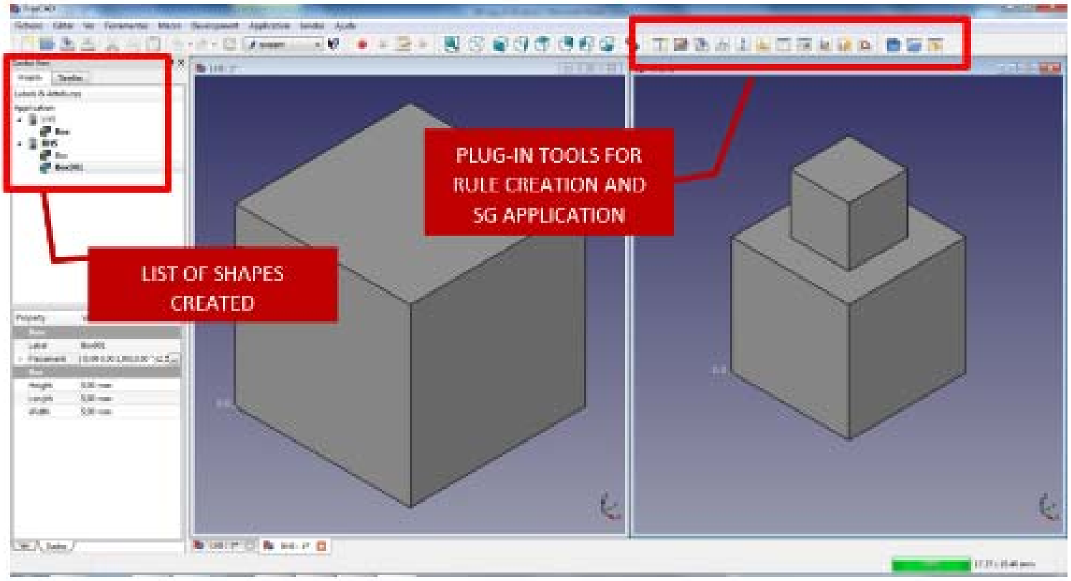


Figure 45 - Left and right side of rules creation (from Spapper available at (http://sourceforge.net/projects/spapper/) accessed September 2014)

Task 3 – SG application:

After creating the rule for the left and right solids, we can apply the rule with the corresponding command. The buttons are not obvious, as it was only possible to understand their function using the manual. It was also only possible to apply the rule saving it, closing the application and re-uploading the saved rule.

The SG application window allows choosing the number of iterations and the number of desired solutions. It also allows rule visualization (Figure 46).

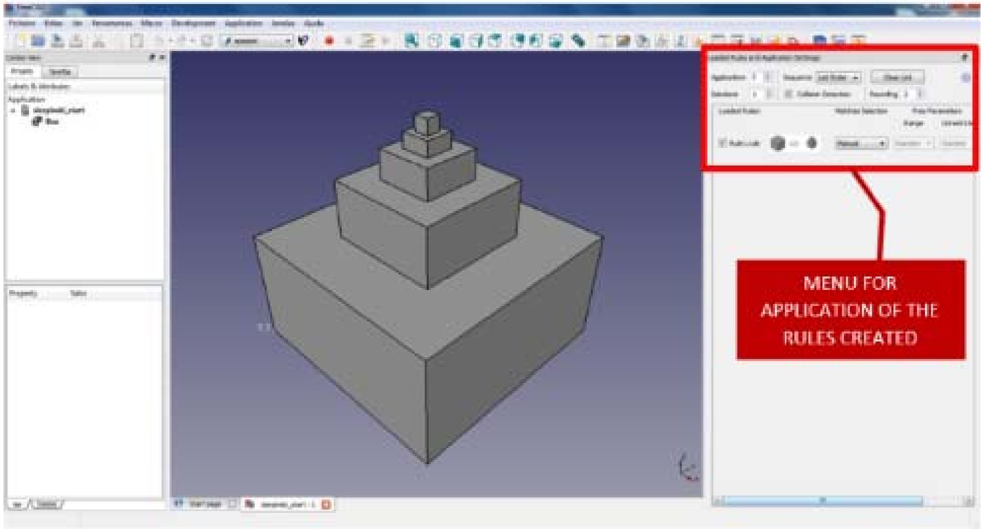


Figure 46 - Shape application (from Spapper available at (http://sourceforge.net/projects/spapper/) accessed September 2014)

Task 4 - Solutions manipulation:

This is the only application that provides full handling of the solutions. It is possible to manipulate the solids obtained with total freedom. This is very interesting for the use of SG as a creative partner and in the design/architecture project.

Task 5 - SG modification:

As in most applications, we cannot directly change the rules created. New rules have to be made with the desired data from scratch.

After completing the CW through these six SG implementations, it is important to stress that all of them seem to be in an early stage of development or just for simple manipulation/visualization of SG. They all seem to be fit for educational purposes, rather than to professional architectural or design projects. The summary of the CW is shown next in Table 6.

***COGNITIVE WALKTHROUGH***

| | Implementation / Authors | **Shaper 2D** (Mcgill, 2001b) | **Subshape** (I Jowers et al., 2008) | **SGI** (T. Trescak, Rodriguez, & Esteva, 2009) | **SGDE** (A. I. Li et al., 2009) | **SD2** (Iestyn Jowers et al., 2010) | **Spapper** (Hoisl, 2012) |
|---|---|---|---|---|---|---|---|
| Tasks | 1.Creation of Initial Shapes | Selection of preexisting shapes | Selection of external drawings in bitmap format | Free draw | Free draw | Free drawing or selection of external drawings in bitmap format | Free draw |
| | 2.Creation of Initial Shapes | Graphical manipulation of the position of shapes | Selection of external drawings in bitmap format or by selecting sections of the initial shape | Free draw | Free draw | Free drawing, selection of external drawings in bitmap format or by selecting sections of the initial shape | Free draw |
| | 3. Application of GF | Immediate visualization with choice of the number of iterations | Recognition of the "match" and then application of an iteration of the rule | Application of the number of iterations chosen | Application of one iteration of the rule at a time, with a preview of the result of the following iteration | Recognition of the match and then applying an iteration of the rule<br>Application of the number of iterations chosen | Application of the number of iterations chosen |
| | 4. Solutions Handling | Direct manipulation of the shapes and rules, immediately changing the results | N/A | N/A | Possibility to regress in number of iterations | N/A | Manipulation of the resulting shapes as GF independent objects |
| | 5. GF Change | Direct manipulation of the shapes and rules, immediately changing the results | Redefinition of rules | Redefinition of shapes and/or rules | Redefinition of shapes and/or rules | Redefinition of rules | Redefinition of shapes and/or rules |

Table 6**.** Preliminary Analysis of the Tasks (Tching et al., 2016)

### 3.1.4 ANALYSIS RESULTS

The present analysis has been carried out in accordance with the purposes of the CW, that is, with the ease of learning and the manipulation of the computer application in sight. However, previously to the conclusions of the scope of this analysis goals, it is interesting to point out a that most of the applications are rather generic and simplified, not only concerning the interface but also in its functioning.

According to the objectives set for this study, we concluded that none of the tested applications would be easily adopted for practical use in design and architecture creative projects. The lack of correspondence between these applications and the ones that are generally used by these groups of users can be pointed as a significant obstacle to its adoption, as shown below.

Since these users are characteristically comfortable to use CAD tools, which have highly qualified interfaces, the non-resemblance with these tools is one relevant aspect to point out. This limitation can be found not only at the interface level but also in the way the tasks are performed. Though working with SG can be rather specific, there is no resemblance, for example, in the way the shapes are drawn in the tested applications, neither there is one in the generic manipulation of the applications when the handling of the computer-assisted drawing is generally established.

Specifically, the integration of Snapper (Hoisl, 2012) with a CAD application is a good solution to address this issue. Considering the SG applications as a plug-in for the CAD tools may be an option, even though it is not reasonable to consider it in our approach to the IM-sgi, as the use of SG is much broader.

The Shaper 2D (Mcgill, 2001c) is the application that stands out because it offers straightforward handling and is very intuitive, making direct object manipulation a reality. The fact that it is effortless to use is a great help.

From the comparison Table (Table 6), we can see that SGDE (A. I. Li et al., 2009) has the general characteristics one would wish on an SG computational application. But, like all the other implementations, there is the need to make the interface more appealing, more user-friendly, and suitable to the designers' needs.

This CW is a crucial step in our research, as it allows us to conclude that we cannot find any resemblance or followed guidelines in the existing SG applications. As we can find a considerable range of SG computational applications, we could understand patterns in the way the authors address the tasks and the interface, but that is not true.

There are common general issues that can be pointed after our CW, and that should guide our definition of an interface model for SG implementations. There are four main important conclusions: (1) no resemblance with the commonly used computer applications in the architectural and design fields. This creates an immediate barrier to the use of these applications for design purposes, as designers would have to learn new ways to manipulate the applications; (2) the interfaces are simple and outdated, lacking in being appealing to the users we want to address, that are sensible to visual style; (3) lack of organization between windows, menus and buttons, many times making the user navigate through different areas of the screen for a simple task; (4) none of the applications show an interface that guides the user through the sequence of tasks he should do to be successful in creating and using SG.

With the CW performed, we have a starting point of important issues that IM-sgi should answer. As defining the interface is an enormously time-consuming task, we believe that with IM-sgi we address the issues that are usually left behind when developers create the SG implementations and help these implementations leap forward, helping SG to leave the research field and enter the design practice field.

# 4 IM-SGI – INTERFACE MODEL FOR SHAPE GRAMMAR IMPLEMENTATIONS

After Chapter 3, where we defined our target users, tasks to work with SG, and studied existing SG implementations in search for use patterns and interface characteristics, we present in this chapter our proposal of a model of interface for SG implementations, that we named IM-sgi.

We start to revisit the knowledge areas identified as precious for our research now in the correct perspective according to the conclusions from the previous chapter.

The design of the UI is a creative process, and often, designers have difficulty thinking like end-users. The usability is linked to learning, efficiency, productivity, ease of memorization, satisfaction, and no errors. Good usability is essential, as it reflects the notion of the quality of the user's system. Good usability allows beginners to become productive more quickly, experts to be more efficient, and produces a reduction in errors. The real needs need to be carefully identified so that the application is successful on the market (B. Myers & Ko, 2009).

As already stated in Chapter 2.2, the interface is essential for the success of SG implementations, as it is the means of communication between the computational tool and the designer's goal. To address the interface, acknowledged its essential role, the UI has grown with the use of models (Martins & Garcia, 2015). The UI Models allows the creation of UI variations for the same application, allowing an easier definition of the interface usability and test of the hypothesis (Van den Bergh et al., 2010). But also, the model itself can serve as validation for the interfaces created, as they help to test the structure and event sequences according to what is expected by the model (Hauptmann, 2012).

IM-sgi definition started with the analysis of the interaction model developed by Scott Chase (Chase, 2002), allowing us to define types of users and types of interaction with the SG tool. All the research and analysis carried out to this point had the objective of gathering the necessary information for the proposal of an interface model for computational implementations of SG, whose goal is to break the barrier that is believed to exist between these implementations and the user.

The interface has become an ever-growing component of a computer system as computers become more powerful and larger groups of people use them. Currently, problem-solving in computer systems is often more related to the interface than to hardware or software (M. Nielsen, Störring, Moeslund, & Granum, 2003).

IM-sgi aims to answer the need for computing ergonomics and suitability to architects work that SG Implementations should respond. In Chapter 4.2, we enter deeply in the characteristics of the users that our model aims to serve – architects/designers, artists, and students.

IM-sgi makes the bridge between the user and the SG implementation, expanding the use of SG to the project practice and beyond the analytical and educational fields where they have mainly been used.

## 4.1 Structure of the Interface Model

As the design of interfaces includes everything visible to the user, it extends in depth to the interactive system of the application as a whole. A good interface cannot be applied to a system only after it is built; it should be part of the system design process from the outset. A correct interface can make a dramatic difference in learning time, performance speed, error reduction, user satisfaction experience, and retention of operations knowledge over time.

The definition of the interface starts with the analysis of the tasks, that is, the understanding of the problem domain and what tasks the user performs. In this way, the main actor is the user, the terminologies, and conceptions used in his work are the important ones, not the programmer ones. A good understanding of the cognitive behavior characteristic of the user groups that use the computational application is essential for the application to function according to their capabilities and limitations. In this way, knowledge of the nature of the work performed by the user is critical (M. Nielsen et al., 2003).

Based on all the knowledge acquired with the SG study, its computational implementations and defining the objectives that are considered essential for SG to be effectively used as partners in the creative project, we propose an interface model so that it can be generically used in applications for SG. The definition of the interface model is based starting on the conclusions reached with the analysis made, that is, highlighting the pertinence of two theses that were chosen as adequate for the reasoning of this proposal and in the HCI discipline, more specifically in the ID area to ensure that the principles of a good computer interface are met. Thus, there are two background studies that helped to define IM-sgi: (1) (Chase, 2002) that deepens the various hypotheses of interaction between different users and the implementation of SG, also defining which type of response to the application is the most adequate according to the needs of those same users: and (2) the implementation of Miranda Mcgill (2001b), whose CW test proved to respond positively to the analyzed criteria and presents an interface with potential for development.

More deeply, the structure of the IM-sgi model has support on the Scott Chase model (Chase, 2002) to define the different types of users, their needs and objectives, the modes of use of the implementation by each user group and how the interface should respond to them. On a more micro-scale, the criteria for the interface model follows the principles of the ID and is based on the Shaper 2D application conclusions, of the various applications analyzed, as it stood out for its intuitive mode, ease of use and direct visualization of actions and results.

ID principles focus on creating user experiences that empower the way people work, communicate, and interact (Rogers et al., 2011), distinguishing itself from software engineering that focuses on the production of software solutions for a particular application.

For ID principles to be successfully applied, it is necessary to understand how users act and react to events and how they communicate and interact. Thus, an ID process is the first step in the identification of requirements and requirements definition and then designing interface design alternatives that comply with these requirements, which must be simulated and tested. This first step is, thus, the definition of our interface model. Next, prototypes are made that comply with the model and can be tested to verify their correct applicability. According to (Rogers et al., 2011), there are six main objectives to be fulfilled by the interface: effectiveness; efficiency; safety; good usability; ease of learning and ease of use memorization. On the user side, the ID defines that the systems created should be enjoyable, funny, helpful, motivating, aesthetically appealing, supportive of creativity, and emotionally rewarding.

In summary, ID is a process that aims at thinking about a design problem, knowledge of user needs, the proposal of conceptual models, the prototyping of these models and the evaluation of them regarding usability and objectives of use by the user. It is an iterative process, and new solutions should be given when the prototype test does not deliver the desired results, rethinking the conceptual model, and if it is suitable for the user's tasks.

We can summarize the ID in four activities: Identification of needs and definition of requirements; development of design alternatives that meet these requirements; construction of interactive versions that can be communicated and tested and, finally, to evaluate these alternatives by measuring their acceptance.

According to the principles of ID, it is essential to create conceptual models when creating an interface. Since early HCI developments, as pointed by David and Liddle (1996), the conceptual model is the key point for creating a good interface design. An interface conceptual model refers to the description of the system through a set of ideas and concepts of what the system should do, how it should function and the appearance it should present so that it is understood and used by users. According to the ID, tasks and domain are analyzed in a user-centered approach, making it the center of development and not any other technical issues.

The development of a conceptual interface model involves product forecasting, in this case, a computational implementation of SG and user needs. This model can be designed through an interface metaphor that defines the basic structure of the model, and interaction paradigms should be used to create the conceptual metaphor of the model.

Conceptual interface models can be of two types: action-based or object-based. In the present proposal, we develop a model based on actions, that is, the IM-sgi describes how the users carry out the tasks in the system. Also, there are several model assumptions as to how users perform the tasks. The most common types of activity are instruction; conversation; manipulation navigation, and exploration navigation.

IM-sgi is a conceptual model based on actions of manipulation and exploration, describing the activities of manipulation of objects and navigation in virtual spaces, exploring the user's knowledge regarding their accomplishment of tasks in the real world (in this particular case, the real world refers to the manipulation of SG in definition and manual drawing). An example of such a conceptual model is the direct manipulation of objects (Shneiderman & Plaisant, 2007). Such examples are the CAD applications commonly used by the user groups considered for this research.

For the definition of an interface model, it is important to consider several levels of abstraction to define the design and implementation of each of these. In this sense, they should respond to four levels of abstraction (M. Nielsen et al., 2003): (1) Conceptual - general description of what the application presents to the user and what potential has; and (2) Semantic - description of the functions performed by the system, focused on the functional requirements of the system and not on how the user performs those functions; and (3) Syntactic - description of the sequence of inputs and outputs required to perform the functions described above, meaning the interaction styles of the application; and (4) Lexical - determination of how the syntactic level is translated from formal application operations.

The Conceptual and Semantic levels of the application were explained throughout the present thesis. At the Conceptual level, the interface refers to computational applications that allow the work with SG and the Semantic level was the one tested by the CW, that is, the system should allow the creation of Shapes and Rules that define SG and the consequent use of them to create results.

The IM-sgi criteria define the Syntactic level of the interface model, describing how the different actions should be accessible to the user and how they relate to each other. The Lexical level lies at the end of the proposed interface model, that is, the interface model reflects the entire definition of Conceptual, Semantic and Syntactic levels, and the Lexical level is reflected in the realization of interfaces that follow the model proposed. This level is tested in this thesis through interface prototypes that follow the proposed model.

In sum, in this chapter, we develop the conceptual, semantic and syntactic levels of the interface model for implementations of SG that we call IM-sgi and in the chapter after, prototypes are produced that meet the defined model, reporting to the Lexical level of the interface.

## 4.2 User interaction / SG computational tool

As previously identified, the IM-sgi model must respond to the four levels of abstraction that allow a clear definition of what interfaces should follow. To define the conceptual and semantic levels, it is essential to understand who the users of the SG applications are, for whom the interface model is proposed and how they interact with the tools.

For this definition, the model of interaction of Scott Chase described in Chapter 2.2.1 of this thesis was used as the foundation (Chase, 2002). Based on the model of this author, an adaptation is developed that replies to the vision of this proposal, in terms of users who can carry out creative projects using SG.

CAD applications are at the forefront with their quite sophisticated interfaces. However, the interfaces of the implementations of SG are in the opposite direction (Chase, 2002). What happens in existing SG applications is that they have a minimal choice in how they interact and allow the control of the system's actions. Although the SG study has been around for decades, researchers have only recently focused on issues related to interaction in SG systems. It is in this context that the interaction model of Scott Chase is of great relevance and originates the present research.

The model proposes the development of an interface model that concretizes the modes of interaction of the user with the SG system in an interface model, outlining how the application and the user communicate to ensure the intended interaction.

In this sense, the model of interaction of Scott Chase, that responds exactly to these aspects, is revealed of important pertinence and relevance. As already mentioned, this model (Chase, 2002) is used to support the definition of user categories and how they interact with the SG system. According to Chase, there are several possible scenarios for controlling the computational applications of SG. The author considers that there are three distinct entities: The creator of the system, the designer, and the computer. Scenarios vary in the level of control of each of these three entities; the control may vary between being wholly on the user's side or totally on the computer side. Thus, with the focus on the user, the user may have full control, partial control (sharing part of the control with the other two entities) or no control.

Chase's model of interaction opens the chance for combination with a model for the Interface of SG implementations. Enhancing the interface as having great importance for the success of SG implementations it allows the connection of the computational tool to the goal to accomplish with it.

According to Chase control scenarios presented in Chapter 2.3.1, there are several possible scenarios for the control of SG implementations.

Scenarios 5 and 6, where the computer system has full control of events and only allows the user to be a spectator of results, without any intervention, were excluded from this study since they fall outside of the scope of studying how to use SG in creative projects.

For each of these groups, the interface should behave distinctly, as the level of manipulation of SG, either in the side of shapes, of the rules, or both is different. The most basic use is more directed to shape handling. In the opposite direction, the greater the knowledge and specific objectives, the biggest is the focuses on rules manipulation.

For a better definition of the stated above, a circular scheme is presented (Figure 47), reflecting the various possibilities for manipulation and control of SG implementations according to these three groups. This scheme considers valid options both for analytical and original SG.

This circular scheme must be read as follows: each circumference section relates to the following inner and out circumference sections that "touch" it. On the other hand, the central axes of each section refer to the most complex action possible for each user group.

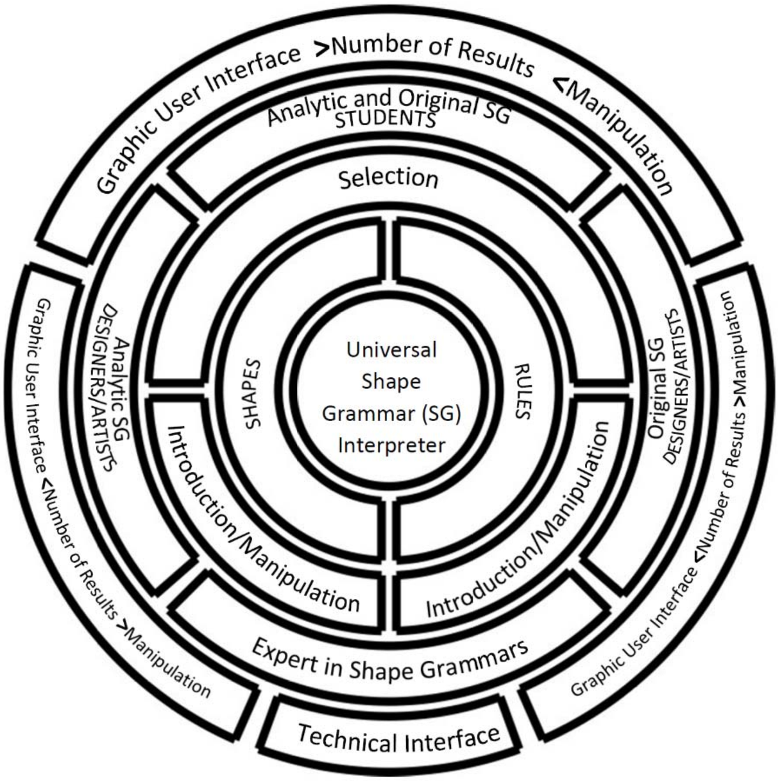

Graphic User Interface >Number of Results <Manipulation
Analytic and Original SG
STUDENTS
Selection
Analytic SG
DESIGNERS/ARTISTS
Graphic User Interface <Number of Results >Manipulation
SHAPES
Universal
Shape
Grammar (SG)
Interpreter
RULES
Original SG
DESIGNERS/ARTISTS
Graphic User Interface <Number of Results >Manipulation
Introduction/Manipulation
Introduction/Manipulation
Expert in Shape Grammars
Technical Interface

Figure 47 - Scheme of relations between users and IM-sgi interface model

The schematic representation of the relationships between users and the interface was done in a circular diagram to symbolize the various layers of information involved. We can verify that the core is the universal SG interpreter and that the outermost layer is the interface. Among these layers we have, from the inside to the outside, the generating elements of SG (shapes and rules), how these elements are worked (selection, introduction/manipulation) and the user groups.

The outer layer defines which type of interface is adequate to each group (Graphical User Interface or Technical Interface). It also defines if we should have more or a smaller number of results and ability to manipulate, with the signals “>” and “>”.

When analyzing a specific user group section, the outer layer that connects to it describes the characterization of the interface for that group, in the aspects of manipulation and results, and the inner layers reflect how the user manipulates Shapes and Rules to create SG.

For better understanding, next is presented the same scheme, but coloring each of the several possible readings, one for each user group.

Below, Figure 48, Figure 49 and Figure 50 illustrate the depth of control for each group of users, illustrating with color the segments that correlate for each group. Most basic users (Students) have no direct access to anything, but the immediately following layer (Figure 48). Intermediate users (Designers/Artists) can skip a layer and directly manipulate “deeper” elements (Figure 49). Finally, expert users (Experts in SG) can skip all layers and directly manipulate the core (Figure 50).

The segmentation of the user actions is not related to the imposition of limitations; it is related to the adjustment of the interface accordingly to the user's needs. This premise allows the interface to be evolutionary and shows levels of complexity according to the tasks that user groups perform. This adaptation allows the same computational application to communicate differently and adjusted to each group of SG users identified in this proposal. This proposal hypothesizes that exposure to too much information is detrimental to understanding and that segmenting the tools can lead to better usability. Considering that the user looks for them in case of need or exploration of new possibilities does not reflect a guarantee (John M Carroll & Rosson, 1986).

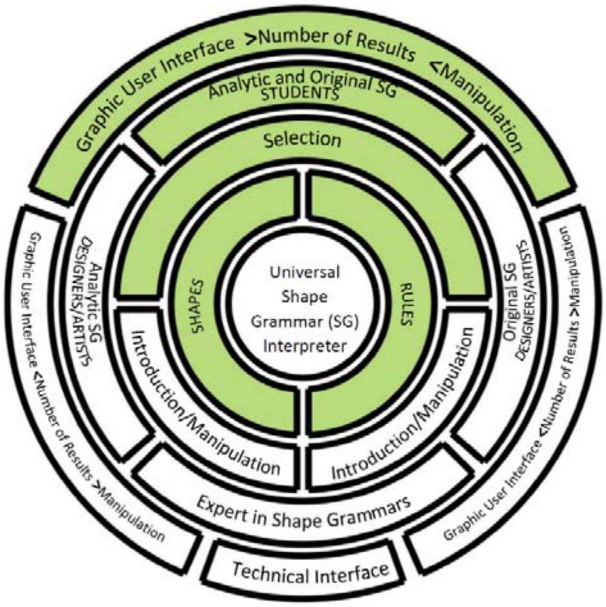

Graphic User Interface >Number of Results <Manipulation
Analytic and Original SG
STUDENTS
Selection
Analytic SG
DESIGNERS/ARTISTS
Graphic User Interface <Number of Results >Manipulation
SHAPES
Universal
Shape
Grammar (SG)
Interpreter
RULES
Original SG
DESIGNERS/ARTISTS
Graphic User Interface <Number of Results >Manipulation
Introduction/Manipulation
Introduction/Manipulation
Expert in Shape Grammars
Technical Interface

Figure 48 - Interaction scheme for "Students" group

In Figure 48 above it is indicated the type of interface that is set to "Students", the most basic of the three, focused on the use of SG in a perspective of learning and exploration. This interface is intended to allow the selection of pre-existing shapes and rules, already existing in the computational application, so that the users can easily learn and understand how they relate and create solutions. With this purpose, the interface must have a greater number of results and fewer possibilities of manipulation.

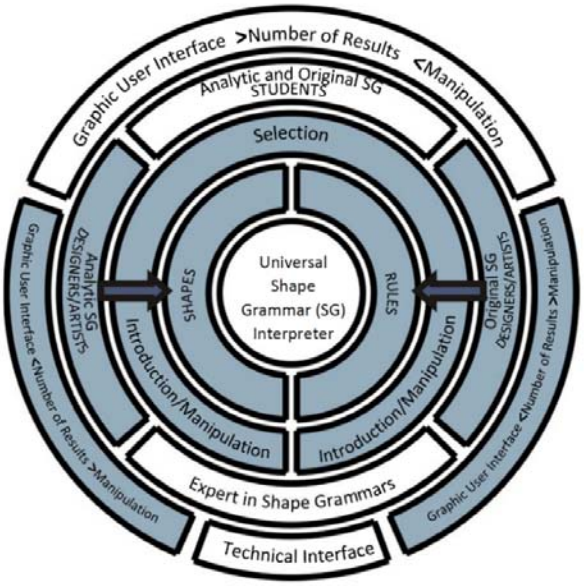

Graphic User Interface >Number of Results <Manipulation
Analytic and Original SG
STUDENTS
Selection
Analytic SG
DESIGNERS/ARTISTS
Graphic User Interface <Number of Results >Manipulation
SHAPES
Universal
Shape
Grammar (SG)
Interpreter
RULES
Original SG
DESIGNERS/ARTISTS
Graphic User Interface <Number of Results >Manipulation
Introduction/Manipulation
Introduction/Manipulation
Expert in Shape Grammars
Technical Interface

Figure 49 - Interaction scheme for "Designers/Artists" group

Figure 49 illustrates the type of interface set to "Designers/Artists" group, which is considered the intermediate mode regarding the knowledge level and need for SG manipulation.

In the outermost layer, it can be seen that the interface needs to present a smaller quantity of results and shall permit greater manipulation. At this level, Shapes and Rules can be created by the user, as preexisting ones may not fill all his needs.

Reading this diagram from the left side, users can handle the application for Analytical SG, focusing on the Shapes, and reading from the right; users can manipulate the application for Original SG requiring handling the Rules. For both the Shapes and Rules, it remains possible to work only by selection, as in the most basic mode of interaction.

This is the privileged interface for users that can work with SG implementations for real architectural, design, or artistic projects. It requires a certain level of SG knowledge, and that's why it is on the center of the scheme, as we can consider it the one that would answer to the objectives of our research – the use of SG in creative projects.

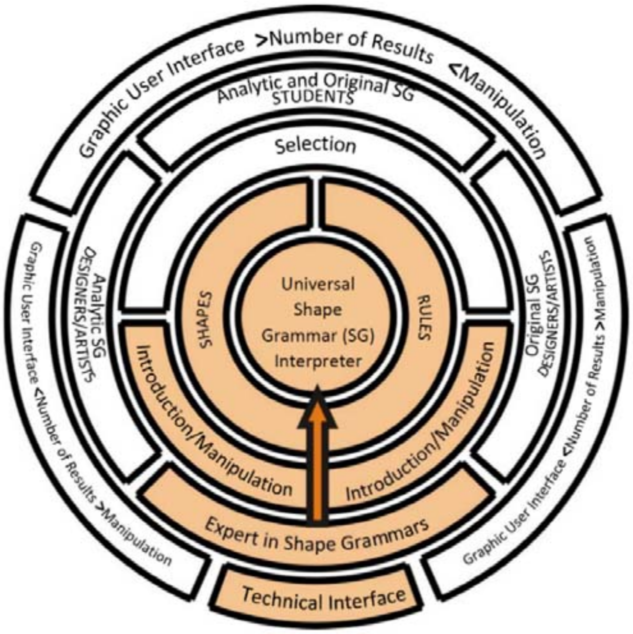

Graphic User Interface >Number of Results <Manipulation
Analytic and Original SG
STUDENTS
Selection
Analytic SG
DESIGNERS/ARTISTS
Original SG
DESIGNERS/ARTISTS
Graphic User Interface <Number of Results >Manipulation
Graphic User Interface <Number of Results >Manipulation
SHAPES
Universal
Shape
Grammar (SG)
Interpreter
RULES
Introduction/Manipulation
Introduction/Manipulation
Expert in Shape Grammars
Technical Interface

Figure 50 - Interaction scheme for "Experts" group

Figure 50 illustrates the type of interface set to "Experts of SG" group, which is considered the expert mode regarding the knowledge level of SG and programming knowledge. These users are considered to be able to manipulate the universal SG interpreter so that they can expand the implementation abilities and range of solutions.

In the creation of a computer application, the interface is a time-consuming task and, frequently, the program developer is not familiar with the true needs and limitations of the end user. A model that specifies categories of users of SG and their objectives, that clarifies how each type of user manipulates SG and what barriers of communication need to be addressed increases the possibilities of success of the use of SG implementations.

With the users and levels of interaction defined, we go to defining the interface model criteria.

## 4.3 Interface Model Criteria

As described above in the State of the Art, the principles of HCI allow the creation and evaluation of computational interfaces according to criteria of good usability of the same ones. Considering that the HCI principles guide the infrastructure of the interface model, specific criteria are needed to define the particular IM-sgi structure.

In this way, the Syntactic level of this model follows the principles of this discipline to reach the desired goals of usability. This level begins by defining which interaction style best fits the applications in question. Examples of interaction styles are command lines, menus, shapes, natural language, direct manipulation, virtual reality, or multimodal combinations.

The CW performed allowed us to conclude that the GUI is the most suitable for the implementations of SG, which is characterized by direct manipulation and that mimics the hand drawing of the architect when exploring design ideas. Choosing the best interaction style for the application is a key step to the effectiveness of the system.

In a GUI-interface style, a set of objects is displayed on the monitor, and the user can perform manipulations on those objects. This means that the user does not need to remember command language, in addition to standardized manipulations. Its main exercise is the recall of available objects and their states, which should be continuously displayed on the monitor (M. Nielsen et al., 2003). By pointing to objects and actions, the user can quickly accomplish the tasks, watch the results immediately, and go back just as quickly. Inputs on the keyboard or use of menus for choices are overridden by cursor movement devices such as the mouse or direct manipulation on touchscreens with virtual pens or hand.

Direct manipulation is appealing to apprentices, easy to memorize for intermediate users, and encourages exploration. It can also be a quick tool for experienced users. In this type of interface, user input actions should be as close as possible to the thoughts that motivate these actions, reducing the distance between the user's intention and the input action in the application. This distance is reduced by analogies to existing human capacities (pointing, grasping, moving objects in space), as opposed to trained behaviors (M. Nielsen et al., 2003). The combination of styles may be appropriate when the needs of the users are diverse and/or complex, and there may be a need to use, for example, menus to control direct manipulation environments when a particular operation is not found.

As early as 1983, Schneiderman pointed to direct manipulation as the best way to generate enthusiasm among users, facilitating their confidence in the ability to perform tasks, achieve mastery in the application and feel satisfaction in using it (Shneiderman, 1983). The author describes direct manipulation through four essential aspects: (1) permanent representation of the object of interest; (2) physical actions (movement and selection by the mouse) or use of subtitled buttons, instead of complex syntaxes; (3) fast, incremental and reversible operations, whose impact on the object of interest is immediately visible; and (4) approach at different levels in relation to learning, allowing the use of the application with minimum knowledge, so that learners can learn what is necessary at a more fundamental level and with exercise and exploration become experts.

In short, the author points out that satisfaction in the use of a computational application arises from the ability to manipulate the object of interest directly and the possibility of generating multiple alternatives quickly. By asserting the four principles outlined above, users experience less anxiety as the system becomes clear, and actions are easily reversible, promoting confidence, feeling control, and facilitating the prediction of system responses. Basic users also quickly learn essential functions; intermittent users easily retain the concept of operations, and expert users can work quickly and even define new functions.

The completed CW on Chapter 3.1.3 identified that a graphical interface helps users to perform the tasks to work with SG. The IM-sgi is a graphic interface of direct manipulation, as the criteria are defined in this chapter. The IM-sgi criteria are the basis for the creation of interfaces for computational applications of SG. The design of the criteria is the definition and operationalization of the usability dimensions (J. M. C. Bastien & Scapin, 1993).

### 4.3.1 System Actions

The mental perception of a system is formed mainly by the interpretation of its visible structure and its actions (D. A. Norman, 1998). The conceptual mental models of the user make him/her have a certain perception of the world, and small clues that refer to these are sufficient for an understanding of the system and a rationalization of its functioning. In this way, visual cues regarding the operations to be performed make the system simple to interpret, preventing frustration and error.

The addition of controls and displays causes the user to lose sight of all the functionalities and concentrate on one or two fixed functions that match their needs, ignoring the purpose of the whole system. In this way, the best way to present the operation of an application is to keep its structure and actions visible. There must be individual controls with singular functions, keeping the system understandable. The relationships between the user's intentions, the necessary actions, and the results are sensitive, not arbitrary, and perceptible (D. A. Norman, 1998). When the possible actions are in greater number than the controls, the application becomes difficult to control. When all functions are visible and directly controllable, the controls themselves serve as a reminder of these functions, requiring less mental effort from the user. The location of the controls should also be in accordance with the sequences of actions required and the relationship between them, facilitating the flow of activities, and again, the memory of what is to be performed by the user.

To perform an action, there are four aspects to consider: (1) the purpose of the action; (2) what is realized in the universe of action; (3) what is this same universe; (4) and finally, the alteration as seen in this universe by the accomplishment of the action.

The action itself, in turn, is divided into two aspects: the execution and the evaluation, that is, the realization of something and the verification of its realization. Combining all these aspects, the author defines seven levels of an action, which are divided into: (1) goal formation; (2) intention formation; (3) specification of the action; (4) execution of the action; (5) perceived state of the universe; (6) interpretation of the state of the universe; and (7) evaluation of the result.

These seven levels serve as guidelines for designing the interfaces, providing a list of issues that must be addressed to ensure that the two aspects of action, implementation, and evaluation are met. On the other hand, the two aspects of actions (execution and evaluation), are revealed in two principles for the interface: (1) visibility - make relevant elements/actions visible, and (2) feedback - give each action an immediate and obvious effect.

In the scope of visibility, the following questions must be answered: what moves, what is fixed, where / how the object is grasped, which part of the object is manipulated, what movements are possible (push, pull, rotate).

In the context of the Feedback, the questions are regarding what changes have occurred, what has changed about the previous state, and where changes can be detected. For these questions to be answered, we return to the scope of visibility, as the changes to be observed must be visible, clearly demarcated, and the outcome of the actions immediately apparent.

In short, the design of the interface should ensure that the user knows what to do and perceives what is happening, that is, it must assured that: (1) it is easy to determine what actions are possible at any time; (2) the elements are visible, including the conceptual model of the system, (3) alternative actions and results of actions are understandable; (4) facilitates the evaluation of the current state of the system; and (5) follow natural links between intentions and necessary actions, actions and results, and between the information that is visible and the interpretation of the state of the system.

Following all the principles outlined above, all these questions are now referred to in IM-sgi. The tasks that the user performs with the SG computer system that were identified for the CW, regardless of their level of mastery, are the same actions that guide the definition of the IM-sgi model.

### 4.3.2 Ergonomic criteria

IM-sgi defines the criteria that the interface for SG implementations should satisfy. Why should we have a specific model for these interfaces and not just follow general guidelines? Because it is not possible to define generally what a good interface is. This is dependent on the tasks to be performed and the users who manipulate the interface. As stated before, Usability measures the quality of the interface elements related to their usefulness. According to Usability Inspection Method (Molich & Nielsen, 1990), Usability is a quality attribute that allows the understanding of how easy UI is to use. To continue developing IM-sgi, after understanding the needs of the users, the next step is to define the criteria that guide the interface to good Usability. To achieve this goal, we gathered existing criteria to evaluate the usability and ergonomics of the interface and interpreted them as keys that clearly define the interface model. We then apply it to the tasks the user needs to perform in an SG Implementation.

Since the development of HCI, countless authors have developed several general design guides, sets of guidelines, checklists, standards, and heuristics to help achieve good UI (J. M. C. Bastien & Scapin, 1993). Some examples are standards like ANSI, DIN and ISO, design guides from (Brown & H., 1988), (Shneiderman & Plaisant, 2007), sets of guidelines from and style guides from (IBM, 1990) and (Apple Computer Inc., 1992), and heuristics like those from (J. M. C. Bastien & Scapin, 1993), (Molich & Nielsen, 1990) and (J. Nielsen & Mack, 1994). Despite the specificity of each set of guidelines, the main aspects of usability are learnability, memorability, low ratio error, efficiency, and satisfaction. More recently, we can also point the new usability criteria based on persuasion developed by (Némery & Brangier, 2014). This means that HCI has been sharing a path with Cognitive theories that also focus on reducing the memory load from the user (Hollender, Hofmann, Deneke, & Schmitz, 2010). Also, both areas acknowledge the importance of the learner characteristics, showing that, to a certain extent, Cognitive concepts and HCI approaches have been integrated.

Bastien and Scapin recognized the limitations and difficulties in evaluating an interface, using any HCI approach individually. So they proposed a set of EC for interface evaluation (J. M. C. Bastien & Scapin, 1993) that intends to be a compilation of the several existing guidelines, making their criteria very specific and oriented to avoid misinterpretations. The term Ergonomic Criteria shows that the authors were interested in the adequacy of the criteria to the users' characteristics, behaviors, and needs. Bastien and Scapin also developed a consistent model of testing these criteria to validate their adequacy, since the existing guidelines and heuristics from other authors hadn't applied them. These criteria have also been developed for use in different types of interfaces, like virtual reality environments (J. M. C. C. Bastien & Scapin, 1995).

As the purpose of IM-sgi is to gather the correct criteria for a very specific type of interface (SG Implementation Interface), Bastien and Scapin's EC was the basis for defining IM-sgi Criteria, together with the knowledge gained with a CW made with existing SG implementations and analysis according to the ISO standards.

Bastien and Scapin produced their EC to create mechanisms that would allow usability to be evaluated based on the interface analysis instead of user testing. This then could be used directly by interface designers and give explicit measurement mechanisms. The EC are organized into eight sections that are divided into 18 EC groups, as follows:

1. Guidance (Elementary criteria are in boldfaced type and are followed by their abbreviations)

    1.1 **Prompting (PROM)**

    1.2 Grouping and distinction of items

        1.2.1 **Grouping and distinction of items by location (GDLO)**

        1.2.2 **Grouping and distinction of items by format (GDFO)**

    1.3 **Immediate feedback (FEED)**

    1.4 **Legibility (LEGI)**

2. User workload

    2.1 Brevity

        2.1.1 **Concision (CONC)**

        2.1.2 **Minimal actions (MIAC)**

    2.2 Information density (INDE)

3. User explicit control

    3.1 **Explicit user actions (EXUA)**

    3.2 **User control (USCO)**

4. Adaptability

    4.1 **Flexibility (FLEX)**

    4.2 **Users' experience management (USEX)**

5. Error Management

    5.1 **Error protection (ERPR)**

    5.2 **Quality of error messages (QUEM)**

    5.3 **Error correction (ERCO)**

6. **Consistency (CONS)**

7. **Significance of codes (SICO)**

8. **Compatibility (COMP)**

By converting these evaluation criteria to define successful interfaces for SG implementations, IM-sgi demonstrates how each criteria should be addressed for a specific SG task so that the steps needed are successfully communicated by the interface. So, instead of general guidelines, IM-sgi clearly defines how the interface should communicate with the user in each task to be performed while working with SG.

Bastien and Scapin defined 18 EC that are organized in eight main sections, showing a level of complexity and knowledge we considered needed and adjusted to the definition of our model. For the definition of IM-sgi, Bastien, and Scapin's criteria have been interpreted as guidelines and transformed into IM-sgi definition criteria. IM-sgi Criteria, although based on Bastien and Scapin's Ergonomic Criteria, are presented in a different order.

IM-sgi Criteria create specific definitions for how the interface should look to guide the user through the main tasks to be performed while working with SG. These five tasks were defined for and subsequently validated by the CW of Chapter 3.1.3. In the previous analysis, the main tasks the users perform were defined according to a hierarchy of tasks that the user takes to create and use an SG.

In summary, each IM-sgi Criteria is divided into six groups of guidelines, one group of general guidelines and one group of guidelines related to each of the above tasks.

Resuming the users and interactions modes defined in the previous chapters, the three types of users and interaction modes that the interface must consider are:

Students – GUI mainly with drawing tools available and with immediate visualization of any action made. This interface should be prepared for reduced control of the user over the implementation, which gives SG results in a more automated way. This is considered a Beginner Mode.

Artists / Designers – GUI with drawing, editing, and exporting tools that follow conventional CAD software, allowing the user to have good control of the implementation. They can then use it with possible integration with other CAD software so that SG can be defined to solve design projects, and its results used for further design project development/detailing. The user control should be higher, and the degree of automation should be less than in the Beginner Mode, as this is an Intermediate Mode.

Experts in SG – GUI combined with programming tools that allow the user to have full control over the implementation. Command-lines can be used to help the user clearly define the Rules of SG to use SG for specific problem solving (Tching et al., 2013). They can also change the application code so that it can create SG in new or different ways. The user shall have full control over the application with a reduced degree of automation, as this is an Expert Mode.

Following the above descriptions of Bastien and Scapin's EC, and then target users and task types, IM-sgi is presented here with the description of the original EC and the adaptation for IM-sgi Criteria. It is important to notice that IM-sgi Criteria were defined so that they can be applied to any type of SG implementation, although not addressing directly all the features that the implementation shall have.

The idealized general SG implementation characterization made by (Mckay et al., 2012) points to 7 requirements: (1) Shape; (2) Orientation of Shapes; (3) Semantics; (4) Definition Interface; (5) Usability by designers / intuitive user interface; (6) Automatic Subshape detection; and (7) Clear interpretation of resulting designs. Our model focuses on the Definition Interface and Usability by Designers with guidelines for these specific requirements.

The criteria in IM-sgi can address different types of generation processes that can be applied to the several actions that the interface must allow for a specific SG implementation. Taking the generation process presented in (Tomas Trescak et al., 2012), several input parameters can be introduced to manipulate an SG, for instance, choosing the number of iterations, applying step-by-step iterations, and generating a list of possible next shapes to be chosen and the use of markers. For expert users, our defined criteria can provide different ways of defining rules, either by graphic shape rules or mathematical rule schemata (Grasl & Economou, 2013).

In summary, we defined IM-sgi with a total of 259 criteria, divided according to the 18 EC that Bastien and Scapin (1993) developed  and organized as follows:

- Compatibility - 17 Criteria
- Adaptability -User's Experience Management - 18 Criteria
- Adaptability -Flexibility - 15 Criteria

- Significance of Codes - 14 Criteria
- Consistency - 7 Criteria
- Guidance-Immediate Feedback - 21 Criteria
- Guidance-Grouping and Distinction of Items by Location - 17 Criteria
- Guidance-Grouping and Distinction of Items by Format - 12 Criteria
- Guidance-Prompting - 18 Criteria
- Guidance-Legibility - 15 Criteria
- User Workload - Brevity - Concision - 12 Criteria
- User Workload - Brevity - Minimal Actions - 18 Criteria
- User Workload - Information Density - 13 Criteria
- Explicit User Actions - 13 Criteria
- User Control - 15 Criteria
- Error Protection - 14 Criteria
- Quality of Error Messages - 9 Criteria
- Error Correction - 11 Criteria

These 18 EC groups provide precise guidelines for interface creation, explicitly defining how the interface communicates with the user through the tasks he/she performs to achieve good usability. Every EC leads to the definition of six groups of criteria, one group with general guidelines and five groups according to the tasks previously defined. All criteria focus on assuring that the interface clearly guides the user on the actions he/she should perform to do the tasks and also guides him through the correct order of steps to successfully create and manipulate SG.

Next, we present an Infographic to illustrate the structure of IM-sgi to help visualize its organization. On Appendix B, we present the criteria on an organized table.

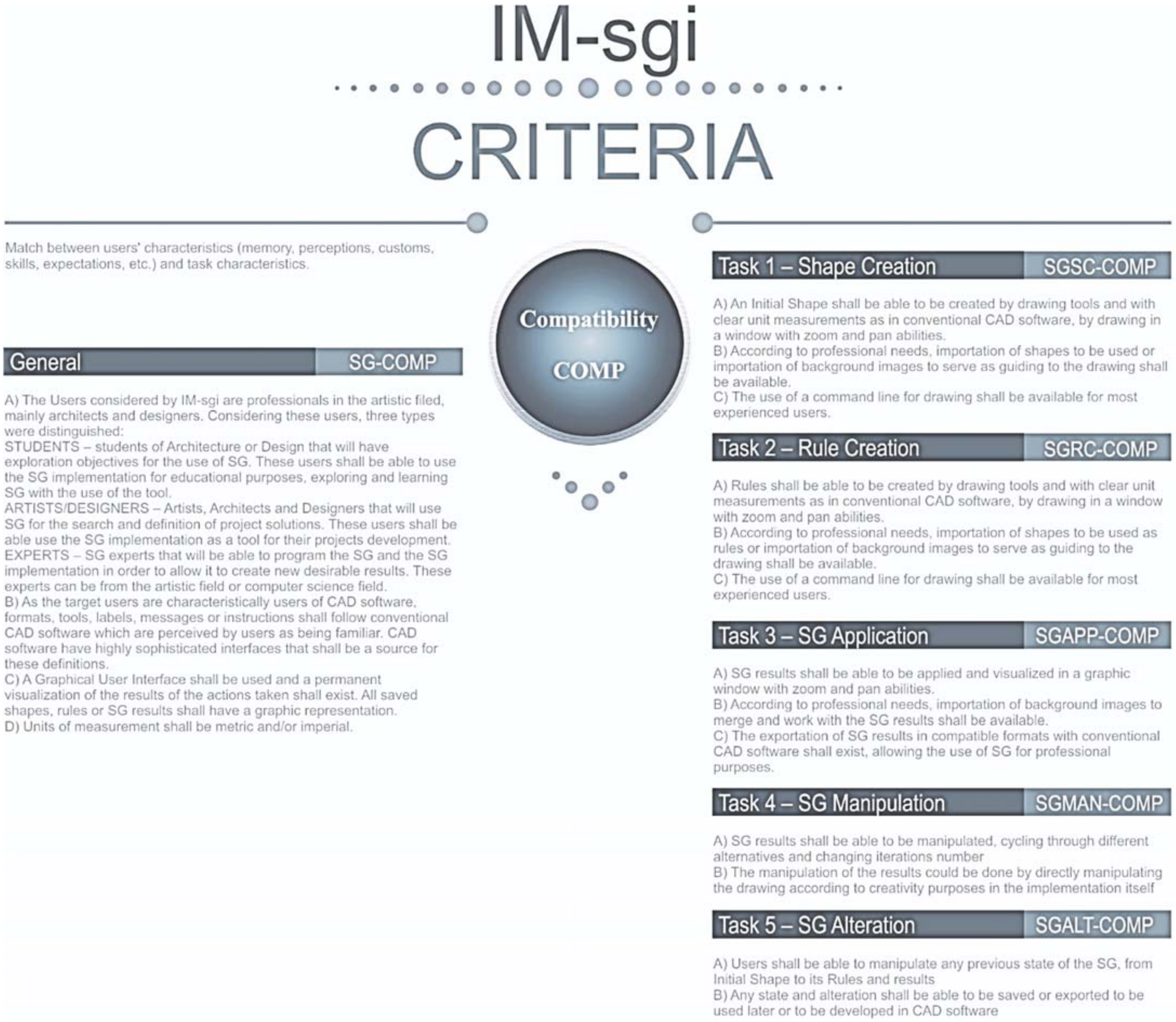
In total, the 18 groups are presented, each one referring to sets of criteria for general and specific tasks. Each Criteria Group has a name and an abbreviation.

Having five types of tasks defined before to work with SG, these are also presented with abbreviations. This means each group of criteria has six subgroups: (1) subgroup for General criteria; and (2) subgroups for tasks 1 to 5, according to Table 7.

| **TASK NUMBER** | **DESCRIPTION** | **ABBREVIATION** |
|---|---|---|
| Task 1 | Shape Creation – initial shape creation | SGSC |
| Task 2 | Rule Creation – rules definition for the creation of the Shape Grammar | SGRC |
| Task 3 | SG Application – application of an iterations' number for Shape Grammar use | SGAPP |
| Task 4 | SG Manipulation – manipulation of the Shape Grammar solutions | SGMAN |
| Task 5 | SG Alteration – alteration of the components of the Shape Grammar, initial shape or rules, to adjust it to the needs of the user | SGALT |

Table 7 - Tasks to be performed by Shape Grammar Implementations

The infographics show on the middle a circle with the name of the Criteria Group, along with its abbreviation; on the left side are shown the criteria applicable in General, and on the right side are presented the criteria applicable to each one of the five Tasks.



# IM-sgi

# CRITERIA

Compatibility

COMP

Match between users' characteristics (memory, perceptions, customs, skills, expectations, etc.) and task characteristics.

## General — SG-COMP

A) The Users considered by IM-sgi are professionals in the artistic filed, mainly architects and designers. Considering these users, three types were distinguished:
STUDENTS – students of Architecture or Design that will have exploration objectives for the use of SG. These users shall be able to use the SG implementation for educational purposes, exploring and learning SG with the use of the tool.
ARTISTS/DESIGNERS – Artists, Architects and Designers that will use SG for the search and definition of project solutions. These users shall be able use the SG implementation as a tool for their projects development.
EXPERTS – SG experts that will be able to program the SG and the SG implementation in order to allow it to create new desirable results. These experts can be from the artistic field or computer science field.
B) As the target users are characteristically users of CAD software, formats, tools, labels, messages or instructions shall follow conventional CAD software which are perceived by users as being familiar. CAD software have highly sophisticated interfaces that shall be a source for these definitions.
C) A Graphical User Interface shall be used and a permanent visualization of the results of the actions taken shall exist. All saved shapes, rules or SG results shall have a graphic representation.
D) Units of measurement shall be metric and/or imperial.

## Task 1 – Shape Creation — SGSC-COMP

A) An Initial Shape shall be able to be created by drawing tools and with clear unit measurements as in conventional CAD software, by drawing in a window with zoom and pan abilities.
B) According to professional needs, importation of shapes to be used or importation of background images to serve as guiding to the drawing shall be available.
C) The use of a command line for drawing shall be available for most experienced users.

## Task 2 – Rule Creation — SGRC-COMP

A) Rules shall be able to be created by drawing tools and with clear unit measurements as in conventional CAD software, by drawing in a window with zoom and pan abilities.
B) According to professional needs, importation of shapes to be used as rules or importation of background images to serve as guiding to the drawing shall be available.
C) The use of a command line for drawing shall be available for most experienced users.

## Task 3 – SG Application — SGAPP-COMP

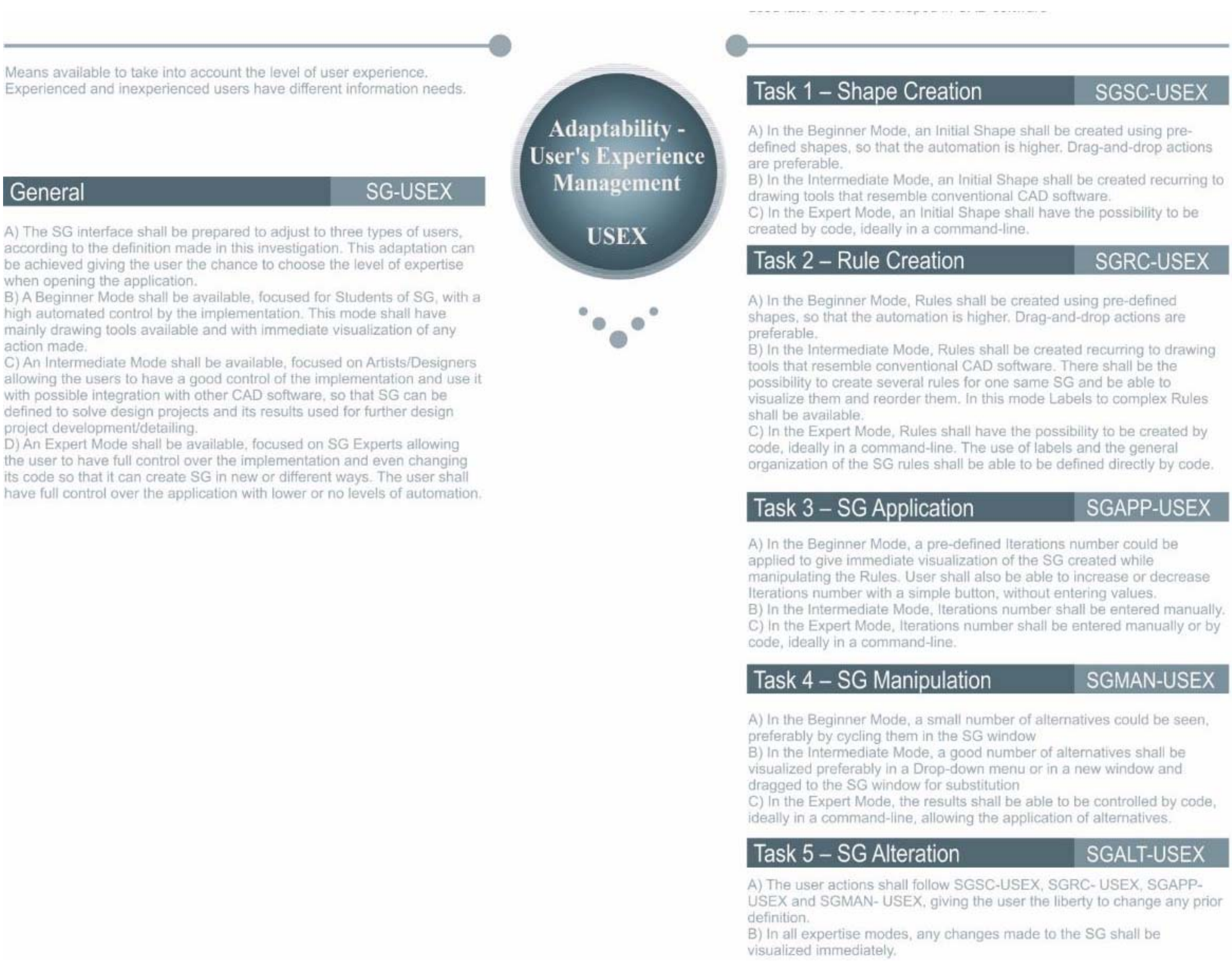
A) SG results shall be able to be applied and visualized in a graphic window with zoom and pan abilities.
B) According to professional needs, importation of background images to merge and work with the SG results shall be available.
C) The exportation of SG results in compatible formats with conventional CAD software shall exist, allowing the use of SG for professional purposes.

## Task 4 – SG Manipulation — SGMAN-COMP

A) SG results shall be able to be manipulated, cycling through different alternatives and changing iterations number
B) The manipulation of the results could be done by directly manipulating the drawing according to creativity purposes in the implementation itself

## Task 5 – SG Alteration — SGALT-COMP

A) Users shall be able to manipulate any previous state of the SG, from Initial Shape to its Rules and results
B) Any state and alteration shall be able to be saved or exported to be used later or to be developed in CAD software



Means available to take into account the level of user experience. Experienced and inexperienced users have different information needs.

## Adaptability - User's Experience Management

## USEX

### General — SG-USEX

A) The SG interface shall be prepared to adjust to three types of users, according to the definition made in this investigation. This adaptation can be achieved giving the user the chance to choose the level of expertise when opening the application.
B) A Beginner Mode shall be available, focused for Students of SG, with a high automated control by the implementation. This mode shall have mainly drawing tools available and with immediate visualization of any action made.
C) An Intermediate Mode shall be available, focused on Artists/Designers allowing the users to have a good control of the implementation and use it with possible integration with other CAD software, so that SG can be defined to solve design projects and its results used for further design project development/detailing.
D) An Expert Mode shall be available, focused on SG Experts allowing the user to have full control over the implementation and even changing its code so that it can create SG in new or different ways. The user shall have full control over the application with lower or no levels of automation.

### Task 1 – Shape Creation — SGSC-USEX

A) In the Beginner Mode, an Initial Shape shall be created using pre-defined shapes, so that the automation is higher. Drag-and-drop actions are preferable.
B) In the Intermediate Mode, an Initial Shape shall be created recurring to drawing tools that resemble conventional CAD software.
C) In the Expert Mode, an Initial Shape shall have the possibility to be created by code, ideally in a command-line.

### Task 2 – Rule Creation — SGRC-USEX

A) In the Beginner Mode, Rules shall be created using pre-defined shapes, so that the automation is higher. Drag-and-drop actions are preferable.
B) In the Intermediate Mode, Rules shall be created recurring to drawing tools that resemble conventional CAD software. There shall be the possibility to create several rules for one same SG and be able to visualize them and reorder them. In this mode Labels to complex Rules shall be available.
C) In the Expert Mode, Rules shall have the possibility to be created by code, ideally in a command-line. The use of labels and the general organization of the SG rules shall be able to be defined directly by code.

### Task 3 – SG Application — SGAPP-USEX

A) In the Beginner Mode, a pre-defined Iterations number could be applied to give immediate visualization of the SG created while manipulating the Rules. User shall also be able to increase or decrease Iterations number with a simple button, without entering values.
B) In the Intermediate Mode, Iterations number shall be entered manually.
C) In the Expert Mode, Iterations number shall be entered manually or by code, ideally in a command-line.

### Task 4 – SG Manipulation — SGMAN-USEX

A) In the Beginner Mode, a small number of alternatives could be seen, preferably by cycling them in the SG window
B) In the Intermediate Mode, a good number of alternatives shall be visualized preferably in a Drop-down menu or in a new window and dragged to the SG window for substitution
C) In the Expert Mode, the results shall be able to be controlled by code, ideally in a command-line, allowing the application of alternatives.

### Task 5 – SG Alteration — SGALT-USEX

A) The user actions shall follow SGSC-USEX, SGRC- USEX, SGAPP-USEX and SGMAN- USEX, giving the user the liberty to change any prior definition.
B) In all expertise modes, any changes made to the SG shall be visualized immediately.

## Adaptability-Flexibility

## FLEX

Means available to the users to customize the interface in order to take into account their working strategies and/or their habits, and the task requirements.
Flexibility is reflected in the number of possible ways of achieving a given goal.

### General — SG-FLEX

A) SG implementation interface shall take into account conventional CAD software, as the target users are mainly from Architecture and Design fields, allowing the user to organize menus, tool boxes, windows or other to resemble the software they are familiar with.
B) Users shall be able to use Buttons, Menus or Commands, o perform the same action, using what fits best their habits
C) Users shall be able to choose windows size, accordingly to the drawing or visualization needs

### Task 1 – Shape Creation — SGSC-FLEX

A) Initial Shape window shall allow resizing, facilitating the drawing actions.
B) Drawing commands can be available using drawing tools, using pre-defined shapes selected or dragged to the window or by a command line.
C) In Expert Mode, user shall be able to define Rules prior to Initial Shape drawing instead of following the general sequence of tasks for SG creation.

### Task 2 – Rule Creation — SGRC-FLEX

A) Rules window can allow resizing, facilitating the drawing actions.
B) Drawing commands can be available using drawing tools, using pre-defined shapes selected or dragged to the window or by a command line.
C) Label tools shall be available, dragging symbols to the drawing or inserting them with button.
D) In Expert Mode, User shall be able to define Rules prior to Initial Shape drawing instead of following the general sequence of tasks for SG creation.

### Task 3 – SG Application — SGAPP-FLEX

A) SG window shall have a big presence in the screen and allow resizing.
B) User shall be able to choose the iterations number by adding the specific number or cycling through pre-defined numbers. The system can also have a pre-defined number applied automatically that can be changed.

### Task 4 – SG Manipulation — SGMAN-FLEX

A) The manipulation of alternatives shall be possible according to SGAPP- FLEX.
B) User shall be able to choose to see alternatives in a new window or cycle through them directly in the SG window, or Drop-down menu or even textually in a command-line.

### Task 5 – SG Alteration — SGALT-FLEX

A) The user choices shall follow SGSC-FLEX, SGRC- FLEX and SGAPP-FLEX.

## Significance of Codes

## SICO

Relationship between a term and/or a sign and its reference.
Codes and names are significant to the users when there is a strong semantic relationship between such codes and the items or actions they refer to.

### General — SG-SICO

A) All window titles shall be clearly related to the tasks and results that will be available
B) Any abbreviations shall be clear and limited to the strictly necessary
C) Menus and Button naming shall follow conventional CAD software and be clearly related to the actions
D) Codes to be used in command lines shall be meaningful and allow the clear definition of the actions

### Task 1 – Shape Creation — SGSC-SICO

A) The term "INITIAL SHAPE" shall be used in the title of the window for this purpose
B) This term shall be used for any message related to Initial Shape definition
C) Drawing tools shall have clear names and follow conventional CAD software

### Task 2 – Rule Creation — SGRC-SICO

A) The term "RULES" shall be used in the title of the window for this purpose
B) This term shall be used for any message related to Rule definition
C) Drawing tools available for Rule creation shall have the same naming as the ones for Initial Shape Drawing

### Task 3 – SG Application — SGAPP-SICO

A) The term "SHAPE GRAMMAR" shall be used in the title of the window for this purpose
B) This term shall be used for any message related to Shape Grammar application
C) The term "ITERATIONS" shall be used for the Data entry field for Iterations application

### Task 4 – SG Manipulation — SGMAN-SICO

A) A term related to "ALTERNATIVES" or "RESULTS" shall be used for menus or windows that allow the choice of Shape Grammar alternatives

### Task 5 – SG Alteration — SGALT-SICO

Not Applicable

## Guidance - Grouping and distinction of items by location

## GDLO

Relative positioning of items organizing them according to their class or distinction.
Users will detect the different items easily if they are correctly, also helping learning and remembering of items.

### General SG-GDLO

A) Three main drawing windows must exist, organized by order:
1 – INITIAL SHAPE or equivalent
2 – RULES or equivalent
3 – SHAPE GRAMMAR or equivalent
They shall be organized linearly, left to right or up to down, or combinations of these, according to reading conventions
B) Any extra windows must be organized according to above reading conventions, next to the related window.
C) The menu options to be used in each drawing window must be organized according to the object they apply and accordingly to the order of commands to be performed.

### Task 1 – Shape Creation SGSC-GDLO

A) Window for Initial Shape creation must be shown as logically the first to be used. All drawing tools for the Initial Shape drawing must be available in that window, in buttons or menus.
B) Any new windows or menus that relate to Initial Shape must be gathered with the main Initial Shape window
C) If a command line exists for the Initial Shape creation, it shall be located on the bottom of the Initial Shape window

### Task 2 – Rule Creation SGRC-GDLO

A) Window for Rule creation must be shown as logically the second one to be used. All the tools for the Rule definition must be available in that window, in buttons or menus.
B) A Window shall exist to show the list of existing rules, next to the Rule main window. Exact same logic if instead of a window for this purpose a Drop-Down menu is opened.
C) Any new windows or menus that relate to Rule creation must be gathered with the main Rule Window
D) If a command line exists for the Rule creation, it shall be located on the bottom of the Initial Shape window

### Task 3 – SG Application SGAPP-GDLO

A) Window for SG application must be shown as logically the third one to be used. All the tools for the SG application must be available in that window, in buttons or menus.
B) The button or data field to add the Iterations number, to apply the SG, must be clearly situated in the SG window
C) Any new windows or menus that relate to SG application must be gathered with the main SG window

### Task 4 – SG Manipulation SGMAN-GDLO

A) SG manipulation made by changing de Iterations number, must follow SGAPP- GDLO- B
B) A window or drop-down menu shall exist to show SG alternatives to be selected. This must be gathered with the main SG window
C) Tools related to save or export results shall be clearly located near the SG window

### Task 5 – SG Alteration SGALT-GDLO

A) SG alteration made by changing de Initial Shape, shall be clear by the application of SGSC- GDLO
B) SG alteration made by changing Rules, shall be clear by the application of SGRC- GDLO

## Guidance - Grouping and distinction of items by format

## GDFO

Graphical features (format, color, etc.) that distinguish items of a given class.
It will be easier for the user to recognize relationships or distinctions between items by their similarities or differences.

### General SG-GDFO

A) Drawing windows for shapes and rule creation shall be graphically similar
B) Drawing tools shall be the same in Shape and Rule window, so they can be easily recognized
C) SG windows shall be graphically identifiable as the main results and manipulation window
D) Secondary windows for Rules list or SG alternatives shall be graphically similar to be identifiable as secondary and have the same position relatively to the main window they refer to. If they are shown by a Drop Down menu, they shall be also similar graphically and in position in the window.

### Task 1 – Shape Creation SGSC-GDFO

A) All drawing tools for shape creation in the Initial Shape window shall be similar, either in a List Menu or in Buttons. These shall also be similar to conventional drawing tools CAD software.
B) If a command line exists for shape definition, it shall be graphically identifiable as this kind of tool

### Task 2 – Rule Creation SGRC-GDFO

A) All drawing tools for shape creation in the Rule window shall be similar, either in a List Menu or in Buttons. These shall also be similar to conventional drawing tools in CAD software.
B) If a command line exists for rule definition, it shall be graphically identifiable as this kind of tool
C) A secondary window for the list of Rules that compose the grammar shall be graphically identifiable as this kind of element. The same way if it is shown in a Drop-down menu.

### Task 3 – SG Application SGAPP-GDFO

A) The button or data field to add the Iterations number shall be graphically relevant and predominant in the SG window

### Task 4 – SG Manipulation SGMAN-GDFO

A) A secondary window for a list of SG alternatives shall be graphically identifiable as this kind of element. The same way if it is shown in a Drop-down menu.

### Task 5 – SG Alteration SGALT-GDFO

A) All windows, menus and buttons shall be graphically identifiable according to SGSC- GDFO and SGRC- GDFO so that it is easy to identify the procedures to change Initial Shape or Rules of the SG.

## Guidance - Grouping and distinction of items by location

## GDLO

Relative positioning of items organizing them according to their class or distinction.
Users will detect the different items easily if they are correctly, also helping learning and remembering of items.

### General — SG-GDLO

A) Three main drawing windows must exist, organized by order:
1 – INITIAL SHAPE or equivalent
2 – RULES or equivalent
3 – SHAPE GRAMMAR or equivalent
They shall be organized linearly, left to right or up to down, or combinations of these, according to reading conventions
B) Any extra windows must be organized according to above reading conventions, next to the related window.
C) The menu options to be used in each drawing window must be organized according to the object they apply and accordingly to the order of commands to be performed.

### Task 1 – Shape Creation — SGSC-GDLO

A) Window for Initial Shape creation must be shown as logically the first to be used. All drawing tools for the Initial Shape drawing must be available in that window, in buttons or menus.
B) Any new windows or menus that relate to Initial Shape must be gathered with the main Initial Shape window
C) If a command line exists for the Initial Shape creation, it shall be located on the bottom of the Initial Shape window

### Task 2 – Rule Creation — SGRC-GDLO

A) Window for Rule creation must be shown as logically the second one to be used. All the tools for the Rule definition must be available in that window, in buttons or menus.
B) A Window shall exist to show the list of existing rules, next to the Rule main window. Exact same logic if instead of a window for this purpose a Drop-Down menu is opened.
C) Any new windows or menus that relate to Rule creation must be gathered with the main Rule Window
D) If a command line exists for the Rule creation, it shall be located on the bottom of the Initial Shape window

### Task 3 – SG Application — SGAPP-GDLO

A) Window for SG application must be shown as logically the third one to be used. All the tools for the SG application must be available in that window, in buttons or menus.
B) The button or data field to add the Iterations number, to apply the SG, must be clearly situated in the SG window
C) Any new windows or menus that relate to SG application must be gathered with the main SG window

### Task 4 – SG Manipulation — SGMAN-GDLO

A) SG manipulation made by changing de Iterations number, must follow SGAPP- GDLO- B
B) A window or drop-down menu shall exist to show SG alternatives to be selected. This must be gathered with the main SG window
C) Tools related to save or export results shall be clearly located near the SG window

### Task 5 – SG Alteration — SGALT-GDLO

A) SG alteration made by changing de Initial Shape, shall be clear by the application of SGSC- GDLO
B) SG alteration made by changing Rules, shall be clear by the application of SGRC- GDLO

## Guidance - Grouping and distinction of items by format

## GDFO

Graphical features (format, color, etc.) that distinguish items of a given class.
It will be easier for the user to recognize relationships or distinctions between items by their similarities or differences.

### General — SG-GDFO

A) Drawing windows for shapes and rule creation shall be graphically similar
B) Drawing tools shall be the same in Shape and Rule window, so they can be easily recognized
C) SG windows shall be graphically identifiable as the main results and manipulation window
D) Secondary windows for Rules list or SG alternatives shall be graphically similar to be identifiable as secondary and have the same position relatively to the main window they refer to. If they are shown by a Drop Down menu, they shall be also similar graphically and in position in the window.

### Task 1 – Shape Creation — SGSC-GDFO

A) All drawing tools for shape creation in the Initial Shape window shall be similar, either in a List Menu or in Buttons. These shall also be similar to conventional drawing tools CAD software.
B) If a command line exists for shape definition, it shall be graphically identifiable as this kind of tool

### Task 2 – Rule Creation — SGRC-GDFO

A) All drawing tools for shape creation in the Rule window shall be similar, either in a List Menu or in Buttons. These shall also be similar to conventional drawing tools in CAD software.
B) If a command line exists for rule definition, it shall be graphically identifiable as this kind of tool
C) A secondary window for the list of Rules that compose the grammar shall be graphically identifiable as this kind of element. The same way if it is shown in a Drop-down menu.

### Task 3 – SG Application — SGAPP-GDFO

A) The button or data field to add the Iterations number shall be graphically relevant and predominant in the SG window

### Task 4 – SG Manipulation — SGMAN-GDFO

A) A secondary window for a list of SG alternatives shall be graphically identifiable as this kind of element. The same way if it is shown in a Drop-down menu.

### Task 5 – SG Alteration — SGALT-GDFO

A) All windows, menus and buttons shall be graphically identifiable according to SGSC- GDFO and SGRC- GDFO so that it is easy to identify the procedures to change Initial Shape or Rules of the SG.

## Guidance - Prompting

## PROM

Guide users through the alternatives when several actions are possible.
Inform about the actual state or context of the system.
Saves users from learning many commands and helps navigate the system easily.

### General SG-PROM

A) All Windows clearly named, numbered according to order of use, if needed
B) For data field entries, display associated label
C) Display measurement units when drawing tools are used
D) Provide drawing status information (shape measurements, shape color, thickness, filling, shape geometry (opened or closed))
E)Provide help tools

### Task 1 – Shape Creation SGSC-PROM

A) Window for Initial Shape Creation name:
1 – INITIAL SHAPE or equivalent
B) Display only available actions and in order of use: drawing tools for shape drawing, editing and saving.
C) If drawing is made by data entry, provide required formats and accepted values.

### Task 2 – Rule Creation SGRC-PROM

A) Window for Rule Creation name:
2 – RULES or equivalent
B) Display only available actions: drawing tools for shape drawing, editing and saving.
C) If drawing is made by data entry, provide required formats and accepted values
D) Provide Drop-down Menu, List or Window with saved rules that will compose the SG

### Task 3 – SG Application SGAPP-PROM

A) Window for SG Application name:
3 – SHAPE GRAMMAR or equivalent
B) Display only available actions: number of iterations to be used, editing and saving.
C) Provide the required format and acceptable values for the iterations number to apply
D) Provide Drop-down Menu, List or Window with SG alternatives

### Task 4 – SG Manipulation SGMAN-PROM

A) Display only available actions: changing number of iterations, choose different alternatives, edit, save and export.

### Task 5 – SG Alteration SGALT-PROM

A) Display only available actions: add/change Initial Shape; add/change rule or rules order

## Guidance - Legibility

## LEGI

Lexical characteristics of the information presented on the screen that may hamper or facilitate the reading of this information.

### General SG-LEGI

A) Window names shall be all equally formatted and aligned, preferably with upper case letters and aligned on the left
B) Menus in all the windows shall be distributed with the same inter-word spacing and preferably with upper case letters
C) Tools and Menus shall follow conventional CAD software

### Task 1 – Shape Creation SGSC-LEGI

A) Drawing tools shall have clear icons if they are buttons and, preferably have an associated label
B) Drawing window shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa. A scalable grid can help understand shapes relations.
C) If a command line exists to enter shape definitions, the text shall have a Font without sheriff and with a good readable size

### Task 2 – Rule Creation SGRC-LEGI

A) Drawing tools shall have clear icons if they are buttons and, preferably have an associated label
B) Drawing window or Rules list window shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa. A scalable grid can help understand shapes relations.
C) If a command line exists to enter shape definitions, the text shall have a Font without sheriff and with a good readable size

### Task 3 – SG Application SGAPP-LEGI

A) SG window shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa. A scalable grid can help understand shapes relations.
B) If a command line exists to enter shape definitions, the text shall have a Font without sheriff and with a good readable size
C) Iterations Button or Data field shall have a velar Font without sheriff and with good readable size.

### Task 4 – SG Manipulation SGMAN-LEGI

A) If a SG alternatives' window exists, it shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa.
B) Iterations button shall be as SGAPP- LEGI- C.

### Task 5 – SG Alteration SGALT-LEGI

A) Tools and Menus needed to manipulate the SG shall be according to SGSC- LEGI, SGRC- LEGI and SGAPP- LEGI.

## User workload – Brevity - Concision

CONC

Perceptual and cognitive workload for individual inputs or outputs. The shorter the entries, the less memorization is needed.

### General SG-CONC

A) Possibly existing Data field shall allow exclusively accepted values
B) If a measurement unit is associated with a particular data field for drawing, include that unit as part of the field label
C) If codes are used in a command-line to activate tools, use short codes with no more than 5 characters
D) In iconic buttons, allow labels with the description to appear when hoovering the cursor.

### Task 1 – Shape Creation SGSC-CONC

A) Drawing tools for Initial Shape drawing shall be the only ones available directly on the Initial Shape window
B) If a command line is used, drawing commands shall be small and clear. Measurements shall automatically have the measurement unit, instead of making the user add them

### Task 2 – Rule Creation SGRC-CONC

A) Drawing tools for Rule drawing shall be the only ones available directly on the Rule window and be similar to the ones to create Initial Shapes
B) If a command line is used, commands shall be small and clear. Measurements shall automatically have the measurement unit, instead of making the user add them
C) The window or Drop-down menu with list of rules shall allow them to be selected or re-ordered by simple click and drag with the cursor

### Task 3 – SG Application SGAPP-CONC

A) Iterations Button or Data field shall accept exclusively numbers and allow only the length acceptable by the implementation

### Task 4 – SG Manipulation SGMAN-CONC

A) The window or Drop-down menu with alternatives shall allow them to be selected by simple click with the cursor

### Task 5 – SG Alteration SGALT-CONC

A) The tools to be used shall follow SGSC- CONC, SGRC- CONC SGAPP- CONC and SGMAN- CONC.

## User workload – Brevity - Minimal actions

MIAC

Workload with respect to the number of actions necessary to accomplish a goal or a task.
Limiting le the steps users must go through to reach a goal decreases the risks of making errors.

### General SG-MIAC

A) Reduce to the minimum necessary actions to create a SG
B) Reduce to the minimum the codes to be used in command-lines

### Task 1 – Shape Creation SGSC-MIAC

A) For the creation of the Initial Shape only drawing tools shall be needed and they shall be immediately available in the Rules window.
B) Manipulation of the shapes shall be done by click and drag with the cursor
C) If a command line exists for the creation of shapes, these shall be easily created by simple commands with dimensions' introduction.
D) Saving and editing tools shall be distinguished as secondary tools

### Task 2 – Rule Creation SGRC-MIAC

A) For the creation of Rules only drawing tools shall be needed and they shall be immediately available in the Rules window.
B) Manipulation of the shapes shall be done by click and drag with the cursor.
C) The Initial shape shall automatically appear in the Rules window to void repeating it's drawing
D) When a SG is made from a list of rules, allow the creation of a new rule starting from the drawing of the previous one
E) The insertion of Labels shall be done by single commands, preferably a button with the label to be used
F) If a command line exists for the creation of shapes, these shall be easily created by simple commands with dimensions' introduction.
G) Saving a rule to the list shall be done by a single command that will add it to the Rules' list

### Task 3 – SG Application SGAPP-MIAC

A) A simple button or data field shall be needed to indicate the Iterations number and apply the SG and no other command shall appear as relevant in the SG window
B) Saving a result shall be done by a single command.

### Task 4 – SG Manipulation SGMAN-MIAC

A) Manipulation of the SG shall be easily done by changing the Iterations number, accordingly to SGAPP- MIAC- A.
B) Choosing alternatives shall be done simply by click and drag with the cursor.

### Task 5 – SG Alteration SGALT-MIAC

A) The tools to be used shall follow SGSC- MIAC, SGRC- MIAC SGAPP- MIAC and SGMAN- MIAC.

## User control

## USCO

Users should always be in control of the system processing (interrupt, cancel, pause and continue).

### General SG-USCO

A) Allow users to pace their entry, rather than controlling the time for each action
B) Allow users to Save the current state of the system to resume it in another time
C) Provide the user with Undo and Redo tools
D) Provide a Cancel option to take the user to the initial state of the system or action

### Task 1 – Shape Creation SGSC-USCO

A) Allow the user to take its time to define the Initial Shape, allowing the drawing to be changed until the user considers it finished
B) Allow users to Save the Initial Shape to be used later

### Task 2 – Rule Creation SGRC-USCO

A) Allow the user to take its time to define Rules, allowing the drawing to be changed until the user considers it finished
B) Allow users to Save Rules to be used later

### Task 3 – SG Application SGAPP-USCO

A) Allow the user to see the Iterations result without timing out
B) Allow users to Save results to be used later

### Task 4 – SG Manipulation SGMAN-USCO

A) Allow the user to try several Iterations numbers until reaching a desirable result
B) Allow the user to try alternatives and select them for further work without timing out
C) Allow users to Save results to be used later

### Task 5 – SG Alteration SGALT-USCO

A) The tools to be used shall follow SGSC- USCO, SGRC- USCO SGAPP- USCO and SGMAN- USCO without timing out
B) Any new state shall be able to be Saved and used later.

## Error Protection

## ERPR

Means available to detect and prevent data entry errors, command errors, or actions with destructive consequences.
It is preferable to detect errors before validation rather than after.

### General SG-ERPR

A) Assure that the SG implementation will deal properly with possible user error, including accidental inputs
B) Use display messages to warn the user about incorrect inputs
C) Display advisory messages before closing the SG implementation if all elements are not correctly saved or if there is a pending action.
D) If the SG computation by the implementation is limited, user definitions that will create computational errors shall be prevented

### Task 1 – Shape Creation SGSC-ERPR

A) Assure user creates an Initial Shape as first action, creating warnings if another tasks are attempted first
B) In the Expert Mode, where the user might start by Rules creation, a message shall appear to make the user confirm that action as first one.
C) If Initial Shapes are defined by code, there shall be prevention of data entry errors, accepting only valid codes and warning if incorrect ones were attempted to be used

### Task 2 – Rule Creation SGRC-ERPR

A) The SG implementation shall have the means to advise the user if the Rules attempted to be made have errors that will lead to a computation error of the SG
B) Unless the defined Rule is valid, it shall not be possible to save it and use it
C) If Rules are defined by code, there shall be prevention of data entry errors, accepting only valid codes and warning if incorrect ones were attempted to be used

### Task 3 – SG Application SGAPP-ERPR

A) The Iterations field shall only allow inputs of integer numbers and warn if incorrect data was attempted to be used
B) If a limited number of iterations is possible for a given SG, there shall be a warning and only valid numbers accepted

### Task 4 – SG Manipulation SGMAN-ERPR

A) The implementation shall allow the user to access alternatives to SG results without destroying the SG rules definition

### Task 5 – SG Alteration SGALT-ERPR

A) The error prevention of user actions to change the SG shall follow SGSC-ERPR, SGRC- ERPR, SGAPP- ERPR and SGMAN- ERPR.

## User Workload - Information density

## INDE

Users' workload from a perceptual and cognitive point of view with regard to the whole set of information presented to the users rather than each individual element or item.
Items that are not related to the task should be removed.

### General SG-INDE

A) The implementation display shall show exclusively the windows referred in SG-GLO- A.
B) Other windows shall be presented as secondary or opened only when needed
C) Command lines shall be integrated in the bottom of the corresponding window
D) Allow shapes, rules and results to be permanently visible, so the user doesn't have to memorize and give them proper emphasis to the rest of the environment.
E) Computation needed for the SG application shall be automatic and without any calculated entry done manually by the user

### Task 1 – Shape Creation SGSC-INDE

A) Initial Shape window shall be a simple drawing area with only drawing tools available.
B) A command line for shape creation can be shown only when needed, in the bottom of the Rules' window.

### Task 2 – Rule Creation SGRC-INDE

A) Rules window shall be a simple drawing area with drawing tools available and a save command to create list of rules for SGs with more than one rule.
B) A window with the graphical visualization of the rules can be opened when needed instead of being always visible
C) A command line for rule creation can be shown only when needed, in the bottom of the Rules' window.

### Task 3 – SG Application SGAPP-INDE

A) SG window shall be a simple drawing area with the iterations number tool as main command.

### Task 4 – SG Manipulation SGMAN-INDE

A) A window with the graphical visualization of alternatives can be opened when needed instead of being always visible

### Task 5 – SG Alteration SGALT-INDE

A) The tools to be used shall follow SGSC- INDE, SGRC- INDE SGAPP-INDE and SGMAN- INDE.

## Explicit user actions

## EXUA

Relationship between the computer processing and the actions of the users.
The computer must process only the actions requested by the users and only when requested to do so.

### General SG-EXUA

A) Request from the user a clear Enter action to initiate the SG process
B)Request from the user a clear click on Menu and Drop-down menu choices
C) Request from the user a clear Enter action when using command-line entries

### Task 1 – Shape Creation SGSC-EXUA

A) When defining the Initial Shape, request a Finish command to declare that the drawing is finished and that the next task can be started
B) If the definition is made in a command line, request an Enter command after each code used

### Task 2 – Rule Creation SGRC-EXUA

A) When defining Rules, request a Finish command to declare that the definition is finished, so that the rule can be added to the SG Rules' list
B) If the definition is made in a command line, request an Enter command after each code used
C) When all rules to be used by the SG are finished, request a Finish command to declare that next task can be started

### Task 3 – SG Application SGAPP-EXUA

A) When choosing the number of Iterations to apply in the SG, request an Enter command for the action to be applied
B) To Save or Export a result, request an Enter command

### Task 4 – SG Manipulation SGMAN-EXUA

A) To change the Iterations number to apply to the SG, request an Enter command for the action to be applied
B) To select an alternative, preferably with Drag-and-Drop from an Alternatives Window or selection from it, request a Click on the desired shape to be dragged or selected to the SG window.

### Task 5 – SG Alteration SGALT-EXUA

A) The tools to be used shall follow SGSC- EXUA, SGRC- EXUA SGAPP- EXUA and SGMAN- EXUA.

## Consistency

## CONS

The way interface design choices (codes, naming, formats, procedures, etc.) are maintained in similar contexts, and are different when applied to

### General — SG-CONS

A) Window titles and formats shall follow the same format and placement in all windows.
B) The screen format shall be identical in all expertise modes, adapting itself to the specifications of each one
C) Menu options, buttons and command-lines shall be identical in all different windows and expertise modes
D) All messages shall follow the same format and position on the screen

### Task 1 – Shape Creation — SGSC-CONS

A) Initial Shape window and tools shall be identical and recognizable in all expertise modes, independently of the additional tools in each one

### Task 2 – Rule Creation — SGRC-CONS

A) Rules window and tools shall be identical and recognizable in all expertise modes, independently of the additional tools in each one

### Task 3 – SG Application — SGAPP-CONS

A) SG window and tools shall be identical and recognizable in all expertise modes, independently of the additional tools in each one

### Task 4 – SG Manipulation — SGMAN-CONS

Not applicable

### Task 5 – SG Alteration — SGALT-CONS

Not applicable

## Guidance - Immediate feed-back

## FEED

System responses to users' actions.
Feedback quality and rapidity help the user have confidence and satisfaction as well as a good understanding of the dialogue.

### General — SG-FEED

A) SG implementation shall have a Graphical User Interface that allows immediate visualization of created shapes, created rules and SG results and alternatives
B) A graphical visualization of the created rules, either by drawing or data entry shall be immediately presented
C) SG Iterations shall be immediately graphically visualized, along with a number of alternatives
D) Undo and Redo tools shall be available and it's results immediately visualized
E) Messages guiding the user shall be delivered

### Task 1 – Shape Creation — SGSC-FEED

A) Initial Shape window shall allow the graphical design with drawing tools, similarly to conventional CAD software.
B) Drawing tools shall be clearly identified and it's results immediately visualized when used
C) Any change made to the shapes shall be easily reversible and all the results immediately visualized.
D) If a command-line exists to define shapes, the results shall be graphically seen in the Initial Shape window

### Task 2 – Rule Creation — SGRC-FEED

A) Rules window shall allow the graphical design with drawing tools, similarly to conventional CAD software.
B) Drawing tools shall be clearly identified and it's results immediately visualized when used
C) Any change made to the shapes and rules shall be easily reversible and all the results immediately visualized.
D) The system shall allow saving and ordering the rules that will compose the SG and allow the user to easily visualize them and manipulate them, preferably in a dedicated Window or Drop-down menu with graphical visualizations of the composed rules
E) If a command-line exists to define rules, the results shall be graphically seen in the Rules' window

### Task 3 – SG Application — SGAPP-FEED

A) The result of the number of Iterations applied shall be immediately seen in the SG window
B) The system shall graphically show alternatives of the SG with the same number of Iterations, preferably in a dedicated Window or Drop-down menu.

### Task 4 – SG Manipulation — SGMAN-FEED

A) In the Beginner Mode, a small number of alternatives could be seen, preferably by cycling them in the SG window
B) In the Intermediate Mode, a good number of alternatives shall be visualized preferably in a Drop-down menu or in a new window and dragged to the SG window for substitution
C) In the Expert Mode, the results shall be able to be controlled by code, ideally in a command-line, allowing the application of alternatives.

### Task 5 – SG Alteration — SGALT-FEED

A) If changes are made in the Initial Shape, the new results shall be immediately graphically visualized
B) If changes are made in the Rules, the new results shall be immediately graphically visualized
C) Any alteration shall be easily undone or redone and it's results visually seen

The next stage of this research used the criteria to define interfaces for SG implementations. As stated above, Appendix B presents all the information on the infographics above as tables, for a more technical presentation of the sets of criteria. Each table shows the Criteria Groups: Criteria subgroups and sets of criteria for General and five specific tasks. This description has been published in (Tching et al., 2018)

## 4.4 IM-SGI APPLICATION

As stated before, the importance and complexity of the user interface, the direct connection between user and application, is so great that today there is widespread use of toolkits (libraries of code written to support the creation of interfaces), and of interface builders (interactive tools consisting of buttons, menus, etc.). And some component architectures allow the creation of interfaces for connecting components written separately. Ultimately, a computer system is created for the function we want to perform, and the functionality of a system is defined by the set of tasks for its user (Karray et al., 2008). The value of the application is visible when the system is used efficiently by the user to meet his/her objectives.

The process for a good interface definition requires a significant amount of time, but time is often scarce when a new application is being developed. This might be one of the reasons why existing SG applications, which require such complex development of algorithms and programming time, still have rudimentary interfaces, as analyzed in our previous work (Tching et al., 2016). This acknowledgment is the basis of our IM-sgi proposal.

Following the Lewis and Rieman (1994) process of 9 stages listed on Chapter 1.3, IM-sgi definition of stages 1 to 4 have already been developed in the previous chapters of this thesis and stages 5 to 9 are developed in this chapter.

Stage 5 that consists of testing a defined interface was adapted to validate our model of interface. Having defined our interface model for SG Implementations, we used two different methods to test it. With these tests, we seek to get conclusions on how the IM-sgi Criteria can help, on one hand, to understand the usability issues of existing SG implementations' interfaces, and on the other hand, to understand if an interface prototype that follows the IM-sgi Criteria meets the purpose of being easily understood and used for SG creation and manipulation.

To achieve this goal, two main methods were used:

(1) A CW, a Usability Inspection Method which does not demand a UI specialist or tests with users (C Wharton et al., 1994) was used again, this time to analyze the SG implementations that were previously tested in our research, to understand if the IM-sgi criteria apply successfully. As none of these SG implementations has ever been truly adopted for creative projects, this analysis allows us to understand what criteria have been generally disregarded and those that might influence its success as a creative design tool.

(2) A Prototype of Interface was created. This prototype, following the three types of users of SG that were defined in our research (Tching et al., 2016), was developed so that Students in the fields of design and architecture could test it. A Workshop entitled "Shape Grammars in Architecture – Interfaces" took place at ISCTE-IUL, Lisbon-Portugal, on the 29th November 2016. Here, a small group of architecture students was introduced to SG, and our research, and had the chance to test our interface prototype.

We used the System Usability Scale (SUS)  advocated by Tullis and Stetson (Tullis & Stetson, 2004) as a very efficient and effective method of evaluating usability issues. Using this SUS questionnaire, a group of 6 students answered how they felt when performing the tasks to create an SG and how well they understood the proposed system. Although the sample was not significant, the questionnaires and debate during the Workshop led to a reformulation of the Prototype, which is also presented next.

In the next chapters, we present the application of these two methods enumerated.

## 4.5 Cognitive Walkthrough on Existing SG Implementations

In IM-sgi all the criteria focus on assuring that the interface guides the user clearly on the actions he/she should perform to do the tasks and also guides him/her through the correct order of steps to successfully create and manipulate SG. We also previously defined five tasks that needed to be performed to work with SG.

These tasks led the CW made at the beginning of our research to illustrate the usability issues in a group of existing SG implementations. These SG implementations are the same for the CW done to check the fulfillment of IM-sgi Criteria.

Following the same steps described in Chapter 3.1.3, we performed a CW to each of the SG implementations studied throughout this research. We were able to evaluate 15 of the 18 criteria categories of IM-sgi, as the ones related to error management could not be observed in any of the implementations.

We sought to validate our previously defined IM-sgi Criteria for existing SG implementations. From the six previously tested implementations, this CW focused only of five. The implementation called Spapper was not included, as it works as a plugin on a CAD application. This means analyzing its interface according to IM-sgi, runs out of the scope. Thus, we tested Shaper 2D; Subshape; SGI; SGDE, and SG2 using the general context and five specific tasks that IM-sgi addresses.

In the following figures, resumed analytics are presented summarizing the percentage of criteria for each of the IM-sgi criteria categories that were positively observed by each SG implementation. These analytics were obtained with the information gathered by performing a CW for each of the SG implementations.

The full information that allowed these analytics can be seen in Appendix C and D.

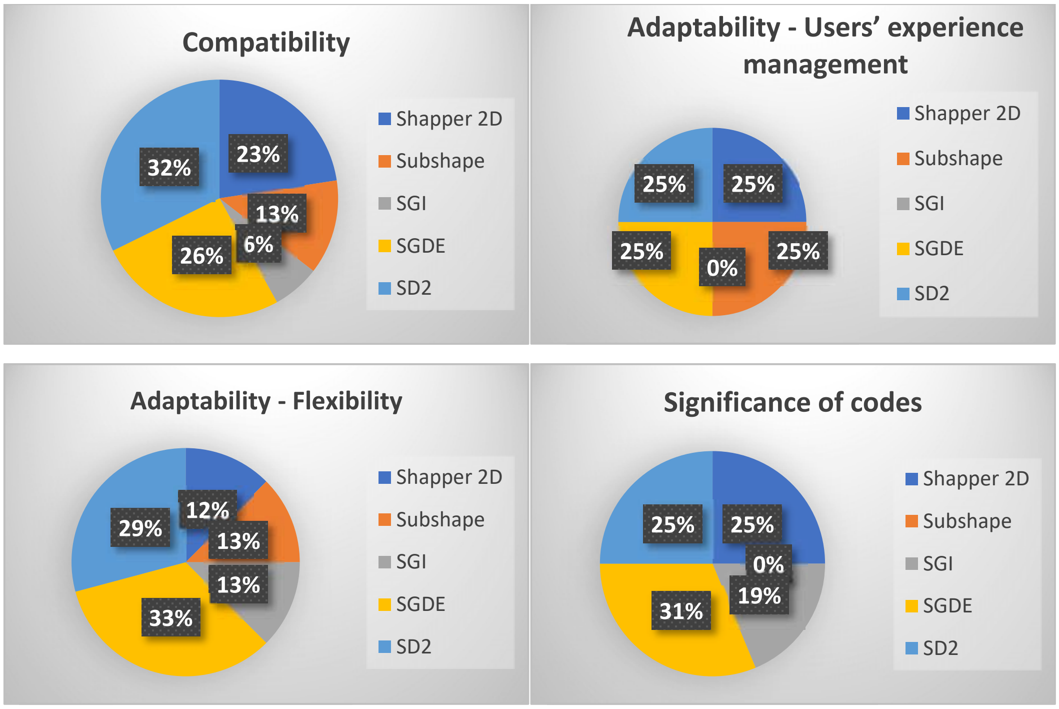


Figure 51 – Summary of the results of the CW, described in Appendix C and D (Part 1)

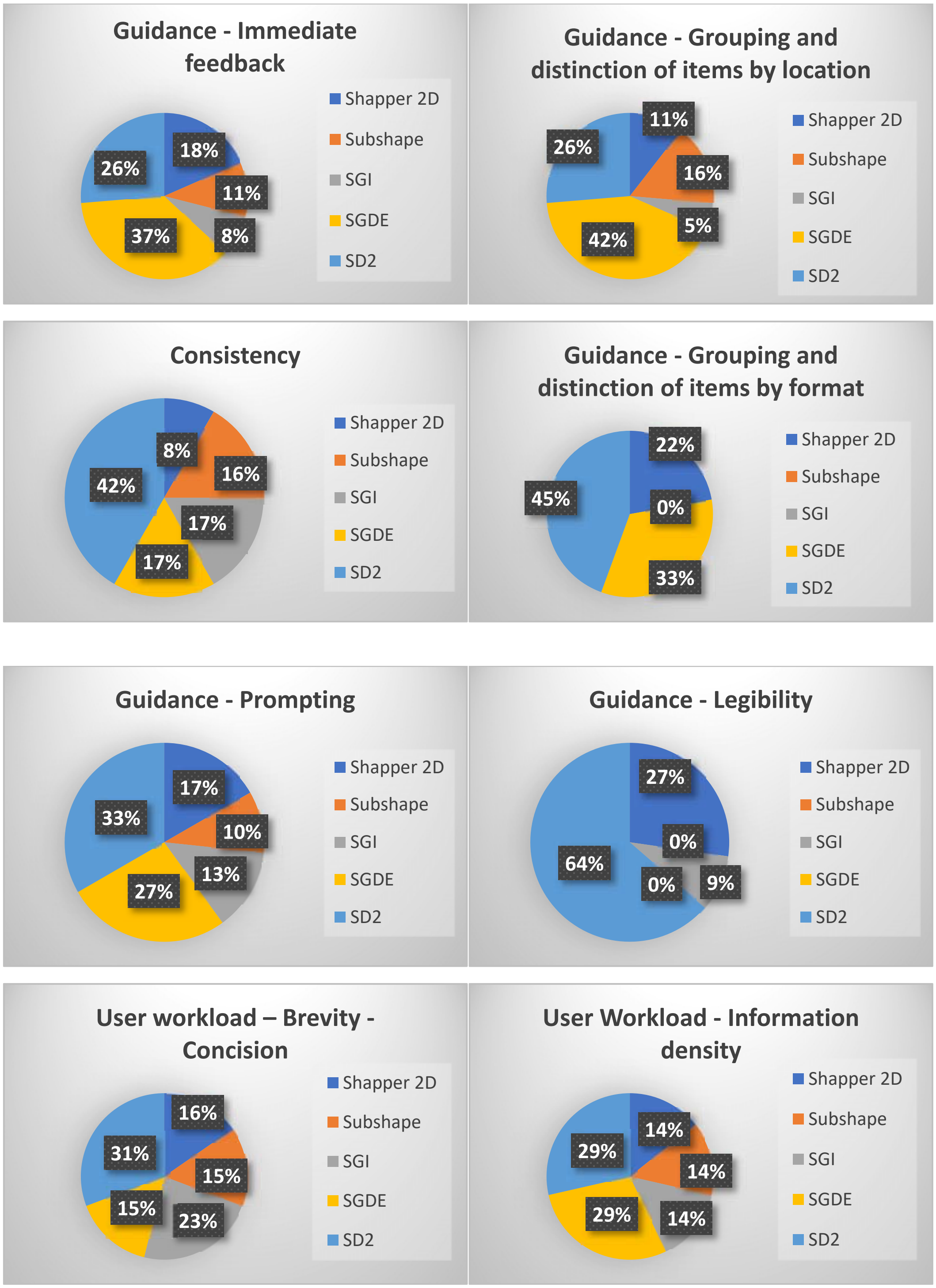


Figure 52 – Summary of the results of the CW, described in Appendix C and D (Part 2)

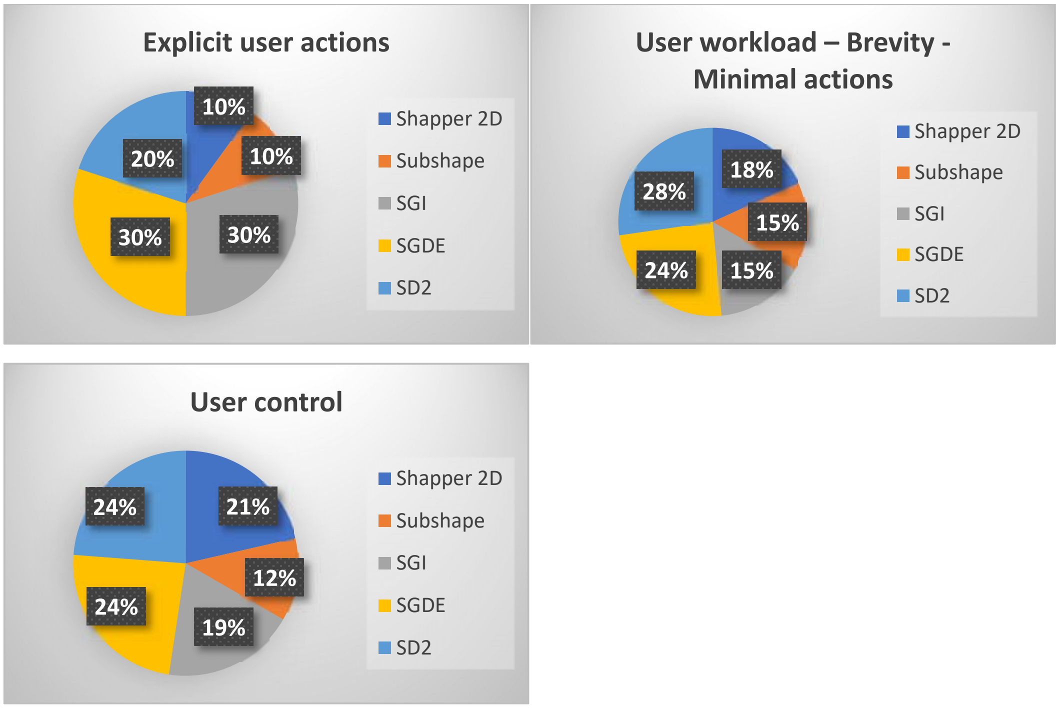


Figure 53 – Summary of the results of the CW, described in Appendix C and D (Part 3)

The above graphics reflect the CW results and all of the graphics show SD2 as the implementation that follows the higher number of IM-sgi criteria. This allows us to understand that the interface from SD2 is the one that has better communication with the user and can be considered as a starting point for prototyping new interfaces for SG computational implementations.

This CW allowed us to verify that more than half of the IM-sgi ergonomic criteria from 15 out of 18 categories. The categories related to Error Protection, Quality of Error Messages, and Error Correction were not tested, as none of the applications were developed on this subject.

Many usability issues can be observed that could justify the misfit of these tools for professional use, as our CW focused on the tasks the user would have to perform to generally work with SG. It is important to state that we ignored any criteria that were directly related to having different interfaces for different types of users, as defined by our research (Tching et al., 2016), as none of these applications take that in consideration.

With the CW performed by studying if the criteria we developed correctly addressed our assumptions, we then performed our second method to test our IM-sgi through prototyping.

## 4.6 PROTOTYPING VERSION 1

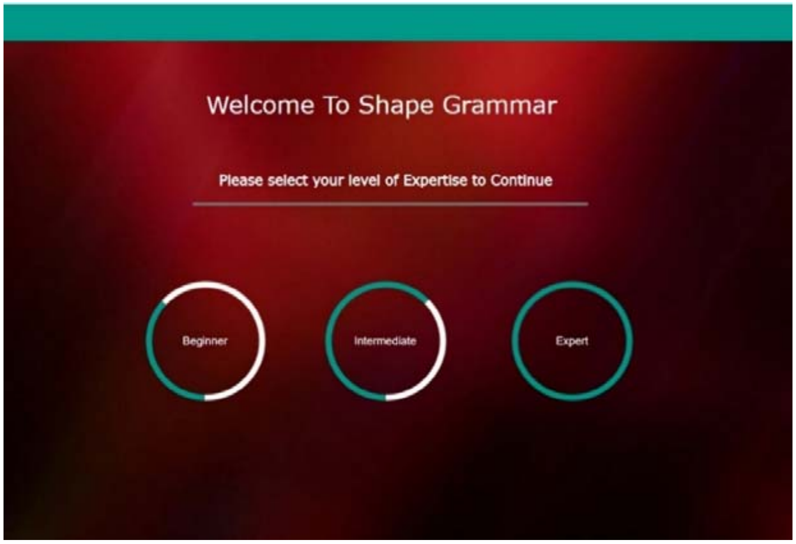
For the application of our IM-sgi, we created a Prototype using Axure, a software design tool for creating applications prototypes, specifications, and diagrams, allowing to plan and prototype without code. This tool allowed us to prototype an interface following the general criteria defined by IM-sgi, without coding the computational application behind it and the interface itself. Thus, this prototype was made with no SG implementation on the background and considering the three types of users defined in our research.

It is essential to state that it was a goal of our research to apply IM-sgi to a real SG implementation. Unfortunately, this was not possible as we could not cooperate with any author presently developing work on this field. It was also not possible to create an SG implementation ourselves as it involves an intricate work that we could not fit in the time limits of our research, or we could not dedicate to the primary purpose, which is the interface.

Although having the three modes for the three types of users, on our prototype, only the simpler mode can be tested. As a starting point, we believe that once the simpler mode is validated, the other ones shall work identically.

**SG Implementation Landing Page**

Our prototype was developed for the three types of users described above. A landing page, shown in Figure 54, is proposed to make the user choose the level of expertise that he/she believes suits his SG knowledge.


Welcome To Shape Grammar

Please select your level of Expertise to Continue

Figure 54 - SG Implementation Landing Page

This definition follows the level of expertise of the users, according to the IM-sgi Criteria SG-COMP and SG-USEX (please see Appendix B for the Abbreviations of the IM-sgi Criteria). This choice originates the loading of the interface that should communicate the user's characteristics.

We then developed the screens following IM-sgi criteria and the general visual look of CAD applications, including dark background for less eye exhaustion, as it happens on most advanced CAD software.

### SG Implementation Beginner Screen

The Beginner screen in Figure 55 shows a GUI according to IM-sgi Criteria SG-COMP, representing the users' characteristics. As the interface is directed mainly for users of design areas, a GUI interface that resembles existing CAD software is more easily understood and accepted by this group, as stated by the IM-sgi Criteria SG-FLEXFEED. Three drawing windows are shown according to IM-sgi Criteria SG-FLEX and SG-SICO.

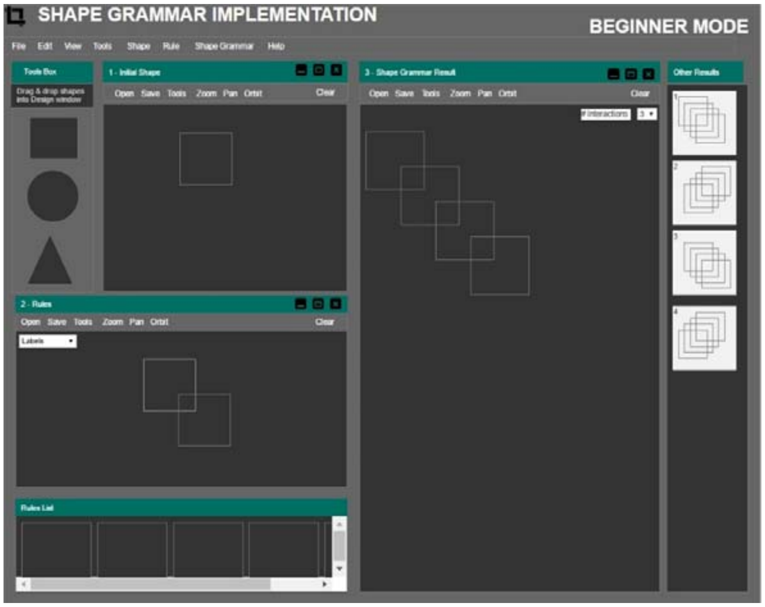

SHAPE GRAMMAR IMPLEMENTATION
BEGINNER MODE
File Edit View Tools Shape Rule Shape Grammar Help
Tools Box
Drag & drop shapes into Design window
1 - Initial Shape
Open Save Tools Zoom Pan Orbit Clear
3 - Shape Grammar Result
Open Save Tools Zoom Pan Orbit Clear
# interactions 3
Other Results
1
2
3
4
2 - Rules
Open Save Tools Zoom Pan Orbit Clear
Labels
Rules List

Figure 55 - *SG Implementation Beginner Screen*

The windows are numbered and named according to IM-sgi Criteria SICO, SG-GDLO, and SG-PROM.

The first task for the Beginner is to define the Initial Shape. In this level of expertise, a toolbox with shapes is shown so that the user may drag them to the Initial Shape window, according to SGSC-USEX-A and SGSC-GDLO-B. When the user manipulates a shape by drag and drop, the Rule window updates in real-time, showing the shape to be used as the basis for Rule definition. This function enables the users to get an immediate, real-time response to their actions. In the Beginner Mode, the use of messages indicating user steps is desirable. This mode may play the role of a tutorial for the use of the application.

The Rules window is the second one to be used, according to SGRC-USEX and SGRC-GDLO-A. A toolbox with a group of Rules is shown graphically so that Rules created are available for selection, from previously saved ones or predefined by the application, according to SGRC-GDLO-B and SGRC-GDLO-C

The use of the SG is made by selecting the iterations number. A pre-defined number can exist to make a solution appear immediately as the Rule is created, according to SGAPP-USEX. As the SG is applied, several alternatives are shown in the Other Results toolbar, which can then be selected to be worked on the SG window, according to SGALT-GDLO. Generally, all criteria that can be verified by a graphic mock-up were considered, except error management.

### SG Implementation Intermediate screen

The Intermediate mode is similar to the Beginner one, but it adds the functionality of free drawing, instead of shape selection. The toolbox with predefined shapes to be selected is eliminated, giving more space for general drawing in the Initial Shape window

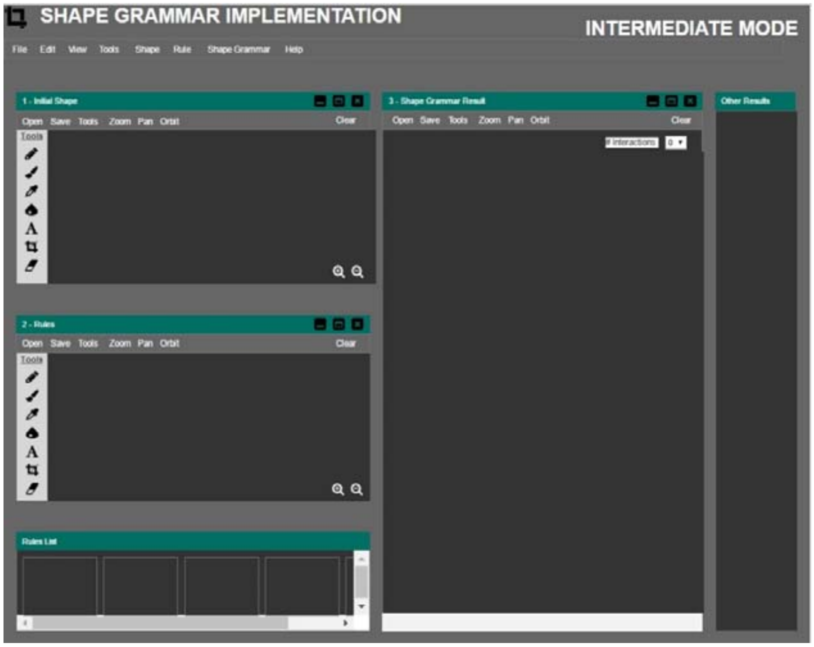

SHAPE GRAMMAR IMPLEMENTATION
INTERMEDIATE MODE
File Edit View Tools Shape Rule Shape Grammar Help
1 - Initial Shape
Open Save Tools Zoom Pan Orbit
Clear
Tools
3 - Shape Grammar Result
Open Save Tools Zoom Pan Orbit
Clear
# interactions
0
Other Results
2 - Rules
Open Save Tools Zoom Pan Orbit
Clear
Tools
Rules List

Figure 56 - SG Implementation Intermediate screen

### SG Implementation Expert Screen

According to SG-USEX, the Expert Mode shown in Figure 57 is for Designers/Artists who use conventional CAD software, as in SG-COMP. Following this, the general display and functioning are similar to the previous mode so that the user can naturally evolve and use the next expert mode without having to interpret the functioning of the application again, according to SG-CONS. In this Expert Mode, according to SGRC- USEX- B and SGRC-FLEX-C, a drop-down list exists in the Rules window for the use of Labels when creating the rules.

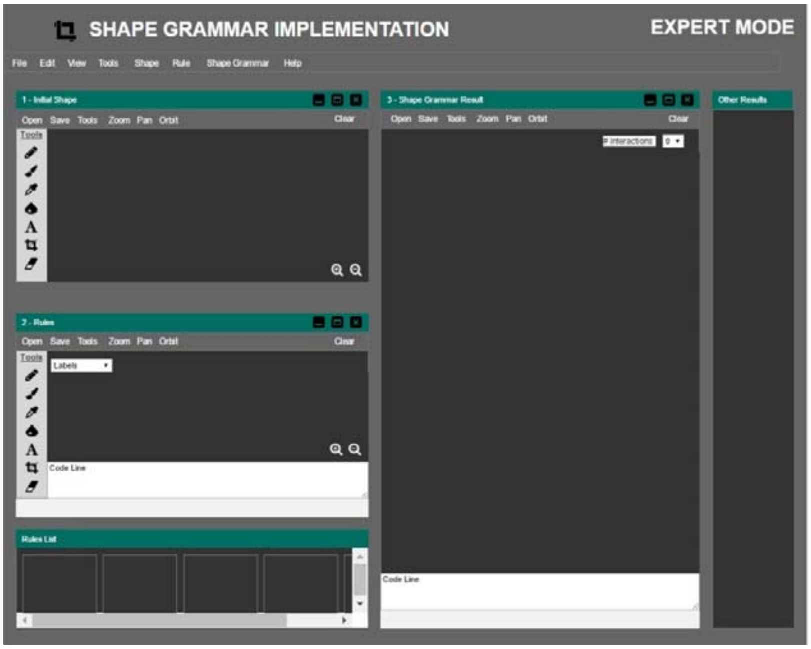

SHAPE GRAMMAR IMPLEMENTATION
EXPERT MODE
File Edit View Tools Shape Rule Shape Grammar Help
1 - Initial Shape
Open Save Tools Zoom Pan Orbit
Clear
Tools
2 - Rules
Open Save Tools Zoom Pan Orbit
Clear
Tools
Labels
Code Line
Rules List
3 - Shape Grammar Result
Open Save Tools Zoom Pan Orbit
Clear
# interactions
0
Code Line
Other Results

Figure 57 - SG Implementation Expert screen

The Expert screen allows the availability of command-lines to define Rules or change the SG through code. These command-lines could be as shown, directly in the window of the task to be performed, or they could appear in the bottom of the entire interface, usable for any task. In this mode, the user is an Expert in SG and/or in the development of the implementation itself, so expert tools are available, according to SG- USEX- D. Please refer to Appendix E for a complete description of the Axure's Prototype created.

## 4.7 TEST 1

After creating the prototype to test our interface model, the nest important step is to test it and validate if it fits the purpose of communicating correctly how to work with SG. Even being just a prototype of interface, without a functional SG implementation to work with, we decided to make a test with users to validate the IM-sgi criteria applied.

.To evaluate the presented Prototype, we offered a workshop for students to use it and then evaluate and assess their user experience. They were selected out of the three categories of users below that are addressed in our prototype. The Workshop was entitled "Shape Grammars in Architecture – Interfaces" and took place at ISCTE-IUL, Lisbon-Portugal, on the 29th November 2016.

The students filled out a questionnaire after using the interface. We used one of the most popular questionnaires, SUS, which is short and does seem to yield reliable results across sample sizes (Tullis & Stetson, 2004).

The SUS includes ten items for consideration using four-point responses (strongly disagree -- strongly agree) to each of these results, as shown in Table 7 below. In summary, except for half of the SUS respondents thinking that they would/would not use this system frequently, the response to usability questions was overwhelmingly positive, helping to validate the usability of our interface, as described in the table and figures below.

| Question | Strongly Disagreed | Disagreed | Agreed | Strongly Agreed |
|---|---|---|---|---|
| 1 - I think that I would like to use this system frequently | 0 | 3 | 3 | 0 |
| 2 - I found the system unnecessarily complex | 1 | 5 | 0 | 0 |
| 3 - I thought the system was easy to use | 0 | 0 | 4 | 2 |
| 4 – I think that I would need the support of a technical person to be able to use this system | 0 | 5 | 0 | 1 |
| 5 - I found the various functions in this system were well integrated | 0 | 2 | 2 | 2 |
| 6 - I thought there was too much inconsistency in this system | 1 | 4 | 1 | 0 |
| 7 - I would imagine that most people would learn to use this system very quickly | 0 | 1 | 3 | 2 |

| | | | | |
|---|---|---|---|---|
| 8 - I found the system very cumbersome to use | 3 | 3 | 0 | 0 |
| 9 - I felt very confident using the system | 0 | 0 | 6 | 0 |
| 10 - I needed to learn a lot of things before I could get going with this system | 2 | 4 | 0 | 0 |

Table 8. SUV Questionnaire questions and answers obtained

The results obtained with the questionnaire and debate held in the Workshop led to the reformulation of the prototype, presented next as Prototype Version 2. The questionnaires can be seen individually in Appendix G.

## 4.8 Prototype Version 2

Taking into account that 50% of the testing users were not confident they would use the system, since that was the main purpose, even though the results about the system usability were very positive, we decided to reformulate the Prototype.

This reformulation was made taking into account the newest online applications to work with 3D shapes like Tinker CAD (“Tinkercad | Create 3D digital designs with online CAD,” n.d.), which show simple and interactive GUI that intend to be self-explanatory.

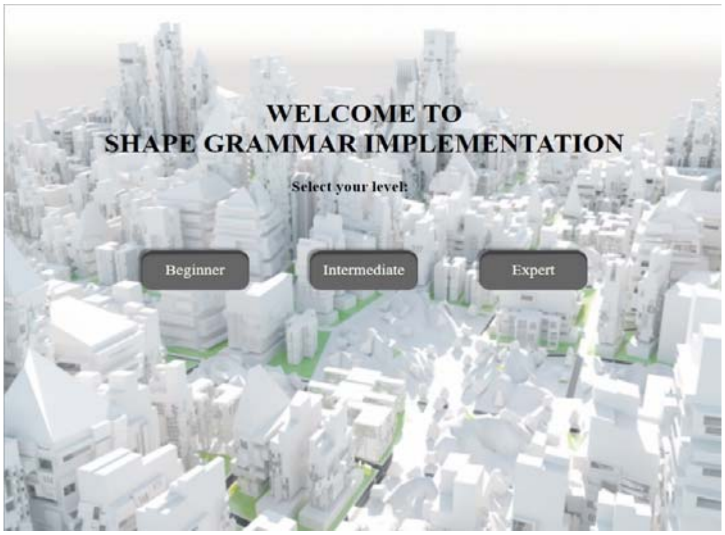
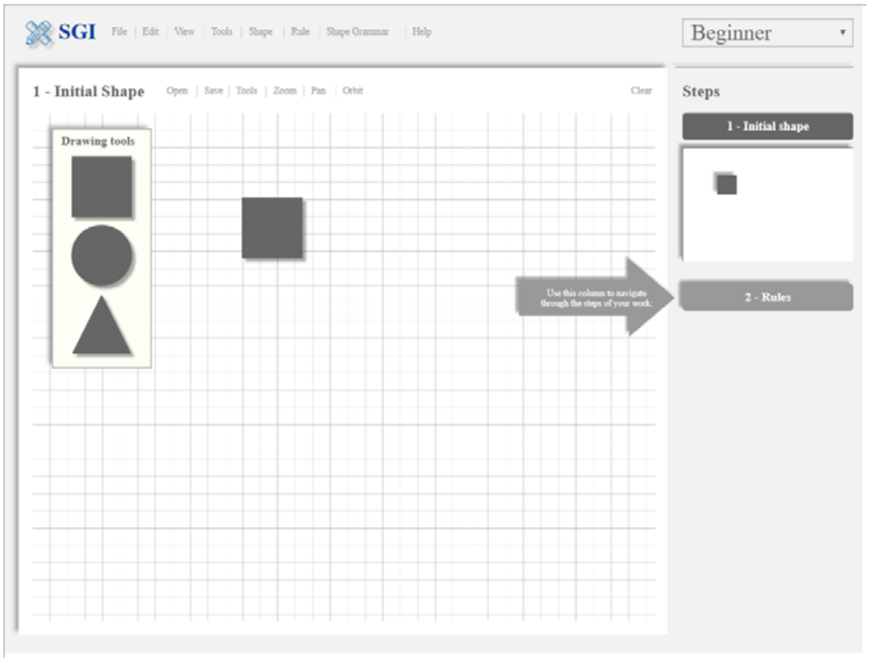


Figure 58 - SG Implementation Landing Page

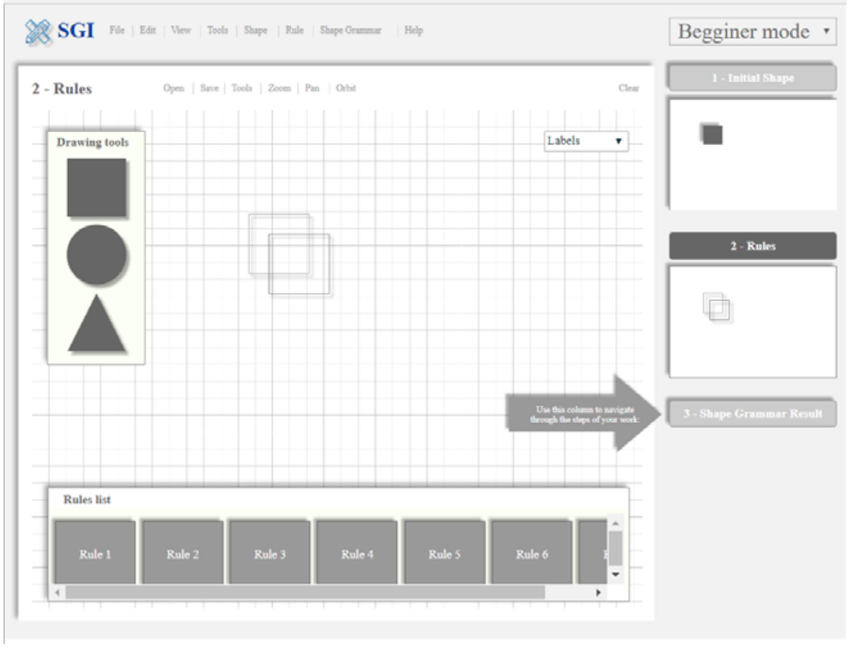
The landing page (Figure 58) of the Prototype Version 2 functions the same way as our first version; only the graphic design was changed, directing it more clearly for designers and architects.

The main difference introduced in Version 2 of the Prototype is the definition of one single window presented to the user at a time and a navigation pane on the right that allows the user to go through the several steps. "Help" messages were also introduced, being time sensitive (they display only for a short period once you enter the new task window). This main difference from the Beginners Mode to the Intermediate Mode and Expert Mode stays the same as in the previous version. In the intermediate mode, it is possible to draw freely, instead of using predefined shapes, and in the expert mode, a Code-Line that is directly written is available for SG creation and manipulation,


SGI
File | Edit | View | Tools | Shape | Rule | Shape Grammar | Help
Beginner ▾
1 - Initial Shape
Open | Save | Tools | Zoom | Pan | Orbit
Clear
Drawing tools
Steps
1 - Initial shape
2 - Rules

Use this column to navigate through the steps of your work.

Figure 59 - Beginners Mode Initial Shape Page (Task 1)



Use this column to navigate through the steps of your work.

Figure 60 - Beginners Mode Rules Page (Task 2)

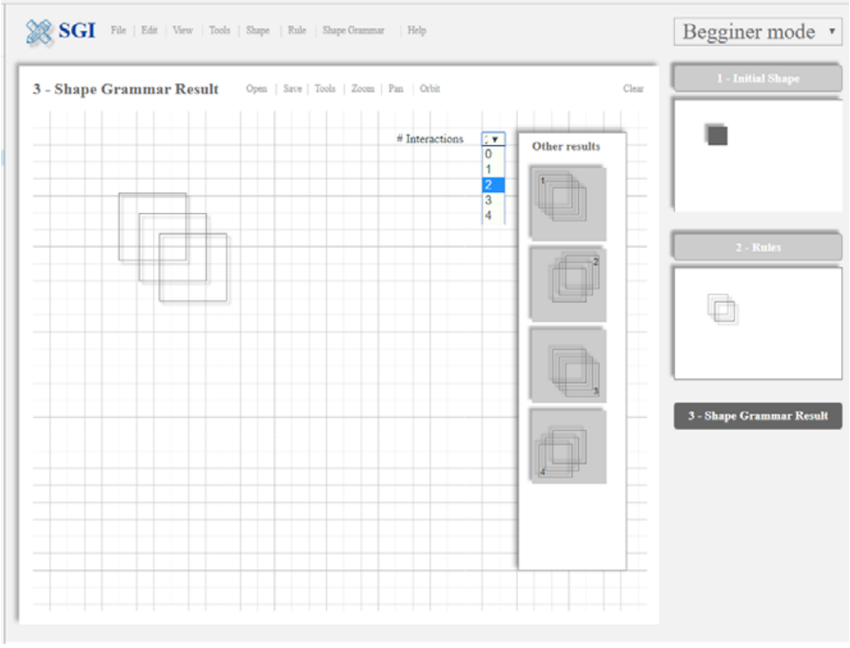


Figure 61 - Beginners Mode Shape Grammars Result Page (Task 3)

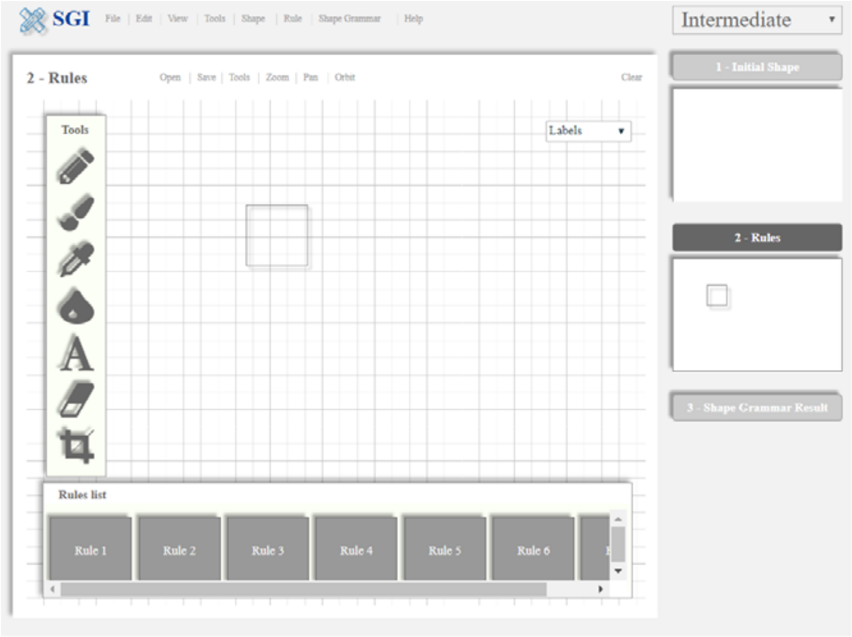


Figure 62 – Intermediate Mode Rules Page (Task 2)

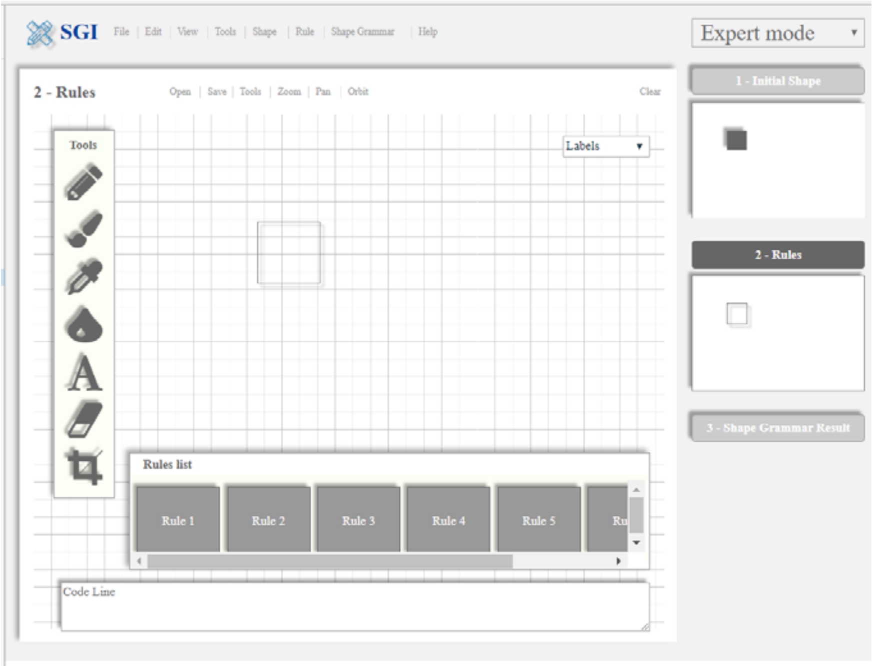


Figure 63 – Expert Mode Rules Page (Task 2)

Please refer to Appendix F for a complete description of the Axure's Prototype created. Again, only the Beginners Mode is ready for testing. It is important to state that, although the Prototypes were made to work with 2D shapes, our interface model does not limit the power of using 3D SG in any way so that the interface can adjust to more complex geometries.

As an interface prototype, specific features of the SG applications are not addressed or are limited. We chose to use 2D shapes representations because, in the field of architecture, they are full of meaning when exploring conceptual ideas. The design plans continue to be the main way of communicating an architectural project and are many times the main way of exploring specific solutions.

Although having the three modes (Beginner, Intermediate, Expert) for the three types of users (Students, Designers/Artists, Experts), only the simpler mode of our Prototype 2 has been developed. As a starting point, we believe that once the simpler mode is validated, the other ones shall work identically.

The appendix A of this thesis refers to examples of how SG could be used for architectural projects (Tching, Reis, & Paio, 2013). As this is our main goal, we developed a simulation on the use of our prototype by an architect to create an SG that draws urbanistic developments, according to specific rules.

## 4.9 TEST 2 AND VALIDATION

This chapter illustrates the use of the prototype in two ways: (1) general use of the tool that would be implemented behind the prototype; (2) illustration of the use to use SG in the composition of an urbanistic project.

For the test of the second prototype we decided to use a different approach than for the first version. As the general functioning of the interface was not changed, as the same IM-sgi criteria were applied, we believed the results would be similar and wouldn't allow us to get more conclusions of its valid use for our thesis purpose (the use of SG for architectural and design projects). With this in mind, we decided to describe carefully how the SG implementation behind the prototype should behave (Step-by-Step) and then simulate that use for a Urbanistic Creative Project, validating how it could apply to our main goal.

### 4.9.1 GENERAL STEP-BY-STEP

As our prototype was functionally developed for the *Beginner Mode*, that is the one that is next presented

1 – Select experience level:

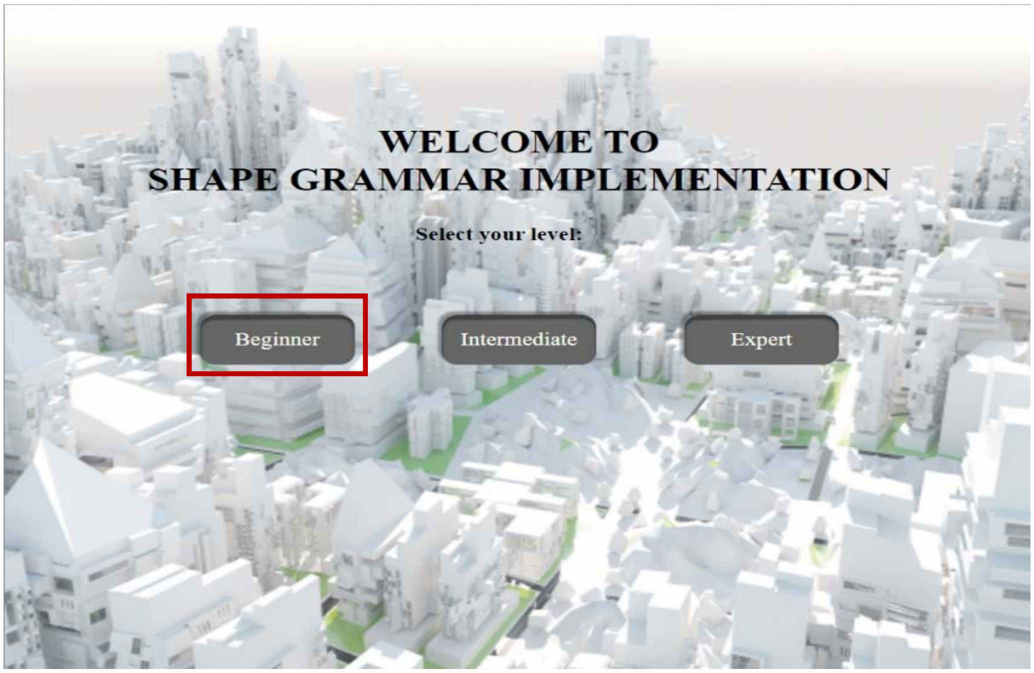


Figure 64 - SG Implementation Landing Page

2 – Use *Drawing Tools* to perform the first step indicated on the right side *1 – Initial Shape*:

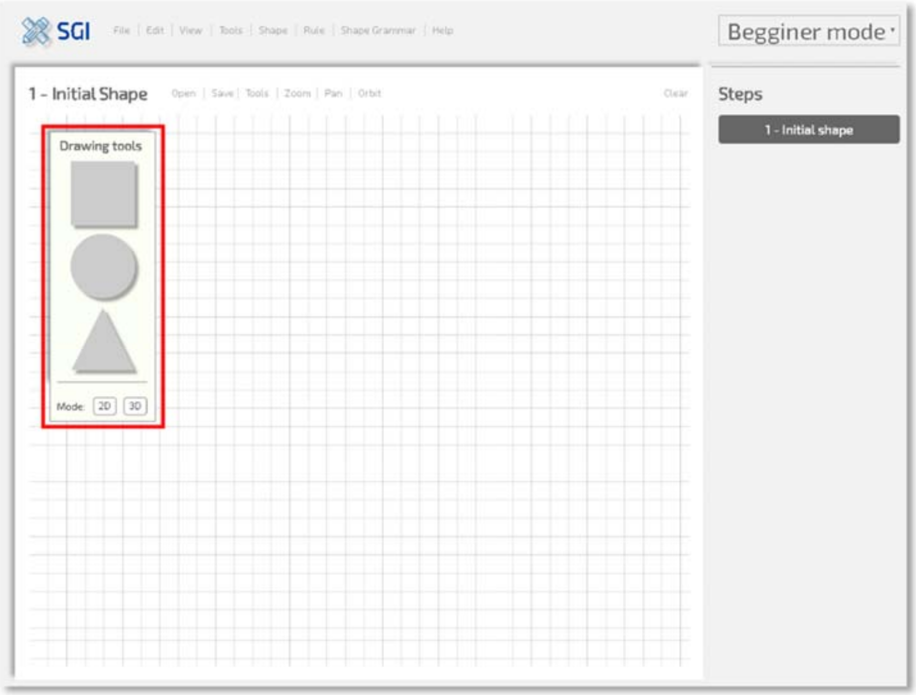

SGI
File | Edit | View | Tools | Shape | Rule | Shape Grammar | Help
Begginer mode
1 - Initial Shape
Open | Save | Tools | Zoom | Pan | Orbit
Clear
Drawing tools
Mode: 2D 3D
Steps
1 - Initial shape

Figure 65 – Choosing the Initial Shape

The introduction of a new button is shown in the *Drawing Tools* box, to illustrate how the interface can adjust for the abilities the SG implementation might have. In this example, we show that the user could choose between 2D and 3D shapes before starting to work.

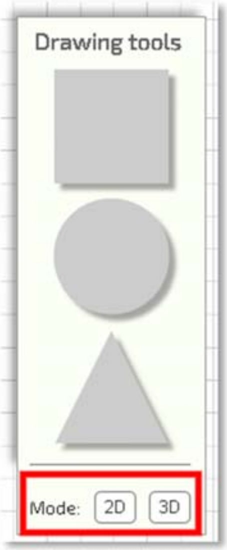


Figure 66 – Choosing the Shape Mode

3 -Create Initial Shape dragging one of the three shapes available to the main drawing window:

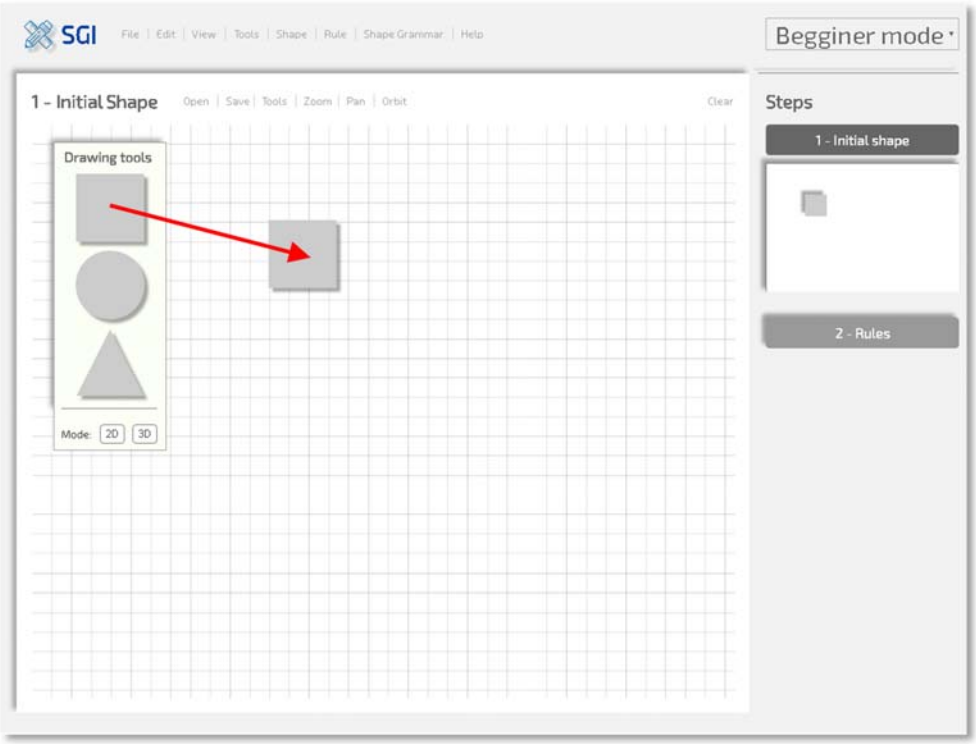

SGI
File | Edit | View | Tools | Shape | Rule | Shape Grammar | Help
Begginer mode
1 - Initial Shape
Open | Save | Tools | Zoom | Pan | Orbit
Clear
Drawing tools
Mode: 2D 3D
Steps
1 - Initial shape
2 - Rules

Figure 67 – Drag Shape to define Initial Shape

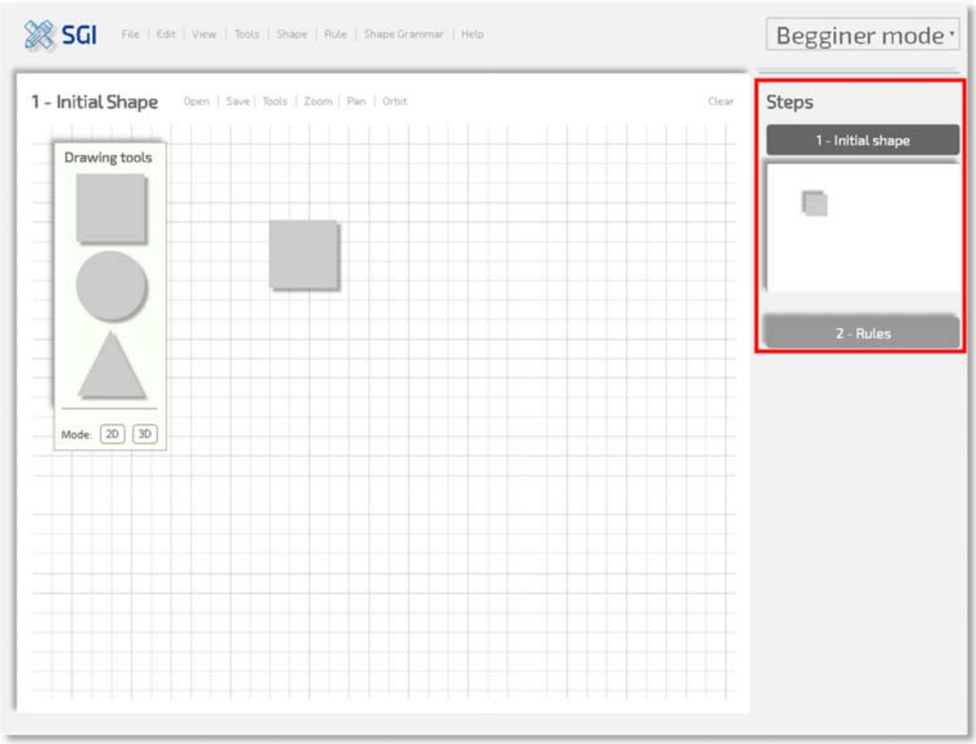

SGI
File | Edit | View | Tools | Shape | Rule | Shape Grammar | Help
Begginer mode
1 - Initial Shape
Open | Save | Tools | Zoom | Pan | Orbit
Clear
Drawing tools
Mode: 2D 3D
Steps
1 - Initial shape
2 - Rules

Figure 68 – Step illustration to help to guide the generation process

To create the initial shape, we only need to drop the shape to be used (Figure 67). At the same moment, the shape is "dropped" in the core of the drawing window, in the right column, a thumbnail appears representative of Step 1, as well as the button to go to the next step *2- Rules* (Figure 68). As shown in the previous chapter, guiding messages also appear.

To change the initial shape, simply drag another to the core of window 1 – Initial shape.

4. Click the *2 - Rules* button to start rule creation

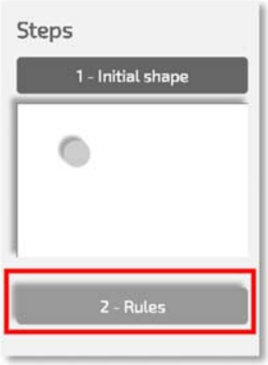


Figure 69 – Press Button to follow the generation process

5. Select Shape that defines the Rule, through spatial relations:

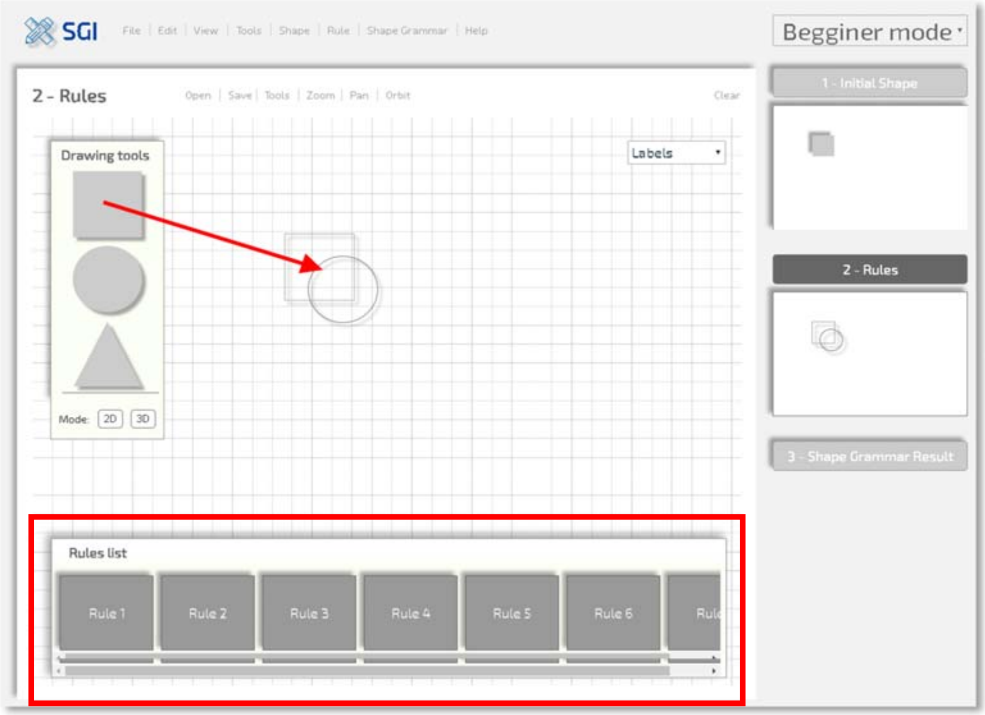


Figure 70 – Creating a Rule dragging a second shape from the *Drawing Tool*box

This prototype allows simulating the operation by combining a square with another, with a circle or with a triangle. To define the rule, we can drag a second shape to combine the two, according to the spatial relation and matching condition desired.

The bottom window shows the possibility of creating multiple rules to compose the SG. This feature is only shown; it is not functional in the prototype.

Note that when the shapes are combined, a thumbnail is reproduced in the right column, to illustrate the rule created. Also, a button for the third step of the generation process appears named *3-Shape Grammar Result* (Figure 71).

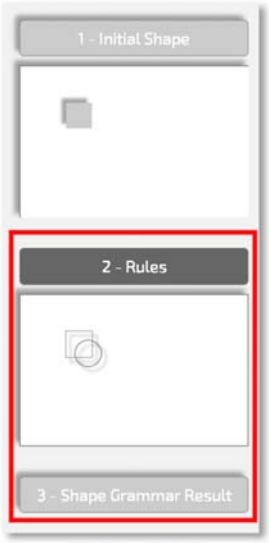


Figure 71 – Illustration of the created Rule and the Button for the next step

6. Apply the SG created by choosing the *Interactions* number to apply

Once the button for the third step is pressed (*3 – Shape Grammar Result*), a new feature given by a dropdown menu (# *Interactions*, marked in the window above) appears to allow the SG application (Figure 72). Changing the number of interactions allows to see the SG result reproduced both in the Window *3 – Shape Grammar Results* as well as in the *Other results* module, that shows several options that the same SG and number of iterations can produce. The *Other Results* window is the open the door to allow emergence and to enable the user to have some level of control on the results choice.

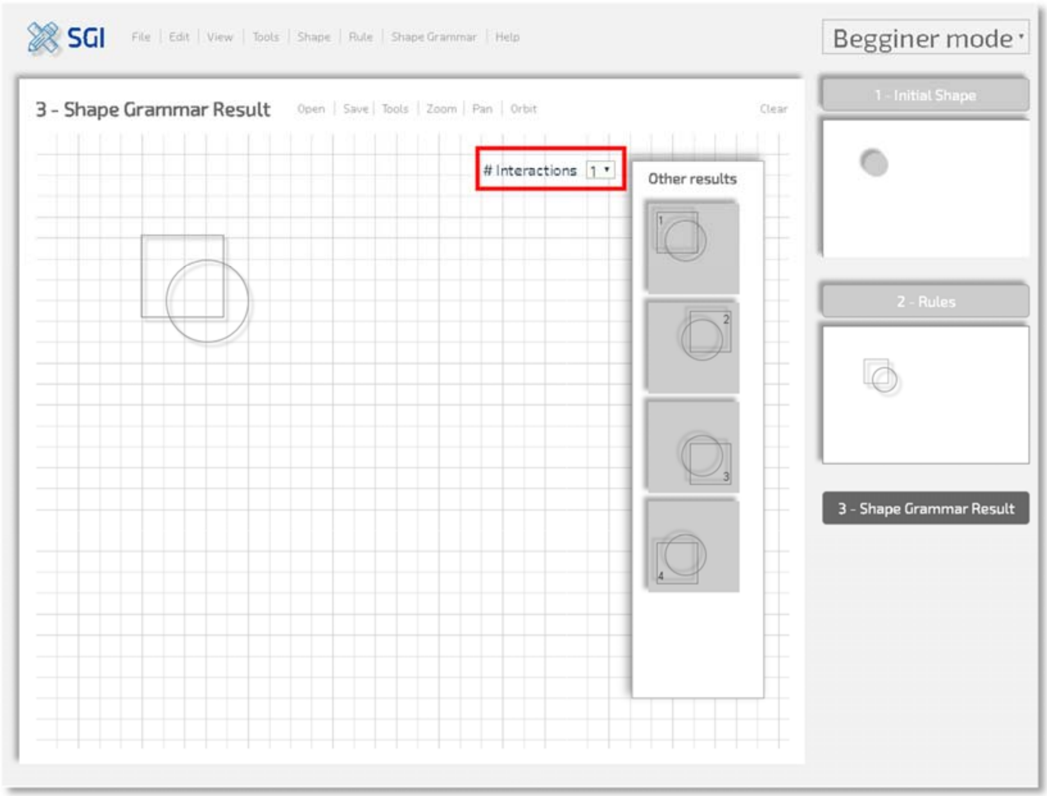


Figure 72 – Interactions choosing Drop Down List

In resume, with very simple actions that only involve dragging shapes, press buttons, and number selection, any beginner user of SG can create simple SG and visualize the results.

### 4.9.2 Using SG for Urbanistic Creative Project

Although our prototype was developed for the Beginner Level, so we could in a less complex way test its functioning and adequacy, as our goal is to help SG enter the architecture and design practice, we made the exercise of simulating the use of the prototype for the definition of an SG for urbanistic design. According to the examples of SG grammar use in the architectural practice published and shown on Appendix A, next, we show the use of the prototype for an SG that creates Lot Definitions, according to building and public space relations.

The rules of the SG presented next express orthogonal definitions for buildings and circulation ways. Other rules can be defined, such as radial intention (as it can be seen in some cities) or be extrapolated to emerging buildings, through a wide range of shape relationships between them.

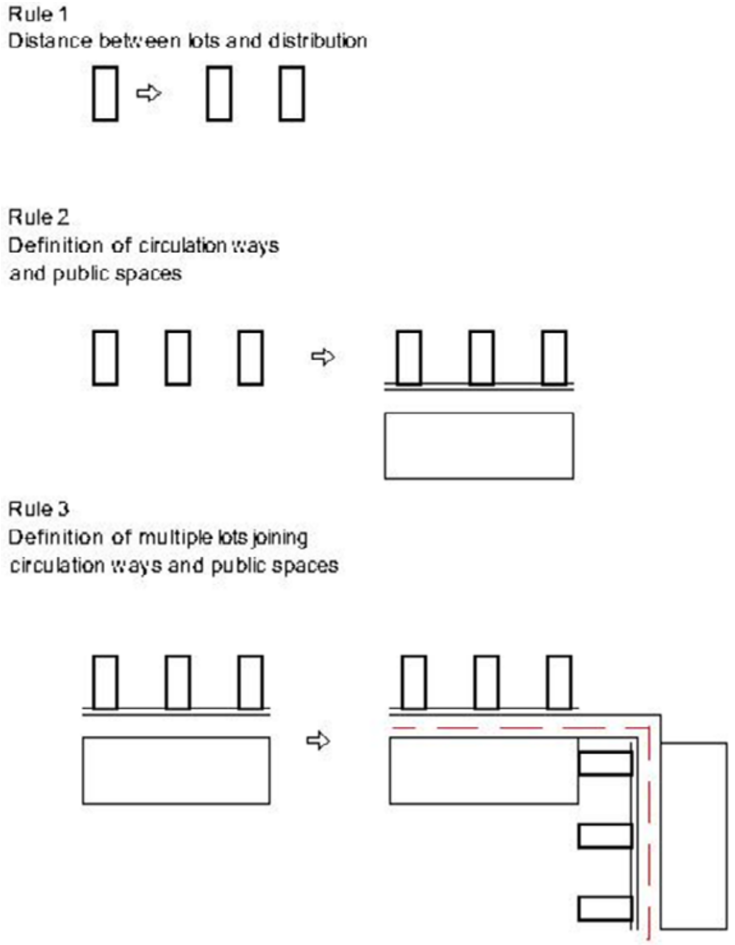

Rule 1
Distance between lots and distribution
Rule 2
Definition of circulation ways
and public spaces
Rule 3
Definition of multiple lots joining
circulation ways and public spaces

Figure 73 – Lot Distribution

Figure 73 illustrates three rules that compose an SG to define buildings with correct distances (smaller rectangles), streets (illustrated by the dashed red line) and public areas (bigger rectangles). The third rule is the multiplication of lots, following the composition generated by the previous ones. We skipped the illustration of these 3 rules creation, as it would follow the Step-by-Step presented previously. We focused on showing the use of the SG composed by these three rules.

The first rule of the SG multiplicate rectangles according to a specific distance, symbolizing distribution of the building lots (Figure 74). The geometry created in the main window shows the  result of applying Rule 1 with 2 iterations.

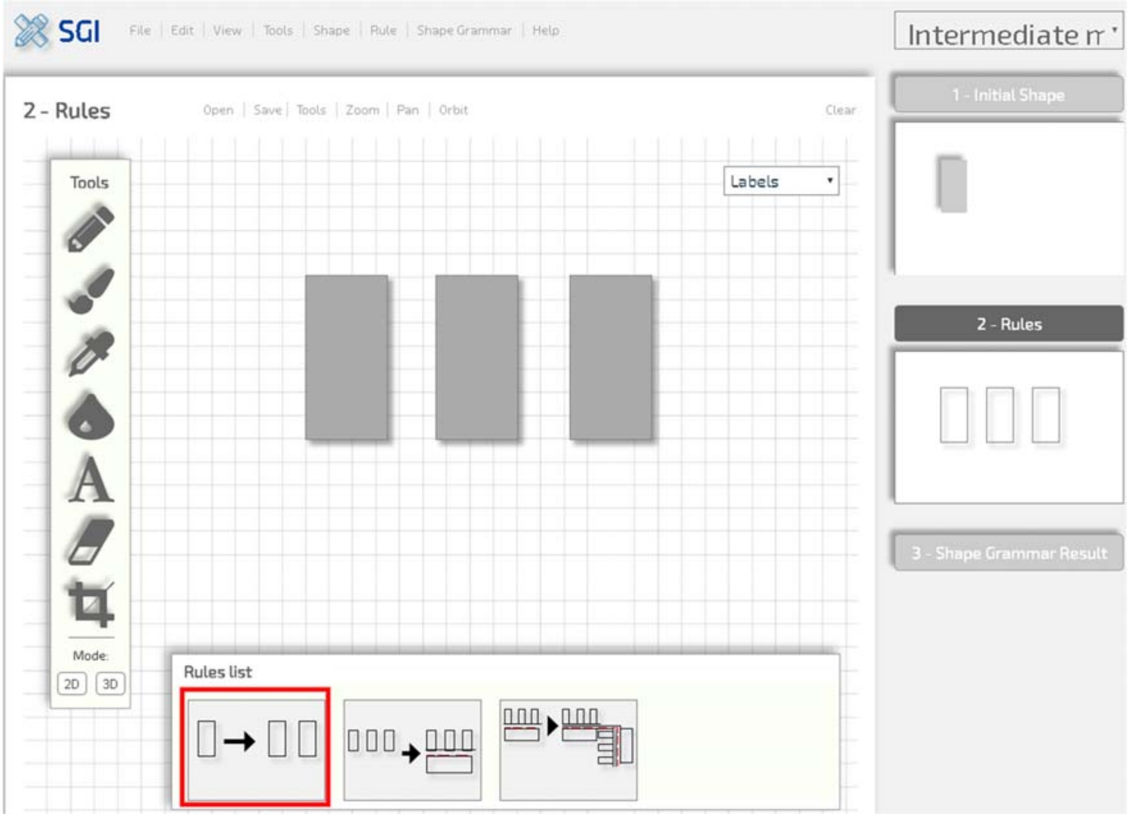


Figure 74 – Applying Rule one

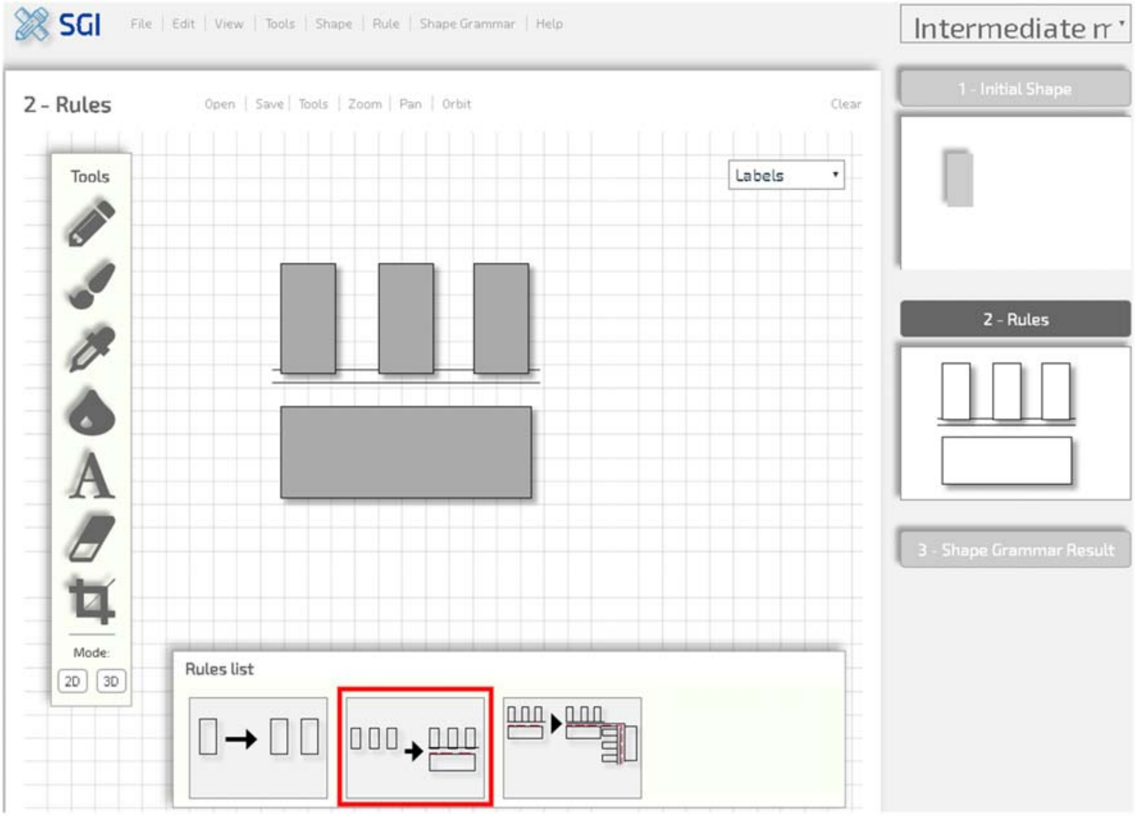


Figure 75 – Streets and Public Spaces

When Rule 1 of the SG application originates three rectangles, Rule 2 can be activated, as it matches the left side of the rule. The result of the use of Rule 2 originates a group of 3 buildings, the sidewalk, and street adjacent to them and public space on the other side of the street .

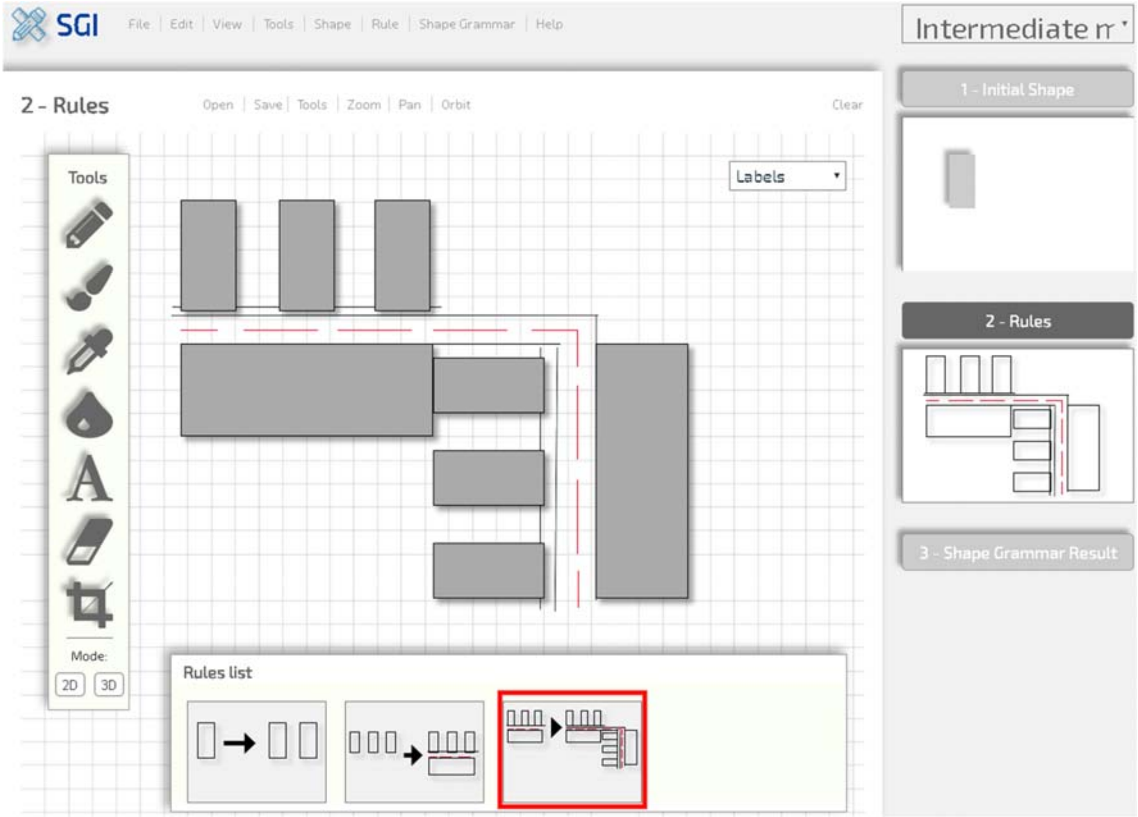


Figure 76 – Combinations of Streets and Public Spaces

Once Rule 2 is applied, Rule 3 can be used, as it matches the combination of groups of buildings and public spaces (Figure 76)

The following image (Figure 77) shows more advanced shapes being displayed in the "Other Results" module to show how Rule 3 can produce different results with the same number of iterations.

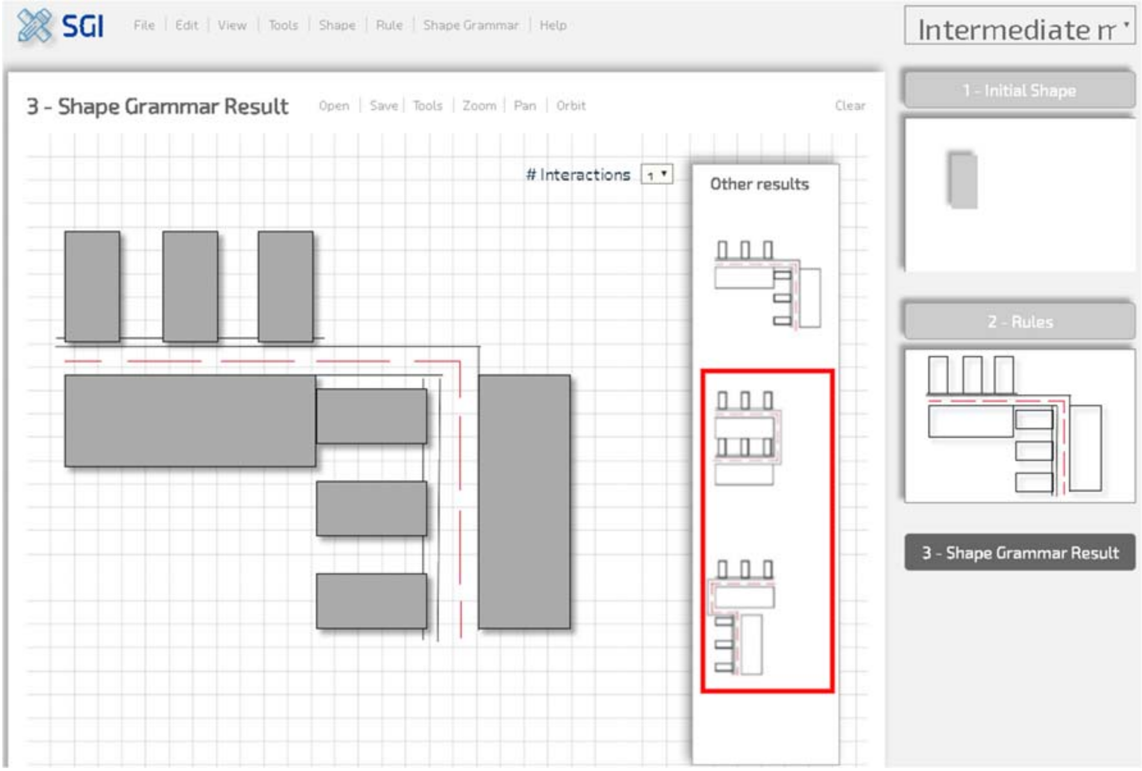


Figure 77 – Results Options

The SG illustrated above shows our vision on the use of SG. The architect's work is based in geometries and relations, and the initial stages of design are many times developed with simplified geometric objects that symbolize complex construction elements.

We also tried to show how the SG implementation behind the prototype would work and help on creative projects, although it was not the scope of this investigation to develop it, as our focus is on interaction and interface communication.

To validate our interface model IM-sgi we used prototypes and illustrated the functionality that the prototype proposes.

We used the HCI method of prototyping to be able to test the criteria of IM-sgi on a GUI interface. The choice of this type of interface was, as referred before, due to the verification that it is well suited to resemble the architect, designer, or artist work when drawing. Also, although several developments are happening in the area, GUI-based software is still prevalent, having evolved mainly in its input methods and response complexity.

It is interesting to state that there are many investigations working around GUI Design Smells, referring to the issues found on the GUI that lead to bad design of the application architecture.

When the GUI relations and interactions with the user are not well suited for the actions needed, the programmer has to come across with challenges on making functionality and interaction to combine and, if the problem is in the interface, it may take him to design the application functioning in a wrong way. This area was not explored, but it reflects the importance of a well-suited interface for each specific application.

To verify a GUI studies have been presented showing that the evaluation of the interface and its usability doesn't imply to exercise the full application functionality (Alkhalid & Labiche, 2018). The use of static analysis, many times recurring to applications that using machine learning simulate the user actions, allows the test of the interaction with the software and simulate its functionalities.

That said, our approach was to create synthetic examples of use, recurring to a prototype of an interface following IM-sgi, focusing the prototype test on the input-output relations, interaction guidance, and clarity of actions and results. We then believe that the prototype verification allows us to show the capabilities of an interface that follows IM-sgi criteria and generalize our results to any SG application whose interface follows IM-gi.

Benedikt Hauptmann (2012) proposes a model-based test recurring to the user where he states that on an interactive system the interface is the mediator between application and the user, we seek the opposite approach he presents. His goal is to define different test types for different goals: (1) pure test logic of the application with no dependence or information on the UI; and (2) pure UI knowledge to execute tests for interaction behaviors. While we propose an interface model, with criteria on how to define an interface that correctly reaches the end users and allows the ease of use to perform the actions, the author proposes UI models to test existing interfaces.

Acknowledging the importance of the UI for the application's success, the use of UI models has been widely studied (Martins & Garcia, 2015). Martins and Garcia presented a literature review on methods to validate UI models and stated that most of them didn't go through validation tests; they just focus on presenting the UI characteristics that the model pursuits. This way, there is not a standard on validation of these models and stated that the technique chosen by the creators of UI models to validate it is related with the objectives of the project and profile of the person that will validate.

In this investigation, we also struggled to understand what the best validation method for our interface model would be. Recurring to prototyping and user testing is a well know HCI method, so we decided to use them.

Although the Workshop performed to make user test on our prototype didn't have a considerable amount of participants, which means the results are not precise, we still believed it was interesting to present the results, and we used the discussion on the Workshop to developed further work and improved the prototype, creating a new version.

Since it was out of scope from this investigation to create an SG implementation, due to its tremendous complexity that would lead us away from our main goal, we tried to illustrate how the system would work on an example of SG for urbanistic projects. Our objective with this was to validate that the criteria on the prototype produce interfaces that adjust to SG use in the architectural project practice.

# 5 Discussion, Conclusion and Future Work

The work of this thesis started with the analysis of existing SG implementations and the evidence that the existing interfaces for these implementations do not allow the user to understand how to use them, or how to control and make use of its results.

With the development of Artificial Intelligence and, within this area, the growth of applications to simulate creativity, the architectural projects gained computational applications which boost not only the technical aspects but also the creative and aesthetic ones. We believe that SG implementations are a good example of AI and we seek to help them enter the creative project practice.

The technology evolution has led to a change in working methods and results achieved in several areas. In Architecture and Design, the representation of the drawing evolved to digital design, carried out with CAD applications, allowing the faster and more faithful representation of the designer's ideas, facilitating his work and making it more efficient, leading to better interpretation by third parties and elimination of errors. Digital Design and CAD applications are widely used today in architectural projects and design. They allowed the production of more complex and ambitious projects, offering new ways of analysis, control, and representation that would not be otherwise available to designers, as more time and unaffordable resources would be required.

However, the contribution of computer technology in the areas of Architecture and Design can surpass simple computerization and the digital representation of what the artist would draw manually.

The recognition of the need to use large amounts of problem-solving knowledge in which the space of alternative paths to solutions is enormous, making the problem computationally intractable, has led to the need to elaborate methods based on exhaustive search. The contribution of AI in this context are methods of explicit representation of knowledge and reasoning, complementing other general methods of knowledge processing, such as those related to Heuristic Search and Logic. In this way, intelligent knowledge-based systems can help the development of SG computational implementations to support the creative project in architecture, design, or even in other artistic fields. SG also allow the possibility of exploring more solutions in a shorter time, allowing the ambition of more complex and daring solutions, as the aid of the computer applications helps the Architect and Designer to test the validity of more and/or better options.

Architectural/design projects, in contrast to other artistic areas, develop in different phases that can be described not only as a need to solve a large number of issues but also as the fulfillment of rules and constraints, whether they are legal, environmental, economic or formal. It is in the solution of all these issues that the final project comes out, and the architect shows his/her creativity, combining all the items in an aesthetic and functional product. Dividing the project into its elementary parts, we see that the architect/designer selects, consciously or intuitively, a set of rules and makes choices that impact the final product (Tching et al., 2013).

Verifying that architects and designers have not adopted existing SG computational implementations, we analyzed those tools and verified that the HCI aspects of the interface were not addressed correctly, making the communication with the user hard to accomplish and not allowing the exploration of using SG for creating new design options.

Thus, our focus was in two objectives: (1) create guidelines for the proper communication between application and user; and (2) help SG enter the architectural and design practice as a partner in search of creative solutions. With these two goals in mind, we understood that our object of study was the interface of SG computational implementations.

Searching to address the interface issues, believing that solving the communication issues will lead architects and designers to verify the potential of working with SG, the proposal of this thesis is an interface model for SG computational implementations, named IM-sgi. This model was developed to answer the need for better communication between the user and SG implementations. We defined IM-sgi with a set of criteria for interfaces, and we used HCI methods to test it, to show that it can lead to the creation of interfaces for SG applications that clearly communicate with the user and guide him in the use of SG in the creative process of a design or aid in architectural/design projects.

To facilitate the computational design process, we have researched historical models of interfaces using SG in the design process. We then proposed and tested a model to address the interface issues related to the use of SG by architects, designers, artists, and students, as they all interact differently and with different purposes in mind. Our interface model IM-sgi aims to help SG implementations to effectively enter into the design practice and become a relevant way of exploring ideas and design solutions. We believe that the creative possibilities that SG can offer to architects/designers are revealed as we further develop our interface. If the usability issues of SG implementations are well addressed, and if the SG implementations interface communicates correctly with the user, then using SG implementations should make it possible for the architects/designers to use SG in their common practice. Our model, which specifies categories of SG users and their objectives, clarifies how each type of user makes use of SG and what barriers of communication need to be addressed (Tching et al., 2016). This increases the possibilities of the success of SG computational implementation use.

Our research was led by the belief that if the usability issues of SG implementations are well addressed, and if the SG implementations Interface communicates correctly with the user, SG can offer creative professionals unrevealed possibilities. In the creation of a computer application, the interface is a time-consuming task, and many times the programmer, or program designer, is not familiar with the true needs and limitations of the end-user. The definition of our interface model was possible through a rigorous study that specifies categories of SG users and their objectives, clarifies how each type of user makes use of SG and what barriers of communication need to be addressed, so that is possible to increase the possibilities of the success of computer SG implementation use.

Based on this research, three publications on the subject took place: (Tching, Paio, & Reis, 2012), (Tching et al., 2013), later invited to be published in (Tching, Reis, & Paio, 2017), and (Tching et al., 2018). A new manuscript was submitted for publication, where the IM-sgi validation is presented and is waiting for a decision.

## 5.1 LIMITATIONS AND FUTURE WORK

For the study, creation, and validation of IM-sgi, several different methods were used, seeking a better demonstration of the criteria applicability and the results that they originate.

First, a CW, which is a Usability Inspection Method that does not demand a user interface specialist or tests with users. The CW was used to analyze existing SG implementations to understand what criteria hadn’t been correctly addressed.

Second, an Interface Prototype was created following IM-sgi criteria and using the knowledge obtained from the previous analysis. This prototype was tested with architecture students in a workshop, using a questionnaire from the System Usability Scale, to understand its usability issues. Although the sample was not significant, it led to a reformulation of the prototype, which is also presented here. As a result, using our established criteria when creating this interface prototype, we have completed the steps needed for creating a new interface that focuses on the success of the task performance, according to (Lewis & Rieman, 1994).

Finally, we present a Step-by-Step use of the SG implementations that the Prototype simulates the creation of an SG for Architectural Projects, more specifically for urbanistic design, describing the functionalities and user actions that allow the definitions of the SG and its use.

For a more deep and complete test of IM-sgi, it should be tested on a wholly programmed SG implementation, allowing the real test of creating shapes, rules, applying, and manipulating SG.

This test should be performed by the types of users defined in this thesis, with a strong focus on designers and architects, to understand their perspective on starting to use SG on their daily practice, as a new way to explore design alternatives and stimulate creativity.

## 5.2 CONTRIBUTIONS

With our research, we seek to contribute both to the Architectural and Design Practice and the fields of Computational Creativity, Computational Design, and HCI.

On the field of HCI, we believe our work contributes to helping software engineers create SG implementations correctly addressing the end user and the communication with him.

Our interface model addresses the types of users and how they relate to SG in terms of SG knowledge and SG final use. Then it presents a set of criteria that clearly defines what actions the SG implementation must allow and how it must show/communicate the results. Assuring that the SG implementation addresses the user needs and correctly communicates with him, we believe he will adopt the use of SG on the daily practice easily, as the creative potential is enormous and the possibilities for developing project ideas are endless.

With these new SG implementations that allow the use of SG by artists, we seek particularly to contribute to the Architectural practice, helping the use of computational applications to happen not only on the final stages of the design process but in the early stages when the hypothesis is being studied.

With SG implementations, the Architect can answer not only to the technical needs but also to creative goals, taking advantage of the synergy between architect and a computational system that can originate projects with new achievements, functional and artistic. SG can define a wide range of solutions that enhance the creative response to a problem and can give the architect creative responses that he would not achieve in another way.

As it is general practice to use computational applications that reproduce the architect's manual design, among other technical aspects of the project (such as automatic measurements, thermal simulations, three-dimensional visualization, etc.), we believe that the next step is the use of SG in the common practice of Architecture, to explore innovative designs. This new way of working is a way of optimizing ideas, using not only computational design but also computational creativity.

Our research sought to contribute to this goal.

# 6 BIBLIOGRAPHY

Agarwal, M., & Cagan, J. A blend of different tastes: The language of coffeemakers, Environment and Planning B: Planning and Design § (1998). https://doi.org/10.1068/b250205

Agarwal, Manish, & Cagan, J. (1997). Shape Grammars and their languages - a methodology for product design and product representation. New York, New York, USA: Proceedings of the 1997 ASME Design Engineering Technical Conferences and Computers in Engineering Conference: Design Theory and Methodology Conference.

Agarwal, Manish, Cagan, J., & Constantine, K. G. (1999). Influencing generative design through continuous evaluation: Associating costs with the coffeemaker shape grammar. *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, *13*(3), 253–275. https://doi.org/10.1017/S0890060499134024

Agarwal, Manish, Cagan, J., & Stiny, G. (2000). A micro language: generating MEMS resonators by using a coupled form - function shape grammar. *Environment and Planning B: Planning and Design*, *27*(4), 615–626. https://doi.org/10.1068/b2619

Akin, O., & Anadol, Z. (1993). What's wrong with CAAD. *4th International Symposium on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany*.

Alkhalid, A., & Labiche, Y. (2018). On Graphical User Interface Verification, (Icsoft), 373–380. https://doi.org/10.5220/0006916603730380

Apple Computer Inc. (1992). Macintosh Human Interface Guidelines. *Addison-Wesley Publishing Company* . https://doi.org/ISBM 0-201-62216-5

Arciszewski, T., Michalski, R. S., & Wnek, J. (1995). Constructive induction: the key to design creativity. *Machine Learning*, (April), 1–30.

Bastien, J. M. C. C., & Scapin, D. L. (1995). Evaluating a user interface with ergonomic criteria. *International Journal of Human-Computer Interaction*, *7*(April 1995), 105–121. https://doi.org/10.1080/10447319509526114

Bastien, J. M. C., & Scapin, D. L. (1993). Ergonomic criteria for the evaluation of human-computer interactions. *INRIA*, (May), 79. Retrieved from hal.inria.fr/inria-00070012/fr/

Biswas, K. K. De. (2006). A Computational Model of Virtual Interpretation. Massachussets Institute of Technology.

Boden, M. A. (2004). *The Creative Mind: Myths and Mechanisms, Second Edition*. Routledge Taylor & Francis Group. https://doi.org/10.4324/9780203508527

Borkowski, A., Gross, M., Heisserman, J., Knight, T., & Nakakoji, K. (1998). Emergence in Design. *AID'98*.

Bottazzi, R. (2018). *Digital Architecture Beyond Computers: Fragments of a Cultural History of Computational Design*. Bloomsbury Visual Arts.

Brown, M. H., & H., M. (1988). Perspectives on algorithm animation. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88* (pp. 33–38). New York, New York, USA: ACM Press. https://doi.org/10.1145/57167.57172

Carroll, J., & Rosson, M. B. (2003). Design Rationale as Theory. *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*, 431–461. https://doi.org/10.1016/B978-155860808-5/50015-0

Carroll, J M. (1997). Human-computer interaction: psychology as a science of design. *Annual Review of Psychology*, *48*, 61–83. https://doi.org/10.1146/annurev.psych.48.1.61

Carroll, John M, & Rosson, M. B. (1986). Paradox of the Active User. *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, *MIT Press*, 80–111. https://doi.org/10.1017/CBO9781107415324.004

Celani, G., Pupo, R., Mendes, G., & Pinhero, É. (2005). Generative Design Systems for Housing. *ECAADe 23*. Retrieved from http://cumincad.scix.net/cgi-bin/works/Show?2005_501

Chase, S. C. (1989). Shapes and shape grammars : from mathematical model to computer implementation. *Environment and Planning B: Urban Analytics and City Science*, *16*(2), 215–242. https://doi.org/10.1068/b160215

Chase, S. C. (1996a). Design Modeling With Shape Algebras and Formal Logic. In F. Ozel &

P. McIntosh (Eds.). *Design Computation: Collaboration, Reasoning, Pedagogy, Proceedings of ACADIA '96*. Association for Computer-Aided Design in Architecture.

Chase, S. C. (1996b). Representing designs with logic formulations of spatial relations. Workshop Notes, Visual Representation, Reasoning and Interaction in Design, Fourth International Conference on Artificial Intelligence in Design.

Chase, S. C. (2002). A model for user interaction in grammar-based design systems. *Elsevier*, *11*(2), 161–172. https://doi.org/10.1016/S0926-5805(00)00101-1

Chase, S. C. (2005). Economies of scale : Generative design tools and adaptability Variety of design alternatives, (May).

Chase, S. C. (2010). Shape grammar implementations: The last 36 years. *Design Computing and Cognition Workshop*. Stuttgart.

Chen, Z. (1999). Computational Intelligence for Decision Support. CRC Press. https://doi.org/10.1201/9781420049145

Chien, S.-F., Donia, M., Snyder, J., & Tsai, W.-J. (1998). Sg-clips: a system to support the automatic generation of designs from grammars. *CAADRIA '98 [Proceedings of The Third Conference on Computer Aided Architectural Design Research in Asia]*, 445–454.

Chiou, S., & Krishnamurti, R. (1995). Grammatical Basis of Chinese TRaditional Architecture.

Chomsky, N. (1955). *The Logical Structure of Linguistic Theory - Noam Chomsky - Google Livros*. (M.I.T. Library, Ed.). Retrieved from https://books.google.pt/books/about/The_Logical_Structure_of_Linguistic_Theo.html?id=WlfbRAAACAAJ&redir_esc=y

Choudhary, R., & Michalek, J. (2005). Design optimization in computer-aided architectural design. *10th International Conference on Computer Aided Architectural Design Research in Asia*, 149–159. Retrieved from http://cumincad.scix.net/data/works/att/caadria2005_b_4a_b.content.pdf

Christian Bastien, J. M., & Scapin, D. L. (1993). Ergonomic Criteria for the Evaluation of Human-Computer Interfaces @BULLET Critères Ergonomiques pour l'Évaluation d’Interfaces Utilisateurs.

Consiglieri, V. (1994). *A morfologia da arquitectura : 1920-1970*. Editorial Estampa. Retrieved from https://www.wook.pt/livro/a-morfologia-da-arquitectura-i-i-victor-consiglieri/84017

Correia, R. C., & Coutinho, R. (2013). DESIGNA -A Shape Grammar Interpreter. *Tecnico Lisboa*. Retrieved from https://fenix.tecnico.ulisboa.pt/downloadFile/395145526538/dissertacao.pdf

Correia, R., Duarte, J. P., & Leitao, A. (2012). GRAMATICA: A general 3D shape grammar interpreter targeting the mass customization of housing. *Proceedings of the 30th ECAADe Conference*, *1*(Knight 2000), 489–496. Retrieved from http://cumincad.scix.net/cgi-bin/works/Show?ecaade2012_273

Dartnall, T. (1994). *Artificial Intelligence and Creativity : an Interdisciplinary Approach*. Springer Netherlands.

Do, E. Y., & Gross, M. D. (2004). Between worlds: visions and views for the future of cad. *GCAD ’04, International Symposium on Generative CAD Systems, Pittsburgh*.

Duarte, J. P. (2005a). A discursive grammar for customizing mass housing: the case of Siza’s houses at Malagueira. *Automation in Construction*, *14*(2), 265–275. https://doi.org/10.1016/j.autcon.2004.07.013

Duarte, J. P. (2005b). Towards the mass customization of housing: the grammar of Siza’s houses at Malagueira. *Environment and Planning B: Planning and Design*, *32*(3), 347–380. https://doi.org/10.1068/b31124

El-Zanfaly, D. (2009). Design by Algorithms : A Generative Design System for Modular Housing Arrangement. In *SIGraDi 2009: Proceedings of the 13th Congress of the Iberoamerican Society of Digital Graphics* (pp. 65–67).

Ertelt, C., & Shea, K. (2009). An Application of Shape Grammars to Planning for CNC Machining. *Volume 5: 35th Design Automation Conference, Parts A and B*, 651–660. https://doi.org/10.1115/DETC2009-86827

Flemming, U. (1986). More Than The Sum Of Parts-the Grammar Of Queen Anne

Houses.pdf.

Gero, J. S. (2006). Creativity, emergence and evolution in design. *Knowledge-Based Systems*, *9*(7), 1–29. https://doi.org/0.1016/S0950-7051(96)01054-4 Gero, J. S. (2006). Creativity, emergence and evolution in design. Knowledge-Based Systems, 9(7), 1–29. https://doi.org/10.1016/S0950-7051(96)01054-4

Gips, J. e Stiny, G. (1975). *The psychological experience of prototyping*.

Gips, J. (1974). A Syntax-Directed Program that performs a Three-dimensional Perceptual Task. *Pattern Recognition*, *6*, 189–199. https://doi.org/10.1016/0031-3203(74)90021-1

Gips, J. (1999). “ Computer Implementation of Shape Grammars ” Computer Implementation of Shape Grammars.

Gips, J., & Stiny, G. (1978). Artificial Intelligence and Aesthetics. *University o f Californi a LOS Angeles, California*, *907*(4), 907–911.

Gips, J., & Stiny, G. (1980). Production systems and grammars: a uniform characterization. *Environment and Planning B: Planning and Design*, *7*, 399–408. https://doi.org/10.1068/b070399

Grasl, T. (2011). GRAPE: A parametric shape grammar implementation. SimAUD’11. Retrieved from https://www.academia.edu/1013581/GRAPE_A_parametric_shape_grammar_implementation?auto=download

Grasl, T., & Economou, A. (2013). From topologies to shapes: parametric shape grammars implemented by graphs. *Environment and Planning B: Planning and Design*, *40*, 905–922. https://doi.org/10.1068/b38156

Hauptmann, B. (2012). Model-based test instantiation for applications with user interfaces, (June 2011), 27–30. https://doi.org/10.1145/2181101.2181108

Heisserman, J. (1991). Generative Geometric Design and boundary solida grammars. *PhD. Dissertation. Carnegie Mellon University*. Pitsburgh, PA.

Heisserman, J., & Callahan, S. (1996). Interactive grammatical design. *AI in Design ’96,* *Workshop Notes on Grammatical Design*. Stanford, CA.

Heisserman, Jeff. (1994). Generative Geometric Design. *Ieee Computer Graphics And Applications*, (March). https://doi.org/10.1109/38.267469

Hewett, T. T. (1999). Human-computer interaction and cognitive psychology in education. In *Proceedings of GVE*.

Hoisl, F. R. (2012). *Visual , Interactive 3D Spatial Grammars in CAD for Computational Design Synthesis*.

Hollender, N., Hofmann, C., Deneke, M., & Schmitz, B. (2010). Integrating cognitive load theory and concepts of human-computer interaction. *Computers in Human Behavior*, *26*(6), 1278–1288. https://doi.org/10.1016/j.chb.2010.05.031

IBM. (1990). *Common User Access – A consistent and usable human-computer interface for the SAA environments*. Retrieved from https://archive.org/details/ibmsj2703E

Jeffries, R., Miller, J. R., Wharton, C., & Uyeda, K. (1991). User interface evaluation in the real world: a comparison of four techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. https://doi.org/10.1145/108844.108862

Jørgensen, A. H., & Myers, B. A. (2008). User interface history. In *Proceeding of the twenty-sixth annual CHI conference extended abstracts on Human factors in computing systems - CHI '08* (p. 2415). New York, New York, USA: ACM Press. https://doi.org/10.1145/1358628.1358696

Jowers, I, Prats, M., Lim, S., Mckay, A., Garner, S., & Chase, S. (2008). Supporting Reinterpretation in Computer-Aided Conceptual Design, (June), 11–13.

Jowers, Iestyn, Hogg, D. C., McKay, A., Chau, H. H., & de Pennington, A. (2010). Shape detection with vision: implementing shape grammars in conceptual design. *Research in Engineering Design*, *21*(4), 235–247. https://doi.org/10.1007/s00163-010-0088-z

Karray, F., Alemzadeh, M., Saleh, J. A., & Arab, M. N. (2008). Human-Computer Interaction : Overview on State of the Art, *International Journal on Smart Sensing and Intelligent Systems. Volume 1 , Issue 1 , ISSN (Online) 1178-5608, DOI: 10.21307/ijssis-2017-283.* 137–159.

Keles, H. Y., O Ë Zkar, M., & Tari, S. (2010). Embedding shapes without predefined parts, *37*, 664–681. https://doi.org/10.1068/b36010

Khan, S., & Tuncer, B. (2019). Speech analysis for conceptual CAD modeling using multi-modal interfaces: An investigation into Architects' and Engineers' speech preferences. *Artificial Intelligence for Engineering Design Analysis and Manufacturing*. https://doi.org/10.1017/S0890060419000015

Kicinger, R., Arciszewski, T., & De Jong, K. (2005). Evolutionary Computation and Structural Design : a Survey of the State of the Art. *Science*, *83*, 1943–1978. https://doi.org/10.1016/j.compstruc.2005.03.002

Kirsch, J. L., & Kirsch, R. A. (1988). The Anatomy of Painting Style: Description with Computer Rules. *Leonardo*, *21*(4), 437. https://doi.org/10.2307/1578708

Kirsch, R. A. (1964). Computer Interpretation of English Text and Picture Patterns.

Kirsch, R. A. (1998). Using Computers to Describe Style. *Technology*, *22*, 153–160.

Knight, T. W. (1999). Applications in Architectural Design, and education and Practice, (April).

Krishnamurti, R, & Giraud, C. (1986). Towards a shape editor: the implementation of a shape generation system. *Environment and Planning B: Planning and Design*, *13*(4), 391–404. https://doi.org/10.1068/b130391

Krishnamurti, Ramesh. (1980). The arithmetic of shapes. *Environment and Planning B: Planning and Design*, *7*(4), 463–484. https://doi.org/10.1068/b070463

Krishnamurti, Ramesh. (1982). SGI: An Interpreter for Shape Grammars. *Technical Report, Centre for Configurational Studies, Design Discipline, The Open University*.

Krishnamurti, Ramesh, & Earl, C. F. (1992). Shape recognition in three dimensions. *Environment and Planning B: Planning and Design*, *19*(5), 585–603. https://doi.org/10.1068/b190585

Leach, N., & Yuan, P. F. (2018). Computational Design (p. 304). Tongji University Press Co., Ltd.

Lewis, C. H., & Rieman, J. (1994). Task-centered user interface design. *Retrieved from Http://Hcibib.Org/Tcuid/Chap-4.Html#4-1*. https://doi.org/10.1017/CBO9781107415324.004

Li, A. I.-K., Lau, A., Kuen, M., & Kuen, L. A. U. M. A. N. (2004). A set-based Shape Grammar Interpreter, with thoughts on Emergence. Retrieved from http://www.andrew.li/downloads/DCC 2004 workshop.pdf

Li, A. I. (2001). A shape grammar for teaching the architectural style of the Yingzao fashi. Retrieved from https://dspace.mit.edu/handle/1721.1/8631

Li, A. I., Chen, L., Wang, Y., & Chau, H. H. (2009). Editing Shapes in a Prototype Two- and Three-dimensional Shape Grammar Environment, (2004), 243–250.

Liddle, & David. (1996). Design of the conceptual model. *Bringing Design to Software*, 17–36. https://doi.org/10.1145/229868.230029

Lim, S., Prats, M., Chase, S. C., & Garner, S. (2003). Sketching in design sketching in design: formalising a transformational process. *Analysis*, 472–478.

Mahatody, T., Sagar, M., & Kolski, C. (2010). State of the Art on the Cognitive Walkthrough Method, Its Variants and Evolutions. *International Journal of Human-Computer Interaction*, *26*(8), 741–785. https://doi.org/10.1080/10447311003781409

Maher, M. Lou. (2008). *Creativity, Computation, and Interaction. University of Sydney and National Science Foundation*.

Martins, L. C. G., & Garcia, R. E. (2015). Validation of User Interface Model: a Systematic Literature Review. *International Conference Software Engineering Research and Practice* , (August), 145–151.

Mayer, R., & Turkienicz, B. (1998). Cognitive Process , Styles and Grammars. *Architecture*, 529–536.

McCormack, J., & D’Inverno, M. (2012). *Computers and creativity*. Springer.

McCormack, J. P., & Cagan, J. (2006). Curve-based shape matching: supporting designers’ hierarchies through parametric shape recognition of arbitrary geometry. *Environment and Planning B: Planning and Design*, *33*(4), 523–540. https://doi.org/10.1068/b31150

McCormack, J. P., Cagan, J., & Vogel, C. M. (2004). Speaking the Buick language: capturing, understanding, and exploring brand identity with shape grammars. *Design Studies*, *25*(1), 1–29. https://doi.org/10.1016/S0142-694X(03)00023-1

Mcgill, M. C. (2001a). A visual Approach for Exploring Computational Design.

Mcgill, M. C. (2001b). Shaper 2d. *Interface*.

Mcgill, M. C. (2001c). Shaper2D Visual Software for Learning Shape Grammars, 1–4.

Mckay, A., Chase, S., Shea, K., Chau, H. H., Hau, A., & Chau, H. H. (2012). Spatial grammar implementation: From theory to useable software. *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, *26*(2), 143–149. https://doi.org/10.1017/S0890060412000042

McLean, A., & Wiggins, G. (2012). Computer Programming in the Creative Arts. In *Computers and Creativity* (pp. 235–252). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-31727-9_9

Menges, A., & Ahlquist, S. (2011). *Computational Design Thinking: Computation Design Thinking*. Wiley.

Michalos, A. C., & Simon, H. A. (2006). *The Sciences of the Artificial*. *Technology and Culture* (Vol. 11). https://doi.org/10.2307/3102825

Minsky, M. (Marvin L. (1967). *Computation : finite and infinite machines*. Prentice-Hall.

Mitchell, W. J. (1975). The theoretical foundation of computer-aided architectural design. *Environment and Planning B: Planning and Design*, *2*(2), 127–150. https://doi.org/10.1068/b020127

Molich, R., & Nielsen, J. (1990). Improving a Human- Computer Dialogue. *Communications of the ACM*, *33*(3), 338–348. https://doi.org/10.1145/77481.77486

Myers, B. A. (1992). State of the Art in User Interface Software Tools, (7597).

Myers, B., & Ko, A. (2009). The Past , Present and Future of Programming in HCI. *Human Factors*. Retrieved from http://repository.cmu.edu/isr

Némery, A., & Brangier, E. (2014). Set of Guidelines for Persuasive Interfaces: Organization and Validation of the Criteria. *Journal of Usability Studies*, *9*(3), 105–128. Retrieved from http://www.usabilityprofessionals.org/upa_publications/jus/2014may/JUS_Nemery_May_2014.pdf

Nielsen, J., & Mack, R. L. (1994). *Usability inspection methods*. Wiley. Retrieved from https://www.nngroup.com/books/usability-inspection-methods/

Nielsen, M., Störring, M., Moeslund, T. B., & Granum, E. (2003). A procedure for developing intuitive and ergonomic gesture interfaces for HCI.

Norman, D. (1988). *The Design of Everydau Things*. Basic Books. Retrieved from www.basickbooks.com

Norman, D. A. (1998). The Design Of Everyday Things. Basic Books. https://doi.org/10.2307/1423268

Novick, D. G., Hollingsed, T., & Martin, L. (2007). Usability inspection methods after 15 years of research and practice after 15 Years of Research and Practice.

Oulasvirta, A., & Hornbæk, K. (2016). HCI Research a s Problem - Solving.

Petrovic, D. (2012). Lecture on Usability, GUI Design and Principles. *San Francisco State University*, *6*(2), 103.

Piazzalunga, U., & Fitzhorn, P. (1998). Note on a three-dimensional shape grammar interpreter. *Environment and Planning B Planning and Design*, *25*(1), 11–30. https://doi.org/10.1068/b250011

Polson, P. G., Lewis, C., Riemen, J., & Wharton, C. (1991). Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces.

Post, E. L. (1943). Formal Reductions of the General Combinatorial Decision Problem. *American Journal of Mathematics*, *65*(2), 197. https://doi.org/10.2307/2371809

Reffat, R. M. (2003). Architectural Exploration and Creativity using Intelligent Design Agents. *ECAADe Digital Design*, *21*, 181–186.

Richard Harper, Tom Rodden, Y. R. and A. S. (2008). *Being in the Year 2020 Being*.

*Microsoft Research*.

Rocker, I. M. (2006). When code matters. *Architectural Design*, *76*(4), 16–25. https://doi.org/10.1002/ad.289

Rogers, Y., Sharp, H., & Preece, J. (2011). Interaction Design. *Wiley*, XXIII, 519.

Romão, L. (2005a). *A study of Illegal Houses of Lisbon Built in 1974 to 1984: From Description to Computation*. School of Architecture and Planning, MIT.

Romão, L. (2005b). SGtools - A Computer Tool for Exploring Designs with Set Grammars. *CAAD Design Futures 2005*, 53–62.

Roth, R. (2017). *User Interface and User Experience (UI/UX) Design*. *Geographic Information Science & Technology Body of Knowledge*. The Geographic Information Science & Technology Body of Knowledge. https://doi.org/10.22224/gistbok/2017.2.5

Russell, S. J., & Norvig, P. (2010). Artifficial Intelligence, A Modern Approach 3rd edition. *Series in Artificial Intelligence, Englewood Cliffs, NJ. The Knowledge Engineering Review*, *11*(1), 78. https://doi.org/https://doi.org/10.1017/S0269888900007724

Saffer, D. (2008). Designing gestural interfaces, *1*, 247. Retrieved from http://www.designinggesturalinterfaces.com/samples/interactivegestures_ch1.pdf

Scott, C., & Sumbul, A. (2005). Grammar Transformations : Using Composite Grammars to Understand Hybridity in Design With an Example from Medieval Islamic Courtyard Buildings.

Shneiderman, B. (1983). Direct Manipulation: A Step Beyond Programming Languages. *IEEE*, *8*(57–69), 57–69. https://doi.org/10.1109/MC.1983.1654471

Shneiderman, B., & Plaisant, C. (2007). *Designing the user interface : strategies for effective human-computer interaction*. Addison-Wesley. https://doi.org/10.1145/25065.950626

Simon, H. A. (1969). *The Sciences of the Artificial*. Karl Taylor Compton Lectures.

Sodiya, A. S. (2009). *User Interface Design and Ergonomics*.

Stiny, G. Ice-Ray: A Note on the Generation of Chinese Lattice Designs, Environment and

Planning B: Planning and Design § (1977). https://doi.org/10.1068/b040089

Stiny, G. Two Exercises In Formal Composition, Environment and Planning B: Planning and Design § (1976). https://doi.org/10.1068/b030187

Stiny, G. (1980). Kindergarten grammars: designing with Froebel's building gifts. *Environment and Planning B: Planning and Design*, *7*(4), 409–462. https://doi.org/10.1068/b070409

Stiny, G. Weights, Environment and Planning B: Planning and Design § (1992). https://doi.org/10.1068/b190413

Stiny, G. & Gips, J. (1972). Shape Grammars and the Generative Specification of Painting and Sculpture. *Information Processing 71 Proceedings of the IFIP Congress 1971*, *2*(71), 1460–1465. https://doi.org//citeulike-article-id:1526281

Stiny, G. & Mitchell, W. J. (1978). The Palladian Grammar. *Environment and Planning B: Planning and Design*. https://doi.org/10.1068/b050005

Strobbe, T., Meyer, R. De, & Campenhout, J. Van. (2013). A Generative Approach towards Performance-Based Design. *Computation and Performance, Proceedings of the 31st ECAADe Conference*, *2*, 627–634.

Sutherland, I. E., Blackwell, A., & Rodden, K. (1980). Sketchpad: A man-machine graphical communication system. Retrieved from http://www.cl.cam.ac.uk/

Symeonidou, I. (2018). Digital Creativity: Embracing New Technologies for Architectural Innovation (pp. 175–189). https://doi.org/10.4018/978-1-5225-3993-3.ch009

Tapia, M. (1999). A visual implementation of a shape grammar system †. *Environment and Planning*, *26*, 59–73. https://doi.org/10.1068/b260059

Tapia, M. A. (1996). From shape to style. Shape grammars: Issues in representation and computation, presentation and selection. *University of Toronto*. Retrieved from http://papers.cumincad.org/cgi-bin/works/Show?5876

Tching, J., Paio, A., & Reis, J. (2012). A Shape Grammar for Self-Built Housing. *SIGRADI 2012*, 486–490. Retrieved from http://papers.cumincad.org/data/works/att/sigradi2012_159.content.pdf

Tching, J., Reis, J., & Paio, A. (2013). Shape grammars for creative decisions in the architectural project. *CISTI*, *5*(2), 84–89. https://doi.org/10.13189/csit.2017.050206

Tching, J., Reis, J., & Paio, A. (2016). A cognitive walkthrough towards an interface model for shape grammar implementations. *Journal of Computer Science and Information Technology*, *4*(3), 92–119. https://doi.org/10.13189/csit.2016.040302

Tching, J., Reis, J., & Paio, A. (2017). Shape Grammars for Creative Decisions in the Architectural Project. *Computer Science and Information Technology*, *5*(2), 84–89. https://doi.org/10.13189/csit.2017.050206

Tching, J., Reis, J., & Paio, A. (2018). IM-sgi: an interface model for shape grammar implementations. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*. https://doi.org/10.1017/S0890060417000695

Tepavcevic, B., & Stojakovic, V. (2012). Shape grammar in contemporary architectural theory and design. *Facta Universitatis - Series: Architecture and Civil Engineering*, *10*(2), 169–178. https://doi.org/10.2298/FUACE1202169T

Terzidis, K. (2003). Expressive form : a conceptual approach to computational design. *Spon Press*, 90.

Tinkercad | Create 3D digital designs with online CAD. (n.d.). Retrieved July 9, 2018, from https://www.tinkercad.com/

Trescak, T., Rodriguez, I., & Esteva, M. (2009). General shape grammar interpreter for intelligent designs generations. *Proceedings of the 2009 6th International Conference on Computer Graphics, Imaging and Visualization: New Advances and Trends, CGIV2009*, (August 2009), 235–240. https://doi.org/10.1109/CGIV.2009.74

Trescak, Tomas, Esteva, M., & Rodriguez, I. (2009). General Shape Grammar Interpreter for Intelligent Designs Generations. *2009 Sixth International Conference on Computer Graphics, Imaging and Visualization*, 235–240. https://doi.org/10.1109/CGIV.2009.74

Trescak, Tomas, Esteva, M., & Rodriguez, I. (2010). A Virtual World Grammar for automatic generation of virtual worlds. *Visual Computer*, *26*(6–8), 521–531.

https://doi.org/10.1007/s00371-010-0473-7

Trescak, Tomas, Esteva, M., & Rodriguez, I. (2012). A shape grammar interpreter for rectilinear forms. *CAD Computer Aided Design*, *44*(7), 657–670. https://doi.org/10.1016/j.cad.2012.02.009

Tullis, T. S., & Stetson, J. N. (2004). A Comparison of Questionnaires for Assessing Website Usability ABSTRACT : Introduction. *Usability Professional Association Conference*.

Turing, A. M. (1950). Computing Machinery and Intelligence. *Source: Mind, New Series*, *59*(236), 433–460.

Turkay, D., & Baykasoglu, A. (2010). Artificial Intelligence in Design and Manufacturing. *University of Gaziantep, Industrial Engineering Department 27310 Gaziantep ABSTRACT*.

Van den Bergh, J., Meixner, G., Breiner, K., Pleuss, A., Sauer, S., & Hussmann, H. (2010). Model-driven development of advanced user interfaces, (May 2014), 4429. https://doi.org/10.1145/1753846.1754166

Wang, Y. (1998). 3D Architecture Form Synthetizer. *S. M. Arch. S., Department of Architecture, Massachusetts Institute of Technology, Cambridge, MA, USA*.

Wharton, C, Rieman, J., Lewis, C., & Polson, P. (1994). Information Processing Theory of Human-Computer Interaction.

Wharton, Cathleen, Rieman, J., Lewis, C., & Polson, P. (1993). The Cognitive Walkthrough Method: A Practitioner’s Guide. *Usability Inspection*. https://doi.org/10.1108/09685220910944731

Winston, P. H. (1993). *Artificial intelligence*. Addison-Wesley Pub. Co.

# APPENDIX A

The following examples are representative of the huge use that SG can take, and how they can be used to achieve solutions or to develop new ideas and were presented in (Joana Tching et al., 2013) and latter invited to be published in (Joana Tching et al., 2017).

Rules applied in each situation will be presented in these examples. It's also important to stress that the architect can insert some others if he wants to. We intend to show how shape grammars can be applied in common phases to the overwhelming majority of urban and building projects.

## DEFINITION OF RULES FOR BUILDINGS, PUBLIC SPACES AND CIRCULATION WAYS

There is a great variety of rules which can be defined concerning the same goals of Figure 1. These rules present orthogonal definitions for buildings and circulation ways. Other rules can be defined such as radial intention (as it can be seen in some cities) or be extrapolated to emerging buildings, through a wide range of shape alphabet relationships between them.

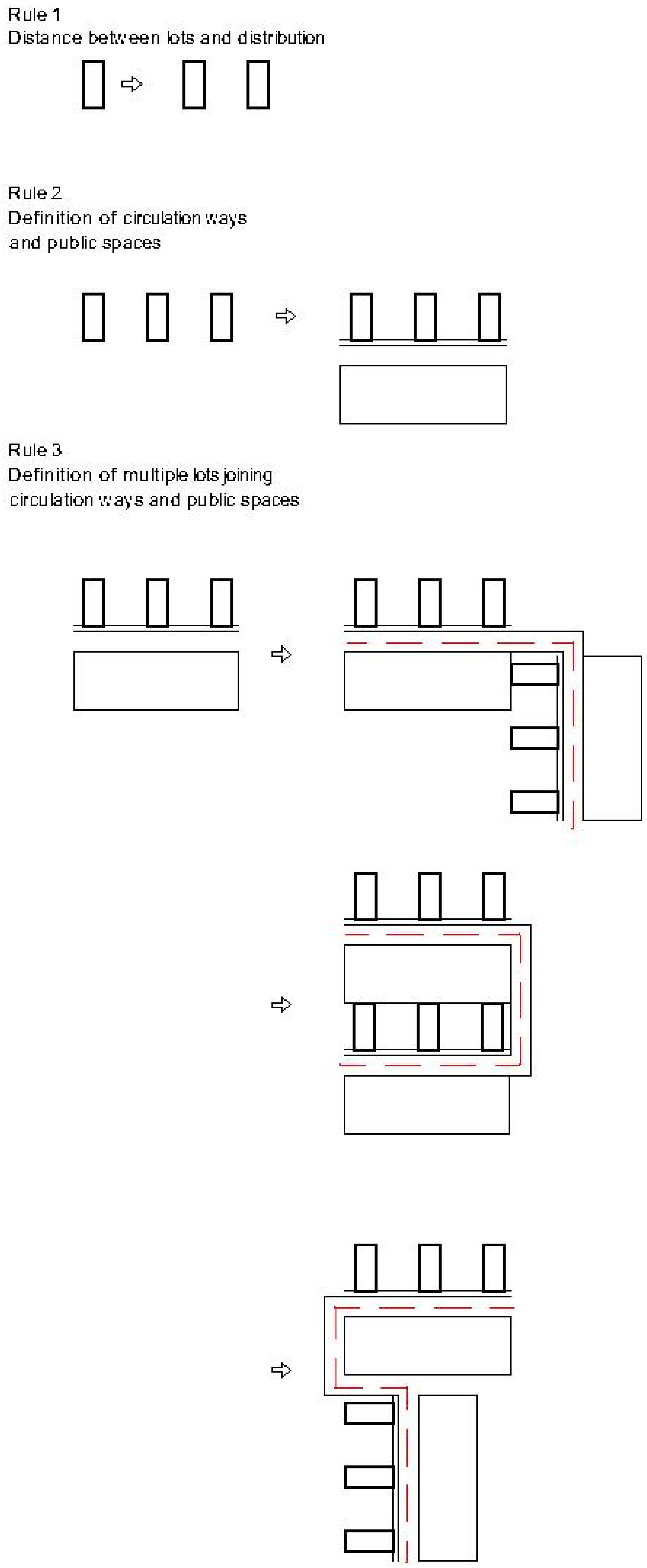

Rule 1
Distance between lots and distribution
Rule 2
Definition of circulation ways
and public spaces
Rule 3
Definition of multiple lots joining
circulation ways and public spaces

Figure 1 – Lot Distribution

## IMPLANTATION POLYGON IN THE LOT

Having a lot to build on, an implantation Polygon of the building has to obey a set of rules as mentioned above and should include choices concerning the building shape. The implantation area has to consider the number of floors of the building, the larger the number of floors, the smaller the implantation area. It should also take into account the organizational logic of the inner spaces, affecting its outer boundary line.

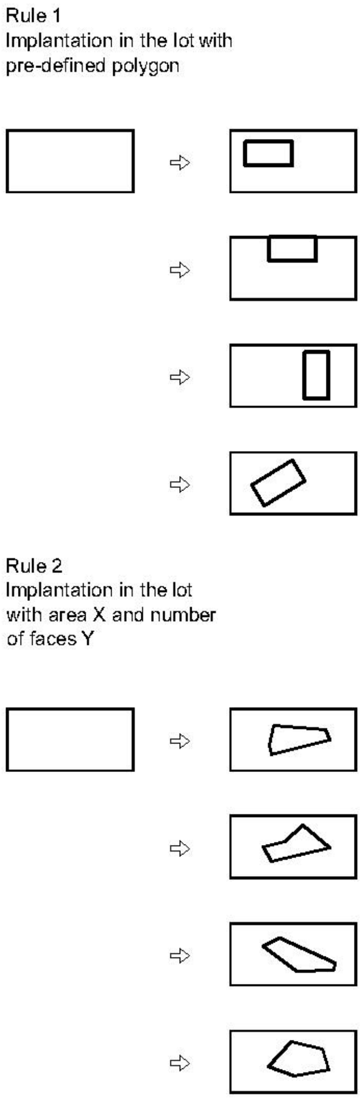

Rule 1
Implantation in the lot with pre-defined polygon
Rule 2
Implantation in the lot with area X and number of faces Y

Figure 2 – Implantation Polygon

## INTEGRATION IN THE LOT – COMPATIBILITY WITH TERRAIN CURVES

Since there are nowadays computational applications to help the architect study the best integration of the building in the lot, minimizing the movement of the top surface and facilitating the construction, the architect's goals might be defined as rules. Project interventions such as taking advantages of the landscape that can be seen from there, favoring high or low areas, and the shape of the level curves can also be a graphic source to combine lines which will define the shape of the building.

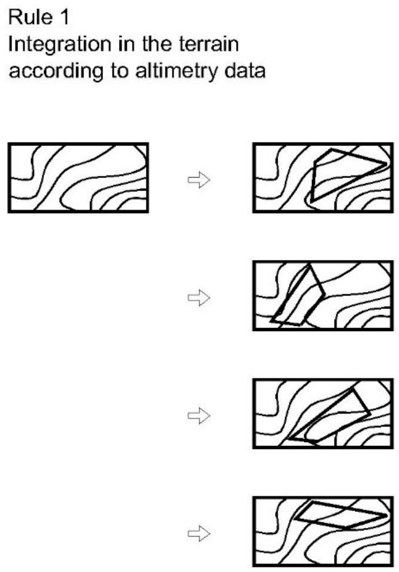


Figure 3– Integration with the terrain

## Circulation Ways

The definition of rules to plan the circulation ways, as presented in Figure 4, can be related, on a larger scale, with the definition of the lot organization and, on a smaller scale, with the definition of the inner communication areas of the building, helping its drawing outline the inner space organization.

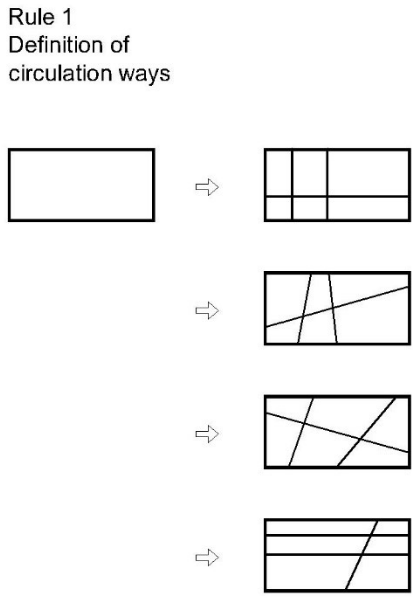


Figure 4 – Circulation ways

## GARDEN DESIGN

We can apply the rules of division of lots and those of circulation ways, using shape grammars which will guarantee that the building areas have the green spaces required and being at the same time formal and unique. In the landscaped areas rules might be defined to plan the distribution of floral and arboreal elements, which might also define aesthetic shapes.

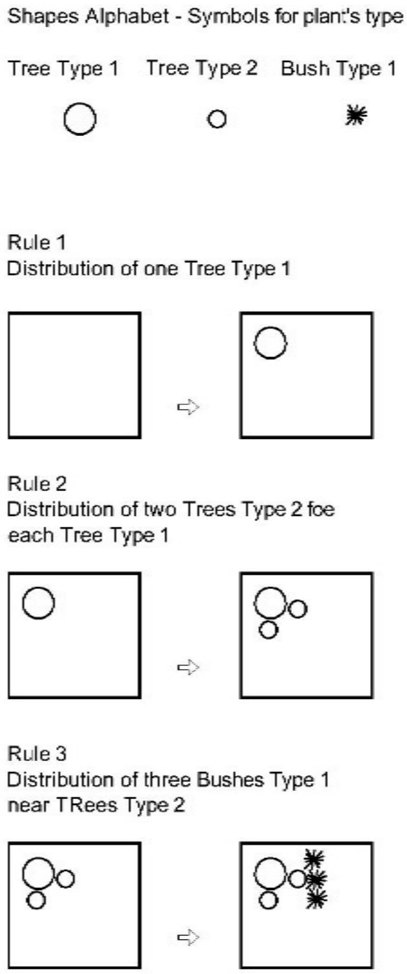

Shapes Alphabet - Symbols for plan:'s type
Tree Type 1
Tree Type 2
Bush Type 1
Rule 1
Distribution of one Tree Type 1
Rule 2
Distribution of two Trees Type 2 foe each Tree Type 1
Rule 3
Distribution of three Bushes Type 1 near TRees Type 2

Figure 5 – Garden Design

## LOT DIVISION

The division of the lots follows a wide range of legal rules such as maximum areas of land occupation, the distance between buildings around, the dimension of circulation ways, green space areas, public space areas, among others. In short, these rules together with the rules the architect intends to insert, concerning the shape of the buildings and the relationships between them may generate grammars which will set the shape distribution in the lot and at the same will define the lot occupation.

### BALCONIES DEFINITION

Being important spaces, as they connect the inner side to the outer side of the building, balconies are elements which affect the shape building and its relationship with the surroundings. Stressing the aesthetic aspects, shape grammars might define rules which will generate balcony designs on the “building shell”, matching technical rules, for instance, maximum suspended area, formal aspects such as exploiting the sights.

### FLOOR DESIGN

Floor design can change from floor to floor. It can be a regular or irregular design, depending on the geometric design we want. Geometric designs can respond to rules relating to the interior organization, to the need of distinguishing areas or, for instance, separation of flats. They might consider the balconies mentioned above and the optimization of a glassed area if people want to enjoy the surroundings.

### INTERIOR SPACE ORGANIZATION

Interior space organization depends on a wide range of rules, related to the building typology and its needs (hotel, hospital, housing, school, everyone has a set of specific rules). All the objective and aesthetic goals should be added to in other to regulate the expected shape.

### DESIGN OF FAÇADE OPENINGS

The design of the façade openings, taking the aspects of floor design mentioned above into account, might be based on rules, such as the definition of the maximum glassed area for each division of the building, for example. They might follow thermal rules, which will guarantee the solar access and shadowing, and geometric rules as well, which will lead to a certain aesthetics of the building

### FORMAL COMBINATION OF MATERIALS

The materials used on façades or in the inner spaces might be subject of graphic, formal/creative spaces, which can be defined by shape grammars.

Following the same logic of the examples described above, different project phases and issues can be found out, which can also be solved through the use of shape grammars.

The architectural project gains its identity on the idea developed by the architect and in the creativity through which he has to solve the issues. He has also to obey a wide range of rules. Overall, the architect's intentions are per se rules, which are not imposed by technical and legal needs of the project but rather by the artist's aesthetic and creative intentions.

Dividing the project into its elementary parts, we see that the architect elects, consciously or intuitively, a set of rules and makes choices which are responsible for the final work. This is the reason why shape grammars can explain design styles, once the rules, which generate a certain shape, are recognized.

Nowadays it's common practice to use computational applications, which reproduce the architect's manual design, among other technical aspects of the project (such as automatic measurements, thermal simulations, three-dimensional visualization, etc.). We anticipate it will be possible to make the architect see in shape grammars and in its applications a common practice and a way of optimizing his ideas, using not only Computational Design but also Computational Creativity.

# Appendix **B – IM-SGI CRITERIA**

| Ergonomic Criteria | EC Abbreviations | EC Definition | SG Tasks | IM-sgi Criteria Abbreviations | IM-sgi Criteria Definitions. IM-sgi – Interface Model for Shape Grammar Implementations |
|---|---|---|---|---|---|

Table B1. Compatibility

| | | | | | |
|---|---|---|---|---|---|
| **Compatibility** | **COMP** | Match between users' characteristics (memory, perceptions, customs, skills, expectations, etc.) and task characteristics. | General | SG- COMP | A) The Users considered by IM-sgi are professionals in the artistic field, mainly architects and designers. Considering these users, three types were distinguished:<br>STUDENTS – students of Architecture or Design that will have exploration objectives for the use of SG. These users shall be able to use the SG implementation for educational purposes, exploring and learning SG with the use of the tool.<br>ARTISTS/DESIGNERS – Artists, Architects and Designers that will use SG for the search and definition of project solutions. These users shall be able use the SG implementation as a tool for to develop their projects.<br>EXPERTS – SG experts that will be able to program the SG and the SG implementation in order to allow it to create newly desired results. These experts can be from the artistic field or computer science field.<br>B) As the target users are characteristically users of CAD software, formats, tools, labels, messages or instructions shall follow conventional CAD software that are perceived by users as being familiar. CAD software has highly sophisticated interfaces that shall be a source for these definitions.<br>C) A Graphical User Interface (GUI) shall be used and a permanent visualization of the results of the actions taken shall exist. All saved shapes, rules or SG results shall have a graphic representation.<br>D) Units of measurement shall be metric and/or imperial. |
| | | | Task 1<br>Shape Creation | SGSC- COMP | A) An Initial Shape shall be able to be created by drawing tools and with clear unit measurements as in conventional CAD software, by drawing in a window with zoom and pan abilities.<br>B) According to professional needs, importation of shapes to be used or importation of background images to serve as a guide to the drawing shall be available.<br>C) The use of a command-line for drawing shall be available for most experienced users. |

Table B1. Compatibility (cont.)

| | | | | | |
|---|---|---|---|---|---|
| **Compatibility** | **COMP** | Match between users' characteristics (memory, perceptions, customs, skills, expectations, etc.) and task characteristics. | Task 2<br>Rule Creation | SGRC- COMP | A) Rules shall be able to be created by drawing tools and with clear unit measurements as in conventional CAD software, by drawing in a window with zoom and pan abilities.<br>B) According to professional needs, importation of shapes to be used as rules or importation of background images to serve as guides to the drawing shall be available.<br>C) The use of a command-line for drawing shall be available for most experienced users. |
| | | | Task 3<br>SG Application | SGAPP- COMP | A) SG results shall be able to be applied and visualized in a graphic window with zoom and pan abilities.<br>B) According to professional needs, importation of background images to merge and work with the SG results shall be available.<br>C) The exportation of SG results in compatible formats with conventional CAD software shall exist, allowing the use of SG for professional purposes. |
| | | | Task 4<br>SG Manipulation | SGMAN- COMP | A) SG results shall be able to be manipulated, cycling through different alternatives and changing iterations number<br>B) The manipulation of the results could be done by directly manipulating the drawing according to creativity purposes in the implementation itself. |
| | | | Task 5<br>SG Alteration | SGALT- COMP | A) Users shall be able to manipulate any previous state of the SG, from Initial Shape to its Rules and results.<br>B) Any state and alteration shall be able to be saved or exported to be used later or to be developed in CAD software. |

Table B2. Adaptability - Users' Experience Management

| | | | | | | |
|---|---|---|---|---|---|---|
| **Adaptability - Users' experience management** | **USEX** | Means available to take into account the level of user experience. Experienced and inexperienced users have different information needs. | General | Users' experience management | SG- USEX | A) The SG interface shall be prepared to adjust to three types of users, according to the definition made in this investigation. This adaptation can be achieved giving the user the chance to choose the level of expertise when opening the application.<br>B) A Beginner Mode shall be available, focused for Students of SG, with high-automated control by the implementation. This mode shall have mainly drawing tools available and with immediate visualization of any action made.<br>C) An Intermediate Mode shall be available, focused on Artists/Designers, allowing the users to have a good control of the implementation and use it with possible integration with other CAD software, so that SG can be defined to solve design projects and its results used for further design project development/detailing.<br>D) An Expert Mode shall be available, focused on SG Experts allowing the user to have full control over the implementation and even changing its code so that it can create SG in new or different ways. The user shall have full control over the application with lower or no levels of automation. |
| | | | Task 1<br>Shape Creation | Task 1<br>Users' experience management | SGSC- USEX | A) In the Beginner Mode, an Initial Shape shall be created using pre-defined shapes, so that the automation is higher. Drag-and-drop actions are preferable.<br>B) In the Intermediate Mode, an Initial Shape shall be created recurring in drawing tools that resemble conventional CAD software.<br>C) In the Expert Mode, an Initial Shape shall have the possibility to be created by code, ideally in a command-line. |
| | | | Task 2<br>Rule Creation | Task 2<br>Users' experience management | SGRC- USEX | A) In the Beginner Mode, Rules shall be created using pre-defined shapes, so that the automation is higher. Drag-and-drop actions are preferable.<br>B) In the Intermediate Mode, Rules shall be created recurring in drawing tools that resemble conventional CAD software. There shall be the possibility to create several rules for one same SG and to be able to visualize them and reorder them. In this mode, Labels to complex Rules shall be available.<br>C) In the Expert Mode, Rules shall have the possibility to be created by code, ideally in a command-line. The use of labels and the general organization of the SG rules shall be able to be defined directly by code. |
| | | | Task 3<br>SG Application | Task 3<br>Users' experience management | SGAPP- USEX | A) In the Beginner Mode, a pre-defined Iterations number could be applied to give immediate visualization of the SG created while manipulating the Rules. User shall also be able to increase or decrease Iterations number with a simple button, without entering values.<br>B) In the Intermediate Mode, the Iterations number shall be entered manually.<br>C) In the Expert Mode, the Iterations number shall be entered manually or by code, ideally in a command-line. |
| | | | Task 4<br>SG Manipulation | Task 4<br>Users' experience management | SGMAN- USEX | A) In the Beginner Mode, a small number of alternatives could be seen, preferably by cycling them in the SG window.<br>B) In the Intermediate Mode, a good number of alternatives shall be visualized, preferably in a Drop-down menu or in a new window and dragged to the SG window for substitution.<br>C) In the Expert Mode, the results shall be able to be controlled by code, ideally in a command-line, allowing the application of alternatives. |

Table B3. Adaptability - Flexibility

| | | | | | |
|---|---|---|---|---|---|
| **Adaptability-Flexibility** | **FLEX** | EC Definition: Means available to the users to customize the interface in order to take into account their working strategies and/or their habits, and the task requirements.<br>Flexibility is reflected in the number of possible ways of achieving a given goal. | General | SG- FLEX | A) SG implementation interface shall take into account conventional CAD software, as the target users are mainly from Architecture and Design fields, allowing the user to organize menus, toolboxes, windows or others to resemble the software they are familiar with.<br>B) Users shall be able to use Buttons, Menus or Commands, or perform the same action, using what suits their habits.<br>C) Users shall be able to choose windows size, accordingly to their drawing or visualization needs. |
| | | | Task 1<br>Shape Creation | SGSC- FLEX | A) Initial Shape window shall allow resizing, facilitating the drawing actions.<br>B) Drawing commands can be available using drawing tools, using pre-defined shapes selected or dragged to the window or by a command-line.<br>C) In Expert Mode, user shall be able to define Rules prior to Initial Shape drawing instead of following the general sequence of tasks for SG creation. |
| | | | Task 2<br>Rule Creation | SGRC- FLEX | A) Rules window can allow resizing, facilitating drawing actions.<br>B) Drawing commands can be available using drawing tools, using pre-defined shapes selected or dragged to the window or by a command-line.<br>C) Label tools shall be available, dragging symbols to the drawing or inserting them with button.<br>D) In Expert Mode, User shall be able to define Rules prior to Initial Shape drawing instead of following the general sequence of tasks for SG creation. |
| | | | Task 3<br>SG Application | SGAPP- FLEX | A) SG window shall have a big presence in the screen and allow resizing.<br>B) User shall be able to choose the iterations number by adding the specific number or cycling through pre-defined numbers. The system can also have a pre-defined number applied automatically that can be changed. |
| | | | Task 4<br>SG Manipulation | SGMAN- FLEX | A) The manipulation of alternatives shall be possible according to SGAPP- FLEX.<br>B) User shall be able to choose to see alternatives in a new window or cycle through them directly in the SG window, or Drop-down menu or even textually in a command-line. |
| | | | Task 5<br>SG Alteration | SGALT- FLEX | A) The user choices shall follow SGSC-FLEX, SGRC- FLEX and SGAPP- FLEX. |

Table B4. Significance of Codes

| | | | | | |
|---|---|---|---|---|---|
| **Significance of codes** | **SICO** | Relationship between a term and/or a sign and its reference.Codes and names are significant to the users when there is a strong semantic relationship between such codes and the items or actions they refer to. | General | SG- SICO | A) All window titles shall be clearly related to the tasks and results that will be available.<br>B) Any abbreviations shall be clear and limited to the strictly necessary.<br>C) Menu and Button naming shall follow conventional CAD software and be clearly related to the actions.<br>D) Codes to be used in command-lines shall be meaningful and allow the clear definition of the actions. |
| | | | Task 1<br>Shape Creation | SGSC- SICO | A) The term "INITIAL SHAPE" shall be used in the title of the window for this purpose.<br>B) This term shall be used for any message related to Initial Shape definition.<br>C) Drawing tools shall have clear names and follow conventional CAD software. |
| | | | Task 2<br>Rule Creation | SGRC- SICO | A) The term "RULES" shall be used in the title of the window for this purpose.<br>B) This term shall be used for any message related to Rule definition.<br>C) Drawing tools available for Rule creation shall have the same naming as the ones for Initial Shape Drawing. |
| | | | Task 3<br>SG Application | SGAPP- SICO | A) The term "SHAPE GRAMMAR" shall be used in the title of the window for this purpose.<br>B) This term shall be used for any message related to Shape Grammar application.<br>C) The term "ITERATIONS" shall be used for the Data entry field for Iterations application. |
| | | | Task 4<br>SG Manipulation | SGMAN- SICO | A) A term related to "ALTERNATIVES" or "RESULTS" shall be used for menus or windows that allow the choice of Shape Grammar alternatives. |
| | | | Task 5<br>SG Alteration | SGALT- SICO | Not applicable |

Table B5. Consistency

| | | | | | |
|---|---|---|---|---|---|
| **Consistency** | **CONS** | **EC Definition:** The way interface design choices (codes, naming, formats, procedures, etc.) are maintained in similar contexts, and are different when applied to different contexts. | General | G- CONS | A) Window titles and formats shall follow the same format and placement in all windows.<br>B) The screen format shall be identical in all expertise modes, adapting itself to the specifications of each one.<br>C) Menu options, buttons and command-lines shall be identical in all different windows and expertise modes.<br>D) All messages shall follow the same format and position on the screen. |
| | | | Task 1<br>Shape Creation | SGSC- CONS | A) Initial Shape window and tools shall be identical and recognizable in all expertise modes, independently of the additional tools in each one. |
| | | | Task 2<br>Rule Creation | SGRC- S CONS | A) Rules window and tools shall be identical and recognizable in all expertise modes, independently of the additional tools in each one. |
| | | | Task 3<br>SG Application | SGAPP- CONS | A) SG window and tools shall be identical and recognizable in all expertise modes, independently of the additional tools in each one. |
| | | | Task 4<br>SG Manipulation | SGMAN- CONS | Not applicable |
| | | | Task 5<br>SG Alteration | SGALT- CONS | Not applicable |

Table B6. Immediate Feedback

| | | | | | |
|---|---|---|---|---|---|
| **Guidance - Immediate Feedback** | **FEED** | System responses to users' actions.<br>Feedback quality and rapidity help the user have confidence and satisfaction as well as a good understanding of the dialogue. | General | SG- FEED | A) SG implementation shall have a GUI that allows immediate visualization of created shapes, created rules and SG results and alternatives.<br>B) A graphical visualization of the created rules, either by drawing or data entry shall be immediately presented.<br>C) SG Iterations shall be immediately graphically visualized, along with a number of alternatives.<br>D) Undo and Redo tools shall be available and its results immediately visualized.<br>E) Messages guiding the user shall be delivered. |
| | | | Task 1<br>Shape Creation | SGSC- FEED | A) Initial Shape window shall allow the graphic design with drawing tools, similar to conventional CAD software.<br>B) Drawing tools shall be clearly identified and its results immediately visualized when used.<br>C) Any change made to the shapes shall be easily reversible and all the results immediately visualized.<br>D) If a command-line exists to define shapes, the results shall be graphically seen in the Initial Shape window. |
| | | | Task 2<br>Rule Creation | SGRC- S FEED | A) Rules window shall allow the graphic design with drawing tools, similar to conventional CAD software.<br>B) Drawing tools shall be clearly identified and its results immediately visualized when used.<br>C) Any change made to the shapes and rules shall be easily reversible and all the results immediately visualized.<br>D) The system shall allow saving and ordering the rules that will compose the SG and allow the user to easily visualize them and manipulate them, preferably in a dedicated Window or Drop-down menu with visualizations of the composed rules.<br>E) If a command-line exists to define rules, the results shall be graphically seen in the Rules window. |
| | | | Task 3<br>SG Application | SGAPP- FEED | A) The result of the number of Iterations applied shall be immediately seen in the SG window.<br>B) The system shall graphically show alternatives of the SG with the same number of Iterations, preferably in a dedicated Window or Drop-down menu. |
| | | | Task 4<br>SG Manipulation | SGMAN- FEED | A) The Iterations tool shall allow immediate change of the number, with immediate visualization of the new result.<br>B) The system shall graphically show alternatives of SG with the same number of Iterations, preferably in a dedicated Window or Drop-down menu, which can be selected and worked on. |
| | | | Task 5<br>SG Alteration | SGALT- FEED | A) If changes are made in the Initial Shape, the new results shall be immediately graphically visualized.<br>B) If changes are made in the Rules, the new results shall be immediately graphically visualized.<br>C) Any alteration shall be easily undone or redone and its results visually seen. |

Table B7. Guidance - Grouping and Distinction of Items by Location

| | | | | | |
|---|---|---|---|---|---|
| **Guidance - Grouping and distinction of items by location** | **GDLO** | Relative positioning of items organizing them according to their class or distinction.<br>Users will detect the different items easily if they are correctly, also helping learning and remembering of items. | General | SG-GDLO | A) Three main drawing windows must exist, organized by order:<br>1 – INITIAL SHAPE or equivalent; 2 – RULES or equivalent; 3 – SHAPE GRAMMAR or equivalent<br>They shall be organized linearly, left to right or up to down, or combinations of these, according to reading conventions.<br>B) Any extra windows must be organized according to above reading conventions, next to the related window.<br>C) The menu options to be used in each drawing window must be organized according to the object they apply to and according to the order of commands to be performed. |
| | | | Task 1<br>Shape Creation | SGSC- GDLO | A) Window for Initial Shape creation must be logically shown as the first to be used. All drawing tools for the Initial Shape drawing must be available in that window, in buttons or menus.<br>B) Any new windows or menus that relate to Initial Shape must be gathered with the main Initial Shape window.<br>C) If a command-line exists for the Initial Shape creation, it shall be located on the bottom of the Initial Shape window. |
| | | | Task 2<br>Rule Creation | SGRC- GDLO | A) Window for Rule creation must be shown as logically the second one to be used. All the tools for the Rule definition must be available in that window, in buttons or menus.<br>B) A Window shall exist to show the list of existing rules, next to the Rule main window. Exact same logic if instead of a window for this purpose, a Drop-Down menu is opened.<br>C) Any new windows or menus that relate to Rule creation must be gathered within the main Rule Window.<br>D) If a command-line exists for the Rule creation, it shall be located on the bottom of the Initial Shape window. |
| | | | Task 3<br>SG Application | SGAPP- GDLO | A) Window for SG application must be logically shown as the third one to be used. All the tools for the SG application must be available in that window, in buttons or menus.<br>B) The button or data field to add the Iterations number, to apply the SG, must be clearly situated in the SG window.<br>C) Any new windows or menus that relate to SG application must be gathered with the main SG window. |
| | | | Task 4<br>SG Manipulation | SGMAN- GDLO | A) SG manipulation made by changing de Iterations number, must follow SGAPP- GDLO- B.<br>B) A window or drop-down menu shall exist to show SG alternatives to be selected. This must be gathered with the main SG window.<br>C) Tools related to save or export results shall be clearly located near the SG window. |
| | | | Task 5<br>SG Alteration | SGALT- GDLO | A) SG alteration made by changing de Initial Shape, shall be clear by the application of SGSC- GDLO<br>B) SG alteration made by changing Rules, shall be clear by the application of SGRC- GDLO |

Table B8. Guidance - Grouping and Distinction of Items by Format

| | | | | | |
|---|---|---|---|---|---|
| **Guidance - Grouping and distinction of items by format** | **GDFO** | Graphical features (format, color, etc.) that distinguish items of a given class.<br>It will be easier for the user to recognize relationships or distinctions between items by their similarities or differences. | General | SG- GDFO | A) Drawing windows for shapes and rule creation shall be graphically similar<br>B) Drawing tools shall be the same in Shape and Rule window, so they can be easily recognized<br>C) SG windows shall be graphically identifiable as the main results and manipulation window<br>D) Secondary windows for Rules list or SG alternatives shall be graphically similar to be identifiable as secondary and have the same position relatively to the main window they refer to. If they are shown by a Drop Down menu, they shall be also similar graphically and in position in the window. |
| | | | Task 1<br>Shape Creation | SGSC- GDFO | A) All drawing tools for shape creation in the Initial Shape window shall be similar, either in a List Menu or in Buttons. These shall also be similar to conventional drawing tools CAD software.<br>B) If a command-line exists for shape definition, it shall be graphically identifiable as this kind of tool |
| | | | Task 2<br>Rule Creation | SGRC- GDFO | A) All drawing tools for shape creation in the Rule window shall be similar, either in a List Menu or in Buttons. These shall also be similar to conventional drawing tools in CAD software.<br>B) If a command-line exists for rule definition, it shall be graphically identifiable as this kind of tool<br>C) A secondary window for the list of Rules that compose the grammar shall be graphically identifiable as this kind of element. The same way if it is shown in a Drop-down menu. |
| | | | Task 3<br>SG Application | SGAPP- GDFO | A) The button or data field to add the Iterations number shall be graphically relevant and predominant in the SG window |
| | | | Task 4<br>SG Manipulation | SGMAN- GDFO | A) A secondary window for a list of SG alternatives shall be graphically identifiable as this kind of element. The same way if it is shown in a Drop-down menu. |
| | | | Task 5<br>SG Alteration | SGALT- GDFO | A) All windows, menus and buttons shall be graphically identifiable according to SGSC- GDFO and SGRC- GDFO so that it is easy to identify the procedures to change Initial Shape or Rules of the SG. |

Table B9. Guidance - Prompting

| | | | | | |
|---|---|---|---|---|---|
| **Guidance - Prompting** | **PROM** | Guide users through the alternatives when several actions are possible.<br>Inform about the actual state or context of the system.<br>Saves users from learning many commands and helps navigate the system easily. | General | SG-PROM | A) All Windows clearly named, numbered according to order of use, if needed.<br>B) For data field entries, display associated label.<br>C) Display measurement units when drawing tools are used.<br>D) Provide drawing status information (shape measurements, shape color, thickness, filling, shape geometry [opened or closed])<br>E) Provide help tools |
| | | | Task 1<br>Shape Creation | SGSC-PROM | A) Window for Initial Shape Creation name: 1 – INITIAL SHAPE or equivalent<br>B) Display only available actions and in order of use: drawing tools for shape drawing, editing and saving.<br>C) If drawing is made by data entry, provide required formats and accepted values. |
| | | | Rule Creation | SGRC-PROM | A) Window for Rule Creation name: 2 – RULES or equivalent<br>B) Display only available actions: drawing tools for shape drawing, editing and saving.<br>C) If drawing is made by data entry, provide required formats and accepted values<br>D) Provide Drop-down Menu, List or Window with saved rules that will compose the SG |
| | | | Task 3<br>SG Application | SGAPP-PROM | A) Window for SG Application name: 3 – SHAPE GRAMMAR or equivalent<br>B) Display only available actions: number of iterations to be used, editing and saving.<br>C) Provide the required format and acceptable values for the iterations number to apply.<br>D) Provide Drop-down Menu, List or Window with SG alternatives |
| | | | Task 4<br>SG Manipulation | SGMAN-PROM | A) Display only available actions: changing number of iterations, choosing different alternatives, edit, save and export. |
| | | | Task 5<br>SG Alteration | SGALT-PROM | A) Display only available actions: add/change Initial Shape; add/change rule or rules order. |

Table B10. Guidance - Legibility

| | | | | | |
|---|---|---|---|---|---|
| **Guidance - Legibility** | **LEGI** | Lexical characteristics of the information presented on the screen that may hamper or facilitate the reading of this information. | General | SG- LEGI | A) Window names shall be all equally formatted and aligned, preferably with upper case letters and aligned on the left<br>B) Menus in all the windows shall be distributed with the same inter-word spacing and preferably with upper case letters.<br>C) Tools and Menus shall follow conventional CAD software. |
| | | | Task 1<br>Shape Creation | SGSC- LEGI | A) Drawing tools shall have clear icons if they are buttons, and preferably, have an associated label.<br>B) Drawing window shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa. A scalable grid can help understand shapes relations.<br>C) If a command-line exists to enter shape definitions, the text shall have a sans serif font and with a good readable size. |
| | | | Task 2<br>Rule Creation | SGRC- LEGI | A) Drawing tools shall have clear icons if they are buttons, and preferably, have an associated label.<br>B) Drawing window or Rules list window shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa. A scalable grid can help understand shapes relations.<br>C) If a command-line exists to enter shape definitions, the text shall have a sans serif font and with a good readable size. |
| | | | Task 3<br>SG Application | SGAPP- LEGI | A) SG window shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa. A scalable grid can help understand shapes relations.<br>B) If a command-line exists to enter shape definitions, the text shall have a sans serif font and with a good readable size.<br>C) Iterations Button or Data field shall have a velar sans serif font and with good readable size. |
| | | | Task 4<br>SG Manipulation | SGMAN- LEGI | A) If a SG alternatives' window exists, it shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa.<br>B) Iterations button shall be as SGAPP- LEGI- C. |
| | | | Task 5<br>SG Alteration | SGALT- LEGI | A) Tools and Menus needed to manipulate the SG shall be according to SGSC- LEGI, SGRC- LEGI and SGAPP- LEGI. |

Table B11. User Workload – Brevity - Concision

| | | | | | |
|---|---|---|---|---|---|
| **User workload – Brevity - Concision** | **CONC** | Perceptual and cognitive workload for individual inputs or outputs. The shorter the entries, the less memorization that is needed. | General | SG- CONC | A) Possibly existing Data field shall allow exclusively accepted values.<br>B) If a measurement unit is associated with a particular data field for drawing, include that unit as part of the field label.<br>C) If codes are used in a command-line to activate tools, use short codes with no more than 5 characters.<br>D) In icon buttons, allow labels with the description to appear when hovering the cursor. |
| | | | Task 1<br>Shape Creation | SGSC- CONC | A) Drawing tools for Initial Shape drawing shall be the only ones available directly on the Initial Shape window.<br>B) If a command-line is used, drawing commands shall be small and clear. Measurements shall automatically have the measurement unit, instead of making the user add them. |
| | | | Task 2<br>Rule Creation | SGRC- CONC | A) Drawing tools for Rule drawing shall be the only ones available directly on the Rule window and be similar to the ones to create Initial Shapes.<br>B) If a command-line is used, commands shall be small and clear. Measurements shall automatically have the measurement unit, instead of making the user add them.<br>C) The window or Drop-down menu with list of rules shall allow them to be selected or re-ordered by simple click and drag with the cursor. |
| | | | Task 3<br>SG Application | SGRC- CONC | A) Iterations Button or Data field shall accept numbers exclusively and allow only the length acceptable by the implementation. |
| | | | Task 4<br>SG Manipulation | SGMAN- CONC | A) The window or Drop-down menu with alternatives shall allow them to be selected by simple click with the cursor. |
| | | | Task 5<br>SG Alteration | SGALT- CONC | A) The tools to be used shall follow SGSC- CONC, SGRC- CONC SGAPP- CONC and SGMAN- CONC. |

Table B12. User Workload – Brevity - Minimal Actions

| | | | | | |
|---|---|---|---|---|---|
| **User workload – Brevity - Minimal actions** | **MIAC** | Workload with respect to the number of actions necessary to accomplish a goal or a task.<br>Limiting le the steps users must go through to reach a goal decreases the risks of making errors. | General | SG- MIAC | A) Reduce to the minimum necessary actions to create a SG.<br>B) Reduce to the minimum the codes to be used in command-lines. |
| | | | Task 1<br>Shape Creation | SGSC- MIAC | A) For the creation of the Initial Shape, only drawing tools shall be needed and they shall immediately be available in the Rules window.<br>B) Manipulation of the shapes shall be done by click and drag with the cursor.<br>C) If a command-line exists for the creation of shapes, these shall be easily created by simple commands with dimensions' introduction.<br>D) Saving and editing tools shall be distinguished as secondary tools. |
| | | | Task 2<br>Rule Creation | SGRC- MIAC | A) For the creation of Rules, only drawing tools shall be needed and they shall be immediately available in the Rules window.<br>B) Manipulation of the shapes shall be done by click and drag with the cursor.<br>C) The Initial shape shall automatically appear in the Rules window to void repeating its drawing.<br>D) When a SG is made from a list of rules, allow the creation of a new rule starting from the drawing of the previous one<br>E) The insertion of Labels shall be done by single commands, preferably a button with the label to be used.<br>F) If a command-line exists for the creation of shapes, these shall be easily created by simple commands with dimensions' introduction.<br>G) Saving a rule to the list shall be done by a single command that will add it to the Rules list |
| | | | Task 3<br>SG Application | SGAPP- MIAC | A) A simple button or data field shall be needed to indicate the Iterations number and apply the SG, and no other command shall appear as relevant in the SG window.<br>B) Saving a result shall be done by a single command. |
| | | | Task 4<br>SG Manipulation | SGMAN- MIAC | A) Manipulation of the SG shall be easily done by changing the Iterations number, accordingly to SGAPP- MIAC- A.<br>B) Choosing alternatives shall be done simply by click and drag with the cursor. |
| | | | Task 5<br>SG Alteration | SGALT- MIAC | A) The tools to be used shall follow SGSC- MIAC, SGRC- MIAC SGAPP- MIAC and SGMAN- MIAC. |

Table B13. User Workload - Information Density

| | | | | | |
|---|---|---|---|---|---|
| **User Workload - Information density** | **INDE** | Users' workload from a perceptual and cognitive point of view with regard to the whole set of information presented to the users rather than each individual element or item. Items that are not related to the task should be removed. | General | SG- INDE | A) The implementation display shall exclusively show the windows referred in SG-GLO- A.<br>B) Other windows shall be presented as secondary or opened only when needed.<br>C) Command-lines shall be integrated in the bottom of the corresponding window.<br>D) Allow shapes, rules and results to be permanently visible, so the user doesn't have to memorize and give them proper emphasis in the rest of the environment.<br>E) Computation needed for the SG application shall be automatic and without any calculated entry done manually by the user. |
| | | | Task 1<br>Shape Creation | SGSC- INDE | A) Initial Shape window shall be a simple drawing area with only drawing tools available.<br>B) A command-line for shape creation can be shown only when needed, in the bottom of the Rules window. |
| | | | Task 2<br>Rule Creation | SGRC- INDE | A) Rules window shall be a simple drawing area with drawing tools available and a save command to create list of rules for SGs with more than one rule.<br>B) A window with the graphical visualization of the rules can be opened when needed, instead of being always visible.<br>C) A command-line for rule creation can be shown only when needed, in the bottom of the Rules window. |
| | | | Task 3<br>SG Application | SGAPP- INDE | A) SG window shall be a simple drawing area with the iterations number tool as main command. |
| | | | Task 4<br>SG Manipulation | SGMAN- INDE | A) A window with the graphic visualization of alternatives can be opened when needed instead of being always visible |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  |  |  | Task 5 – SG Alteration | SGALT- INDE | A) The tools to be used shall follow SGSC- INDE, SGRC- INDE SGAPP- INDE and SGMAN- INDE. |

Table B14. Explicit User Actions

| | | | | | |
|---|---|---|---|---|---|
| **Explicit user actions** | **EXUA** | Relationship between the computer processing and the actions of the users.<br>The computer must process only the actions requested by the users and only when requested to do so. | General | SG- EXUA | A) Request from the user for a clear Enter action to initiate the SG process.<br>B) Request from the user for a clear click on Menu and Drop-down menu choices.<br>C) Request from the user for a clear Enter action when using command-line entries. |
| | | | Task 1<br>Shape Creation | SGSC- EXUA | A) When defining the Initial Shape, request a Finish command to declare that the drawing is finished and that the next task can be started.<br>B) If the definition is made in a command-line, request an Enter command after each code used. |
| | | | Task 2<br>Rule Creation | SGRC- EXUA | A) When defining Rules, request a Finish command to declare that the definition is finished, so that the rule can be added to the SG Rules list<br>B) If the definition is made in a command-line, request an Enter command after each code used.<br>C) When all rules to be used by the SG are finished, request a Finish command to declare that next task can be started. |
| | | | Task 3<br>SG Application | SGAPP- EXUA | A) When choosing the number of Iterations to apply in the SG, request an Enter command for the action to be applied.<br>B) To Save or Export a result, request an Enter command. |
| | | | Task 4<br>SG Manipulation | SGMAN- EXUA | A) To change the Iterations number to apply to the SG, request an Enter command for the action to be applied.<br>B) To select an alternative, preferably with Drag-and-Drop from an Alternatives Window or selection from it, request a Click on the desired shape to be dragged or selected to the SG window. |
| | | | Task 5<br>SG Alteration | SGALT- EXUA | A) The tools to be used shall follow SGSC- EXUA, SGRC- EXUA SGAPP- EXUA and SGMAN- EXUA. |

Table B15. User Control

| | | | | | |
|---|---|---|---|---|---|
| **User control** | **USCO** | Users should always be in control of the system processing (interrupt, cancel, pause and continue). | General | SG- USCO | A) Allow users to pace their entry, rather than controlling the time for each action.<br>B) Allow users to Save the current state of the system to resume it in another time.<br>C) Provide the user with Undo and Redo tools.<br>D) Provide a Cancel option to take the user to the initial state of the system or action. |
| | | | Task 1<br>Shape Creation | SGSC- USCO | A) Allow the user to take its time to define the Initial Shape, allowing the drawing to be changed until the user considers it finished.<br>B) Allow users to Save the Initial Shape to be used later. |
| | | | Task 2<br>Rule Creation | SGRC- USCO | A) Allow the user to take his/her time to define Rules, allowing the drawing to be changed until the user considers it finished.<br>B) Allow users to Save Rules to be used later. |
| | | | Task 3<br>SG Application | SGAPP- USCO | A) Allow the user to see the Iterations result without timing out.<br>B) Allow users to Save results to be used later. |
| | | | Task 4<br>SG Manipulation | SGMAN- USCO | A) Allow the user to try several Iterations numbers until reaching a desirable result.<br>B) Allow the user to try alternatives and select them for further work without timing out.<br>C) Allow users to Save results to be used later. |
| | | | Task 5<br>SG Alteration | SGALT- USCO | A) The tools to be used shall follow SGSC- USCO, SGRC- USCO SGAPP- USCO and SGMAN- USCO without timing out.<br>B) Any new state shall be able to be Saved and used later. |

Table B16. Error Protection

| | | | | | |
|---|---|---|---|---|---|
| **Error Protection** | **ERPR** | Means available to detect and prevent data entry errors, command errors, or actions with destructive consequences. It is preferable to detect errors before validation rather than after. | General | SG- ERPR | A) Assure that the SG implementation will deal properly with possible user error, including accidental inputs.<br>B) Use display messages to warn the user about incorrect inputs.<br>C) Display advisory messages before closing the SG implementation if all elements are not correctly saved or if there is a pending action.<br>D) If the SG computation by the implementation is limited, user definitions that will create computational errors shall be prevented. |
| | | | Task 1<br>Shape Creation | SGSC- ERPR | A) Assure user creates an Initial Shape as first action, creating warnings if another tasks are attempted first.<br>B) In the Expert Mode, where the user might start by Rules creation, a message shall appear to make the user confirm that action is first one.<br>C) If Initial Shapes are defined by code, there shall be prevention of data entry errors, accepting only valid codes and warning if incorrect ones were tried. |
| | | | Task 2<br>Rule Creation | SGRC- ERPR | A) The SG implementation shall have the means to advise the user if the Rules attempted to be made have errors that will lead to a computation error of the SG.<br>B) Unless the defined Rule is valid, it shall not be possible to save it and use it.<br>C) If Rules are defined by code, there shall be prevention of data entry errors, accepting only valid codes and warning if incorrect ones were tried. |
| | | | Task 3<br>SG Application | SGAPP- ERPR | A) The Iterations field shall only allow inputs of integer numbers and warn if incorrect data was input.<br>B) If a limited number of iterations is possible for a given SG, there shall be a warning and only valid numbers accepted. |
| | | | Task 4<br>SG Manipulation | SGMAN- ERPR | A) The implementation shall allow the user to access alternatives to SG results without destroying the SG rules definition. |
| | | | Task 5<br>SG Alteration | SGALT- ERPR | A) The error prevention of user actions to change the SG shall follow SGSC-ERPR, SGRC- ERPR, SGAPP- ERPR and SGMAN- ERPR. |

Table B17. Quality of Error Messages

| | | | | | |
|---|---|---|---|---|---|
| **Quality of error messages** | **E QUEM** | Phrasing and the content of error messages, that is: their relevance, readability, and specificity about the nature of the errors (syntax, format, etc.) and the actions needed to correct them. | General | SG- QUEM | **IM-sgi Criteria Definitions. IM-sgi – Interface Model for Shape Grammar Implementations**<br>A) If the user takes an incorrect action, this shall take no effect and an error message shall be displayed.<br>B) Error messages shall have task-oriented wording.<br>C) The error messages shall be specific and brief, informing clearly the error made and the correct action to take.<br>D) Different error messages shall be used according to the expertise mode in function. |
| | | | Task 1<br>Shape Creation | SGSC- QUEM | A) Any error message regarding Initial Shape shall follow the next template, or equivalent:<br>“INITIAL SHAPE: Please make sure to….” |
| | | | Task 2<br>Rule Creation | SGRC- QUEM | A) Any error message regarding Rules shall follow the next template, or equivalent:<br>“RULE CREATION: Please make sure to….” |
| | | | Task 3<br>SG Application | SGAPP- QUEM | A) Any error message regarding the SG by its Iterations application shall follow the next template, or equivalent:<br>“SG ITERATIONS: Please make sure to….” |
| | | | Task 4<br>SG Manipulation | SGMAN- QUEM | A) Any error message regarding the SG alternatives shall follow the next template, or equivalent:<br>“SG ALERNATIVES: Please make sure to….” |
| | | | Task 5<br>SG Alteration | SGALT- QUEM | A) Any error messages needed when the user changes prior definitions shall follow the corresponding templates, according to SGSC-QUEM, SGRC- QUEM, SGAPP- QUEM and SGMAN- QUEM. |

Table B18. Error Correction

| | | | | | |
|---|---|---|---|---|---|
| **Error correction** | **ERCO** | Means available to the users to correct their errors. | General | SG- ERCO | A) Users shall be allowed to edit an action before taking the explicit Enter entry, allowing corrections during composition.<br>B) After an Error Message, the user shall be able to take the correct action or correct the error directly and immediately. |
| | | | Task 1<br>Shape Creation | SGSC- ERCO | A) Allow the user to make any changes to the Initial Shape drawing until a valid shape is accepted.<br>B) Tools to Edit the drawing shall be available to allow the user any correction to the drawing needed.<br>C) If the Initial Shape is done by code, its code shall be accessible and changeable at all times. |
| | | | Task 2<br>Rule Creation | SGRC- ERCO | A) Allow the user to make any changes to the Rules definition until a valid Rule is accepted.<br>B) Tools to Edit the drawing shall be available to allow the user any correction to the drawing needed.<br>C) If the Rules are done by code, their code shall be accessible and changeable at all times. |
| | | | Task 3<br>SG Application | SGAPP- ERCO | A) Allow the user to change the Iterations number if an incorrect entry is done. |
| | | | Task 4<br>SG Manipulation | SGMAN- ERCO | A) Allow the user to manipulate alternatives without losing visibility or access to the ones used before or the ones never used. |
| | | | Task 5<br>SG Alteration | ASGALT- ERCO | A) Any prior action shall be accessible and changeable according to SGSC-ERCO, SGRC- ERCO, SGAPP- ERCO and SGMAN- ERCO. |

# Appendix C - Cognitive Walkthrough Results – List of Criteria

| Ergonomic Criteria Group | SG Tasks | IM-sgi Criteria Abbreviations | IM-sgi Criteria Definition | SHAPER 2D | SUBSHAPE | SGI | SGDE | SD2 |
|---|---|---|---|---|---|---|---|---|
| Compatibility | General | SG- COMP | A) The Users considered by IM-sgi are professionals in the artistic filed, mainly architects and designers. Considering these users, three types were distinguished:<br>STUDENTS – students of Architecture or Design that will have exploration objectives for the use of SG. These users shall be able to use the SG implementation for educational purposes, exploring and learning SG with the use of the tool.<br>ARTISTS/DESIGNERS – Artists, Architects, and Designers that will use SG for the search and definition of project solutions. These users shall be able to use the SG implementation as a tool for the development of their projects.<br>EXPERTS – SG experts that will be able to program the SG and the SG implementation in order to allow it to create new desirable results. These experts can be from the artistic field or computer science field. | N | N | N | N | N |
| | | | B) As the target users are characteristically users of CAD software, formats, tools, labels, messages or instructions shall follow conventional CAD software which is perceived by users as being familiar. CAD software has highly sophisticated interfaces that shall be a source for these definitions. | N | N | N | N | N |
| | | | C) A Graphical User Interface shall be used, and a permanent visualization of the results of the actions taken shall exist. All saved shapes, rules or SG results shall have a graphic representation. | Y | Y | Y | Y | Y |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | D) Units of measurement shall be metric and/or imperial. | N | N | N | N | N |
| | Task 1 – Shape Creation | SGSC- COMP | A) An Initial Shape shall be able to be created by drawing tools and with clear unit measurements as in conventional CAD software, by drawing in a window with zoom and pan abilities. | N | N | N | Y | Y |
| | | | B) According to professional needs, importation of shapes to be used or importation of background images to serve as a guide to the drawing shall be available. | Y | N | N | Y | Y |
| | | | C) The use of a command line for drawing shall be available for most experienced users. | N | N | N | N | N |
| | Task 2 – Rule Creation | SGRC- COMP | A) Rules shall be able to be created by drawing tools and with clear unit measurements as in conventional CAD software, by drawing in a window with zoom and pan abilities. | N | N | N | N | Y |
| | | | B) According to professional needs, importation of shapes to be used as rules or importation of background images to serve as a guide to the drawing shall be available. | Y | N | N | Y | Y |

|  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
|  |  |  | C) The use of a command line for drawing shall be available for most experienced users. | N | N | N | N | N |
|  | Task 3 – SG Application | SGAPP- COMP | A) SG results shall be able to be applied and visualized in a graphic window with zoom and pan abilities. | N | N | N | Y | Y |
|  |  |  | B) According to professional needs, importation of background images to merge and work with the SG results shall be available. | Y | Y | N | N | Y |
|  |  |  | C) The exportation of SG results in compatible formats with conventional CAD software shall exist, allowing the use of SG for professional purposes. | NA | NA | NA | NA | NA |
|  | Task 4 – SG Manipulation | SGMAN- COMP | A) SG results shall be able to be manipulated, cycling through different alternatives and changing iterations number | Y | Y | N | Y | Y |
|  |  |  | B) The manipulation of the results could be done by directly manipulating the drawing according to creativity purposes in the implementation itself | Y | Y | Y | Y | Y |
|  | Task 5 – SG Alteration | SGALT- COMP | A) Users shall be able to manipulate any previous state of the SG, from Initial Shape to its Rules and results | Y | N | N | Y | Y |
|  |  |  | B) Any state and alteration shall be able to be saved or exported to be used later or to be developed in CAD software | NA | NA | NA | NA | NA |
| Adaptability - Users' experience management | General | SG- USEX | A) The SG interface shall be prepared to adjust to three types of users, according to the definition made in this investigation. This adaptation can be achieved giving the user the chance to choose the level of expertise when opening the application. | N | N | N | N | N |
|  |  |  | B) A Beginner Mode shall be available, focused for Students of SG, with a highly automated control by the implementation. This mode shall have mainly drawing tools available and with immediate visualization of any action made. | N | N | N | N | N |
|  |  |  | C) An Intermediate Mode shall be available, focused on Artists/Designers allowing the users to have good control of the implementation and use it with possible integration with other CAD software so that SG can be | N | N | N | N | N |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | defined to solve design projects and its results used for further design project development/detailing. | | | | | |
| | | | D) An Expert Mode shall be available, focused on SG Experts allowing the user to have full control over the implementation and even changing its code so that it can create SG in new or different ways. The user shall have full control over the application with lower or no levels of automation. | N | N | N | N | N |
| | Task 1 – Shape Creation | SGSC- USEX | A) In the Beginner Mode, an Initial Shape shall be created using pre-defined shapes, so that the automation is higher. Drag-and-drop actions are preferable. | N | N | N | N | N |
| | | | B) In the Intermediate Mode, an Initial Shape shall be created recurring to drawing tools that resemble conventional CAD software. | N | N | N | N | N |
| | | | C) In the Expert Mode, an Initial Shape shall have the possibility to be created by code, ideally in a command-line. | N | N | N | N | N |
| | Task 2 – Rule Creation | SGRC- USEX | A) In the Beginner Mode, Rules shall be created using pre-defined shapes, so that the automation is higher. Drag-and-drop actions are preferable. | N | N | N | N | N |
| | | | B) In the Intermediate Mode, Rules shall be created recurring to drawing tools that resemble conventional CAD software. There shall be the possibility to create several rules for one same SG and be able to visualize them and reorder them. In this mode Labels to complex Rules shall be available. | N | N | N | N | N |
| | | | C) In the Expert Mode, Rules shall have the possibility to be created by code, ideally in a command-line. The use of labels and the general organization of the SG rules shall be able to be defined directly by code. | N | N | N | N | N |
| | Task 3 – SG Application | SGAPP- USEX | A) In the Beginner Mode, a pre-defined Iterations number could be applied to give an immediate visualization of the SG created while manipulating the Rules. User shall also be able to increase or decrease | N | N | N | N | N |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Iterations number with a simple button, without entering values. | | | | | |
| | | | B) In the Intermediate Mode, Iterations number shall be entered manually. | N | N | N | N | N |
| | | | C) In the Expert Mode, Iterations number shall be entered manually or by code, ideally in a command-line. | N | N | N | N | N |
| | Task 4 – SG Manipulation | SGMAN- USEX | A) In the Beginner Mode, a small number of alternatives could be seen, preferably by cycling them in the SG window | N | N | N | N | N |
| | | | B) In the Intermediate Mode, a good number of alternatives shall be visualized preferably in a Drop-down menu or in a new window and dragged to the SG window for substitution | N | N | N | N | N |
| | | | C) In the Expert Mode, the results shall be able to be controlled by code, ideally in a command-line, allowing the application of alternatives. | N | N | N | N | N |
| | Task 5 – SG Alteration | SGALT- USEX | A) The user actions shall follow SGSC-USEX, SGRC-USEX, SGAPP- USEX and SGMAN- USEX, giving the user the liberty to change any prior definition. | NA | NA | NA | NA | NA |
| | | | B) In all expertise modes, any changes made to the SG shall be visualized immediately. | Y | Y | N | Y | Y |
| Adaptability-Flexibility | General | SG- FLEX | A) SG implementation interface shall take into account conventional CAD software, as the target users are mainly from Architecture and Design fields, allowing the user to organize menus, tool boxes, windows or other to resemble the software they are familiar with. | N | N | N | N | N |
| | | | B) Users shall be able to use Buttons, Menus or Commands, o perform the same action, using what fits best their habits | N | Y | N | Y | Y |
| | | | C) Users shall be able to choose windows size, accordingly to the drawing or visualization needs | N | N | Y | Y | Y |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Task 1 – Shape Creation | SGSC- FLEX | A) Initial Shape window shall allow resizing, facilitating the drawing actions. | N | N | Y | N | Y |
| | | | B) Drawing commands can be available using drawing tools, using pre-defined shapes selected or dragged to the window or by a command line. | Y | N | N | Y | N |
| | | | C) In Expert Mode, the user shall be able to define Rules prior to Initial Shape drawing instead of following the general sequence of tasks for SG creation. | N | N | N | N | N |
| | Task 2 – Rule Creation | SGRC- FLEX | A) Rules window can allow resizing, facilitating the drawing actions. | N | N | N | N | Y |
| | | | B) Drawing commands can be available using drawing tools, using pre-defined shapes selected or dragged to the window or by a command line. | Y | N | N | Y | N |
| | | | C) Label tools shall be available, dragging symbols to the drawing or inserting them with a button. | N | N | N | Y | N |
| | | | D) In Expert Mode, User shall be able to define Rules prior to Initial Shape drawing instead of following the general sequence of tasks for SG creation. | N | N | N | N | N |
| | Task 3 – SG Application | SGAPP- FLEX | A) SG window shall have a big presence in the screen and allow resizing. | N | N | Y | Y | Y |
| | | | B) User shall be able to choose the iterations number by adding the specific number or cycling through pre-defined numbers. The system can also have a pre-defined number applied automatically that can be changed. | Y | Y | N | Y | Y |
| | Task 4 – SG Manipulation | SGMAN- FLEX | A) The manipulation of alternatives shall be possible according to SGAPP- FLEX. | NA | NA | NA | NA | NA |
| | | | B) User shall be able to choose to see alternatives in a new window or cycle through them directly in the SG window, or Drop-down menu or even textually in a command-line. | N | Y | N | Y | Y |
| | Task 5 – SG Alteration | SGALT- FLEX | A) The user choices shall follow SGSC-FLEX, SGRC-FLEX, and SGAPP- FLEX. | NA | NA | NA | NA | NA |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Significance of codes | General | SG- SICO | A) All window titles shall be clearly related to the tasks and results that will be available | Y | N | N | Y | Y |
| | | | B) Any abbreviations shall be clear and limited to the strictly necessary | Y | N | Y | N | N |
| | | | C) Menus and Button naming shall follow conventional CAD software and be clearly related to the actions | N | N | N | N | N |
| | | | D) Codes to be used in command lines shall be meaningful and allow the clear definition of the actions | N | N | N | N | N |
| | Task 1 – Shape Creation | SGSC- SICO | A) The term "INITIAL SHAPE" shall be used in the title of the window for this purpose | N | N | N | Y | Y |
| | | | B) This term shall be used for any message related to Initial Shape definition | N | N | N | Y | N |
| | | | C) Drawing tools shall have clear names and follow conventional CAD software | N | N | N | N | N |
| | Task 2 – Rule Creation | SGRC- SICO | A) The term "RULES" shall be used in the title of the window for this purpose | Y | N | Y | Y | Y |
| | | | B) This term shall be used for any message related to Rule definition | N | N | Y | Y | Y |
| | | | C) Drawing tools available for Rule creation shall have the same naming as the ones for Initial Shape Drawing | N | N | N | N | N |
| | Task 3 – SG Application | SGAPP- SICO | A) The term "SHAPE GRAMMAR" shall be used in the title of the window for this purpose | N | N | N | N | N |
| | | | B) This term shall be used for any message related to Shape Grammar application | N | N | N | N | N |
| | | | C) The term "ITERATIONS" shall be used for the Data entry field for Iterations application | Y | N | N | N | N |
| | Task 4 – SG Manipulatio n | SGMAN- SICO | A) A term related to "ALTERNATIVES" or "RESULTS" shall be used for menus or windows that allow the choice of Shape Grammar alternatives | N | N | N | N | N |
| | Task 5 – SG Alteration | SGALT- SICO | Not applicable | NA | NA | NA | NA | NA |

|  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
| Consistency | General | SG- CONS | A) Window titles and formats shall follow the same format and placement in all windows. | Y | Y | Y | Y | Y |
|  |  |  | B) The screen format shall be identical in all expertise modes, adapting itself to the specifications of each one | N | N | N | N | N |
|  |  |  | C) Menu options, buttons, and command-lines shall be identical in all different windows and expertise modes | N | N | N | N | N |
|  |  |  | D) All messages shall follow the same format and position on the screen | N | N | Y | Y | Y |
|  | Task 1 – Shape Creation | SGSC- CONS | A) Initial Shape window and tools shall be identical and recognizable in all expertise modes, independently of the additional tools in each one | N | Y | N | N | Y |
|  | Task 2 – Rule Creation |  | A) Rules window and tools shall be identical and recognizable in all expertise modes, independently of the additional tools in each one | N | N | N | N | Y |
|  | Task 3 – SG Application | SGRC- CONS | A) SG window and tools shall be identical and recognizable in all expertise modes, independently of the additional tools in each one | N | N | N | N | Y |
|  | Task 4 – SG Manipulation | SGMAN- CONS | Not applicable | NA | NA | NA | NA | NA |
|  | Task 5 – SG Alteration | SGALT- CONS | Not applicable | NA | NA | NA | NA | NA |
| Guidance - Immediate feedback | General | SG- FEED | A) SG implementation shall have a Graphical User Interface that allows immediate visualization of created shapes, created rules and SG results and alternatives | Y | Y | N | Y | Y |
|  |  |  | B) Graphical visualization of the created rules, either by drawing or data entry shall be immediately presented | Y | Y | N | Y | Y |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | C) SG Iterations shall be immediately graphically visualized, along with a number of alternatives | Y | Y | N | Y | Y |
| | | | D) Undo and Redo tools shall be available and its results immediately visualized | N | N | Y | Y | N |
| | | | E) Messages guiding the user shall be delivered | N | N | N | N | N |
| | Task 1 – Shape Creation | SGSC- FEED | A) Initial Shape window shall allow the graphical design with drawing tools, similarly to conventional CAD software. | N | N | N | N | N |
| | | | B) Drawing tools shall be clearly identified, and it's results immediately visualized when used | N | N | N | Y | Y |
| | | | C) Any change made to the shapes shall be easily reversible, and all the results immediately visualized. | N | N | Y | Y | N |
| | | | D) If a command-line exists to define shapes, the results shall be graphically seen in the Initial Shape window | N | N | N | N | N |
| | Task 2 – Rule Creation | SGRC- FEED | A) Rules window shall allow the graphical design with drawing tools, similarly to conventional CAD software. | N | N | N | N | N |
| | | | B) Drawing tools shall be clearly identified, and its results immediately visualized when used | N | N | N | Y | Y |
| | | | C) Any change made to the shapes and rules shall be easily reversible, and all the results immediately visualized. | N | N | N | Y | N |
| | | | D) The system shall allow saving and ordering the rules that will compose the SG and allow the user to easily visualize them and manipulate them, preferably in a dedicated Window or Drop-down menu with graphical visualizations of the composed rules | N | N | N | Y | Y |
| | | | E) If a command-line exists to define rules, the results shall be graphically seen in the Rules' window | N | N | N | N | N |
| | Task 3 – SG Application | SGAPP-FEED | A) The result of the number of Iterations applied shall be immediately seen in the SG window | Y | N | N | Y | Y |
| | | | B) The system shall graphically show alternatives of the SG with the same number of Iterations, preferably in a dedicated Window or Drop-down menu. | N | N | N | N | N |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Task 4 – SG Manipulation | SGMAN- FEED | A) The Iterations tool shall allow immediate change of the number, with immediate visualization of the new result. | Y | N | N | Y | Y |
| | | | B) The system shall graphically show alternatives of the SG with the same number of Iterations, preferably in a dedicated Window or Drop-down menu, which can be selected and worked on. | N | N | N | N | N |
| | Task 5 – SG Alteration | SGALT- FEED | A) If changes are made in the Initial Shape, the new results shall be immediately graphically visualized | Y | N | N | Y | Y |
| | | | B) If changes are made in the Rules, the new results shall be immediately graphically visualized | Y | Y | N | Y | Y |
| | | | C) Any alteration shall be easily undone or redone, and its results visually seen | N | N | Y | Y | N |
| Guidance - Grouping, and distinction of items by location | General | SG-GDLO | A) Three main drawing windows must exist, organized by order:<br>1 – INITIAL SHAPE or equivalent<br>2 – RULES or equivalent<br>3 – SHAPE GRAMMAR or equivalent<br>They shall be organized linearly, left to right or up to down, or combinations of these, according to reading conventions | N | Y | N | Y | N |
| | | | B) Any extra windows must be organized according to the above reading conventions, next to the related window. | N | Y | N | N | Y |
| | | | C) The menu options to be used in each drawing window must be organized according to the object they apply and accordingly to the order of commands to be performed. | N | N | N | Y | N |
| | Task 1 – Shape Creation | SGSC- GDLO | A) Window for Initial Shape creation must be shown as logically the first to be used. All drawing tools for the Initial Shape drawing must be available in that window, in buttons or menus. | Y | N | N | Y | N |
| | | | B) Any new windows or menus that relate to Initial Shape must be gathered with the main Initial Shape window | N | N | N | Y | Y |
| | | | C) If a command line exists for the Initial Shape creation, it shall be located on the bottom of the Initial Shape window | N | N | N | N | N |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Task 2 – Rule Creation | SGRC- GDLO | A) Window for Rule creation must be shown as logically the second one to be used. All the tools for the Rule definition must be available in that window, in buttons or menus. | Y | Y | N | Y | N |
| | | | B) A Window shall exist to show the list of existing rules, next to the Rule main window. Exact same logic if instead of a window for this purpose a Drop-Down menu is opened. | N | N | Y | Y | Y |
| | | | C) Any new windows or menus that relate to Rule creation must be gathered with the main Rule Window | N | N | N | Y | N |
| | | | D) If a command line exists for the Rule creation, it shall be located on the bottom of the Initial Shape window | N | N | N | N | N |
| | Task 3 – SG Application | SGAPP- GDLO | A) Window for SG application must be shown as logically the third one to be used. All the tools for the SG application must be available in that window, in buttons or menus. | N | N | N | Y | N |
| | | | B) The button or data field to add the Iterations number, to apply the SG, must be clearly situated in the SG window | N | N | N | N | N |
| | | | C) Any new windows or menus that relate to SG application must be gathered with the main SG window | N | N | N | N | Y |
| | Task 4 – SG Manipulation | SGMAN- GDLO | A) SG manipulation made by changing de Iterations number, must follow SGAPP- GDLO- B | NA | NA | NA | NA | NA |
| | | | B) A window or drop-down menu shall exist to show SG alternatives to be selected. This must be gathered with the main SG window | N | N | N | N | N |
| | | | C) Tools related to saving or export results shall be clearly located near the SG window | N | N | N | N | Y |
| | Task 5 – SG Alteration | SGALT- GDLO | A) SG alteration made by changing de Initial Shape shall be clear by the application of SGSC- GDLO | NA | NA | NA | NA | NA |
| | | | B) SG alteration made by changing Rules shall be clear by the application of SGRC- GDLO | NA | NA | NA | NA | NA |
| Guidance - Grouping, and distinction of | General | SG- GDFO | A) Drawing windows for shapes and rule creation shall be graphically similar | N | N | N | N | Y |
| | | | B) Drawing tools shall be the same in Shape and Rule window, so they can be easily recognized | N | N | N | N | Y |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| items by format | | | C) SG windows shall be graphically identifiable as the main results and manipulation window | N | N | N | Y | Y |
| | | | D) Secondary windows for Rules list or SG alternatives shall be graphically similar to be identifiable as secondary and have the same position relatively to the main window they refer to. If they are shown by a Drop Down menu, they shall also be similar graphically and in position in the window. | N | N | N | Y | N |
| | Task 1 – Shape Creation | SGSC- GDFO | A) All drawing tools for shape creation in the Initial Shape window shall be similar, either in a List Menu or in Buttons. These shall also be similar to conventional drawing tools CAD software. | Y | N | N | N | N |
| | | | B) If a command line exists for shape definition, it shall be graphically identifiable as this kind of tool | N | N | N | N | N |
| | Task 2 – Rule Creation | SGRC- GDFO | A) All drawing tools for shape creation in the Rule window shall be similar, either in a List Menu or in Buttons. These shall also be similar to conventional drawing tools in CAD software. | Y | N | N | N | N |
| | | | B) If a command line exists for rule definition, it shall be graphically identifiable as this kind of tool | N | N | N | N | N |
| | | | C) A secondary window for the list of Rules that compose the grammar shall be graphically identifiable as this kind of element. The same way if it is shown in a Drop-down menu. | N | N | N | Y | Y |
| | Task 3 – SG Application | SGAPP-GDFO | A) The button or data field to add the Iterations number shall be graphically relevant and predominant in the SG window | N | N | N | N | N |
| | Task 4 – SG Manipulation | SGMAN-GDFO | A) A secondary window for a list of SG alternatives shall be graphically identifiable as this kind of element. The same way if it is shown in a Drop-down menu. | N | N | N | N | N |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Task 5 – SG Alteration | SGALT-GDFO | A) All windows, menus, and buttons shall be graphically identifiable according to SGSC- GDFO and SGRC- GDFO so that it is easy to identify the procedures to change Initial Shape or Rules of the SG. | NA | NA | NA | NA | NA |
| Guidance - Prompting | General | SG-PROM | A) All Windows clearly named, numbered according to the order of use, if needed | N | N | N | N | N |
| | | | B) For data field entries, display associated label | Y | Y | N | N | Y |
| | | | C) Display measurement units when drawing tools are used | N | N | N | N | N |
| | | | D) Provide drawing status information (shape measurements, shape color, thickness, filling, shape geometry (opened or closed)) | N | N | N | N | N |
| | | | E)Provide help tools | N | N | N | N | N |
| | Task 1 – Shape Creation | SGSC-PROM | A) Window for Initial Shape Creation name: 1 – INITIAL SHAPE or equivalent | N | N | N | Y | Y |
| | | | B) Display only available actions and in order of use: drawing tools for shape drawing, editing, and saving. | N | N | N | Y | Y |
| | | | C) If drawing is made by data entry, provide required formats and accepted values. | N | N | N | N | NA |
| | Task 2 – Rule Creation | SGRC-PROM | A) Window for Rule Creation name: 2 – RULES or equivalent | Y | N | Y | Y | Y |
| | | | B) Display only available actions: drawing tools for shape drawing, editing, and saving. | N | N | N | Y | Y |
| | | | C) If drawing is made by data entry, provide required formats and accepted values | N | N | N | N | NA |
| | | | D) Provide Drop-down Menu, List or Window with saved rules that will compose the SG | N | N | Y | Y | Y |
| | Task 3 – SG Application | SGAPP-PROM | A) Window for SG Application name: 3 – SHAPE GRAMMAR or equivalent | Y | N | Y | N | Y |
| | | | B) Display only available actions: number of iterations to be used, editing and saving. | Y | N | N | Y | Y |
| | | | C) Provide the required format and acceptable values for the iterations number to apply | Y | N | N | N | N |
| | | | D) Provide Drop-down Menu, List or Window with SG alternatives | N | N | N | N | N |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Task 4 – SG Manipulation | SGMAN-PROM | A) Display only available actions: changing number of iterations, choose different alternatives, edit, save and export. | N | Y | N | Y | Y |
| | Task 5 – SG Alteration | SGALT-PROM | B) Display only available actions: add/change Initial Shape; add/change rule or rules order | N | Y | Y | Y | Y |
| Guidance - Legibility | General | SG- LEGI | A) Window names shall be all equally formatted and aligned, preferably with upper case letters and aligned on the left | Y | N | N | N | Y |
| | | | B) Menus in all the windows shall be distributed with the same inter-word spacing and preferably with upper case letters | Y | N | N | N | Y |
| | | | C) Tools and Menus shall follow conventional CAD software | N | N | N | N | N |
| | Task 1 – Shape Creation | SGSC- LEGI | A) Drawing tools shall have clear icons if they are buttons and, preferably have an associated label | Y | N | N | N | N |
| | | | B) Drawing window shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa. A scalable grid can help understand shapes relations. | N | N | N | N | Y |
| | | | C) If a command line exists to enter shape definitions, the text shall have a Font without sheriff and with a good readable size | N | N | N | N | N |
| | Task 2 – Rule Creation | SGRC- LEGI | A) Drawing tools shall have clear icons if they are buttons and, preferably have an associated label | N | N | N | N | N |
| | | | B) Drawing window or Rules list window shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa. A scalable grid can help understand shapes relations. | N | N | N | N | Y |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | C) If a command line exists to enter shape definitions, the text shall have a Font without sheriff and with a good readable size | N | N | N | N | N |
| | Task 3 – SG Application | SGAPP- LEGI | A) SG window shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa. A scalable grid can help understand shapes relations. | N | N | N | N | Y |
| | | | B) If a command line exists to enter shape definitions, the text shall have a Font without sheriff and with a good readable size | N | N | N | N | N |
| | | | C) Iterations Button or Data field shall have a velar Font without sheriff and with good readable size. | N | N | Y | N | Y |
| | Task 4 – SG Manipulation | SGMAN- LEGI | A) If a SG alternatives' window exists, it shall have a contrasting background with the drawing lines. Ideally black background and white shapes or vice versa. | N | N | N | N | Y |
| | | | B) Iterations button shall be as SGAPP- LEGI- C. | NA | NA | NA | NA | NA |
| | Task 5 – SG Alteration | SGALT- LEGI | A) Tools and Menus needed to manipulate the SG shall be according to SGSC- LEGI, SGRC- LEGI and SGAPP- LEGI. | NA | NA | NA | NA | NA |
| User workload – Brevity - Concision | General | SG- CONC | A) Possibly existing Data field shall allow exclusively accepted values | Y | Y | Y | Y | Y |
| | | | B) If a measurement unit is associated with a particular data field for drawing, include that unit as part of the field label | N | N | N | N | N |
| | | | C) If codes are used in a command-line to activate tools, use shortcodes with no more than 5 characters | N | N | N | N | N |
| | | | D) In iconic buttons, allow labels with the description to appear when hovering the cursor. | N | NA | Y | NA | NA |
| | Task 1 – Shape Creation | SGSC- CONC | A) Drawing tools for Initial Shape drawing shall be the only ones available directly on the Initial Shape window | N | N | N | Y | Y |
| | | | B) If a command line is used, drawing commands shall be small and clear. Measurements shall automatically have the measurement unit, instead of making the user add them | N | N | N | N | N |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Task 2 – Rule Creation | SGRC- CONC | A) Drawing tools for Rule drawing shall be the only ones available directly on the Rule window and be similar to the ones to create Initial Shapes | N | N | N | N | Y |
| | | | B) If a command line is used, commands shall be small and clear. Measurements shall automatically have the measurement unit, instead of making the user add them | N | N | N | N | N |
| | | | C) The window or Drop-down menu with a list of rules shall allow them to be selected or re-ordered by simple click and drag with the cursor | N | N | N | N | N |
| | Task 3 – SG Application | SGRC- CONC | A) Iterations Button or Data field shall accept numbers exclusively and allow only the length acceptable by the implementation | Y | Y | Y | N | Y |
| | Task 4 – SG Manipulation | SGMAN- CONC | A) The window or Drop-down menu with alternatives shall allow them to be selected by a simple click with the cursor | N | N | N | N | N |
| | Task 5 – SG Alteration | SGALT- CONC | A) The tools to be used shall follow SGSC- CONC, SGRC- CONC SGAPP- CONC and SGMAN- CONC. | NA | NA | NA | NA | NA |
| User workload – Brevity - Minimal actions | General | SG- MIAC | A) Reduce to the minimum necessary actions to create a SG | N | N | N | N | N |
| | | | B) Reduce to the minimum the codes to be used in command-lines | N | N | N | N | N |
| | Task 1 – Shape Creation | SGSC- MIAC | A) For the creation of the Initial Shape only drawing tools shall be needed and they shall be immediately available in the Rules window. | N | N | N | Y | Y |
| | | | B) Manipulation of the shapes shall be done by click and drag with the cursor | Y | Y | Y | Y | Y |
| | | | C) If a command line exists for the creation of shapes, these shall be easily created by simple commands with dimensions' introduction. | N | N | N | N | N |
| | | | D) Saving and editing tools shall be distinguished as secondary tools | Y | Y | Y | Y | Y |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Task 2 – Rule Creation | SGRC- MIAC | A) For the creation of Rules only drawing tools shall be needed and they shall be immediately available in the Rules window. | N | N | N | N | Y |
| | | | B) Manipulation of the shapes shall be done by click and drag with the cursor. | Y | Y | Y | Y | Y |
| | | | C) The Initial shape shall automatically appear in the Rules window to void repeating it's drawing | Y | N | N | N | N |
| | | | D) When a SG is made from a list of rules, allow the creation of a new rule starting from the drawing of the previous one | N | N | N | Y | Y |
| | | | E) The insertion of Labels shall be done by single commands, preferably a button with the label to be used | N | N | N | Y | N |
| | | | F) If a command line exists for the creation of shapes, these shall be easily created by simple commands with dimensions' introduction. | N | N | N | N | N |
| | | | G) Saving a rule to the list shall be done by a single command that will add it to the Rules' list | N | N | N | Y | Y |
| | Task 3 – SG Application | SGAPP- MIAC | A) A simple button or data field shall be needed to indicate the Iterations number and apply the SG, and no other command shall appear as relevant in the SG window | Y | Y | Y | N | Y |
| | | | B) Saving a result shall be done by a single command. | Y | Y | Y | Y | Y |
| | Task 4 – SG Manipulatio n | SGMAN- MIAC | A) Manipulation of the SG shall be easily done by changing the Iterations number, accordingly to SGAPP- MIAC- A. | NA | NA | NA | NA | NA |
| | | | B) Choosing alternatives shall be done simply by click and drag with the cursor. | N | N | N | N | N |
| | Task 5 – SG Alteration | SGALT- MIAC | A) The tools to be used shall follow SGSC- MIAC, SGRC- MIAC SGAPP- MIAC and SGMAN- MIAC. | NA | NA | NA | NA | NA |
| User Workload - Information density | General | SG- INDE | A) The implementation display shall show exclusively the windows referred in SG-GLO- A. | NA | NA | NA | NA | NA |
| | | | B) Other windows shall be presented as secondary or opened only when needed | N | N | N | Y | N |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | C) Command lines shall be integrated into the bottom of the corresponding window | N | N | N | N | N |
| | | | D) Allow shapes, rules, and results to be permanently visible, so the user doesn't have to memorize and give them proper emphasis to the rest of the environment. | Y | Y | Y | N | Y |
| | | | E) Computation needed for the SG application shall be automatic and without any calculated entry done manually by the user | Y | Y | Y | Y | Y |
| | Task 1 – Shape Creation | SGSC- INDE | A) Initial Shape window shall be a simple drawing area with only drawing tools available. | N | N | N | Y | Y |
| | | | B) A command line for shape creation can be shown only when needed, in the bottom of the Rules' window. | N | N | N | N | N |
| | Task 2 – Rule Creation | SGRC- INDE | A) Rules window shall be a simple drawing area with drawing tools available, and a save command to create a list of rules for SGs with more than one rule. | N | N | N | N | N |
| | | | B) A window with the graphical visualization of the rules can be opened when needed instead of being always visible | N | N | N | Y | Y |
| | | | C) A command line for rule creation can be shown only when needed, in the bottom of the Rules' window. | N | N | N | N | N |
| | Task 3 – SG Application | SGAPP- INDE | A) SG window shall be a simple drawing area with the iterations number tool as the main command. | N | N | N | N | N |
| | Task 4 – SG Manipulation | SGMAN- INDE | A) A window with the graphical visualization of alternatives can be opened when needed instead of being always visible | N | N | N | N | N |
| | Task 5 – SG Alteration | SGALT- INDE | A) The tools to be used shall follow SGSC- INDE, SGRC- INDE SGAPP- INDE and SGMAN- INDE. | NA | NA | NA | NA | NA |
| Explicit user actions | General | SG - EXU | A) Request from the user a clear Enter action to initiate the SG process | N | N | N | N | N |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | B)Request from the user a clear click on Menu and Drop-down menu choices | Y | Y | Y | Y | Y |
| | | | C) Request from the user a clear Enter action when using command-line entries | N | N | N | N | N |
| | Task 1 – Shape Creation | SGSC- EXUA | A) When defining the Initial Shape, request a Finish command to declare that the drawing is finished and that the next task can be started | N | N | N | Y | Y |
| | | | B) If the definition is made in a command line, request an Enter command after each code used | N | N | N | N | N |
| | Task 2 – Rule Creation | SGRC- EXUA | A) When defining Rules, request a Finish command to declare that the definition is finished, so that the rule can be added to the SG Rules' list | N | N | N | Y | N |
| | | | B) If the definition is made in a command line, request an Enter command after each code used | N | N | N | N | N |
| | | | C) When all rules to be used by the SG are finished, request a Finish command to declare that the next task can be started | N | N | N | N | N |
| | Task 3 – SG Application | SGAPP- EXUA | A) When choosing the number of Iterations to apply in the SG, request an Enter command for the action to be applied | N | N | Y | N | N |
| | | | B) To Save or Export a result, request an Enter command | N | N | N | N | N |
| | Task 4 – SG Manipulatio n | SGMAN- EXUA | A) To change the Iterations number to apply to the SG, request an Enter command for the action to be applied | N | N | Y | N | N |
| | | | B) To select an alternative, preferably with Drag-and-Drop from an Alternatives Window or selection from it, request a Click on the desired shape to be dragged or selected to the SG window. | N | N | N | N | N |
| | Task 5 – SG Alteration | SGALT- EXUA | A) The tools to be used shall follow SGSC- EXUA, SGRC- EXUA SGAPP- EXUA and SGMAN- EXUA. | NA | NA | NA | NA | NA |
| User control | General | SG- USCO | A) Allow users to pace their entry, rather than controlling the time for each action | N | N | Y | N | N |
| | | | B) Allow users to Save the current state of the system to resume it in another time | N | N | N | N | Y |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | C) Provide the user with Undo and Redo tools | N | N | N | Y | N |
| | | | D) Provide a Cancel option to take the user to the initial state of the system or action | N | N | N | Y | N |
| | Task 1 – Shape Creation | SGSC- USCO | A) Allow the user to take its time to define the Initial Shape, allowing the drawing to be changed until the user considers it finished | Y | N | Y | Y | Y |
| | | | B) Allow users to Save the Initial Shape to be used later | Y | Y | Y | Y | Y |
| | Task 2 – Rule Creation | SGRC- USCO | A) Allow the user to take its time to define Rules, allowing the drawing to be changed until the user considers it finished | Y | N | Y | Y | Y |
| | | | B) Allow users to Save Rules to be used later | Y | N | N | Y | Y |
| | Task 3 – SG Application | SGAPP - USCO | A) Allow the user to see the Iterations result without timing out | Y | N | Y | Y | Y |
| | | | B) Allow users to Save results to be used later | Y | Y | Y | Y | Y |
| | Task 4 – SG Manipulatio n | SGMAN- USCO | A) Allow the user to try several Iterations numbers until reaching a desirable result | Y | Y | Y | N | Y |
| | | | B) Allow the user to try alternatives and select them for further work without timing out | N | N | N | N | N |
| | | | C) Allow users to Save results to be used later | Y | Y | Y | Y | Y |
| | Task 5 – SG Alteration | SGALT- USCO | A) The tools to be used shall follow SGSC- USCO, SGRC- USCO SGAPP- USCO and SGMAN- USCO without timing out | NA | N | NA | NA | NA |
| | | | B) Any new state shall be able to be Saved and used later. | Y | Y | N | Y | Y |
| Error Protection | General | SG- ERPR | A) Assure that the SG implementation will deal properly with possible user error, including accidental inputs | NA | NA | NA | NA | NA |
| | | | B) Use display messages to warn the user about incorrect inputs | NA | NA | NA | NA | NA |
| | | | C) Display advisory messages before closing the SG implementation if all elements are not correctly saved or if there is a pending action. | NA | NA | NA | NA | NA |
| | | | D) If the SG computation by the implementation is limited, user definitions that will create computational errors shall be prevented | NA | NA | NA | NA | NA |

|  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
|  | Task 1 – Shape Creation | SGSC- ERPR | A) Assure user creates an Initial Shape as the first action, creating warnings if other tasks are attempted first | NA | NA | NA | NA | NA |
|  |  |  | B) In the Expert Mode, where the user might start by Rules creation, a message shall appear to make the user confirm that action as the first one. | NA | NA | NA | NA | NA |
|  |  |  | C) If Initial Shapes are defined by code, there shall be the prevention of data entry errors, accepting only valid codes and warning if incorrect ones were attempted to be used | NA | NA | NA | NA | NA |
|  | Task 2 – Rule Creation | SGRC- ERPR | A) The SG implementation shall have the means to advise the user if the Rules attempted to be made have errors that will lead to a computation error of the SG | NA | NA | NA | NA | NA |
|  |  |  | B) Unless the defined Rule is valid, it shall not be possible to save it and use it | NA | NA | NA | NA | NA |
|  |  |  | C) If Rules are defined by code, there shall be the prevention of data entry errors, accepting only valid codes and warning if incorrect ones were attempted to be used | NA | NA | NA | NA | NA |
|  | Task 3 – SG Application | SGAPP- ERPR | A) The Iterations field shall only allow inputs of integer numbers and warn if incorrect data was attempted to be used | NA | NA | NA | NA | NA |
|  |  |  | B) If a limited number of iterations is possible for a given SG, there shall be a warning and only valid numbers accepted | NA | NA | NA | NA | NA |
|  | Task 4 – SG Manipulation | SGMAN- ERPR | A) The implementation shall allow the user to access alternatives to SG results without destroying the SG rules definition | NA | NA | NA | NA | NA |
|  | Task 5 – SG Alteration | SGALT- ERPR | A) The error prevention of user actions to change the SG shall follow SGSC-ERPR, SGRC- ERPR, SGAPP-ERPR and SGMAN- ERPR. | NA | NA | NA | NA | NA |
| Quality of error messages | General | SG- QUEM | A) If the user takes an incorrect action, this shall take no effect, and an error message shall be displayed | NA | NA | NA | NA | NA |
|  |  |  | B) Error messages shall have task-oriented wording. | NA | NA | NA | NA | NA |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | C) The error messages shall be specific and brief, informing clearly the error made and the correct action to take | NA | NA | NA | NA | NA |
| | | | D) Different error messages shall be used according to the expertise mode in function | NA | NA | NA | NA | NA |
| | Task 1 – Shape Creation | SGSC-QUEM | A) Any error message regarding Initial Shape shall follow the next template, or equivalent:<br>“INITIAL SHAPE: Please make sure to ….” | NA | NA | NA | NA | NA |
| | Task 2 – Rule Creation | SGRC-QUEM | A) Any error message regarding Rules shall follow the next template, or equivalent:<br>“RULE CREATION: Please make sure to ….” | NA | NA | NA | NA | NA |
| | Task 3 – SG Application | SGAPP-QUEM | A) Any error message regarding the SG by its Iterations application shall follow the next template, or equivalent:<br>“SG ITERATIONS: Please make sure to ….” | NA | NA | NA | NA | NA |
| | Task 4 – SG Manipulation | SGMAN-QUEM | A) Any error message regarding the SG alternatives shall follow the next template, or equivalent:<br>“SG ALTERNATIVES: Please make sure to ….” | NA | NA | NA | NA | NA |
| | Task 5 – SG Alteration | SGALT-QUEM | A) Any error messages needed when the user changes prior definitions shall follow the corresponding templates, according to SGSC-QUEM, SGRC- QUEM, SGAPP- QUEM and SGMAN- QUEM. | NA | NA | NA | NA | NA |
| Error correction | General | SG- ERCO | A) Users shall be allowed to edit an action before taking the explicit Enter entry, allowing corrections during composition | NA | NA | NA | NA | NA |
| | | | B) After an Error Message, the user shall be able to take the correct action or correct the error directly and immediately | NA | NA | NA | NA | NA |
| | | SGSC-ER | A) Allow the user to make any changes to the Initial Shape drawing until a valid shape is accepted | NA | NA | NA | NA | NA |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Task 1 – Shape Creation | | B) Tools to Edit the drawing shall be available to allow the user any correction to the drawing needed | NA | NA | NA | NA | NA |
| | | | C) If the Initial Shape is done by code, it’s code shall be accessible and changeable at all times | NA | NA | NA | NA | NA |
| | Task 2 – Rule Creation | SGRC- ERCO | A) Allow the user to make any changes to the Rules definition until a valid Rule is accepted | NA | NA | NA | NA | NA |
| | | | B) Tools to Edit the drawing shall be available to allow the user any correction to the drawing needed | NA | NA | NA | NA | NA |
| | | | C) If the Rules are made by code, their code shall be accessible and changeable at all times | NA | NA | NA | NA | NA |
| | Task 3 – SG Application | SGAPP- ERCO | A) Allow the user to change the Iterations number if an incorrect entry is done | NA | NA | NA | NA | NA |
| | Task 4 – SG Manipulation | SGMAN- ERCO | A) Allow the user to manipulate alternatives without losing visibility or access to the ones used before or the ones never used | NA | NA | NA | NA | NA |
| | Task 5 – SG Alteration | SGALT- ERCO | A) Any prior action shall be accessible and changeable according to SGSC-ERCO, SGRC- ERCO, SGAPP- ERCO, and SGMAN- ERCO. | NA | NA | NA | NA | NA |

# IM-sgi Prototype V01

## VERSION 01

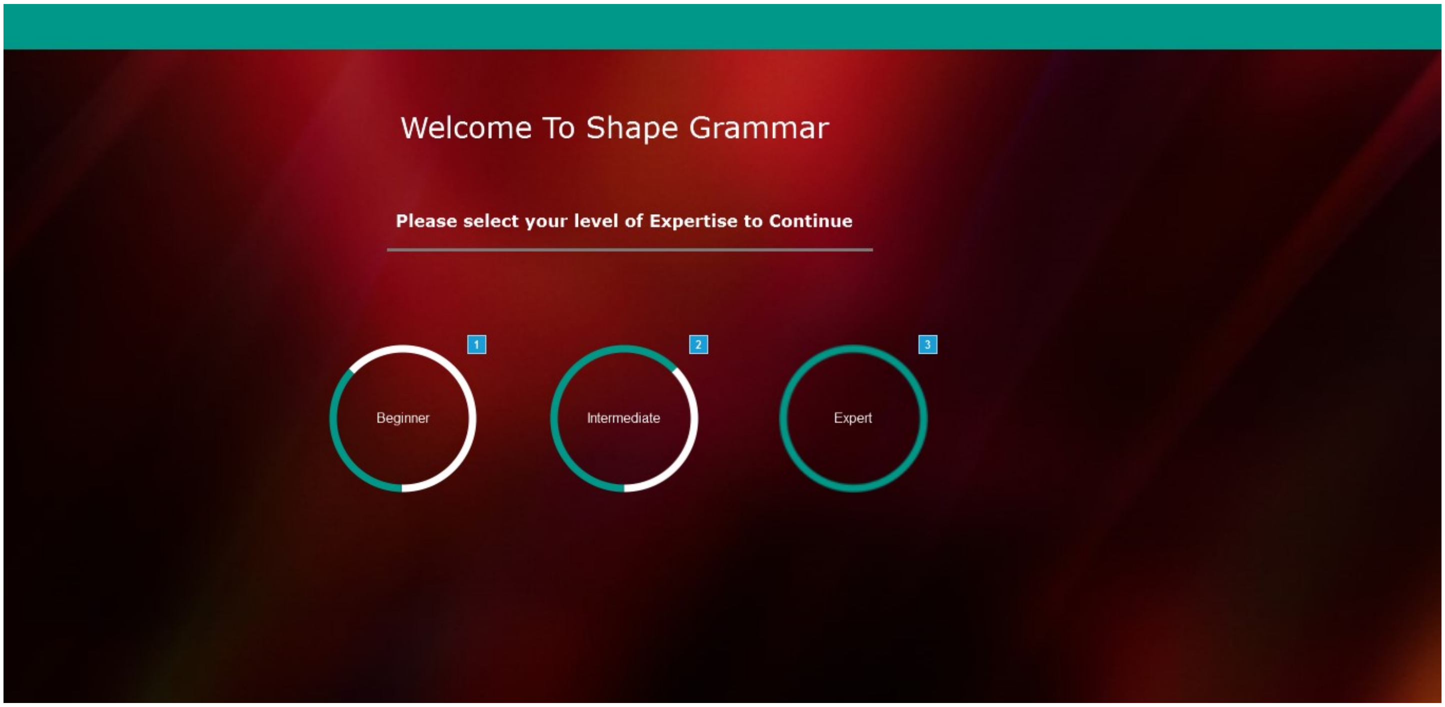
# 1. Pages

## 1.1. Page Tree

Home
- Shape Grammar - Begginer
- Shape Grammar - Intermediate
- Shape Grammar - Expert

## 1.2. Home

### 1.2.1. User Interface



Welcome To Shape Grammar

Please select your level of Expertise to Continue

### 1.2.2. Widget Table

| Footnote | Interactions |
|---|---|
| 1 | OnClick:<br>Case 1:<br>Open Shape Grammar - Begginer in Current Window |
| 2 | OnClick:<br>Case 1:<br>Open Shape Grammar - Intermediate in Current Window |
| 3 | OnClick:<br>Case 1:<br>Open Shape Grammar - Expert in Current Window |

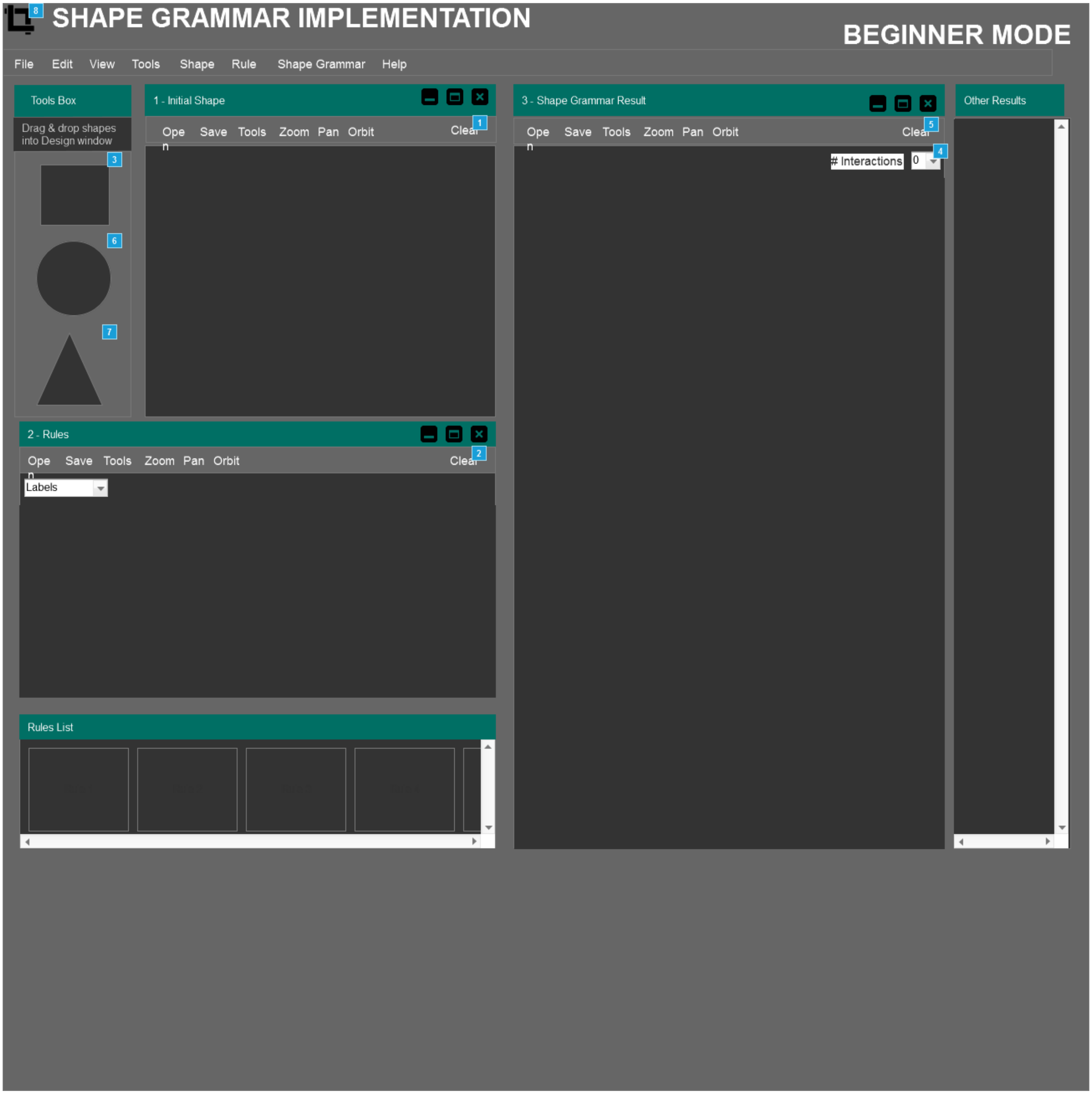
## 1.3.

## 1.4.

## 1.5.

## 1.6. Shape Grammar - Begginer

### 1.6.1. User Interface


SHAPE GRAMMAR IMPLEMENTATION
BEGINNER MODE
File Edit View Tools Shape Rule Shape Grammar Help
Tools Box
Drag & drop shapes into Design window
1 - Initial Shape
Ope n Save Tools Zoom Pan Orbit Clear
3 - Shape Grammar Result
# Interactions 0
Other Results
2 - Rules
Labels
Rules List
1
2
3
4
5
6
7
8

### 1.6.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Set Initial shape to Vazio |

| Footnote | Name | Interactions |
|---|---|---|
| 2 | | OnClick:<br>Case 1:<br>Set Rules to Vazio |
| 3 | quadrado | OnDrag:<br>Case 1:<br>Move This with drag<br>OnDragDrop:<br>Case 1<br>(If area of This is over area of Initial shape and state of Rules equals Vazio):<br>Set Initial shape to Quadrado,<br>Rules to Quadrado<br>Move This to x,y before drag<br>Case 2<br>(Else If area of This is over area of Initial shape and state of Rules equals Quadrado):<br>Set Initial shape to Quadrado,<br>Rules to Quadrado + Quadrado<br>Move This to x,y before drag<br>Case 3<br>(Else If area of This is over area of Initial shape and state of Rules equals Circulo):<br>Set Initial shape to Quadrado,<br>Rules to Circulo + Quadrado<br>Move This to x,y before drag<br>Case 4<br>(Else If area of This is over area of Initial shape and state of Rules equals Triangulo):<br>Set Initial shape to Quadrado,<br>Rules to Triangulo + Quadrado<br>Move This to x,y before drag<br>Case 5<br>(Else If area of This is over area of Rules and state of Rules equals Vazio):<br>Set Rules to Quadrado<br>Move This to x,y before drag<br>Case 6<br>(Else If area of This is over area of Rules and state of Rules equals Quadrado):<br>Set Rules to Quadrado + Quadrado<br>Move This to x,y before drag<br>Case 6<br>(Else If area of This is over area of Rules and state of Rules equals Circulo):<br>Set Rules to Circulo + Quadrado<br>Move This to x,y before drag<br>Case 7<br>(Else If area of This is over area of Rules and state of Rules equals Triangulo):<br>Set Rules to Triangulo + Quadrado<br>Move This to x,y before drag |

| Footnote | Name | Interactions |
|---|---|---|
| 4 | | OnSelectionChange:<br>Case 1<br>(If state of Rules equals Quadrado + Quadrado and selected option of This equals 1):<br>Set Shape Grammar Result to Dois quadrados<br>Set other_results to Dois quadrados<br>Case 2<br>(Else If state of Rules equals Quadrado + Quadrado and selected option of This equals 2):<br>Set Shape Grammar Result to Três quadrados<br>Set other_results to Três quadrados<br>Case 3<br>(Else If state of Rules equals Quadrado + Quadrado and selected option of This equals 3):<br>Set Shape Grammar Result to Quatro quadrados<br>Set other_results to Quatro quadrados<br>Case 4<br>(Else If state of Rules equals Quadrado + Quadrado and selected option of This equals 4):<br>Set Shape Grammar Result to Cinco quadrados<br>Set other_results to Cinco quadrados<br>Case 6<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 1):<br>Set Shape Grammar Result to Quadrado + um circulo<br>Set other_results to Quadrado + um circulo<br>Case 7<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 2):<br>Set Shape Grammar Result to Quadrado + dois circulo,<br>other_results to Quadrado + dois circulos<br>Case 8<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 3):<br>Set Shape Grammar Result to Quadrado + três circulo,<br>other_results to Quadrado + três circulos<br>Case 9<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 4):<br>Set Shape Grammar Result to Quadrado + quatro circulo,<br>other_results to Quadrado + quatro circulos |
| 5 | | OnClick:<br>Case 1:<br>Set Shape Grammar Result to Vazio,<br>other_results to Vazio |

| Footnote | Name | Interactions |
|---|---|---|
| 6 | circulo | OnDrag:<br>Case 1:<br>Move This with drag<br>OnDragDrop:<br>Case 1<br>(If area of This is over area of Initial shape and state of Rules equals Vazio):<br>Set Initial shape to Circulo,<br>Rules to Circulo<br>Move This to x,y before drag<br>Case 2<br>(Else If area of This is over area of Initial shape and state of Rules equals Quadrado):<br>Set Initial shape to Circulo,<br>Rules to Quadrado + Circulo<br>Move This to x,y before drag<br>Case 3<br>(Else If area of This is over area of Initial shape and state of Rules equals Circulo):<br>Set Initial shape to Circulo,<br>Rules to Circulo + Circulo<br>Move This to x,y before drag<br>Case 4<br>(Else If area of This is over area of Initial shape and state of Rules equals Triangulo):<br>Set Initial shape to Circulo,<br>Rules to Triangulo + Circulo<br>Move This to x,y before drag<br>Case 5<br>(Else If area of This is over area of Rules and state of Rules equals Vazio):<br>Set Rules to Circulo<br>Move This to x,y before drag<br>Case 6<br>(Else If area of This is over area of Rules and state of Rules equals Quadrado):<br>Set Rules to Quadrado + Circulo<br>Move This to x,y before drag<br>Case 6<br>(Else If area of This is over area of Rules and state of Rules equals Circulo):<br>Set Rules to Circulo + Circulo<br>Move This to x,y before drag<br>Case 7<br>(Else If area of This is over area of Rules and state of Rules equals Triangulo):<br>Set Rules to Triangulo + Circulo<br>Move This to x,y before drag |

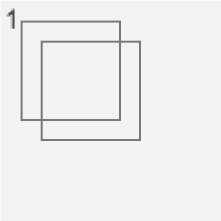
| Footnote | Name | Interactions |
|---|---|---|
| 7 | triangulo | OnDrag:<br>Case 1:<br>Move This with drag<br>OnDragDrop:<br>Case 1<br>(If area of This is over area of Initial shape and state of Rules equals Vazio):<br>Set Initial shape to Triangulo,<br>Rules to Triangulo<br>Move This to x,y before drag<br>Case 2<br>(Else If area of This is over area of Initial shape and state of Rules equals Quadrado):<br>Set Initial shape to Triangulo,<br>Rules to Quadrado + Triangulo<br>Move This to x,y before drag<br>Case 3<br>(Else If area of This is over area of Initial shape and state of Rules equals Circulo):<br>Set Initial shape to Triangulo,<br>Rules to Circulo + Triangulo<br>Move This to x,y before drag<br>Case 4<br>(Else If area of This is over area of Initial shape and state of Rules equals Triangulo):<br>Set Initial shape to Triangulo,<br>Rules to Triangulo + Triangulo<br>Move This to x,y before drag<br>Case 5<br>(Else If area of This is over area of Rules and state of Rules equals Vazio):<br>Set Rules to Triangulo<br>Move This to x,y before drag<br>Case 6<br>(Else If area of This is over area of Rules and state of Rules equals Quadrado):<br>Set Rules to Quadrado + Triangulo<br>Move This to x,y before drag<br>Case 6<br>(Else If area of This is over area of Rules and state of Rules equals Circulo):<br>Set Rules to Circulo + Triangulo<br>Move This to x,y before drag<br>Case 7<br>(Else If area of This is over area of Rules and state of Rules equals Triangulo):<br>Set Rules to Triangulo + Triangulo<br>Move This to x,y before drag |
| 8 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |

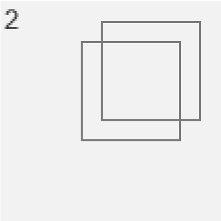
### 1.6.3. other_results

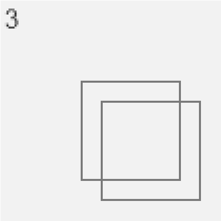
#### *1.6.3.1. Vazio*

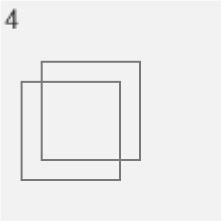
#### *1.6.3.2. Dois quadrados*

#### *1.6.3.3. User Interface*



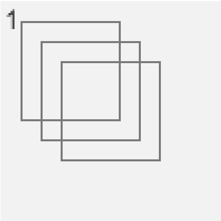


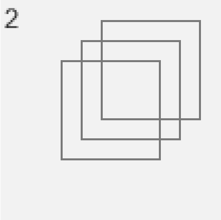


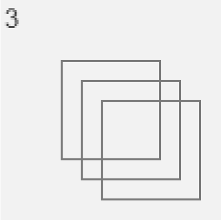


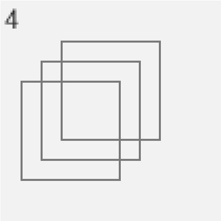
### 1.6.3.4. Três quadrados

### 1.6.3.5. User Interface



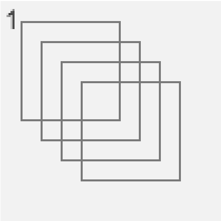


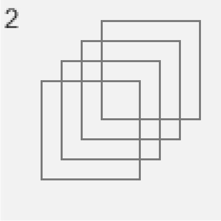


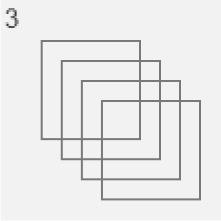


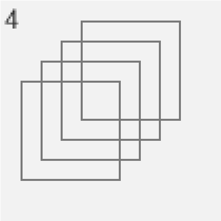
#### *1.6.3.6. Quatro quadrados*

#### *1.6.3.7. User Interface*



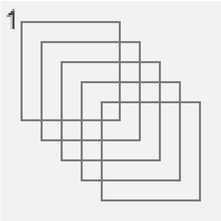


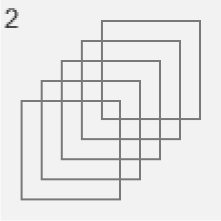


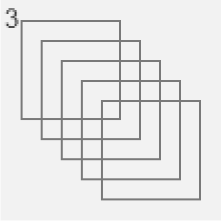


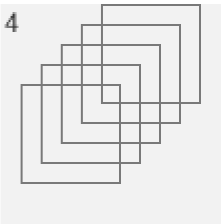
### 1.6.3.8. Cinco quadrados

### 1.6.3.9. User Interface



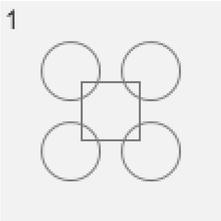


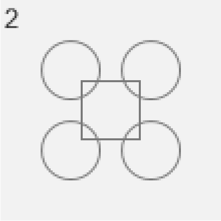


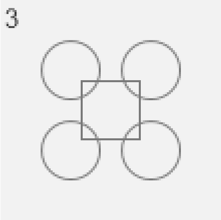


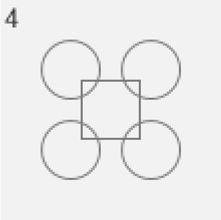
### *1.6.3.10. Quadrado + quatro circulos*

### *1.6.3.11. User Interface*



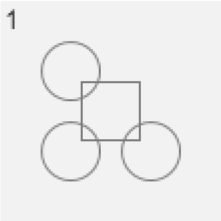


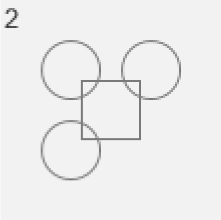


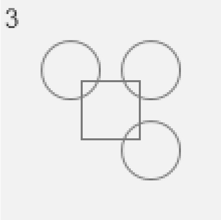


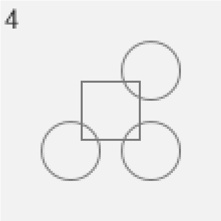
### *1.6.3.12. Quadrado + três circulos*

### *1.6.3.13. User Interface*



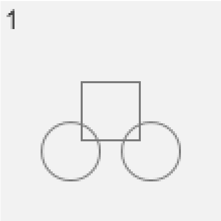


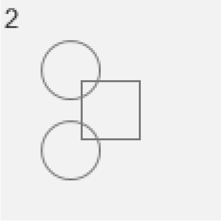


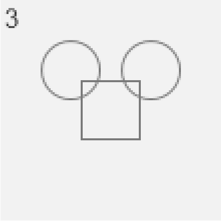


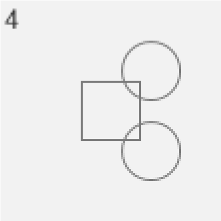
### *1.6.3.14. Quadrado + dois circulos*

### *1.6.3.15. User Interface*



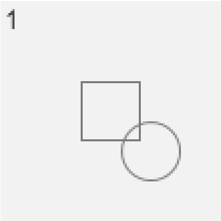


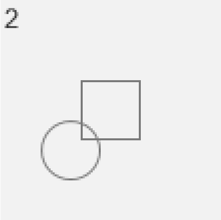


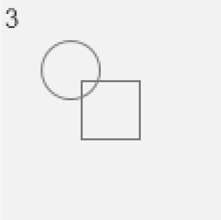


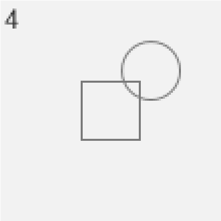
#### *1.6.3.16. Quadrado + um circulo*

#### *1.6.3.17. User Interface*









### 1.6.4. Rules list

#### *1.6.4.1. State1*

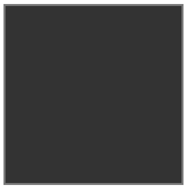
#### *1.6.4.2. User Interface*

### 1.6.5. Initial shape

#### *1.6.5.1. Vazio*

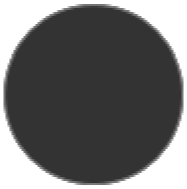
#### *1.6.5.2. Quadrado*

#### *1.6.5.3. User Interface*



#### *1.6.5.4. Circulo*

#### *1.6.5.5. User Interface*



#### *1.6.5.6. Triangulo*

#### *1.6.5.7. User Interface*

### 1.6.6. quadrado

#### *1.6.6.1. State1*

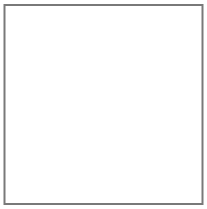
#### *1.6.6.2. User Interface*

### 1.6.7. Shape Grammar Result

#### *1.6.7.1. Vazio*

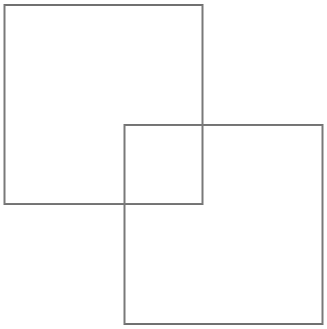
#### *1.6.7.2. Um quadrado*

#### *1.6.7.3. User Interface*



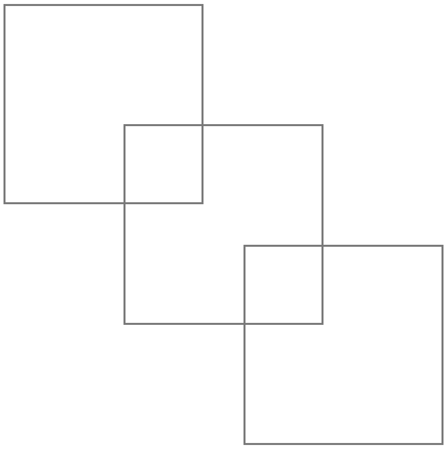
#### *1.6.7.4. Dois quadrados*

#### *1.6.7.5. User Interface*



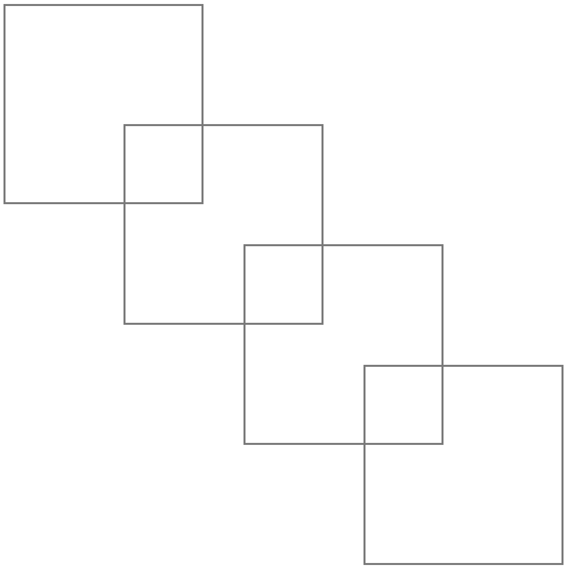
#### *1.6.7.6. Três quadrados*

#### *1.6.7.7. User Interface*



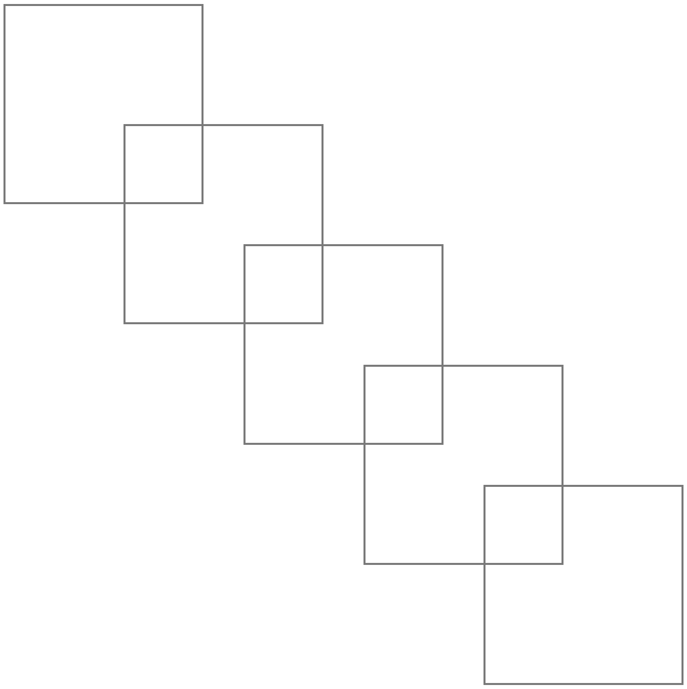
### 1.6.7.8. Quatro quadrados

### 1.6.7.9. User Interface



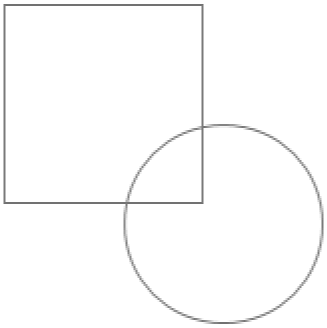
### 1.6.7.10. Cinco quadrados

### 1.6.7.11. User Interface



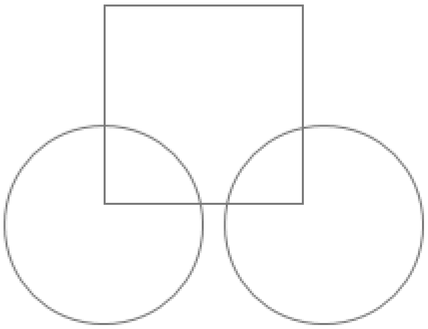
### 1.6.7.12. Quadrado + um circulo

### 1.6.7.13. User Interface



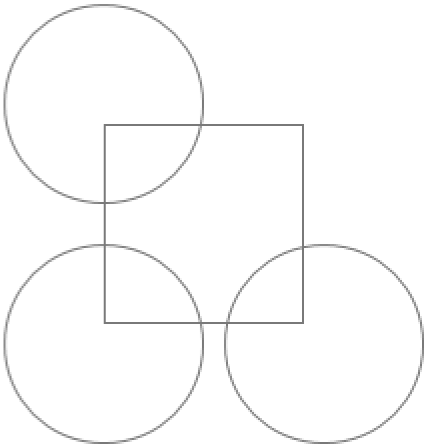
### 1.6.7.14. Quadrado + dois circulo

### 1.6.7.15. User Interface



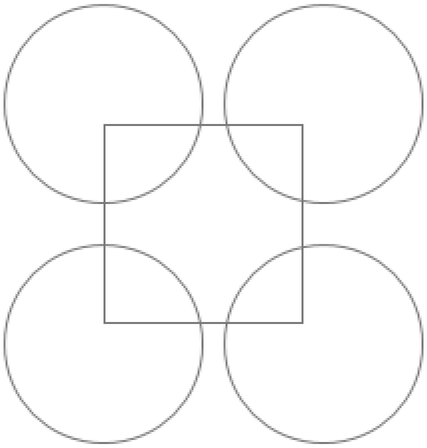
### 1.6.7.16. Quadrado + três circulo

### 1.6.7.17. User Interface



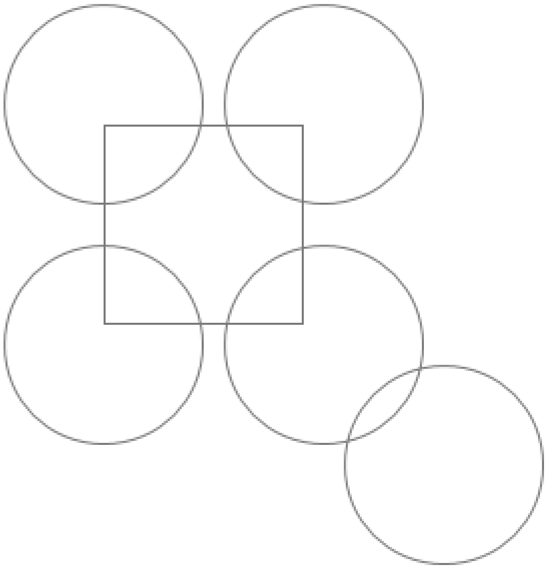
### 1.6.7.18. Quadrado + quatro circulo

### 1.6.7.19. User Interface



#### *1.6.7.20. Quadrado + cinco circulos*

#### *1.6.7.21. User Interface*

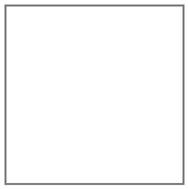


### 1.6.8. Rules

#### *1.6.8.1. Vazio*

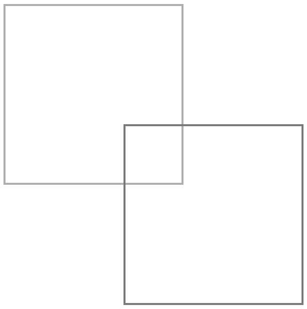
#### *1.6.8.2. Quadrado*

#### *1.6.8.3. User Interface*



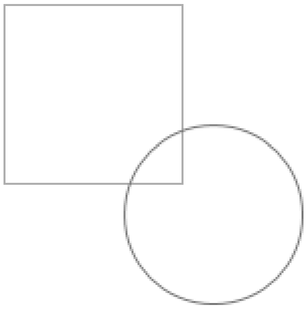
### 1.6.8.4. Quadrado + Quadrado

### 1.6.8.5. User Interface



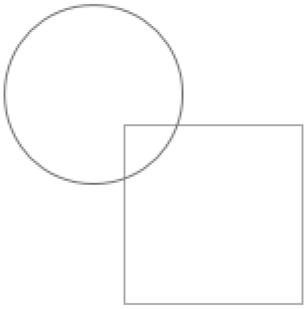
### 1.6.8.6. Quadrado + Circulo

### 1.6.8.7. User Interface



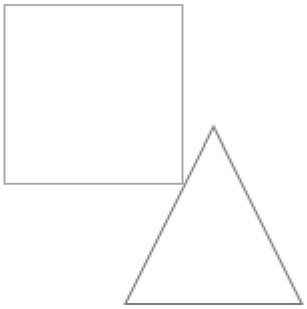
### 1.6.8.8. Circulo + Quadrado

### 1.6.8.9. User Interface



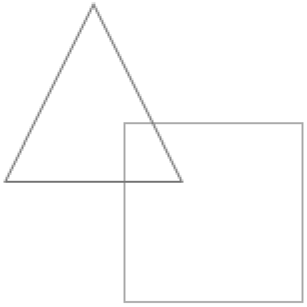
### *1.6.8.10. Quadrado + Triangulo*

### *1.6.8.11. User Interface*



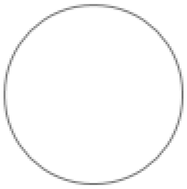
### *1.6.8.12. Triangulo + Quadrado*

### *1.6.8.13. User Interface*



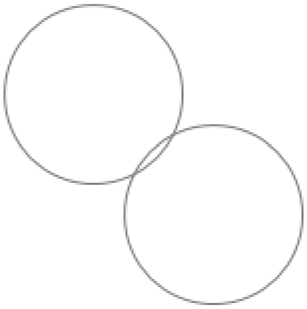
### *1.6.8.14. Circulo*

### *1.6.8.15. User Interface*



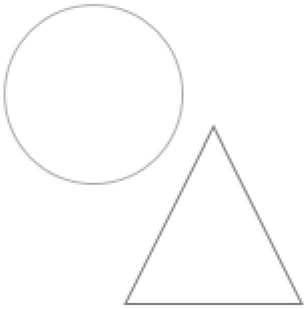
### *1.6.8.16. Circulo + Circulo*

### *1.6.8.17. User Interface*



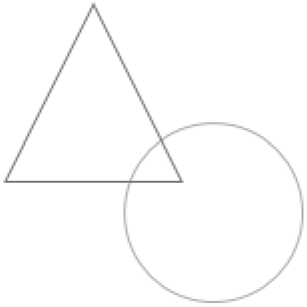
#### *1.6.8.18. Circulo + Triangulo*

#### *1.6.8.19. User Interface*



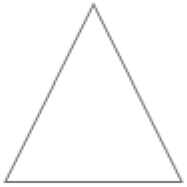
#### *1.6.8.20. Triangulo + Circulo*

#### *1.6.8.21. User Interface*



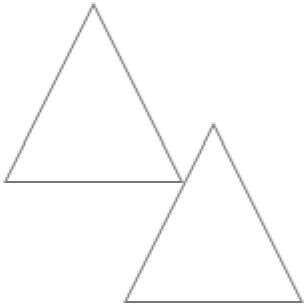
#### *1.6.8.22. Triangulo*

#### *1.6.8.23. User Interface*



#### *1.6.8.24. Triangulo + Triangulo*

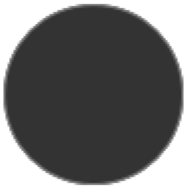
#### *1.6.8.25. User Interface*



### 1.6.9. circulo

#### *1.6.9.1. State1*

#### *1.6.9.2. User Interface*



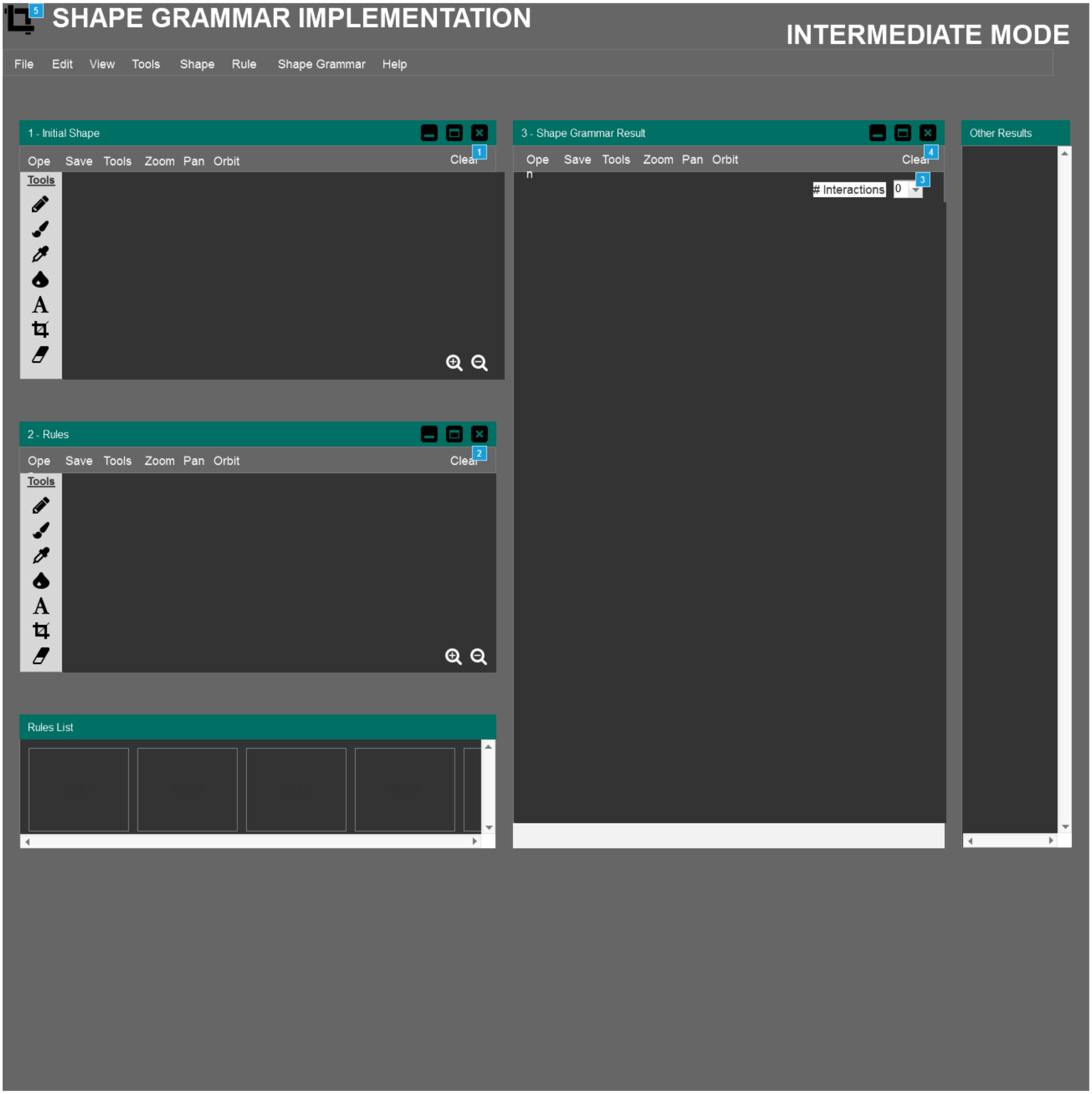
### 1.6.10. triangulo

#### *1.6.10.1. State1*

#### *1.6.10.2. User Interface*

## 1.7. Shape Grammar - Intermediate

### 1.7.1. User Interface


5
SHAPE GRAMMAR IMPLEMENTATION
INTERMEDIATE MODE
File Edit View Tools Shape Rule Shape Grammar Help
1 - Initial Shape
Ope Save Tools Zoom Pan Orbit
Clear
1
Tools
A
2 - Rules
Ope Save Tools Zoom Pan Orbit
Clear
2
Tools
A
Rules List
3 - Shape Grammar Result
Ope Save Tools Zoom Pan Orbit
n
Clear
4
# Interactions
0
3
Other Results

### 1.7.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Set Initial shape to Vazio |

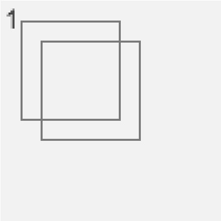
| Footnote | Name | Interactions |
|---|---|---|
| 2 | | OnClick:<br>Case 1:<br>Set Rules to Vazio |
| 3 | | OnSelectionChange:<br>Case 1<br>(If state of Rules equals Quadrado + Quadrado and selected option of This equals 1):<br>Set Shape Grammar Result to Dois quadrados<br>Set other_results to Dois quadrados<br>Case 2<br>(Else If state of Rules equals Quadrado + Quadrado and selected option of This equals 2):<br>Set Shape Grammar Result to Três quadrados<br>Set other_results to Três quadrados<br>Case 3<br>(Else If state of Rules equals Quadrado + Quadrado and selected option of This equals 3):<br>Set Shape Grammar Result to Quatro quadrados<br>Set other_results to Quatro quadrados<br>Case 4<br>(Else If state of Rules equals Quadrado + Quadrado and selected option of This equals 4):<br>Set Shape Grammar Result to Cinco quadrados<br>Set other_results to Cinco quadrados<br>Case 6<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 1):<br>Set Shape Grammar Result to Quadrado + um circulo<br>Set other_results to Quadrado + um circulo<br>Case 7<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 2):<br>Set Shape Grammar Result to Quadrado + dois circulo, other_results to Quadrado + dois circulos<br>Case 8<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 3):<br>Set Shape Grammar Result to Quadrado + três circulo, other_results to Quadrado + três circulos<br>Case 9<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 4):<br>Set Shape Grammar Result to Quadrado + quatro circulo, other_results to Quadrado + quatro circulos |
| 4 | | OnClick:<br>Case 1:<br>Set Shape Grammar Result to Vazio, other_results to Vazio |
| 5 | | OnClick:<br>Case 1:<br>Open Home in Current Window |

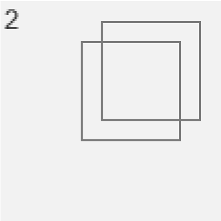
### 1.7.3. other_results

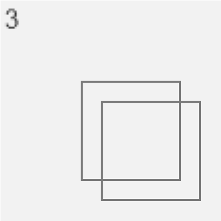
#### *1.7.3.1. Vazio*

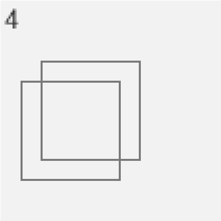
#### *1.7.3.2. Dois quadrados*

#### *1.7.3.3. User Interface*



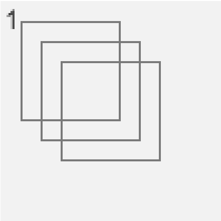


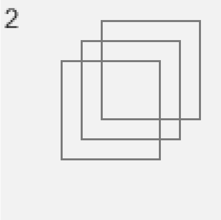


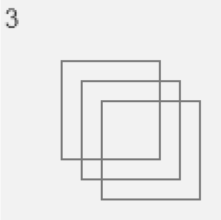


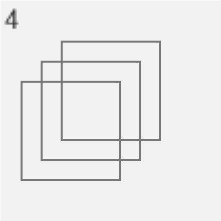
#### 1.7.3.4. Três quadrados

#### 1.7.3.5. User Interface



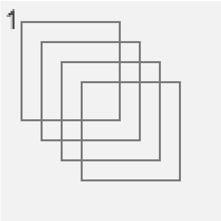


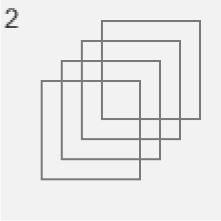


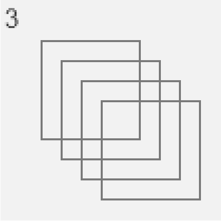


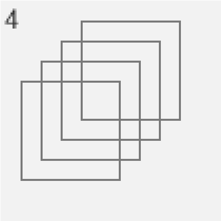
### 1.7.3.6. Quatro quadrados

### 1.7.3.7. User Interface



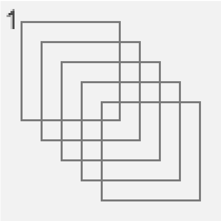


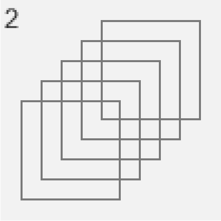


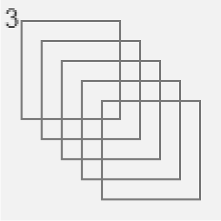


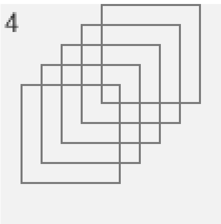
### *1.7.3.8. Cinco quadrados*

### *1.7.3.9. User Interface*



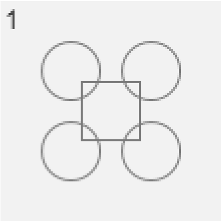


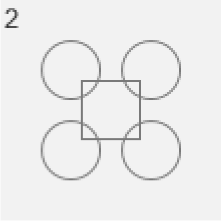


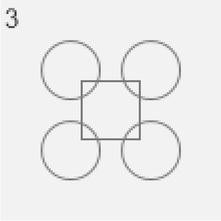


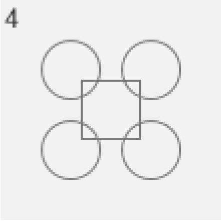
### *1.7.3.10. Quadrado + quatro circulos*

### *1.7.3.11. User Interface*



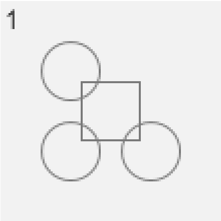


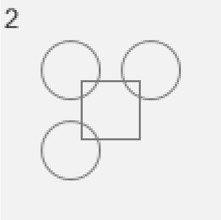


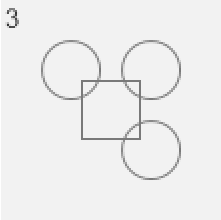


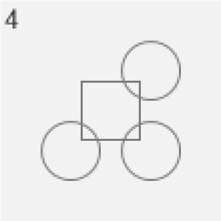
### 1.7.3.12. Quadrado + três circulos

### 1.7.3.13. User Interface



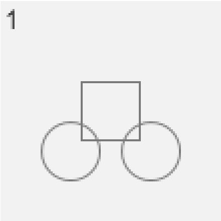


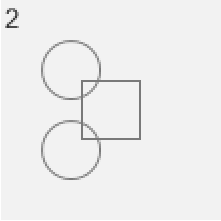


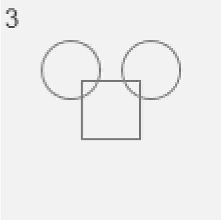


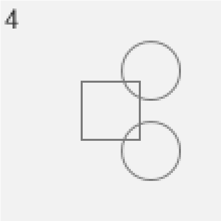
### *1.7.3.14. Quadrado + dois circulos*

### *1.7.3.15. User Interface*



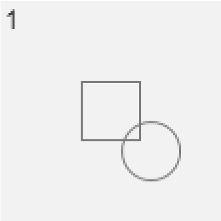


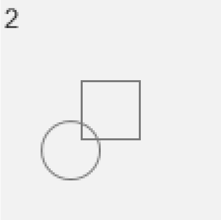


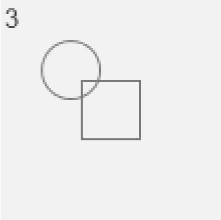


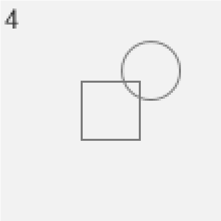
### *1.7.3.16. Quadrado + um circulo*

### *1.7.3.17. User Interface*









## 1.7.4. Rules list

### *1.7.4.1. State1*

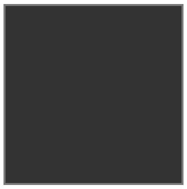
### *1.7.4.2. User Interface*

### 1.7.5. Initial shape

#### *1.7.5.1. Vazio*

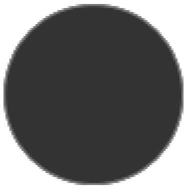
#### *1.7.5.2. Quadrado*

#### *1.7.5.3. User Interface*



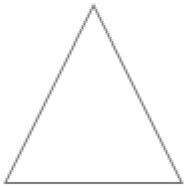
#### *1.7.5.4. Circulo*

#### *1.7.5.5. User Interface*



#### *1.7.5.6. Triangulo*

#### *1.7.5.7. User Interface*

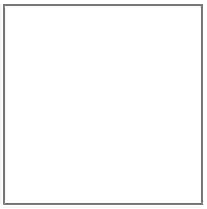


### 1.7.6. Shape Grammar Result

#### *1.7.6.1. Vazio*

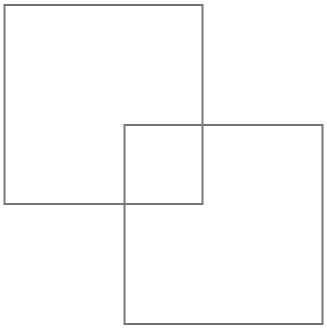
#### *1.7.6.2. Um quadrado*

#### *1.7.6.3. User Interface*



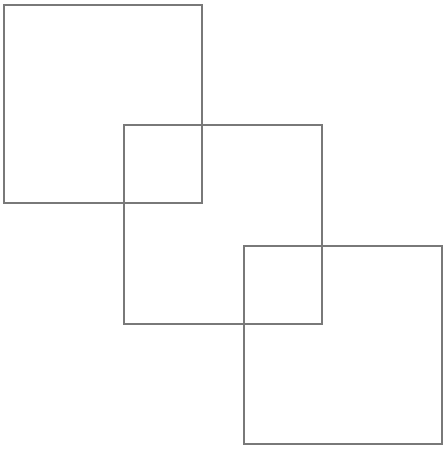
#### *1.7.6.4. Dois quadrados*

#### *1.7.6.5. User Interface*



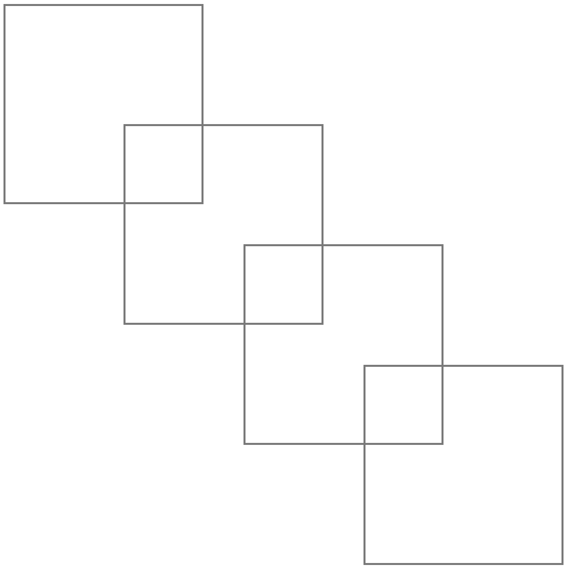
#### *1.7.6.6. Três quadrados*

#### *1.7.6.7. User Interface*



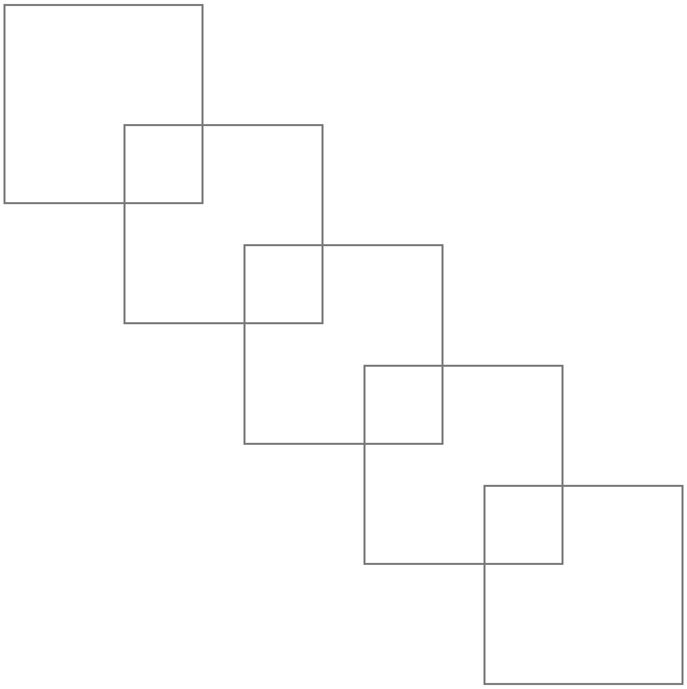
#### 1.7.6.8. Quatro quadrados

#### 1.7.6.9. User Interface



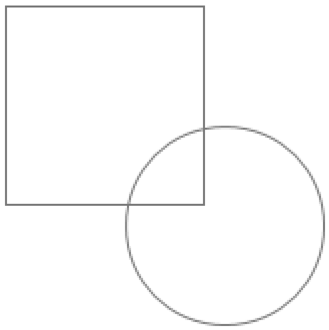
#### 1.7.6.10. Cinco quadrados

#### 1.7.6.11. User Interface



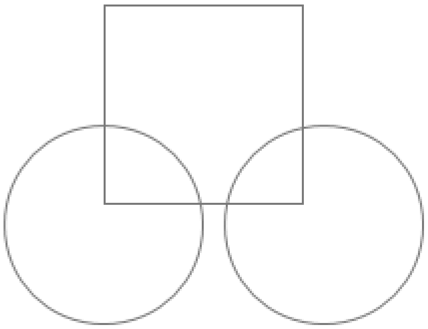
### 1.7.6.12. Quadrado + um circulo

### 1.7.6.13. User Interface



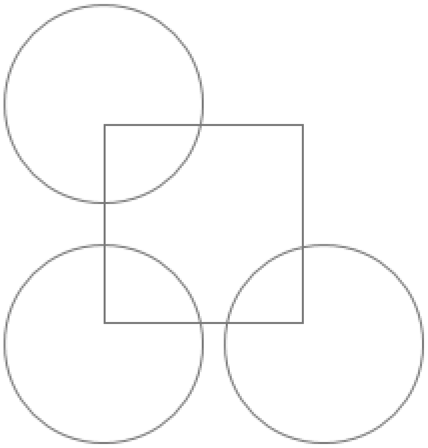
### 1.7.6.14. Quadrado + dois circulo

### 1.7.6.15. User Interface



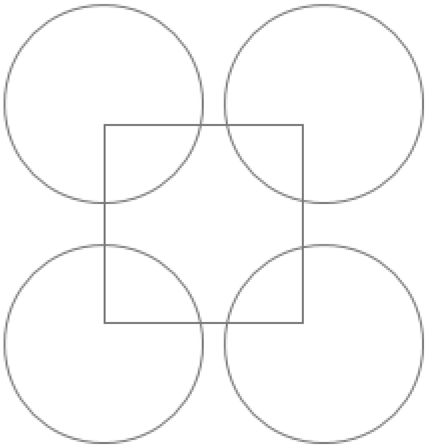
### *1.7.6.16. Quadrado + três circulo*

### *1.7.6.17. User Interface*



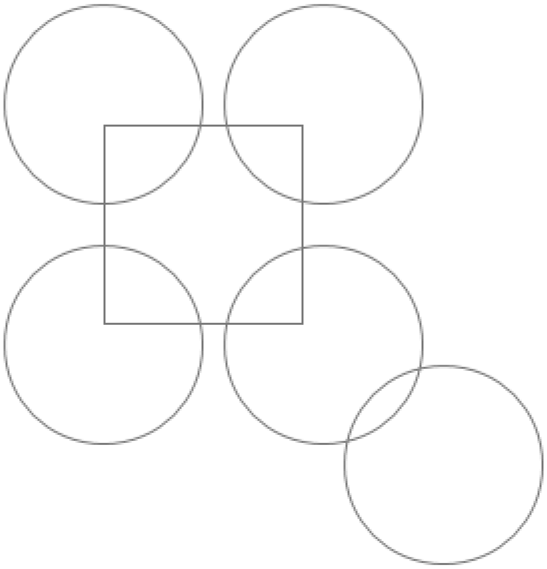
### 1.7.6.18. Quadrado + quatro circulo

### 1.7.6.19. User Interface



#### *1.7.6.20. Quadrado + cinco circulos*

#### *1.7.6.21. User Interface*

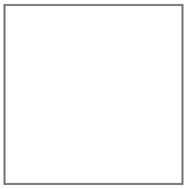


### 1.7.7. Rules

#### *1.7.7.1. Vazio*

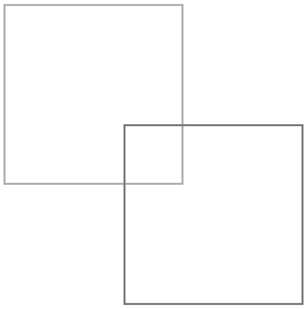
#### *1.7.7.2. Quadrado*

#### *1.7.7.3. User Interface*



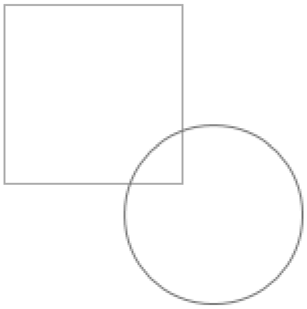
### 1.7.7.4. Quadrado + Quadrado

### 1.7.7.5. User Interface



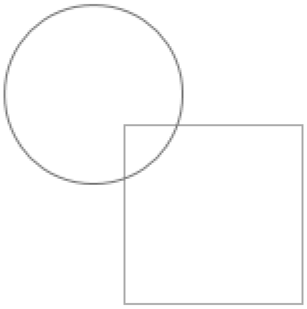
### 1.7.7.6. Quadrado + Circulo

### 1.7.7.7. User Interface



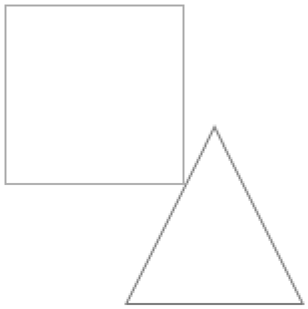
### 1.7.7.8. Circulo + Quadrado

### 1.7.7.9. User Interface



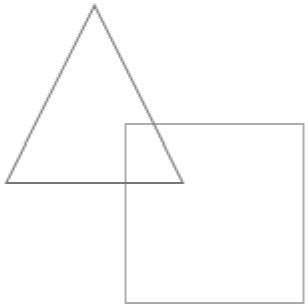
### *1.7.7.10. Quadrado + Triangulo*

### *1.7.7.11. User Interface*



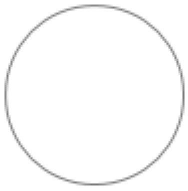
### *1.7.7.12. Triangulo + Quadrado*

### *1.7.7.13. User Interface*



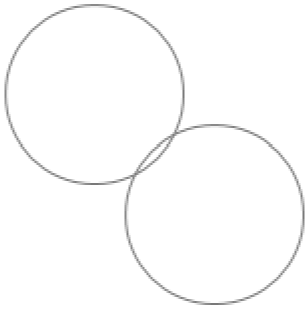
### *1.7.7.14. Circulo*

### *1.7.7.15. User Interface*



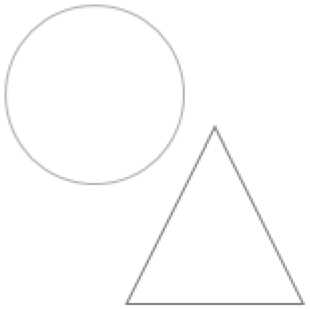
### *1.7.7.16. Circulo + Circulo*

### *1.7.7.17. User Interface*



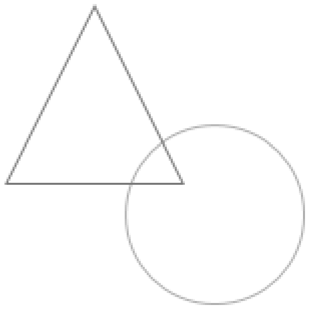
### 1.7.7.18. Circulo + Triangulo

### 1.7.7.19. User Interface



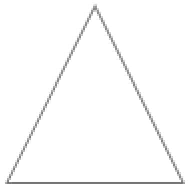
### 1.7.7.20. Triangulo + Circulo

### 1.7.7.21. User Interface



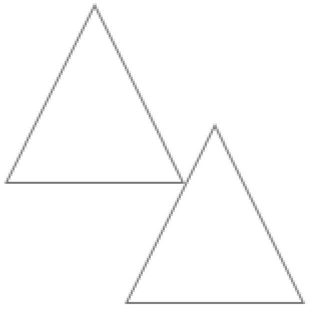
### 1.7.7.22. Triangulo

### 1.7.7.23. User Interface



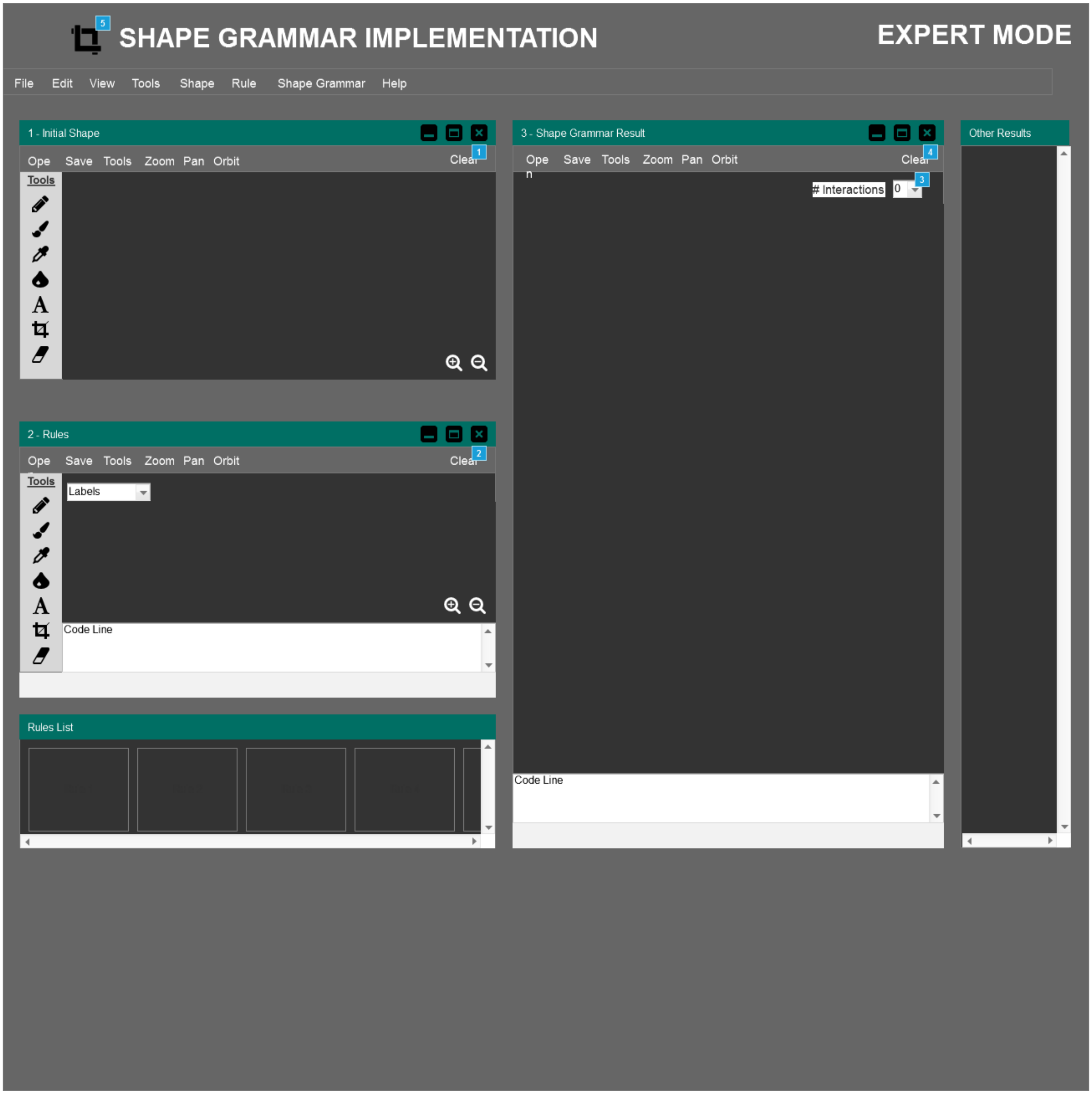
### 1.7.7.24. Triangulo + Triangulo

### 1.7.7.25. User Interface



## 1.8. Shape Grammar - Expert

### 1.8.1. User Interface



### 1.8.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Set Initial shape to Vazio |

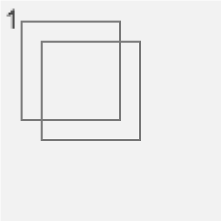
| Footnote | Name | Interactions |
|---|---|---|
| 2 | | OnClick:<br>Case 1:<br>Set Rules to Vazio |
| 3 | | OnSelectionChange:<br>Case 1<br>(If state of Rules equals Quadrado + Quadrado and selected option of This equals 1):<br>Set Shape Grammar Result to Dois quadrados<br>Set other_results to Dois quadrados<br>Case 2<br>(Else If state of Rules equals Quadrado + Quadrado and selected option of This equals 2):<br>Set Shape Grammar Result to Três quadrados<br>Set other_results to Três quadrados<br>Case 3<br>(Else If state of Rules equals Quadrado + Quadrado and selected option of This equals 3):<br>Set Shape Grammar Result to Quatro quadrados<br>Set other_results to Quatro quadrados<br>Case 4<br>(Else If state of Rules equals Quadrado + Quadrado and selected option of This equals 4):<br>Set Shape Grammar Result to Cinco quadrados<br>Set other_results to Cinco quadrados<br>Case 6<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 1):<br>Set Shape Grammar Result to Quadrado + um circulo<br>Set other_results to Quadrado + um circulo<br>Case 7<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 2):<br>Set Shape Grammar Result to Quadrado + dois circulo, other_results to Quadrado + dois circulos<br>Case 8<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 3):<br>Set Shape Grammar Result to Quadrado + três circulo, other_results to Quadrado + três circulos<br>Case 9<br>(Else If state of Rules equals Quadrado + Circulo and selected option of This equals 4):<br>Set Shape Grammar Result to Quadrado + quatro circulo, other_results to Quadrado + quatro circulos |
| 4 | | OnClick:<br>Case 1:<br>Set Shape Grammar Result to Vazio, other_results to Vazio |
| 5 | | OnClick:<br>Case 1:<br>Open Home in Current Window |

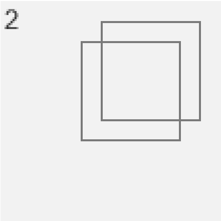
### 1.8.3. other_results

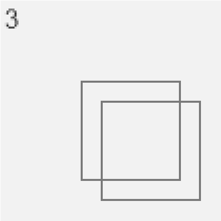
#### *1.8.3.1. Vazio*

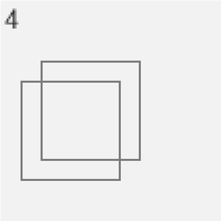
#### *1.8.3.2. Dois quadrados*

#### *1.8.3.3. User Interface*



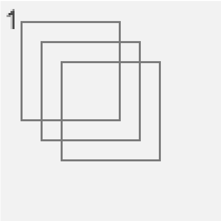


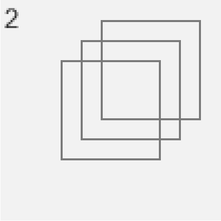


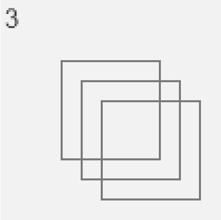


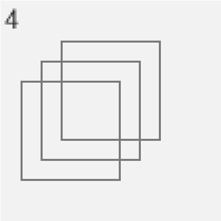
### 1.8.3.4. Três quadrados

### 1.8.3.5. User Interface



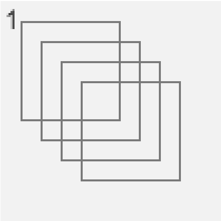


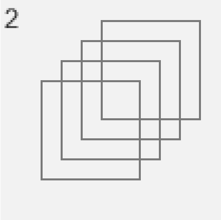


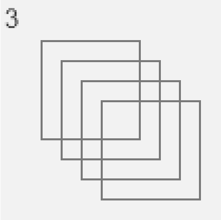


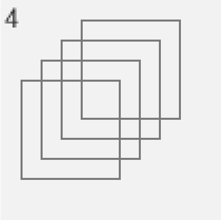
### 1.8.3.6. Quatro quadrados

### 1.8.3.7. User Interface



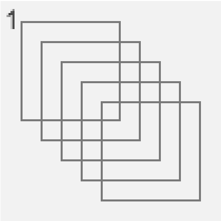


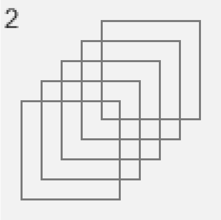


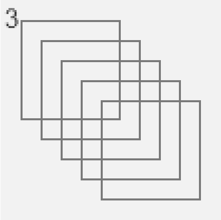


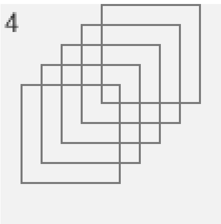
### 1.8.3.8. Cinco quadrados

### 1.8.3.9. User Interface



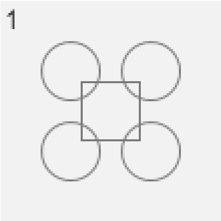


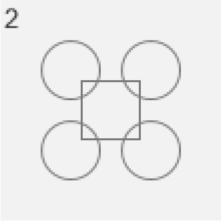


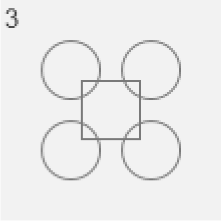


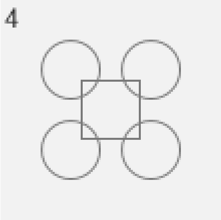
### 1.8.3.10. Quadrado + quatro circulos

### 1.8.3.11. User Interface



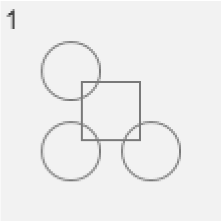


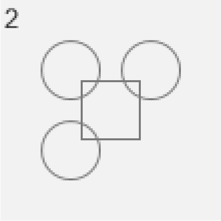


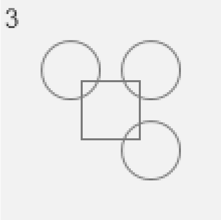


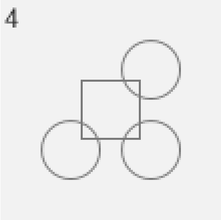
### *1.8.3.12. Quadrado + três circulos*

### *1.8.3.13. User Interface*



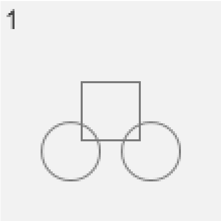


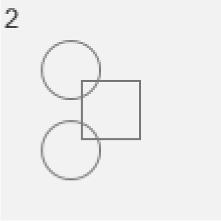


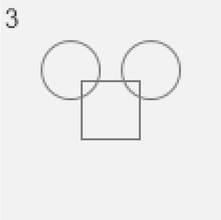


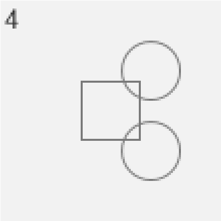
### 1.8.3.14. Quadrado + dois circulos

### 1.8.3.15. User Interface



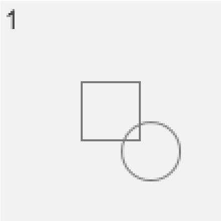


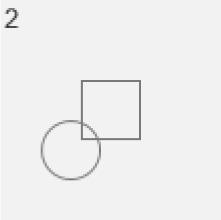


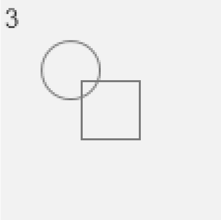


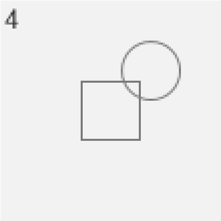
### 1.8.3.16. Quadrado + um circulo

### 1.8.3.17. User Interface









## 1.8.4. Rules list

### 1.8.4.1. State1

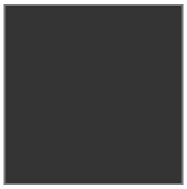
### 1.8.4.2. User Interface

### 1.8.5. Initial shape

#### *1.8.5.1. Vazio*

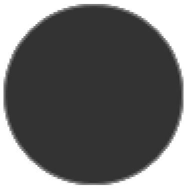
#### *1.8.5.2. Quadrado*

#### *1.8.5.3. User Interface*



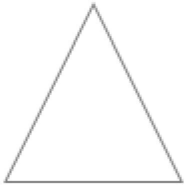
#### *1.8.5.4. Circulo*

#### *1.8.5.5. User Interface*



#### *1.8.5.6. Triangulo*

#### *1.8.5.7. User Interface*

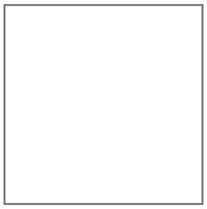


### 1.8.6. Shape Grammar Result

#### *1.8.6.1. Vazio*

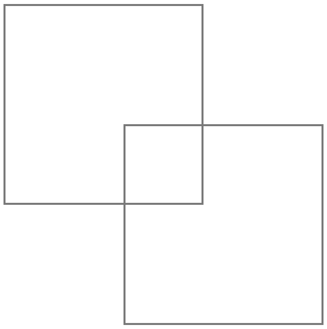
#### *1.8.6.2. Um quadrado*

#### *1.8.6.3. User Interface*



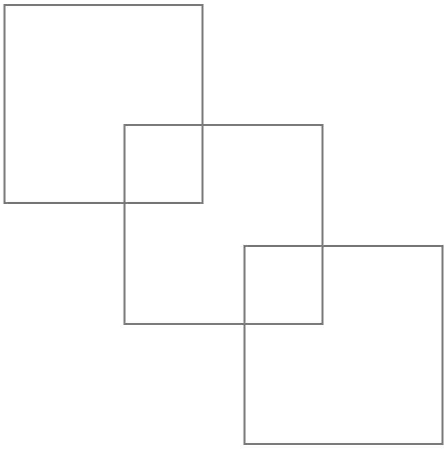
#### *1.8.6.4. Dois quadrados*

#### *1.8.6.5. User Interface*



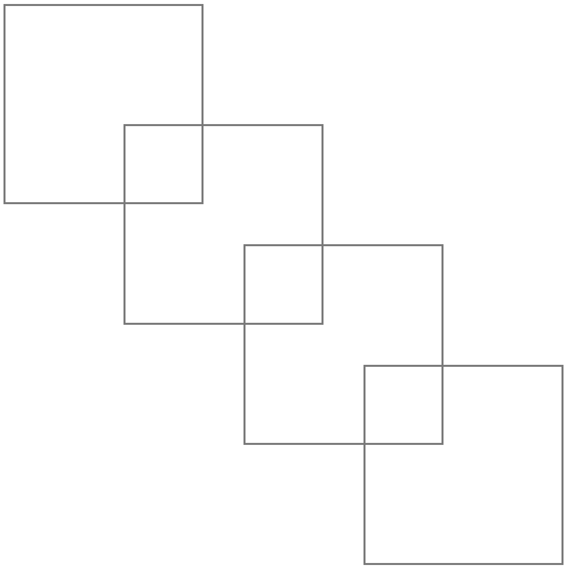
#### *1.8.6.6. Três quadrados*

#### *1.8.6.7. User Interface*



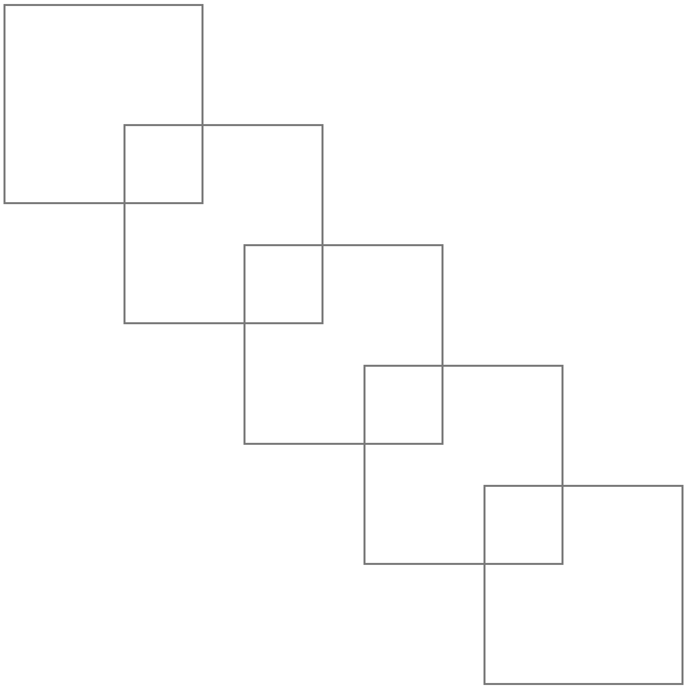
#### 1.8.6.8. Quatro quadrados

#### 1.8.6.9. User Interface



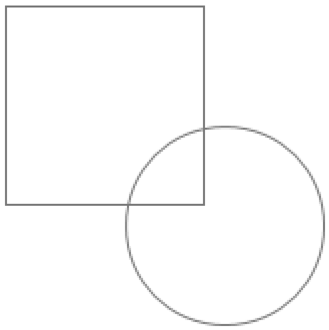
#### 1.8.6.10. Cinco quadrados

#### 1.8.6.11. User Interface



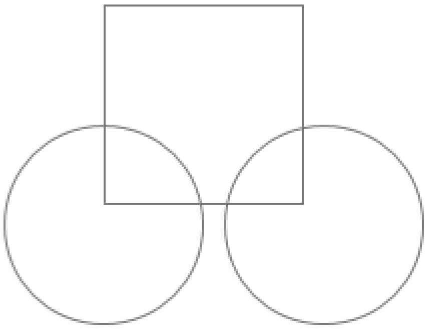
### *1.8.6.12. Quadrado + um circulo*

### *1.8.6.13. User Interface*



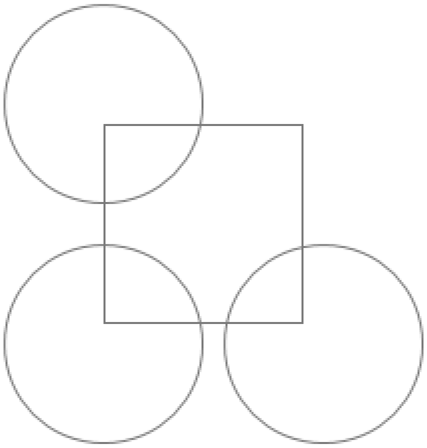
### 1.8.6.14. Quadrado + dois circulo

### 1.8.6.15. User Interface



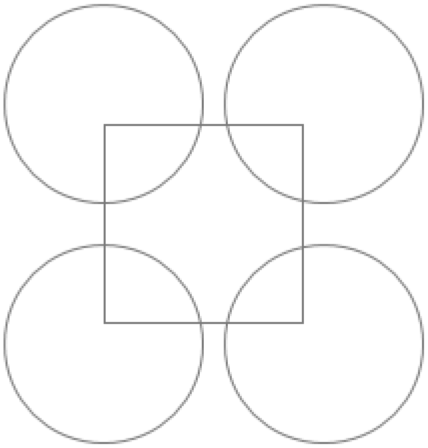
### 1.8.6.16. Quadrado + três circulo

### 1.8.6.17. User Interface



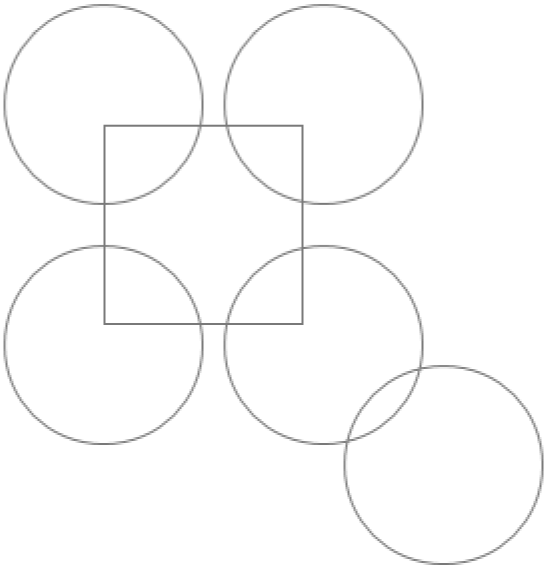
### 1.8.6.18. Quadrado + quatro circulo

### 1.8.6.19. User Interface



#### *1.8.6.20. Quadrado + cinco circulos*

#### *1.8.6.21. User Interface*

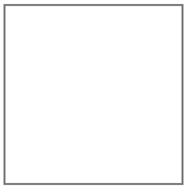


### 1.8.7. Rules

#### *1.8.7.1. Vazio*

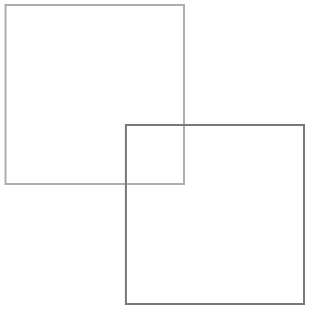
#### *1.8.7.2. Quadrado*

#### *1.8.7.3. User Interface*



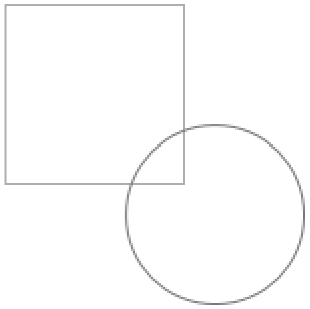
### 1.8.7.4. Quadrado + Quadrado

### 1.8.7.5. User Interface



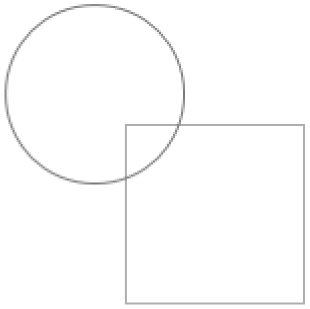
### 1.8.7.6. Quadrado + Circulo

### 1.8.7.7. User Interface



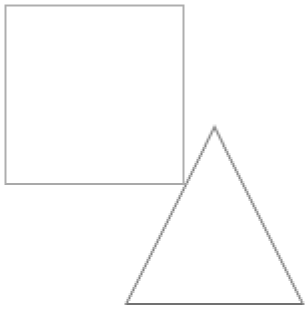
### 1.8.7.8. Circulo + Quadrado

### 1.8.7.9. User Interface



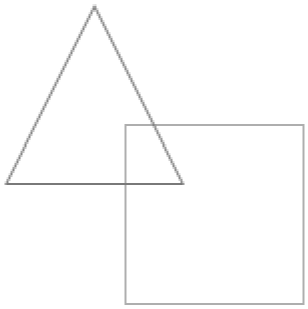
### *1.8.7.10. Quadrado + Triangulo*

### *1.8.7.11. User Interface*



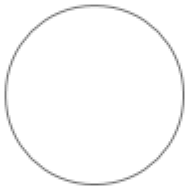
### *1.8.7.12. Triangulo + Quadrado*

### *1.8.7.13. User Interface*



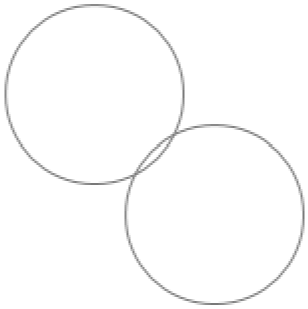
### *1.8.7.14. Circulo*

### *1.8.7.15. User Interface*



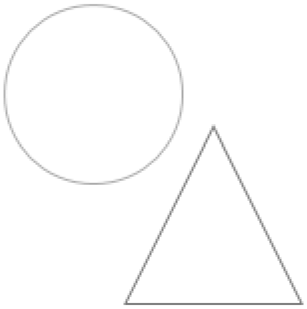
### *1.8.7.16. Circulo + Circulo*

### *1.8.7.17. User Interface*



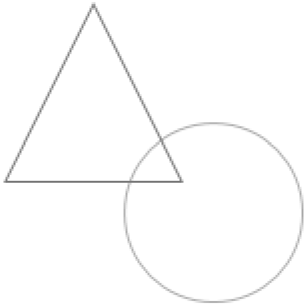
### 1.8.7.18. Circulo + Triangulo

### 1.8.7.19. User Interface



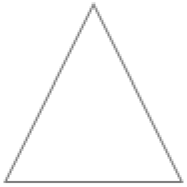
### 1.8.7.20. Triangulo + Circulo

### 1.8.7.21. User Interface



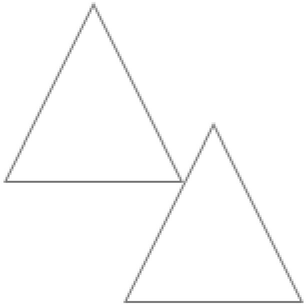
### 1.8.7.22. Triangulo

### 1.8.7.23. User Interface



### 1.8.7.24. Triangulo + Triangulo

### 1.8.7.25. User Interface



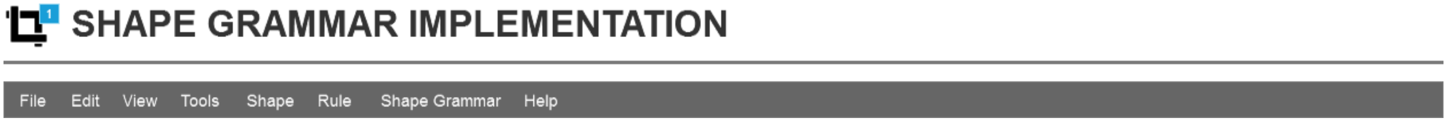
# 2. Masters

## 2.1. Master List

Topo + menu

## 2.2. Topo + menu

### 2.2.1. User Interface



### 2.2.2. Widget Table

| Footnote | Interactions |
|---|---|
| 1 | OnClick:<br>Case 1:<br>Open Home in Current Window |

# IM-sgi Prototype V02

## VERSION 02

## 1. Pages

### 1.1. Page Tree

Home
- Beginner
  - 1 - Initial Shape
  - 2 - Rules
  - 3 - Shape Grammar Result
- Intermediate
  - 1 - Initial Shape
  - 2 - Rules
  - 3 - Shape Grammar Result
- Expert
  - 1 - Initial Shape
  - 2 - Rules
  - 3 - Shape Grammar Result

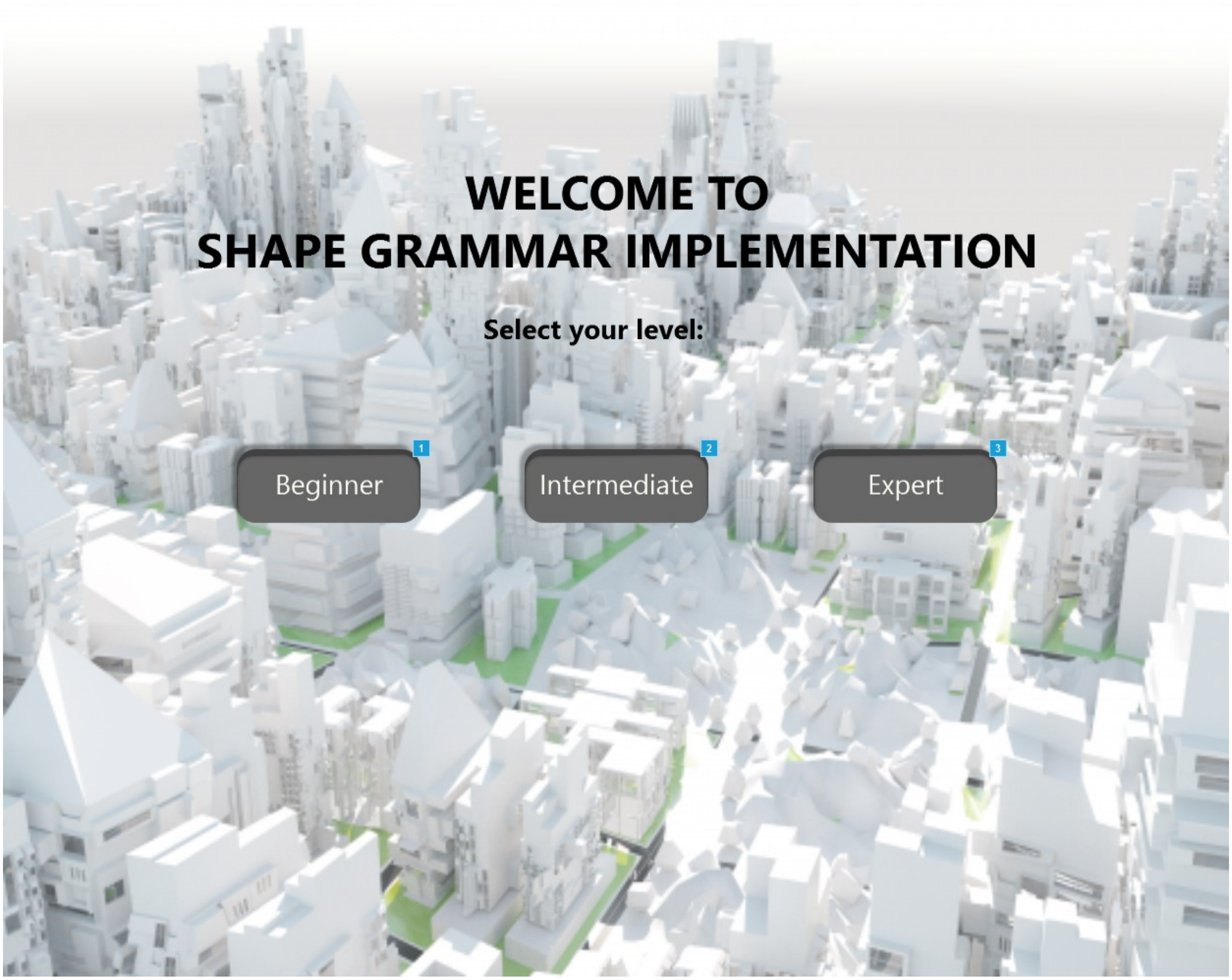
### 1.2.

## 1.3.

## 1.4.

## 1.5. Home

### 1.5.1. User Interface



WELCOME TO
SHAPE GRAMMAR IMPLEMENTATION

Select your level:

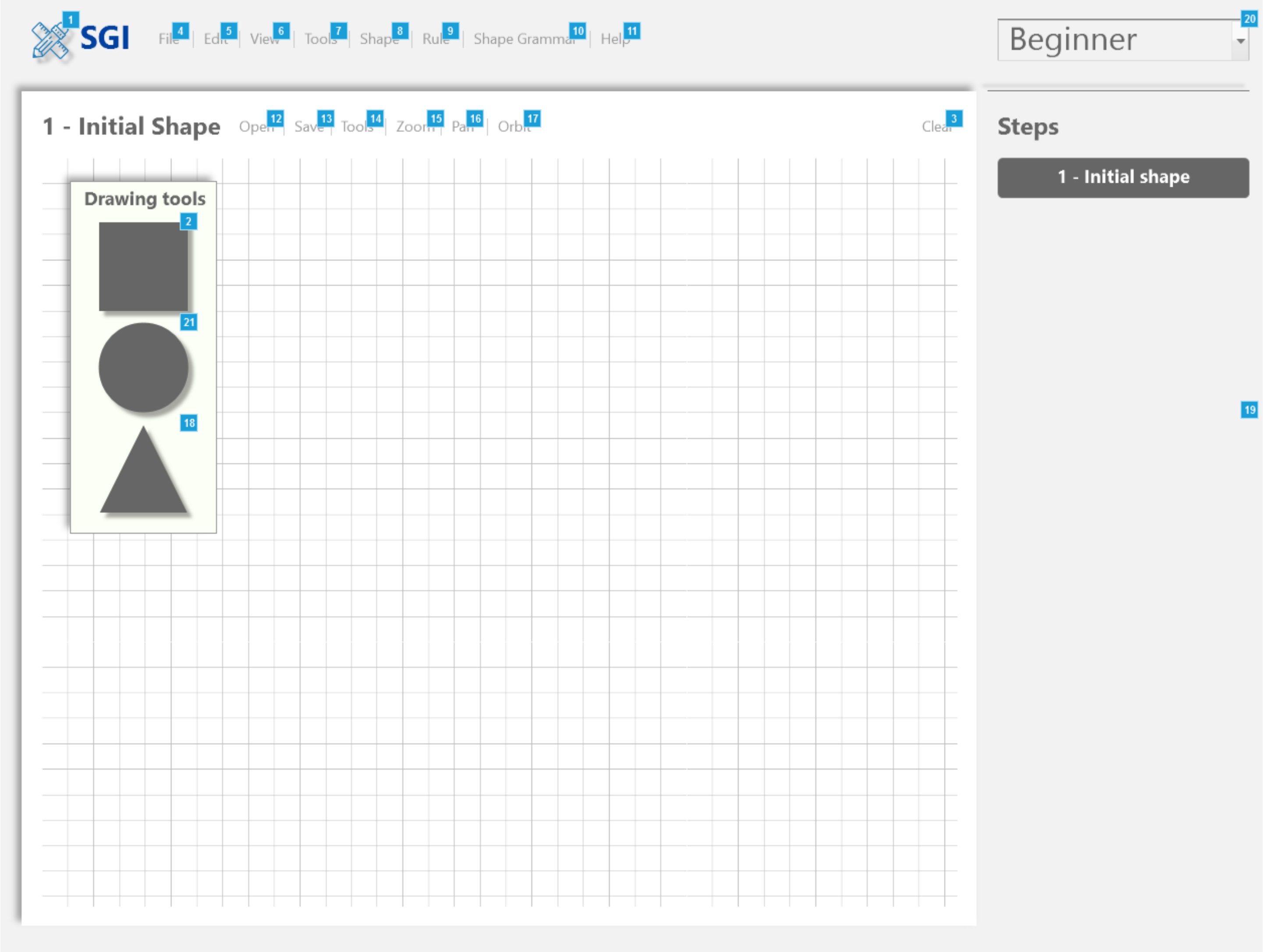
### 1.5.2. Widget Table

| Footnote | Interactions |
|---|---|
| 1 | OnClick:<br>Case 1:<br>Open 1 - Initial Shape in Current Window |
| 2 | OnClick:<br>Case 1:<br>Open 1 - Initial Shape in Current Window |
| 3 | OnClick:<br>Case 1:<br>Open 1 - Initial Shape in Current Window |

## 1.6. Beginner

## 1.7. 1 - Initial Shape

### 1.7.1. User Interface



### 1.7.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 2 | quadrado | OnDrag:<br>Case 1:<br>Move This with drag<br>OnDragDrop:<br>Case 1<br>(If area of This is over area of Initial shape):<br>Set Initial shape to Quadrado<br>Move This to x,y before drag<br>Set Telinha #1 to Quadradinho<br>Show Next step #2 fade 500 ms,<br>Telinha 1<br>Set value of f equal to "q"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms |

| Footnote | Name | Interactions |
|---|---|---|
| 3 | | OnClick:<br>Case 1:<br>Set Initial shape to Vazio<br>Hide Telinha 1 fade 500 ms,<br>Next step #2 fade 500 ms |
| 4 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 5 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 6 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 7 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 8 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 9 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 10 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 11 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 12 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 13 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 14 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 15 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 16 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 17 | | OnClick:<br>Case 1:<br>Open Home in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 18 | triangulo | OnDrag:<br>Case 1:<br>Move This with drag<br>OnDragDrop:<br>Case 1<br>(If area of This is over area of Initial shape):<br>Set Initial shape to Triangulo<br>Move This to x,y before drag<br>Set Telinha #1 to Triangulinho<br>Show Next step #2 fade 500 ms,<br>Telinha 1 fade 500 ms<br>Set value of f equal to "t"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms |
| 19 | Next step #2 | OnClick:<br>Case 1:<br>Open 2 - Rules in Current Window |
| 20 | | OnSelectionChange:<br>Case 1<br>(If selected option of This equals  Intermediate mode):<br>Open 1 - Initial Shape in Current Window<br>Case 2<br>(Else If selected option of This equals  Expert mode):<br>Open 1 - Initial Shape in Current Window<br>Case 3<br>(Else If selected option of This equals  Begginer mode):<br>Open 1 - Initial Shape in Current Window |
| 21 | circulo | OnDrag:<br>Case 1:<br>Move This with drag<br>OnDragDrop:<br>Case 1<br>(If area of This is over area of Initial shape):<br>Set Initial shape to Circulo<br>Move This to x,y before drag<br>Set Telinha #1 to Circulinho<br>Show Next step #2 fade 500 ms,<br>Telinha 1 fade 500 ms<br>Set value of f equal to "c"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms |

### 1.7.3. Unnamed

#### *1.7.3.1. State1*

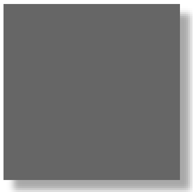
#### *1.7.3.2. User Interface*

### 1.7.4. Initial shape

#### *1.7.4.1. Vazio*

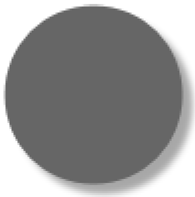
#### *1.7.4.2. Quadrado*

#### *1.7.4.3. User Interface*



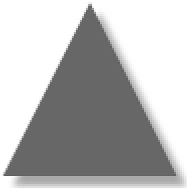
#### *1.7.4.4. Circulo*

#### *1.7.4.5. User Interface*



#### *1.7.4.6. Triangulo*

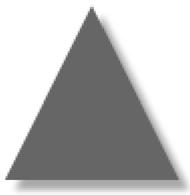
#### *1.7.4.7. User Interface*



### 1.7.5. triangulo

#### *1.7.5.1. State1*

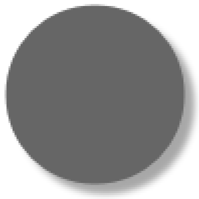
#### *1.7.5.2. User Interface*



### 1.7.6. circulo

#### *1.7.6.1. State1*

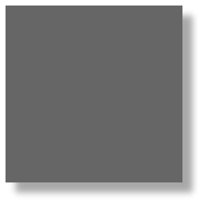
#### *1.7.6.2. User Interface*



### 1.7.7. quadrado

#### *1.7.7.1. State1*

#### *1.7.7.2. User Interface*



### 1.7.8. Telinha #1

#### *1.7.8.1. Limpo*

#### *1.7.8.2. User Interface*

#### 1.7.8.3. Quadradinho

#### 1.7.8.4. User Interface

#### *1.7.8.5. Circulinho*

#### *1.7.8.6. User Interface*

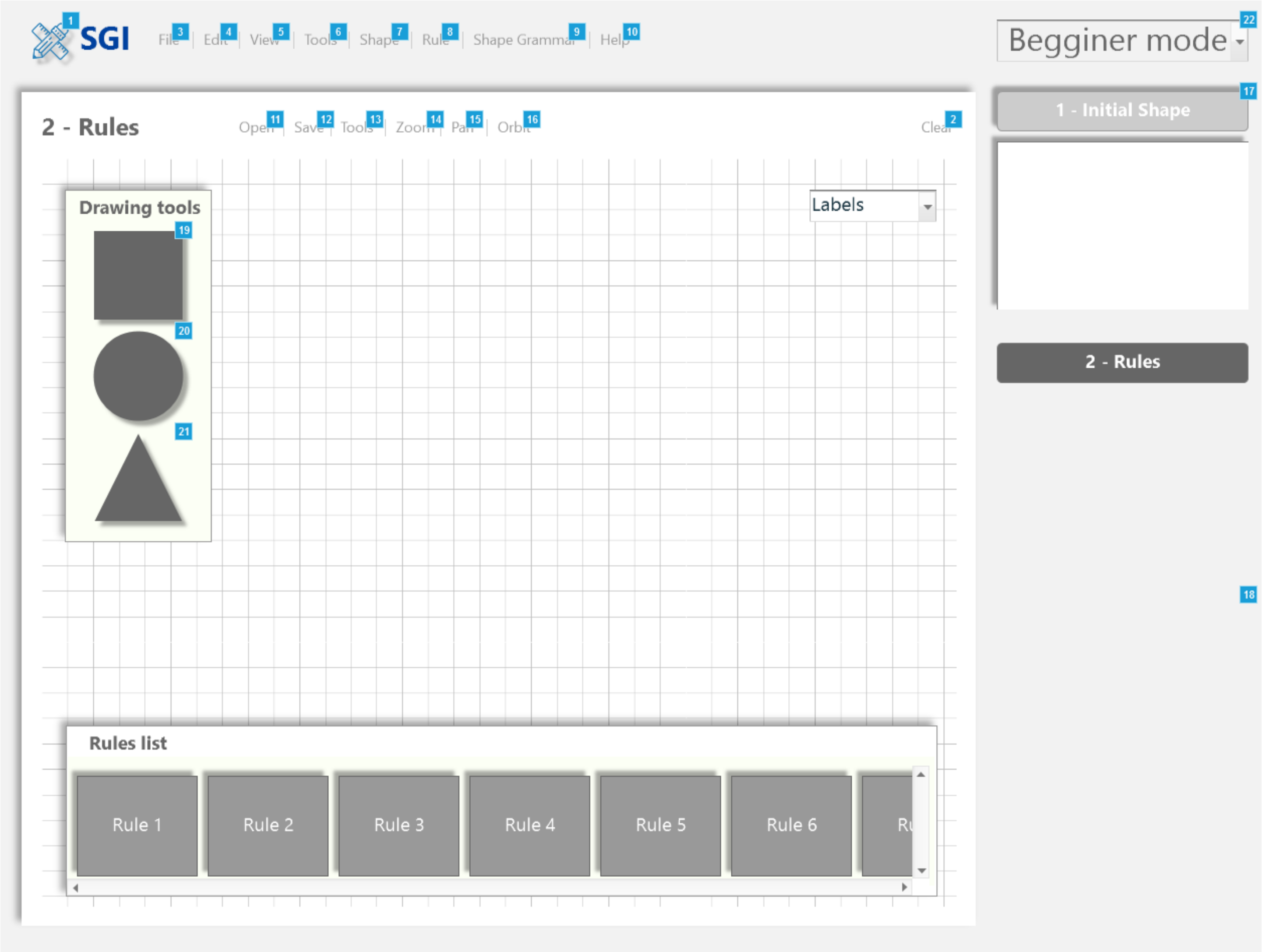
#### *1.7.8.7. Triangulinho*

#### *1.7.8.8. User Interface*

## 1.8. 2 - Rules

```
OnPageLoad:
  Case 1
 (If value of f equals "q"):
    Set Rules to Quadrado
    Set Telinha #1 to Quadradinho
  Case 2
 (Else If value of f equals "c"):
    Set Rules to Circulo,
Telinha #1 to Circulinho
  Case 3
 (Else If value of f equals "t"):
    Set Rules to Triangulo,
Telinha #1 to Triangulinho
```

### 1.8.1. User Interface



### 1.8.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Open Home in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 2 | | OnClick:<br>Case 1:<br>Set Panel to State |
| 3 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 4 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 5 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 6 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 7 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 8 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 9 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 10 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 11 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 12 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 13 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 14 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 15 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 16 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 17 | Next step | OnClick:<br>Case 1:<br>Open 1 - Initial Shape in Current Window |
| 18 | Next step #3 | OnClick:<br>Case 1:<br>Open 3 - Shape Grammar Result in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 19 | quadrado | OnDrag:<br>Case 1:<br>Move This with drag<br>OnDragDrop:<br>Case 1<br>(If state of Rules equals Vazio):<br>Set Rules to Quadrado<br>Move This to x,y before drag<br>Set Telinha #2 to Quadrado<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "q"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms<br>Case 2<br>(Else If state of Rules equals Quadrado):<br>Set Rules to Quadrado + Quadrado<br>Move This to x,y before drag<br>Set Telinha #2 to Quadrado + Quadrado<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "qq"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms<br>Case 3<br>(Else If state of Rules equals Circulo):<br>Set Rules to Quadrado + Circulo<br>Move This to x,y before drag<br>Set Telinha #2 to Quadrado + Circulo<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "qc"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms<br>Case 4<br>(Else If state of Rules equals Circulo + Triangulo):<br>Set Rules to Quadrado + Circulo + Triangulo<br>Move This to x,y before drag<br>Set Telinha #2 to Quadrado + Circulo + Triangulo<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "qtc"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms<br>Case 5<br>(Else If state of Rules equals Triangulo):<br>Set Rules to Quadrado + Triangulo<br>Move This to x,y before drag<br>Set Telinha #2 to Quadrado + Triangulo<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "qt"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms |

| Footnote | Name | Interactions |
|---|---|---|
| 20 | circulo | OnDrag:<br>Case 1:<br>Move This with drag<br>OnDragDrop:<br>Case 1<br>(If state of Rules equals Vazio):<br>Set Rules to Circulo<br>Move This to x,y before drag<br>Set Telinha #2 to Circulo<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "c"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms<br>Case 2<br>(Else If state of Rules equals Quadrado):<br>Set Rules to Quadrado + Circulo<br>Move This to x,y before drag<br>Set Telinha #2 to Quadrado + Circulo<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "qc"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms<br>Case 3<br>(Else If state of Rules equals Triangulo):<br>Set Rules to Circulo + Triangulo<br>Move This to x,y before drag<br>Set Telinha #2 to Circulo + Triangulo<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "ct"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms |

| Footnote | Name | Interactions |
|---|---|---|
| 21 | triangulo | OnDrag:<br>Case 1:<br>Move This with drag<br>OnDragDrop:<br>Case 1<br>(If state of Rules equals Vazio):<br>Set Rules to Triangulo<br>Move This to x,y before drag<br>Set Telinha #2 to Triangulo<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "t"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms<br>Case 2<br>(Else If state of Rules equals Quadrado):<br>Set Rules to Quadrado + Triangulo<br>Move This to x,y before drag<br>Set Telinha #2 to Quadrado + Triangulo<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "qt"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms<br>Case 3<br>(Else If state of Rules equals Circulo):<br>Set Rules to Circulo + Triangulo<br>Move This to x,y before drag<br>Set Telinha #2 to Circulo + Triangulo<br>Show Next step #3,<br>Telinha #2<br>Set value of f equal to "ct"<br>Show (Right Arrow) fade 500 ms<br>Wait 5000 ms<br>Hide (Right Arrow) fade 500 ms |
| 22 |  | OnSelectionChange:<br>Case 1<br>(If selected option of This equals Intermediate mode):<br>Open 1 - Initial Shape in Current Window<br>Case 2<br>(Else If selected option of This equals Expert mode):<br>Open 1 - Initial Shape in Current Window<br>Case 3<br>(Else If selected option of This equals Begginer mode):<br>Open 1 - Initial Shape in Current Window |

## 1.8.3. Unnamed

### *1.8.3.1. State1*

### *1.8.3.2. User Interface*

### 1.8.4. Rules

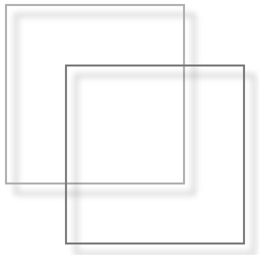
#### *1.8.4.1. Vazio*

#### *1.8.4.2. Quadrado*

#### *1.8.4.3. User Interface*

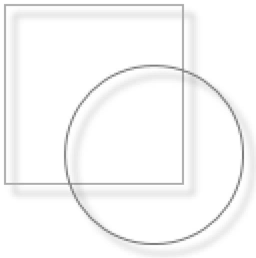
#### *1.8.4.4. Quadrado + Quadrado*

#### *1.8.4.5. User Interface*



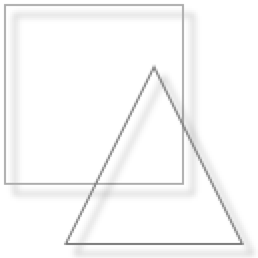
#### 1.8.4.6. Quadrado + Circulo

#### 1.8.4.7. User Interface



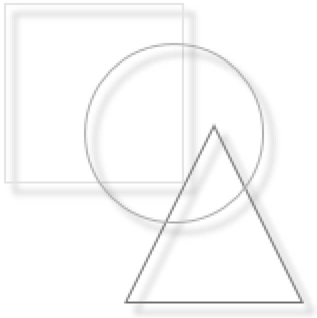
#### 1.8.4.8. Quadrado + Triangulo

#### 1.8.4.9. User Interface



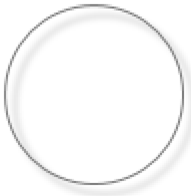
### *1.8.4.10. Quadrado + Circulo + Triangulo*

### *1.8.4.11. User Interface*



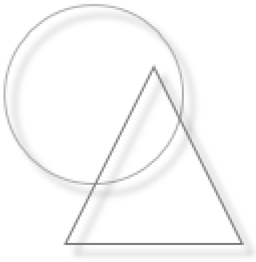
### *1.8.4.12. Circulo*

### *1.8.4.13. User Interface*



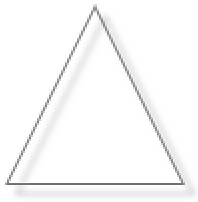
### *1.8.4.14. Circulo + Triangulo*

### *1.8.4.15. User Interface*



#### *1.8.4.16. Triangulo*

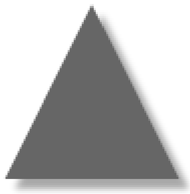
#### *1.8.4.17. User Interface*



### 1.8.5. triangulo

#### *1.8.5.1. State1*

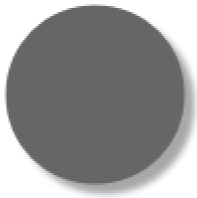
#### *1.8.5.2. User Interface*



### 1.8.6. circulo

#### *1.8.6.1. State1*

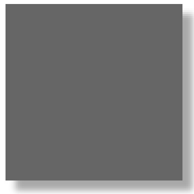
#### *1.8.6.2. User Interface*



### 1.8.7. quadrado

#### *1.8.7.1. State1*

#### *1.8.7.2. User Interface*



### 1.8.8. Telinha #1

#### *1.8.8.1. Limpo*

#### *1.8.8.2. User Interface*

#### 1.8.8.3. Quadradinho

#### 1.8.8.4. User Interface

#### 1.8.8.5. Circulinho

#### 1.8.8.6. User Interface

### 1.8.8.7. Triangulinho

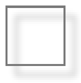
### 1.8.8.8. User Interface

### 1.8.9. Telinha #2

#### *1.8.9.1. Vazio*

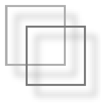
#### *1.8.9.2. Quadrado*

#### *1.8.9.3. User Interface*



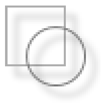
#### *1.8.9.4. Quadrado + Quadrado*

#### *1.8.9.5. User Interface*



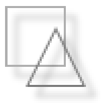
#### *1.8.9.6. Quadrado + Circulo*

#### *1.8.9.7. User Interface*



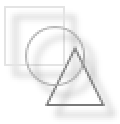
#### *1.8.9.8. Quadrado + Triangulo*

#### *1.8.9.9. User Interface*



#### *1.8.9.10. Quadrado + Circulo + Triangulo*

#### *1.8.9.11. User Interface*

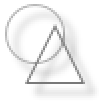


#### *1.8.9.12. Circulo*

#### *1.8.9.13. User Interface*

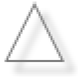
#### *1.8.9.14. Circulo + Triangulo*

#### *1.8.9.15. User Interface*



#### *1.8.9.16. Triangulo*

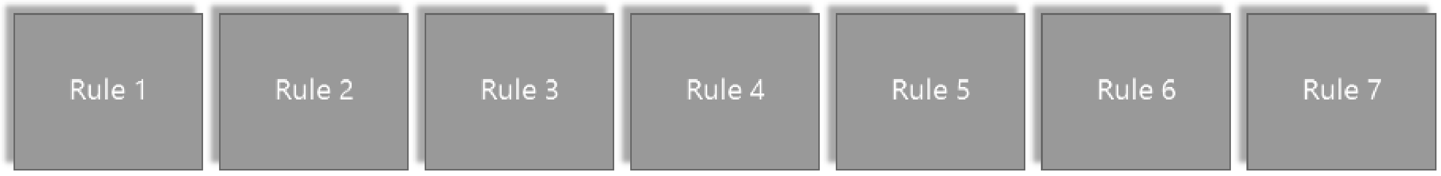
#### *1.8.9.17. User Interface*



### 1.8.10. Rules list

#### *1.8.10.1. State1*

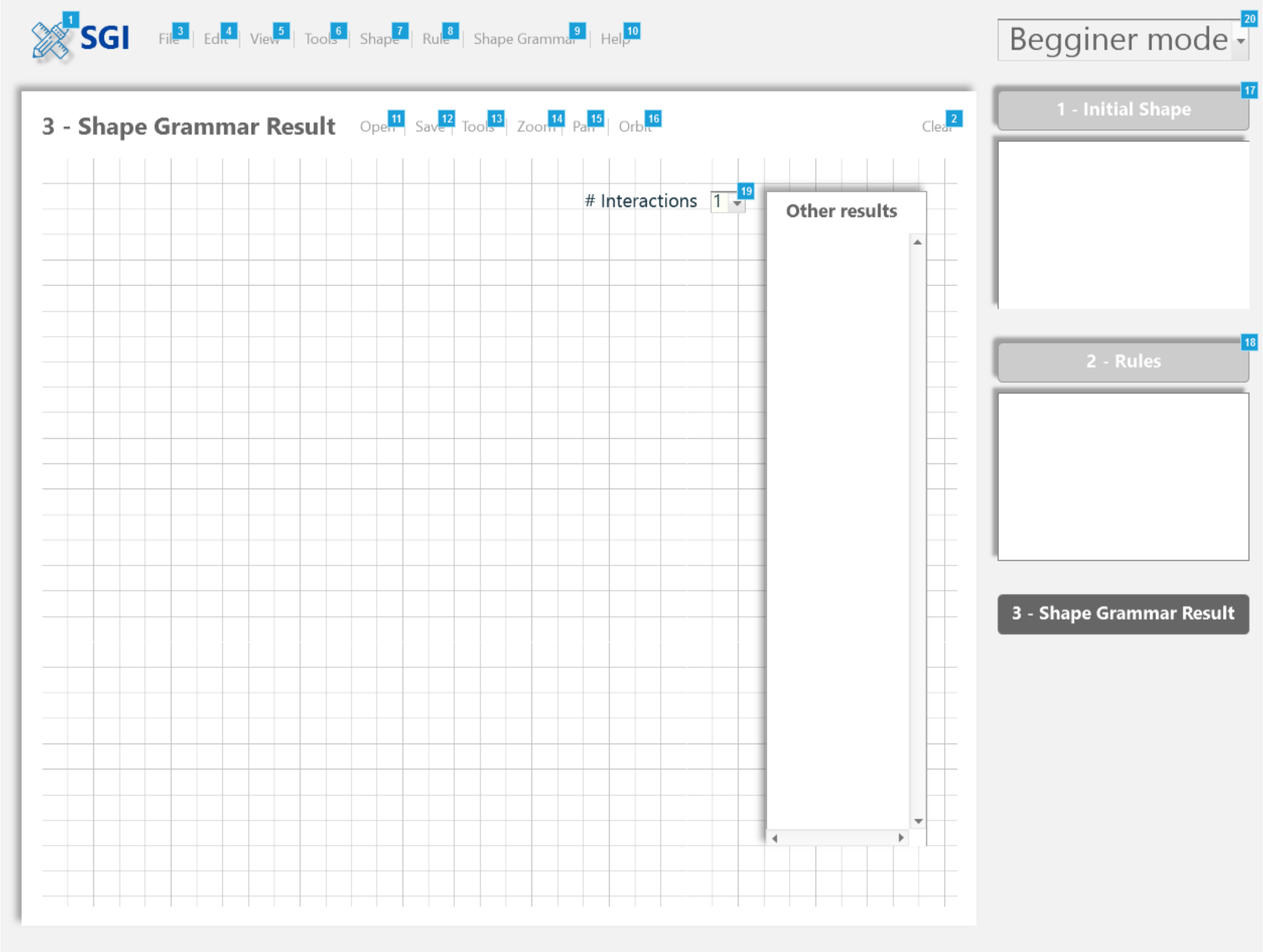
#### *1.8.10.2. User Interface*



## 1.9. 3 - Shape Grammar Result

```
OnPageLoad:
  Case 1
 (If value of f equals "q"):
    Set Telinha #2 to Quadrado,
Telinha #1 to Quadradinho,
Shape Grammar Result to Um quadrado,
other_results to Vazio
  Case 2
 (Else If value of f equals "qq"):
    Set Telinha #2 to Quadrado + Quadrado,
Telinha #1 to Quadradinho,
Shape Grammar Result to Dois quadrados,
other_results to Dois quadrados
  Case 3
 (Else If value of f equals "qc"):
    Set Telinha #2 to Quadrado + Circulo,
Telinha #1 to Circulinho,
Shape Grammar Result to Quadrado + um circulo,
other_results to Quadrado + Circulo
  Case 4
 (Else If value of f equals "qtc"):
    Set Telinha #2 to Quadrado + Circulo + Triangulo,
Telinha #1 to Quadradinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 5
 (Else If value of f equals "qt"):
    Set Telinha #2 to Quadrado + Triangulo,
Telinha #1 to Quadradinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 6
 (Else If value of f equals "c"):
    Set Telinha #2 to Circulo,
Telinha #1 to Circulinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 7
 (Else If value of f equals "ct"):
    Set Telinha #2 to Circulo + Triangulo,
Telinha #1 to Circulinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 7
 (Else If value of f equals "t"):
    Set Telinha #2 to Triangulo,
Telinha #1 to Triangulinho,
Shape Grammar Result to Vazio,
other_results to Vazio
```

### 1.9.1. User Interface



### 1.9.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 2 | | OnClick:<br>Case 1:<br>Set Panel to State |
| 3 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 4 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 5 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 6 | | OnClick:<br>Case 1:<br>Open Home in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 7 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 8 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 9 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 10 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 11 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 12 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 13 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 14 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 15 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 16 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 17 | Next step | OnClick:<br>Case 1:<br>Open 1 - Initial Shape in Current Window |
| 18 | Next step | OnClick:<br>Case 1:<br>Open 2 - Rules in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 19 | | OnSelectionChange:<br>Case 1<br>(If selected option of This equals 1 and value of f equals "qq"):<br>Set Shape Grammar Result to Dois quadrados,<br>other_results to Dois quadrados<br>Case 2<br>(Else If selected option of This equals 2 and value of f equals "qq"):<br>Set Shape Grammar Result to Três quadrados,<br>other_results to Três quadrados<br>Case 3<br>(Else If selected option of This equals 3 and value of f equals "qq"):<br>Set Shape Grammar Result to Quatro quadrados,<br>other_results to Quatro quadrados<br>Case 4<br>(Else If selected option of This equals 4 and value of f equals "qq"):<br>Set Shape Grammar Result to Cinco quadrados,<br>other_results to Cinco quadrados<br>Case 5<br>(Else If selected option of This equals 1 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + um circulo,<br>other_results to Quadrado + Circulo<br>Case 6<br>(Else If selected option of This equals 2 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + dois circulo,<br>other_results to Quadrado + dois circulos<br>Case 7<br>(Else If selected option of This equals 3 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + três circulo,<br>other_results to Quadrado + tres circulos<br>Case 8<br>(Else If selected option of This equals 4 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + quatro circulo,<br>other_results to Quadrado + quatro circulos |
| 20 | | OnSelectionChange:<br>Case 1<br>(If selected option of This equals  Intermediate mode):<br>Open 1 - Initial Shape in Current Window<br>Case 2<br>(Else If selected option of This equals  Expert mode):<br>Open 1 - Initial Shape in Current Window<br>Case 3<br>(Else If selected option of This equals  Begginer mode):<br>Open 1 - Initial Shape in Current Window |

### 1.9.3. Unnamed

#### *1.9.3.1. State1*

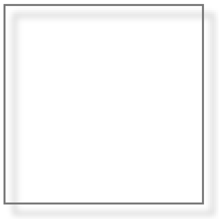
#### *1.9.3.2. User Interface*

### 1.9.4. Shape Grammar Result

#### *1.9.4.1. Vazio*

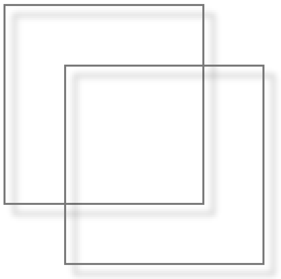
#### *1.9.4.2. Um quadrado*

#### *1.9.4.3. User Interface*



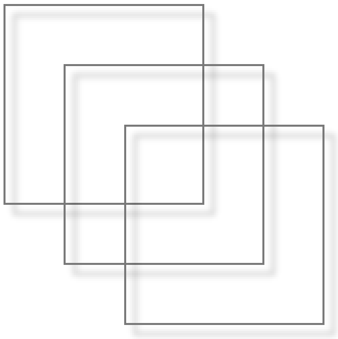
#### *1.9.4.4. Dois quadrados*

#### *1.9.4.5. User Interface*



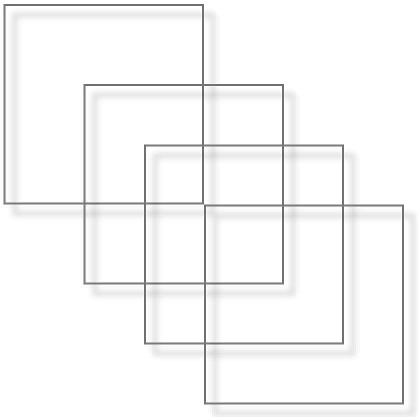
#### 1.9.4.6. Três quadrados

#### 1.9.4.7. User Interface



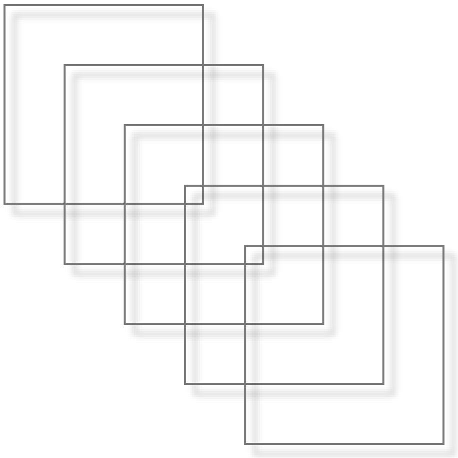
#### 1.9.4.8. Quatro quadrados

#### 1.9.4.9. User Interface



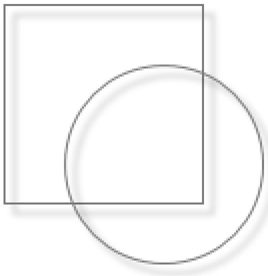
### 1.9.4.10. Cinco quadrados

### 1.9.4.11. User Interface



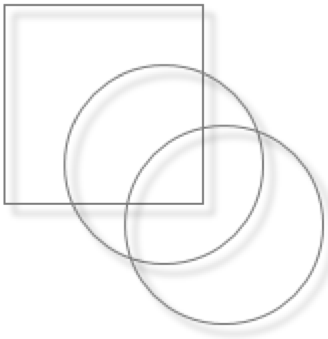
### 1.9.4.12. Quadrado + um circulo

### 1.9.4.13. User Interface



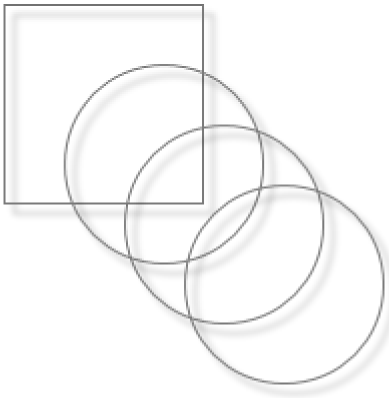
#### 1.9.4.14. Quadrado + dois circulo

#### 1.9.4.15. User Interface



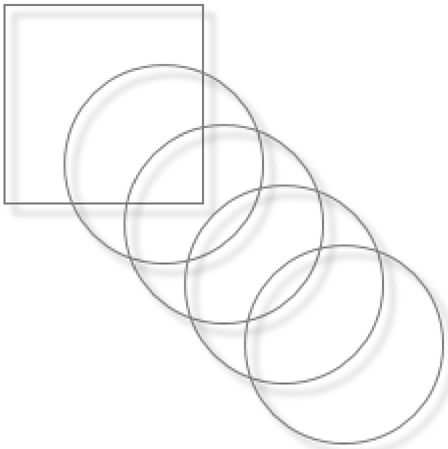
#### 1.9.4.16. Quadrado + três circulo

#### 1.9.4.17. User Interface



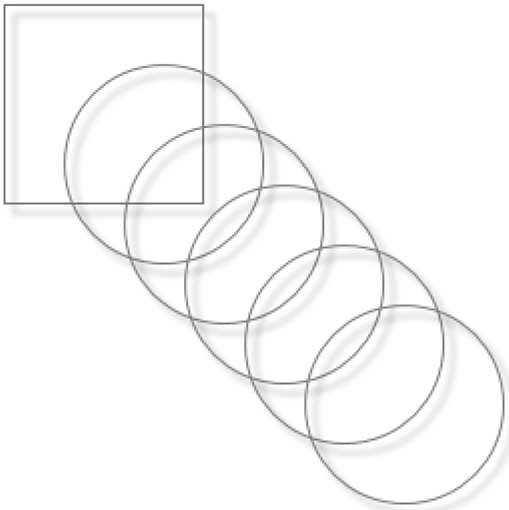
### *1.9.4.18. Quadrado + quatro circulo*

### *1.9.4.19. User Interface*



### *1.9.4.20. Quadrado + cinco circulos*

### *1.9.4.21. User Interface*



### 1.9.5. Telinha #1

#### *1.9.5.1. Limpo*

#### *1.9.5.2. User Interface*

### 1.9.5.3. Quadradinho

### 1.9.5.4. User Interface

### 1.9.5.5. Circulinho

### 1.9.5.6. User Interface

### *1.9.5.7. Triangulinho*

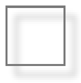
### *1.9.5.8. User Interface*

### 1.9.6. Telinha #2

#### *1.9.6.1. Vazio*

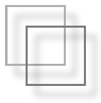
#### *1.9.6.2. Quadrado*

#### *1.9.6.3. User Interface*



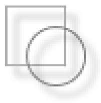
#### *1.9.6.4. Quadrado + Quadrado*

#### *1.9.6.5. User Interface*



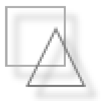
#### *1.9.6.6. Quadrado + Circulo*

#### *1.9.6.7. User Interface*



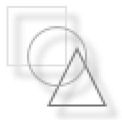
#### *1.9.6.8. Quadrado + Triangulo*

#### *1.9.6.9. User Interface*



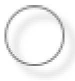
#### *1.9.6.10. Quadrado + Circulo + Triangulo*

#### *1.9.6.11. User Interface*



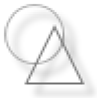
#### 1.9.6.12. Circulo

#### 1.9.6.13. User Interface



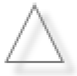
#### 1.9.6.14. Circulo + Triangulo

#### 1.9.6.15. User Interface



#### 1.9.6.16. Triangulo

#### 1.9.6.17. User Interface

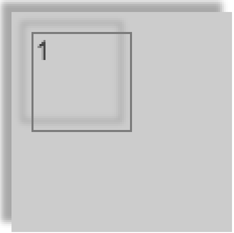


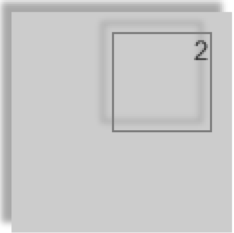
### 1.9.7. other_results

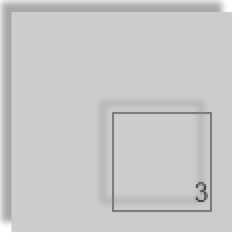
#### *1.9.7.1. Vazio*

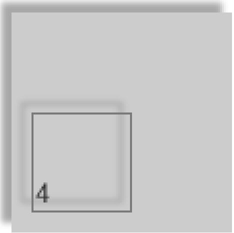
#### *1.9.7.2. Um quadrado*

#### *1.9.7.3. User Interface*



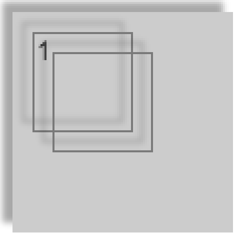


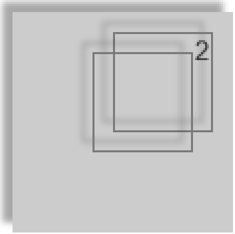


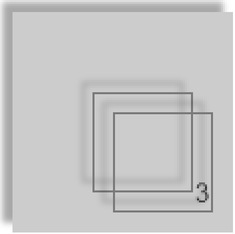


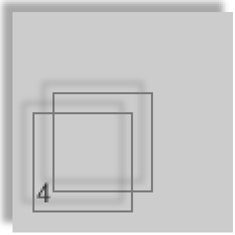
### 1.9.7.4. Dois quadrados

### 1.9.7.5. User Interface



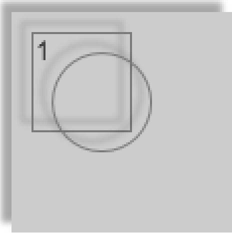


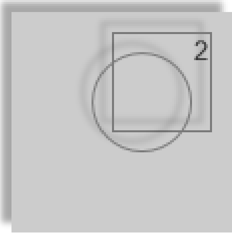


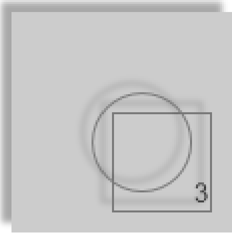


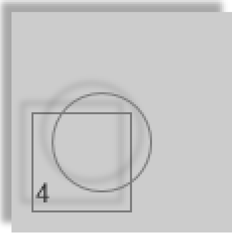
### *1.9.7.6. Quadrado + Circulo*

### *1.9.7.7. User Interface*



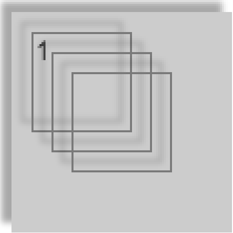


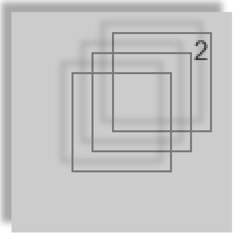


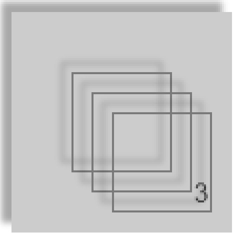


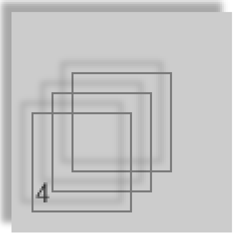
### *1.9.7.8. Três quadrados*

### *1.9.7.9. User Interface*



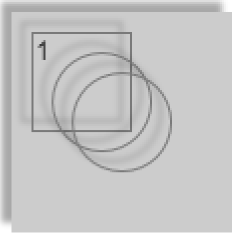


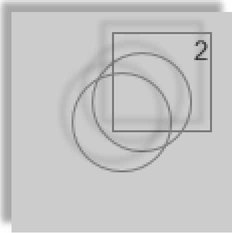


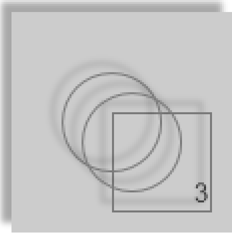


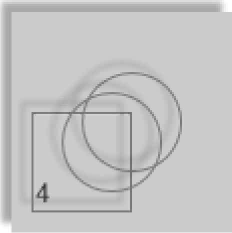
### 1.9.7.10. Quadrado + dois circulos

### 1.9.7.11. User Interface



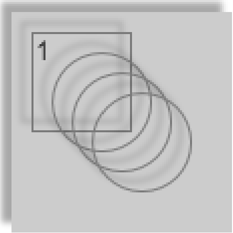


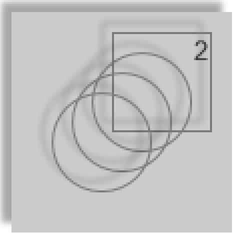


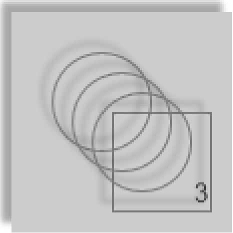


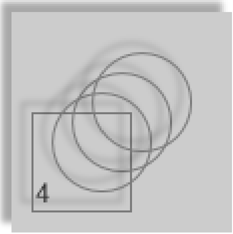
### *1.9.7.12. Quadrado + tres circulos*

### *1.9.7.13. User Interface*



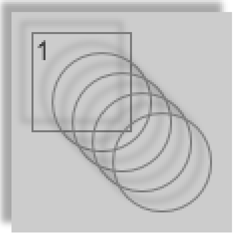


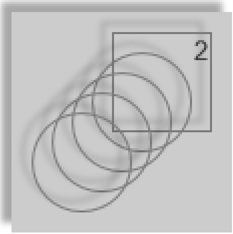


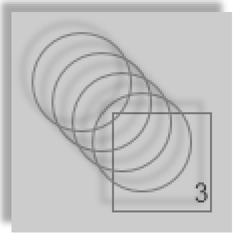


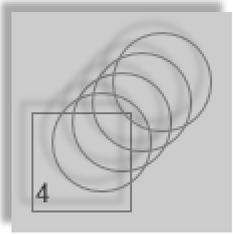
### 1.9.7.14. Quadrado + quatro circulos

### 1.9.7.15. User Interface



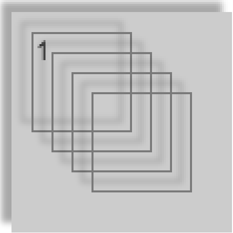


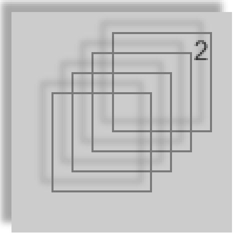


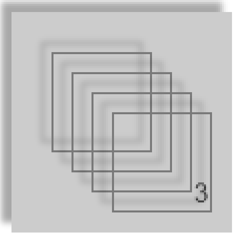


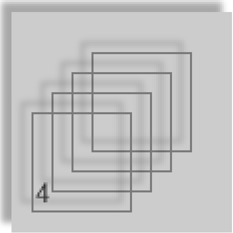
### 1.9.7.16. Quatro quadrados

### 1.9.7.17. User Interface



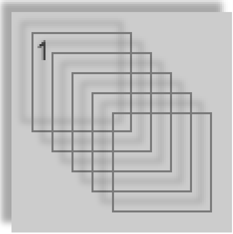


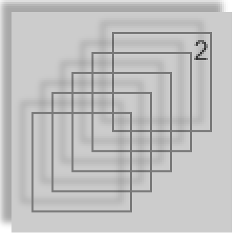


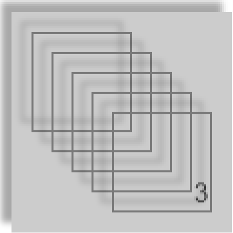


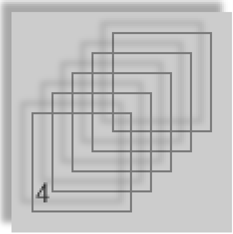
### 1.9.7.18. Cinco quadrados

### 1.9.7.19. User Interface







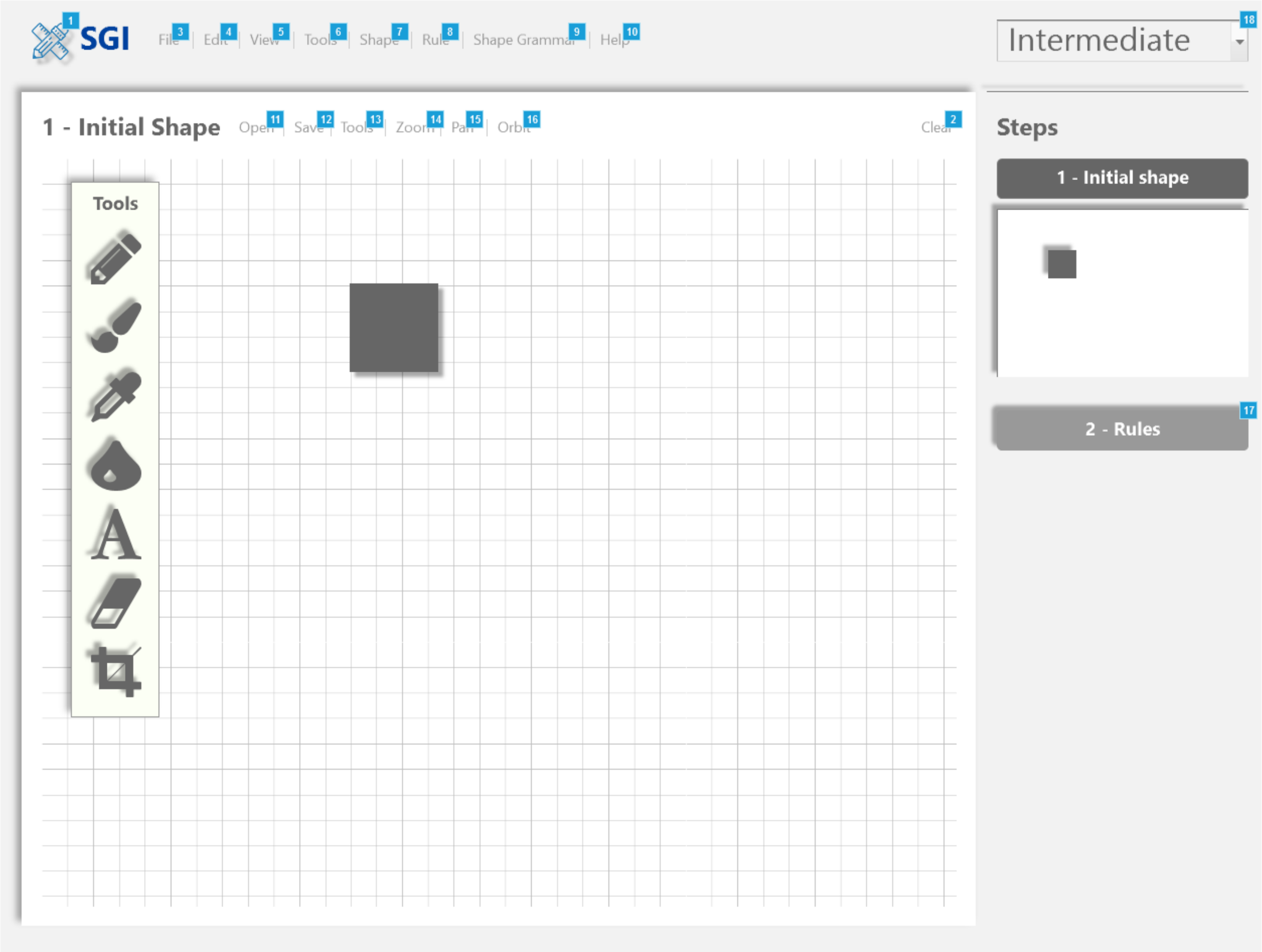


## 1.10. Intermediate

## 1.11. 1 - Initial Shape

```
OnPageLoad:
  Case 1:
    Wait 5000 ms
    Hide (Right Arrow) fade 500 ms
```

### 1.11.1. User Interface



### 1.11.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 2 | | OnClick:<br>Case 1:<br>Set Initial shape to Vazio<br>Hide Telinha 1 fade 500 ms,<br>Next step #2 fade 500 ms |
| 3 | | OnClick:<br>Case 1:<br>Open Home in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 4 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 5 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 6 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 7 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 8 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 9 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 10 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 11 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 12 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 13 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 14 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 15 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 16 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 17 | Next step #2 | OnClick:<br>Case 1:<br>Open 2 - Rules in Current Window |
| 18 | | OnSelectionChange:<br>Case 1<br>(If selected option of This equals Intermediate mode):<br>Open 1 - Initial Shape in Current Window<br>Case 2<br>(Else If selected option of This equals Expert mode):<br>Open 1 - Initial Shape in Current Window<br>Case 3<br>(Else If selected option of This equals Begginer mode):<br>Open 1 - Initial Shape in Current Window |

### 1.11.3. Unnamed

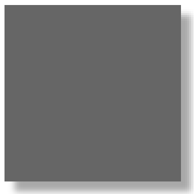
#### *1.11.3.1. State1*

#### *1.11.3.2. User Interface*

### 1.11.4. Initial shape

#### *1.11.4.1. Quadrado*

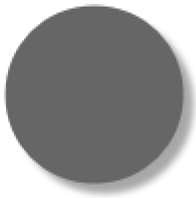
#### *1.11.4.2. User Interface*



#### *1.11.4.3. Vazio*

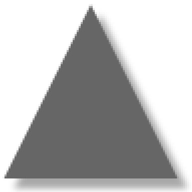
#### *1.11.4.4. Circulo*

#### *1.11.4.5. User Interface*



#### *1.11.4.6. Triangulo*

#### *1.11.4.7. User Interface*



## 1.11.5. Telinha #1

### *1.11.5.1. Limpo*

### *1.11.5.2. User Interface*

#### 1.11.5.3. Quadradinho

#### 1.11.5.4. User Interface

### 1.11.5.5. Circulinho

### 1.11.5.6. User Interface

### 1.11.5.7. Triangulinho

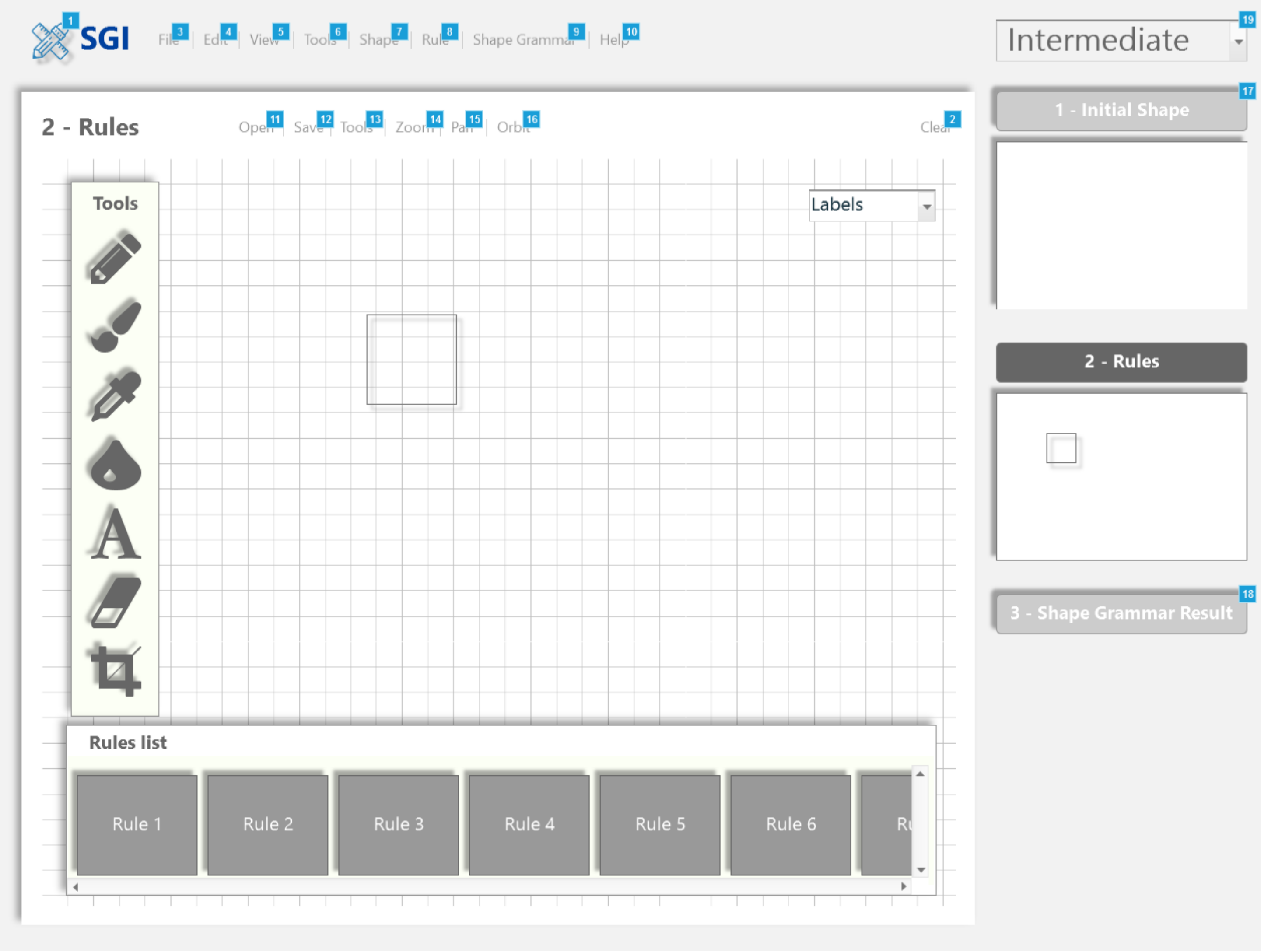
### 1.11.5.8. User Interface

▲

## 1.12. 2 - Rules

```
OnPageLoad:
  Case 1
 (If value of f equals "q"):
    Set Rules to Quadrado
    Set Telinha #1 to Quadradinho
  Case 2
 (Else If value of f equals "c"):
    Set Rules to Circulo,
Telinha #1 to Circulinho
  Case 3
 (Else If value of f equals "t"):
    Set Rules to Triangulo,
Telinha #1 to Triangulinho
  Case 4
 (Else If True):
    Wait 5000 ms
    Hide (Right Arrow) fade 500 ms
```

### 1.12.1. User Interface



### 1.12.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 2 |  | OnClick:<br>Case 1:<br>Set Panel to State |
| 3 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 4 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 5 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 6 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 7 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 8 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 9 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 10 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 11 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 12 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 13 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 14 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 15 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 16 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 17 | Next step | OnClick:<br>Case 1:<br>Open 1 - Initial Shape in Current Window |
| 18 | Next step #3 | OnClick:<br>Case 1:<br>Open 3 - Shape Grammar Result in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 19 | | OnSelectionChange:<br>Case 1<br>(If selected option of This equals  Intermediate mode):<br>Open 1 - Initial Shape in Current Window<br>Case 2<br>(Else If selected option of This equals  Expert mode):<br>Open 1 - Initial Shape in Current Window<br>Case 3<br>(Else If selected option of This equals  Begginer mode):<br>Open 1 - Initial Shape in Current Window |

## 1.12.3. Unnamed

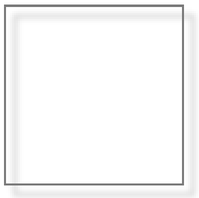
### *1.12.3.1. State1*

### *1.12.3.2. User Interface*

### 1.12.4. Rules

#### *1.12.4.1. Quadrado*

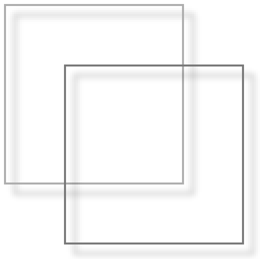
#### *1.12.4.2. User Interface*



#### *1.12.4.3. Vazio*

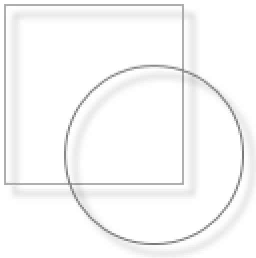
#### *1.12.4.4. Quadrado + Quadrado*

#### *1.12.4.5. User Interface*



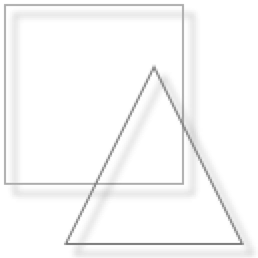
#### 1.12.4.6. Quadrado + Circulo

#### 1.12.4.7. User Interface



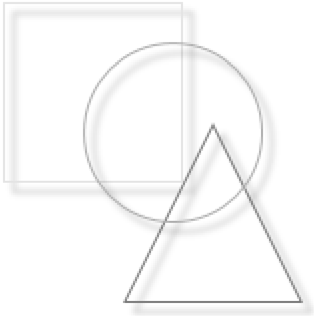
#### 1.12.4.8. Quadrado + Triangulo

#### 1.12.4.9. User Interface



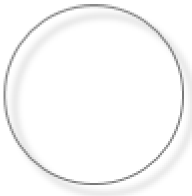
### 1.12.4.10. Quadrado + Circulo + Triangulo

### 1.12.4.11. User Interface



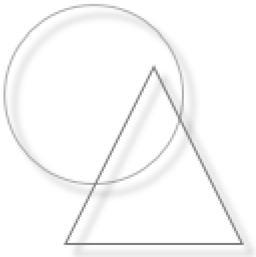
### 1.12.4.12. Circulo

### 1.12.4.13. User Interface



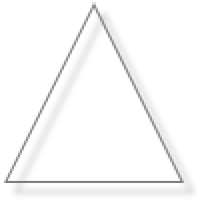
### 1.12.4.14. Circulo + Triangulo

### 1.12.4.15. User Interface



#### 1.12.4.16. Triangulo

#### 1.12.4.17. User Interface



### 1.12.5. Telinha #1

#### *1.12.5.1. Limpo*

#### *1.12.5.2. User Interface*

### 1.12.5.3. Quadradinho

### 1.12.5.4. User Interface

### 1.12.5.5. Circulinho

### 1.12.5.6. User Interface

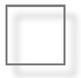
#### 1.12.5.7. Triangulinho

#### 1.12.5.8. User Interface

### 1.12.6. Telinha #2

#### *1.12.6.1. Quadrado*

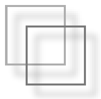
#### *1.12.6.2. User Interface*



#### *1.12.6.3. Vazio*

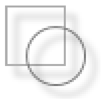
#### *1.12.6.4. Quadrado + Quadrado*

#### *1.12.6.5. User Interface*



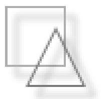
#### *1.12.6.6. Quadrado + Circulo*

#### *1.12.6.7. User Interface*



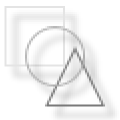
#### *1.12.6.8. Quadrado + Triangulo*

#### *1.12.6.9. User Interface*



#### *1.12.6.10. Quadrado + Circulo + Triangulo*

#### *1.12.6.11. User Interface*

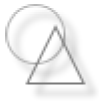


### 1.12.6.12. Circulo

### 1.12.6.13. User Interface

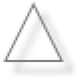
### 1.12.6.14. Circulo + Triangulo

### 1.12.6.15. User Interface



### 1.12.6.16. Triangulo

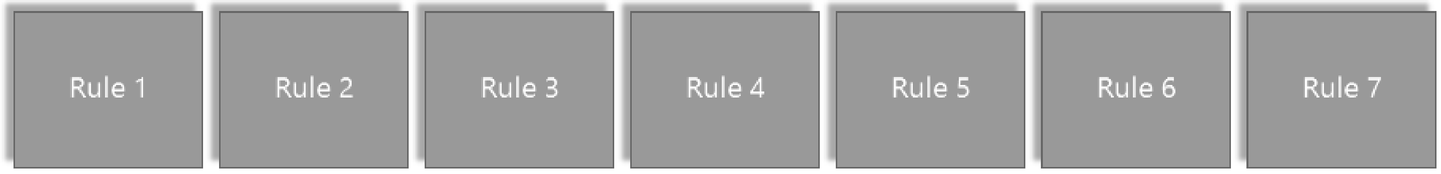
### 1.12.6.17. User Interface



## 1.12.7. Rules list

### 1.12.7.1. State1

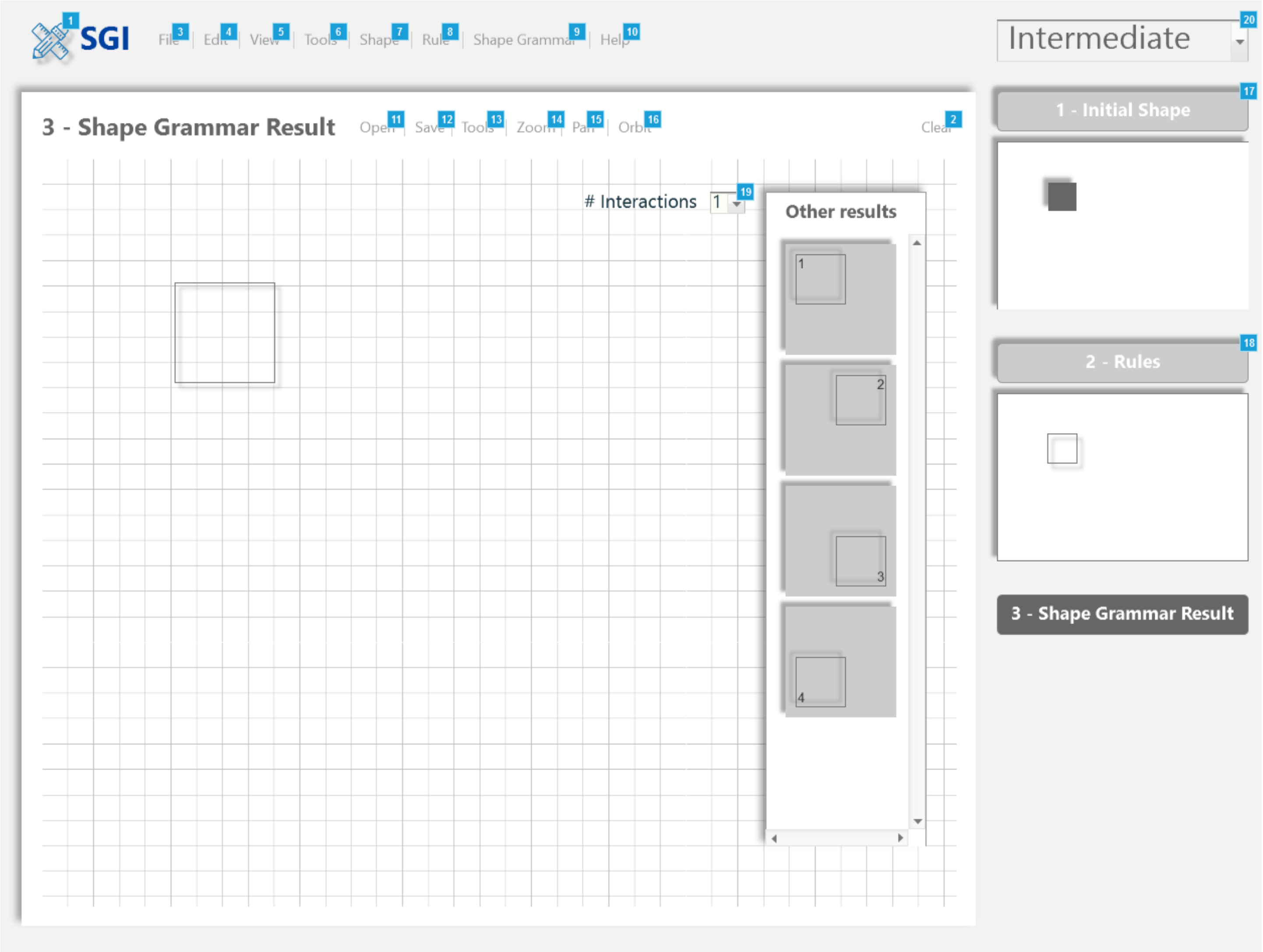
### 1.12.7.2. User Interface



## 1.13. 3 - Shape Grammar Result

```
OnPageLoad:
  Case 1
 (If value of f equals "q"):
    Set Telinha #2 to Quadrado,
Telinha #1 to Quadradinho,
Shape Grammar Result to Um quadrado,
other_results to Vazio
  Case 2
 (Else If value of f equals "qq"):
    Set Telinha #2 to Quadrado + Quadrado,
Telinha #1 to Quadradinho,
Shape Grammar Result to Dois quadrados,
other_results to Dois quadrados
  Case 3
 (Else If value of f equals "qc"):
    Set Telinha #2 to Quadrado + Circulo,
Telinha #1 to Circulinho,
Shape Grammar Result to Quadrado + um circulo,
other_results to Quadrado + Circulo
  Case 4
 (Else If value of f equals "qtc"):
    Set Telinha #2 to Quadrado + Circulo + Triangulo,
Telinha #1 to Quadradinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 5
 (Else If value of f equals "qt"):
    Set Telinha #2 to Quadrado + Triangulo,
Telinha #1 to Quadradinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 6
 (Else If value of f equals "c"):
    Set Telinha #2 to Circulo,
Telinha #1 to Circulinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 7
 (Else If value of f equals "ct"):
    Set Telinha #2 to Circulo + Triangulo,
Telinha #1 to Circulinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 7
 (Else If value of f equals "t"):
    Set Telinha #2 to Triangulo,
Telinha #1 to Triangulinho,
Shape Grammar Result to Vazio,
other_results to Vazio
```

### 1.13.1. User Interface



### 1.13.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 2 | | OnClick:<br>Case 1:<br>Set Panel to State |
| 3 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 4 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 5 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 6 | | OnClick:<br>Case 1:<br>Open Home in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 7 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 8 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 9 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 10 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 11 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 12 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 13 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 14 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 15 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 16 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 17 | Next step | OnClick:<br>Case 1:<br>Open 1 - Initial Shape in Current Window |
| 18 | Next step | OnClick:<br>Case 1:<br>Open 2 - Rules in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 19 | | OnSelectionChange:<br>Case 1<br>(If selected option of This equals 1 and value of f equals "qq"):<br>Set Shape Grammar Result to Dois quadrados,<br>other_results to Dois quadrados<br>Case 2<br>(Else If selected option of This equals 2 and value of f equals "qq"):<br>Set Shape Grammar Result to Três quadrados,<br>other_results to Três quadrados<br>Case 3<br>(Else If selected option of This equals 3 and value of f equals "qq"):<br>Set Shape Grammar Result to Quatro quadrados,<br>other_results to Quatro quadrados<br>Case 4<br>(Else If selected option of This equals 4 and value of f equals "qq"):<br>Set Shape Grammar Result to Cinco quadrados,<br>other_results to Cinco quadrados<br>Case 5<br>(Else If selected option of This equals 1 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + um circulo,<br>other_results to Quadrado + Circulo<br>Case 6<br>(Else If selected option of This equals 2 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + dois circulo,<br>other_results to Quadrado + dois circulos<br>Case 7<br>(Else If selected option of This equals 3 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + três circulo,<br>other_results to Quadrado + tres circulos<br>Case 8<br>(Else If selected option of This equals 4 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + quatro circulo,<br>other_results to Quadrado + quatro circulos |
| 20 | | OnSelectionChange:<br>Case 1<br>(If selected option of This equals Intermediate mode):<br>Open 1 - Initial Shape in Current Window<br>Case 2<br>(Else If selected option of This equals Expert mode):<br>Open 1 - Initial Shape in Current Window<br>Case 3<br>(Else If selected option of This equals Begginer mode):<br>Open 1 - Initial Shape in Current Window |

### 1.13.3. Unnamed

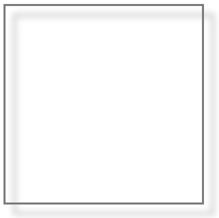
#### *1.13.3.1. State1*

#### *1.13.3.2. User Interface*

#### 1.13.4. Shape Grammar Result

##### *1.13.4.1. Um quadrado*

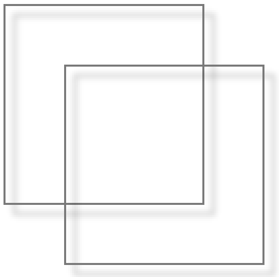
##### *1.13.4.2. User Interface*



##### *1.13.4.3. Vazio*

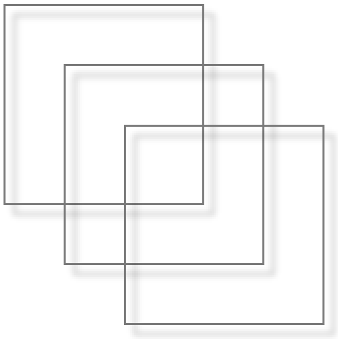
##### *1.13.4.4. Dois quadrados*

##### *1.13.4.5. User Interface*



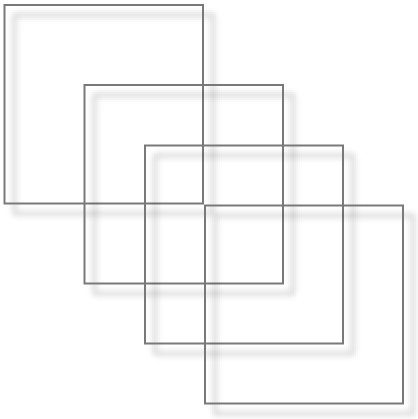
#### 1.13.4.6. Três quadrados

#### 1.13.4.7. User Interface



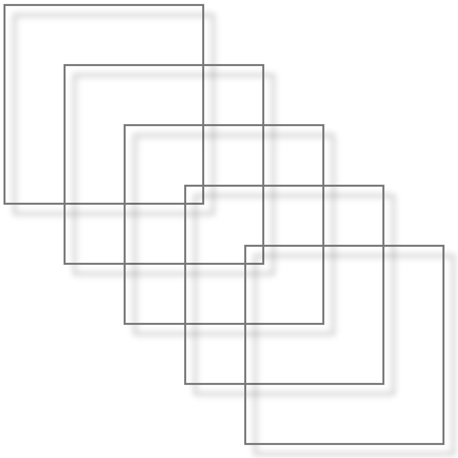
#### 1.13.4.8. Quatro quadrados

#### 1.13.4.9. User Interface



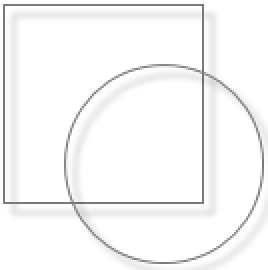
### *1.13.4.10. Cinco quadrados*

### *1.13.4.11. User Interface*



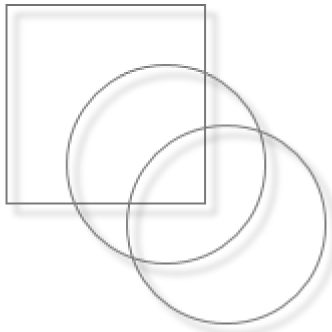
### *1.13.4.12. Quadrado + um circulo*

### *1.13.4.13. User Interface*



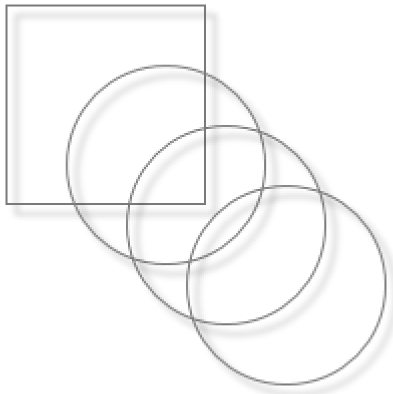
### *1.13.4.14. Quadrado + dois circulo*

### *1.13.4.15. User Interface*



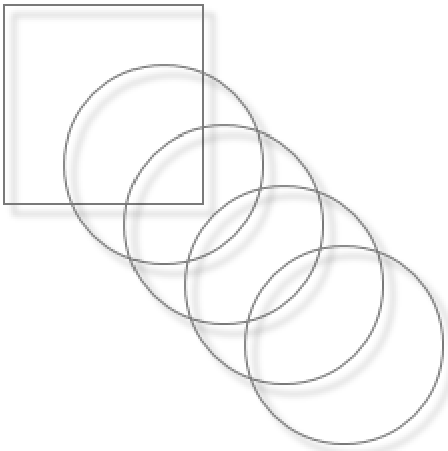
### *1.13.4.16. Quadrado + três circulo*

### *1.13.4.17. User Interface*



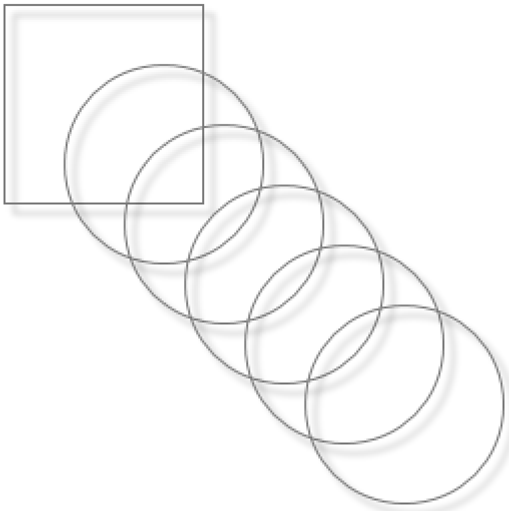
#### *1.13.4.18. Quadrado + quatro circulo*

#### *1.13.4.19. User Interface*



#### *1.13.4.20. Quadrado + cinco circulos*

#### *1.13.4.21. User Interface*



### 1.13.5. Telinha #1

#### *1.13.5.1. Quadradinho*

#### *1.13.5.2. User Interface*

#### 1.13.5.3. Limpo

#### 1.13.5.4. User Interface

### 1.13.5.5. Circulinho

### 1.13.5.6. User Interface

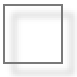
#### 1.13.5.7. Triangulinho

#### 1.13.5.8. User Interface

### 1.13.6. Telinha #2

#### *1.13.6.1. Quadrado*

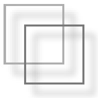
#### *1.13.6.2. User Interface*



#### *1.13.6.3. Vazio*

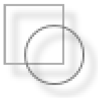
#### *1.13.6.4. Quadrado + Quadrado*

#### *1.13.6.5. User Interface*



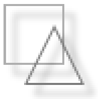
#### *1.13.6.6. Quadrado + Circulo*

#### *1.13.6.7. User Interface*



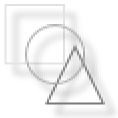
#### *1.13.6.8. Quadrado + Triangulo*

#### *1.13.6.9. User Interface*



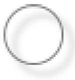
#### *1.13.6.10. Quadrado + Circulo + Triangulo*

#### *1.13.6.11. User Interface*



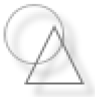
#### *1.13.6.12. Circulo*

#### *1.13.6.13. User Interface*



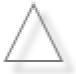
#### *1.13.6.14. Circulo + Triangulo*

#### *1.13.6.15. User Interface*



#### *1.13.6.16. Triangulo*

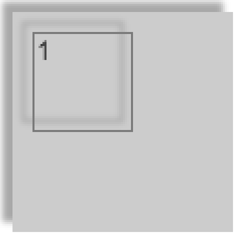
#### *1.13.6.17. User Interface*

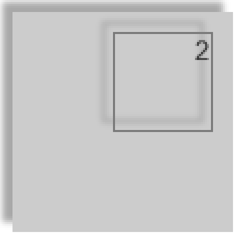


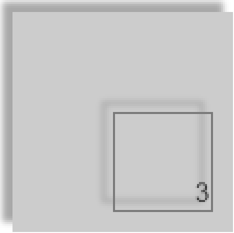
### 1.13.7. other_results

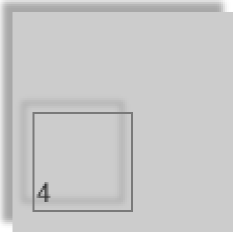
#### *1.13.7.1. Um quadrado*

#### *1.13.7.2. User Interface*





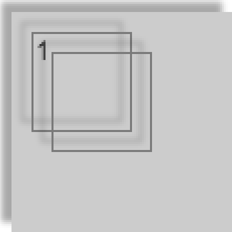


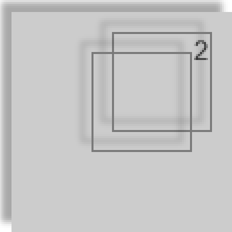


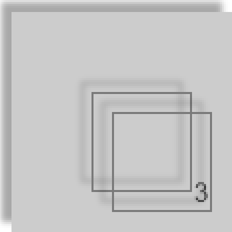
#### *1.13.7.3. Vazio*

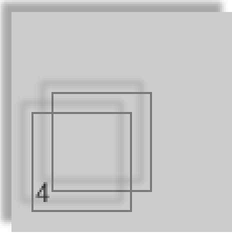
#### *1.13.7.4. Dois quadrados*

#### *1.13.7.5. User Interface*



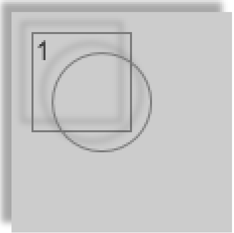


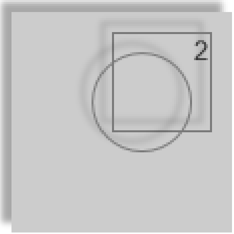


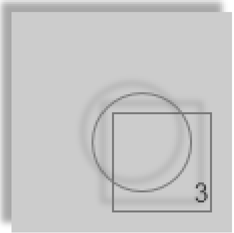


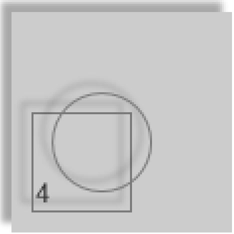
### 1.13.7.6. Quadrado + Circulo

### 1.13.7.7. User Interface



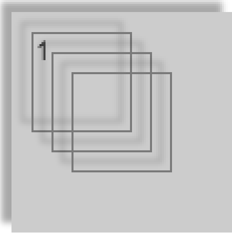


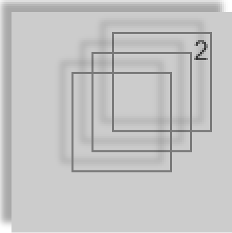


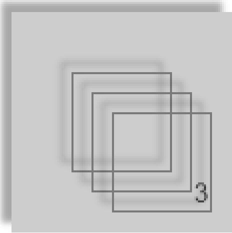


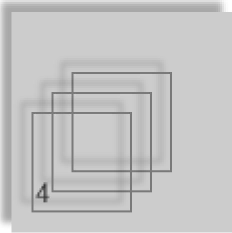
### 1.13.7.8. Três quadrados

### 1.13.7.9. User Interface



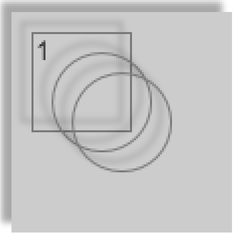


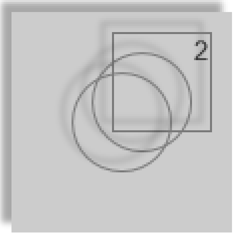


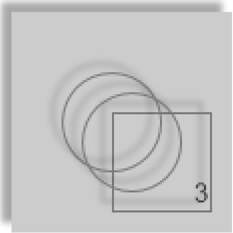


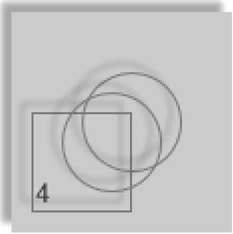
### 1.13.7.10. Quadrado + dois circulos

### 1.13.7.11. User Interface



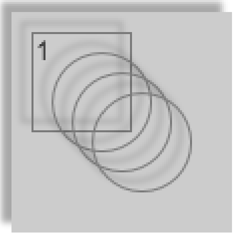


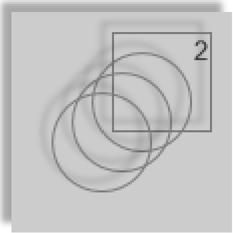


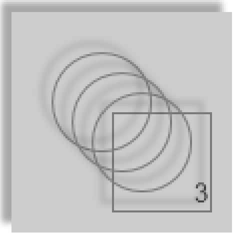


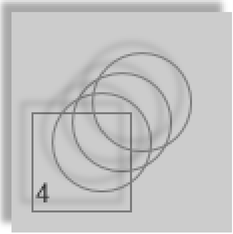
### *1.13.7.12. Quadrado + tres circulos*

### *1.13.7.13. User Interface*



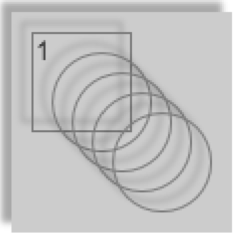


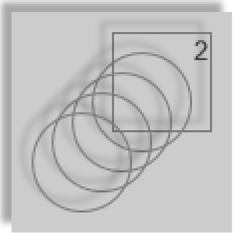


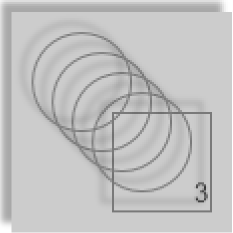


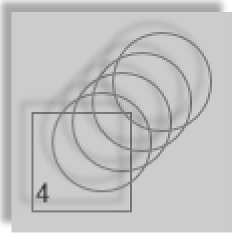
### *1.13.7.14. Quadrado + quatro circulos*

### *1.13.7.15. User Interface*



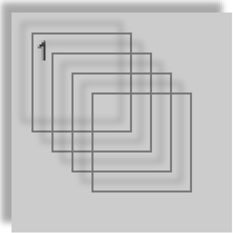


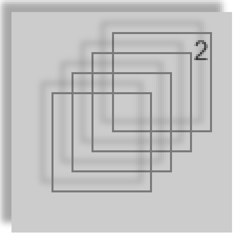


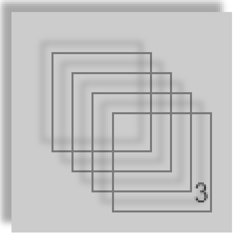


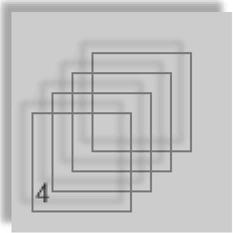
### 1.13.7.16. Quatro quadrados

### 1.13.7.17. User Interface



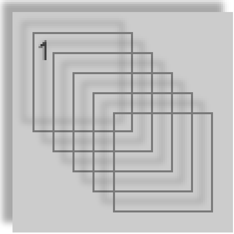


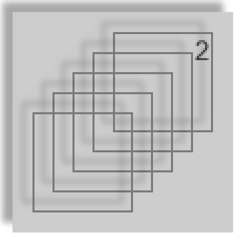


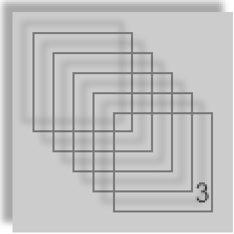


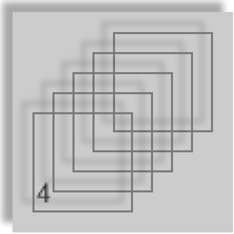
### 1.13.7.18. Cinco quadrados

### 1.13.7.19. User Interface







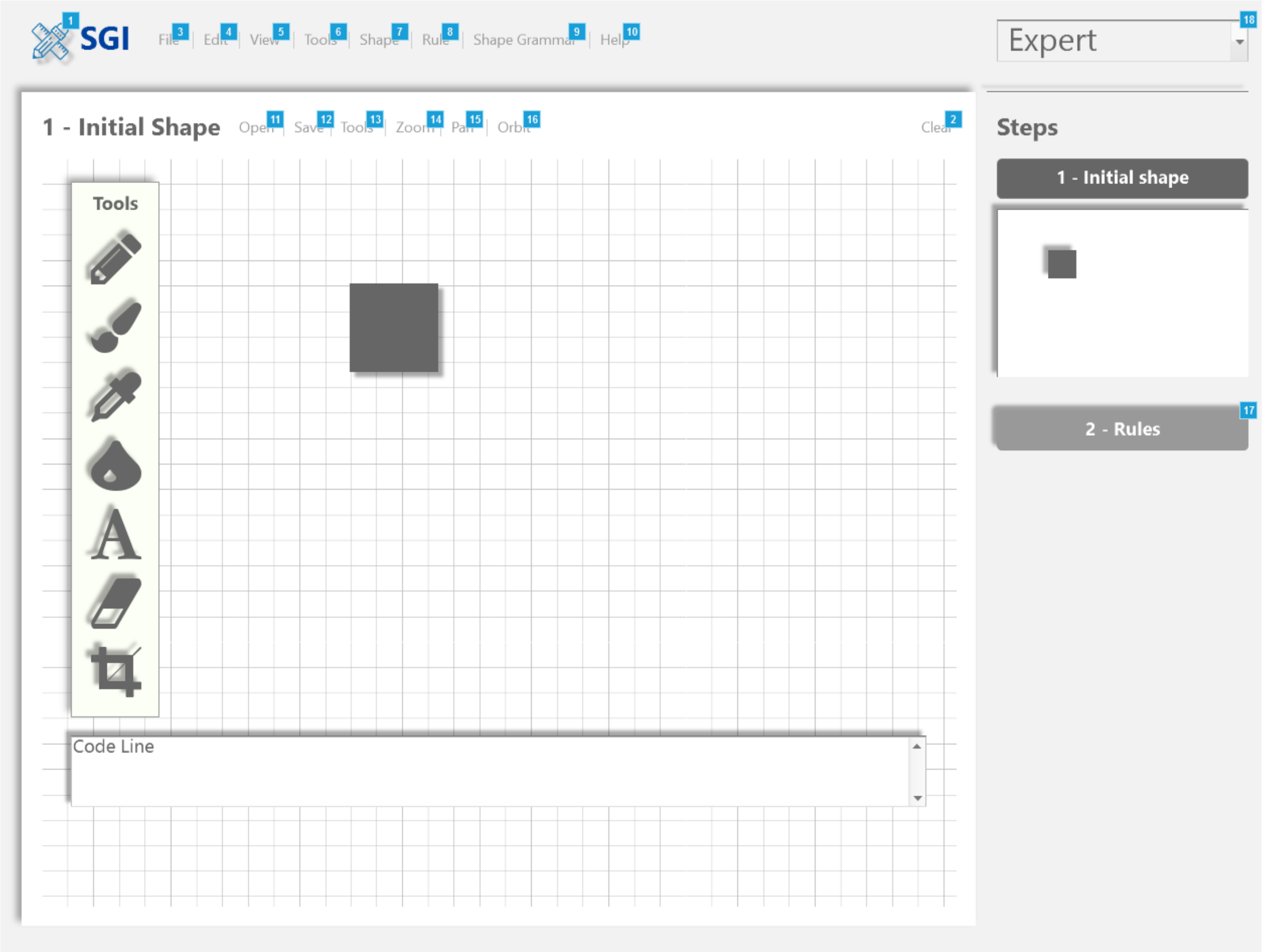


## 1.14. Expert

## 1.15. 1 - Initial Shape

OnPageLoad:
  Case 1:
    Wait 5000 ms
    Hide (Right Arrow) fade 500 ms

### 1.15.1. User Interface



### 1.15.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 2 | | OnClick:<br>Case 1:<br>Set Initial shape to Vazio<br>Hide Telinha 1 fade 500 ms,<br>Next step #2 fade 500 ms |
| 3 | | OnClick:<br>Case 1:<br>Open Home in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 4 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 5 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 6 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 7 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 8 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 9 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 10 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 11 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 12 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 13 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 14 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 15 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 16 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 17 | Next step #2 | OnClick:<br>Case 1:<br>Open 2 - Rules in Current Window |
| 18 | | OnSelectionChange:<br>Case 1<br>(If selected option of This equals  Intermediate mode):<br>Open 1 - Initial Shape in Current Window<br>Case 2<br>(Else If selected option of This equals  Expert mode):<br>Open 1 - Initial Shape in Current Window<br>Case 3<br>(Else If selected option of This equals  Begginer mode):<br>Open 1 - Initial Shape in Current Window |

## 1.15.3. Unnamed

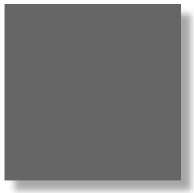
### *1.15.3.1. State1*

### *1.15.3.2. User Interface*

### 1.15.4. Initial shape

#### *1.15.4.1. Quadrado*

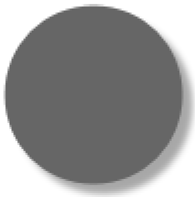
#### *1.15.4.2. User Interface*



#### *1.15.4.3. Vazio*

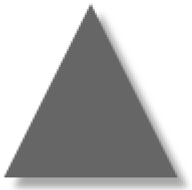
#### *1.15.4.4. Circulo*

#### *1.15.4.5. User Interface*



#### *1.15.4.6. Triangulo*

#### *1.15.4.7. User Interface*



### 1.15.5. Telinha #1

#### *1.15.5.1. Limpo*

#### *1.15.5.2. User Interface*

#### 1.15.5.3. Quadradinho

#### 1.15.5.4. User Interface

### *1.15.5.5. Circulinho*

### *1.15.5.6. User Interface*

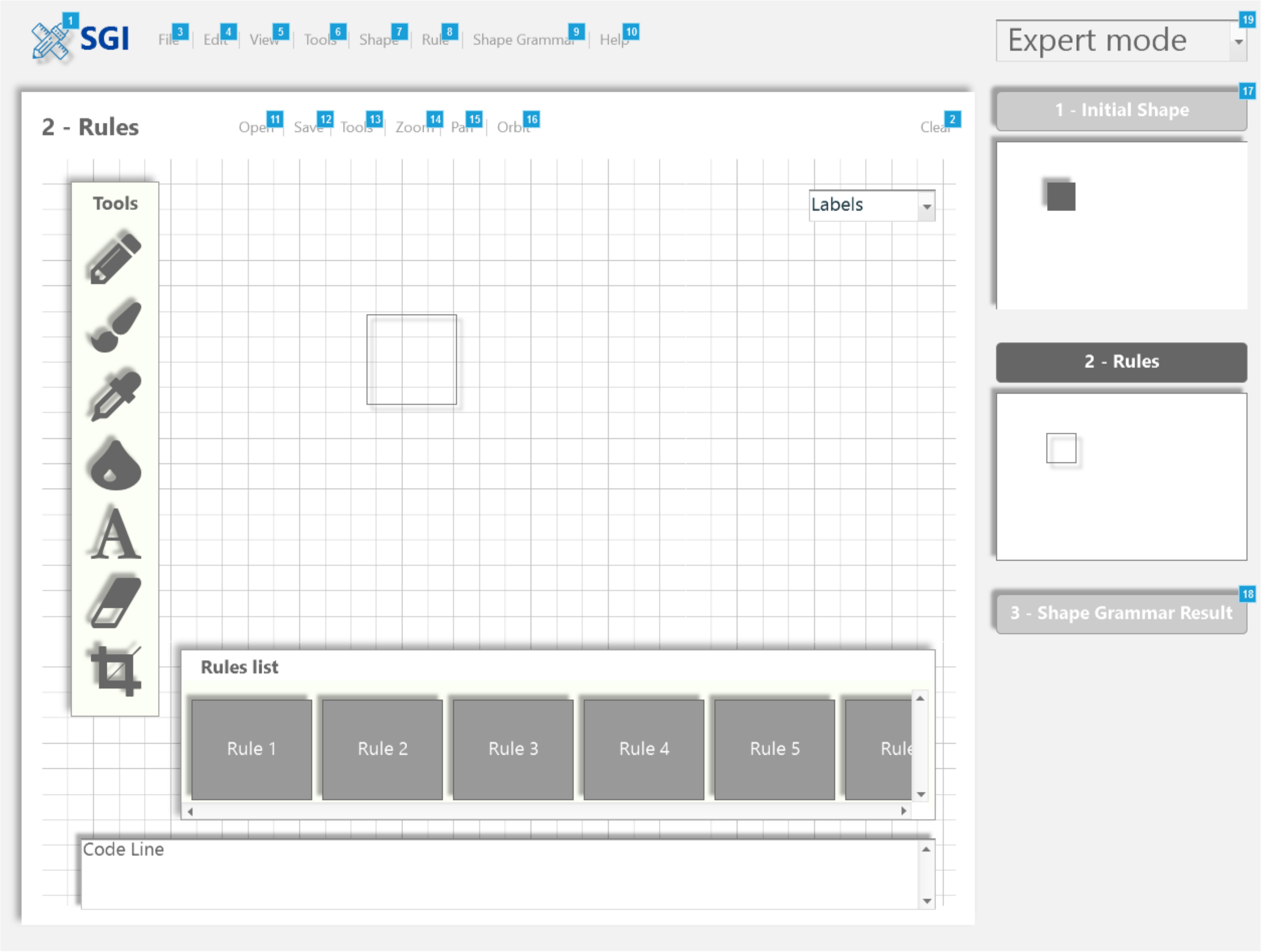
#### 1.15.5.7. Triangulinho

#### 1.15.5.8. User Interface

## 1.16. 2 - Rules

```
OnPageLoad:
  Case 1
 (If value of f equals "q"):
    Set Rules to Quadrado
    Set Telinha #1 to Quadradinho
  Case 2
 (Else If value of f equals "c"):
    Set Rules to Circulo,
Telinha #1 to Circulinho
  Case 3
 (Else If value of f equals "t"):
    Set Rules to Triangulo,
Telinha #1 to Triangulinho
  Case 4
 (Else If True):
    Wait 5000 ms
    Hide (Right Arrow) fade 500 ms
```

### 1.16.1. User Interface



### 1.16.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 2 | | OnClick:<br>Case 1:<br>Set Panel to State |
| 3 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 4 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 5 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 6 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 7 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 8 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 9 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 10 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 11 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 12 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 13 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 14 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 15 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 16 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 17 | Next step | OnClick:<br>Case 1:<br>Open 1 - Initial Shape in Current Window |
| 18 | Next step #3 | OnClick:<br>Case 1:<br>Open 3 - Shape Grammar Result in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 19 | | OnSelectionChange:<br>  Case 1<br> (If selected option of This equals  Intermediate mode):<br>    Open 1 - Initial Shape in Current Window<br>  Case 2<br> (Else If selected option of This equals  Expert mode):<br>    Open 1 - Initial Shape in Current Window<br>  Case 3<br> (Else If selected option of This equals  Begginer mode):<br>    Open 1 - Initial Shape in Current Window |

## 1.16.3. Unnamed

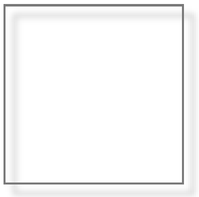
### *1.16.3.1. State1*

### *1.16.3.2. User Interface*

### 1.16.4. Rules

#### *1.16.4.1. Quadrado*

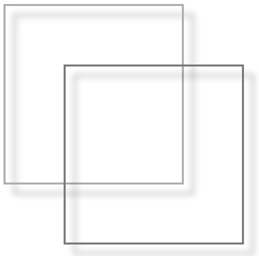
#### *1.16.4.2. User Interface*



#### *1.16.4.3. Vazio*

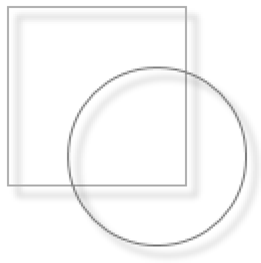
#### *1.16.4.4. Quadrado + Quadrado*

#### *1.16.4.5. User Interface*



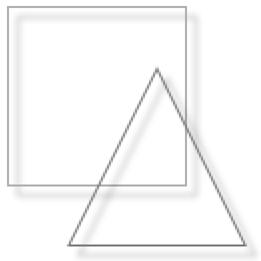
#### 1.16.4.6. Quadrado + Circulo

#### 1.16.4.7. User Interface



#### 1.16.4.8. Quadrado + Triangulo

#### 1.16.4.9. User Interface

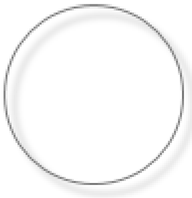


#### *1.16.4.10. Quadrado + Circulo + Triangulo*

#### *1.16.4.11. User Interface*

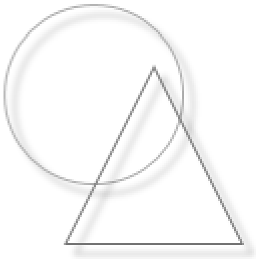
#### *1.16.4.12. Circulo*

#### *1.16.4.13. User Interface*



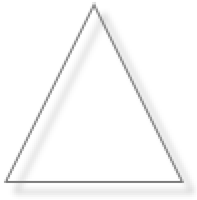
#### *1.16.4.14. Circulo + Triangulo*

#### *1.16.4.15. User Interface*



#### *1.16.4.16. Triangulo*

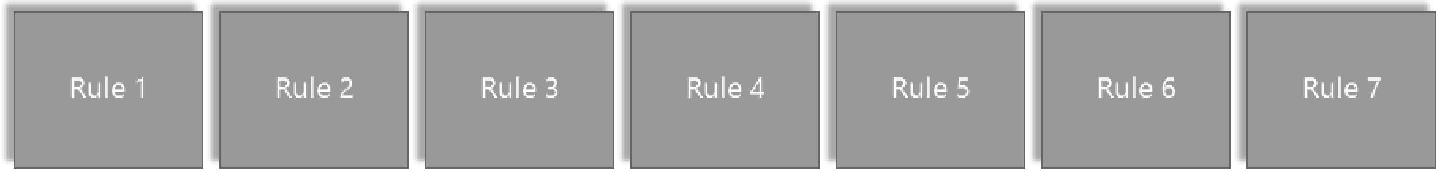
#### *1.16.4.17. User Interface*



### 1.16.5. Rules list

#### *1.16.5.1. State1*

#### *1.16.5.2. User Interface*



## 1.16.6. Telinha #1

### *1.16.6.1. Quadradinho*

### *1.16.6.2. User Interface*

#### *1.16.6.3. Limpo*

#### *1.16.6.4. User Interface*

### 1.16.6.5. Circulinho

### 1.16.6.6. User Interface

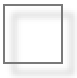
### *1.16.6.7. Triangulinho*

### *1.16.6.8. User Interface*

### 1.16.7. Telinha #2

#### *1.16.7.1. Quadrado*

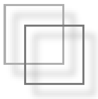
#### *1.16.7.2. User Interface*



#### *1.16.7.3. Vazio*

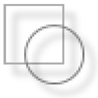
#### *1.16.7.4. Quadrado + Quadrado*

#### *1.16.7.5. User Interface*



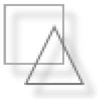
#### *1.16.7.6. Quadrado + Circulo*

#### *1.16.7.7. User Interface*



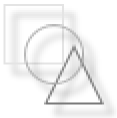
#### *1.16.7.8. Quadrado + Triangulo*

#### *1.16.7.9. User Interface*



#### *1.16.7.10. Quadrado + Circulo + Triangulo*

#### *1.16.7.11. User Interface*



#### *1.16.7.12. Circulo*

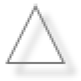
#### *1.16.7.13. User Interface*

#### *1.16.7.14. Circulo + Triangulo*

#### *1.16.7.15. User Interface*

#### *1.16.7.16. Triangulo*

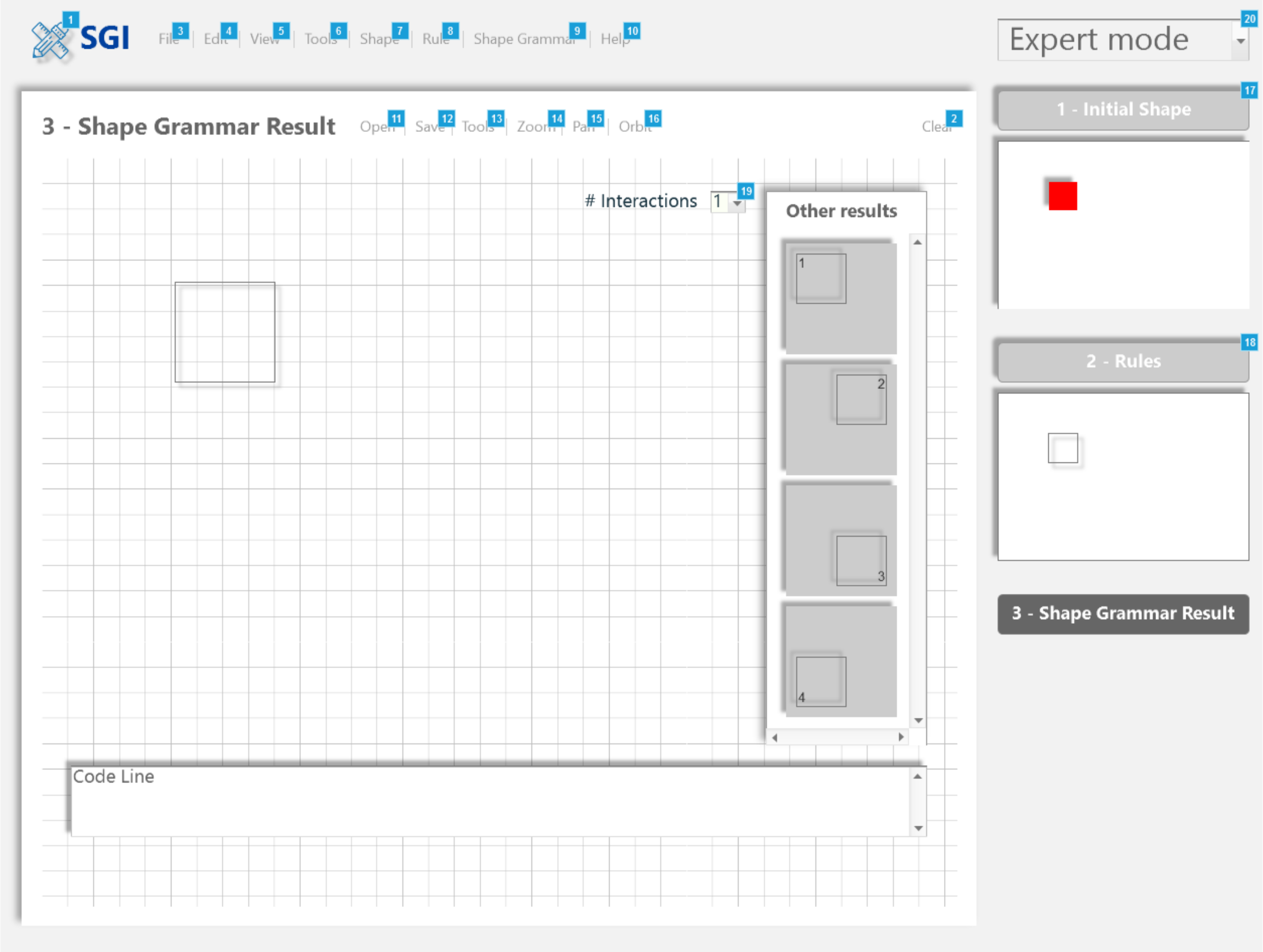
#### *1.16.7.17. User Interface*



## 1.17. 3 - Shape Grammar Result

```
OnPageLoad:
  Case 1
 (If value of f equals "q"):
    Set Telinha #2 to Quadrado,
Telinha #1 to Quadradinho,
Shape Grammar Result to Um quadrado,
other_results to Vazio
  Case 2
 (Else If value of f equals "qq"):
    Set Telinha #2 to Quadrado + Quadrado,
Telinha #1 to Quadradinho,
Shape Grammar Result to Dois quadrados,
other_results to Dois quadrados
  Case 3
 (Else If value of f equals "qc"):
    Set Telinha #2 to Quadrado + Circulo,
Telinha #1 to Circulinho,
Shape Grammar Result to Quadrado + um circulo,
other_results to Quadrado + Circulo
  Case 4
 (Else If value of f equals "qtc"):
    Set Telinha #2 to Quadrado + Circulo + Triangulo,
Telinha #1 to Quadradinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 5
 (Else If value of f equals "qt"):
    Set Telinha #2 to Quadrado + Triangulo,
Telinha #1 to Quadradinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 6
 (Else If value of f equals "c"):
    Set Telinha #2 to Circulo,
Telinha #1 to Circulinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 7
 (Else If value of f equals "ct"):
    Set Telinha #2 to Circulo + Triangulo,
Telinha #1 to Circulinho,
Shape Grammar Result to Vazio,
other_results to Vazio
  Case 7
 (Else If value of f equals "t"):
    Set Telinha #2 to Triangulo,
Telinha #1 to Triangulinho,
Shape Grammar Result to Vazio,
other_results to Vazio
```

### 1.17.1. User Interface



### 1.17.2. Widget Table

| Footnote | Name | Interactions |
|---|---|---|
| 1 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 2 | | OnClick:<br>Case 1:<br>Set Panel to State |
| 3 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 4 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 5 | | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 6 | | OnClick:<br>Case 1:<br>Open Home in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 7 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 8 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 9 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 10 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 11 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 12 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 13 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 14 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 15 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 16 |  | OnClick:<br>Case 1:<br>Open Home in Current Window |
| 17 | Next step | OnClick:<br>Case 1:<br>Open 1 - Initial Shape in Current Window |
| 18 | Next step | OnClick:<br>Case 1:<br>Open 2 - Rules in Current Window |

| Footnote | Name | Interactions |
|---|---|---|
| 19 | | OnSelectionChange:<br>Case 1<br>(If selected option of This equals 1 and value of f equals "qq"):<br>Set Shape Grammar Result to Dois quadrados, other_results to Dois quadrados<br>Case 2<br>(Else If selected option of This equals 2 and value of f equals "qq"):<br>Set Shape Grammar Result to Três quadrados, other_results to Três quadrados<br>Case 3<br>(Else If selected option of This equals 3 and value of f equals "qq"):<br>Set Shape Grammar Result to Quatro quadrados, other_results to Quatro quadrados<br>Case 4<br>(Else If selected option of This equals 4 and value of f equals "qq"):<br>Set Shape Grammar Result to Cinco quadrados, other_results to Cinco quadrados<br>Case 5<br>(Else If selected option of This equals 1 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + um circulo, other_results to Quadrado + Circulo<br>Case 6<br>(Else If selected option of This equals 2 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + dois circulo, other_results to Quadrado + dois circulos<br>Case 7<br>(Else If selected option of This equals 3 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + três circulo, other_results to Quadrado + tres circulos<br>Case 8<br>(Else If selected option of This equals 4 and value of f equals "qc"):<br>Set Shape Grammar Result to Quadrado + quatro circulo, other_results to Quadrado + quatro circulos |
| 20 | | OnSelectionChange:<br>Case 1<br>(If selected option of This equals  Intermediate mode):<br>Open 1 - Initial Shape in Current Window<br>Case 2<br>(Else If selected option of This equals  Expert mode):<br>Open 1 - Initial Shape in Current Window<br>Case 3<br>(Else If selected option of This equals  Begginer mode):<br>Open 1 - Initial Shape in Current Window |

## 1.17.3. Unnamed

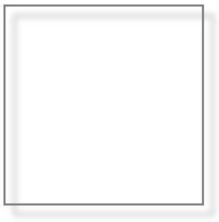
### *1.17.3.1. State1*

### *1.17.3.2. User Interface*

### 1.17.4. Shape Grammar Result

#### *1.17.4.1. Um quadrado*

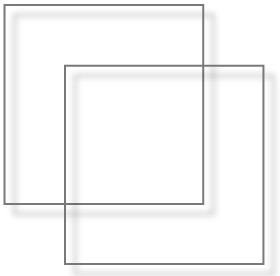
#### *1.17.4.2. User Interface*



#### *1.17.4.3. Vazio*

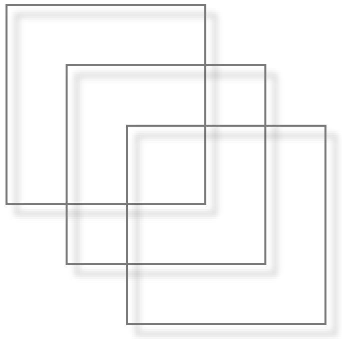
#### *1.17.4.4. Dois quadrados*

#### *1.17.4.5. User Interface*



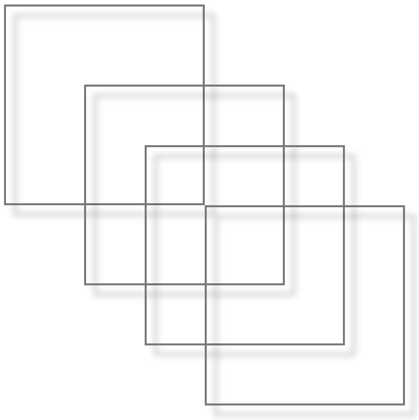
#### 1.17.4.6. Três quadrados

#### 1.17.4.7. User Interface



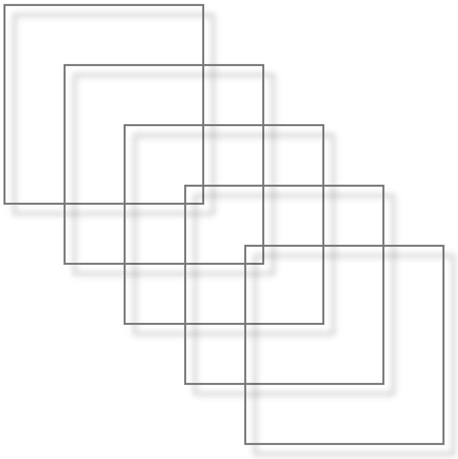
#### 1.17.4.8. Quatro quadrados

#### 1.17.4.9. User Interface



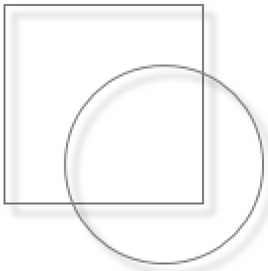
### *1.17.4.10. Cinco quadrados*

### *1.17.4.11. User Interface*



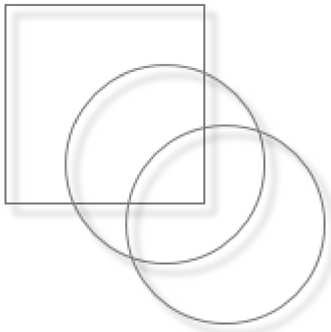
### *1.17.4.12. Quadrado + um circulo*

### *1.17.4.13. User Interface*



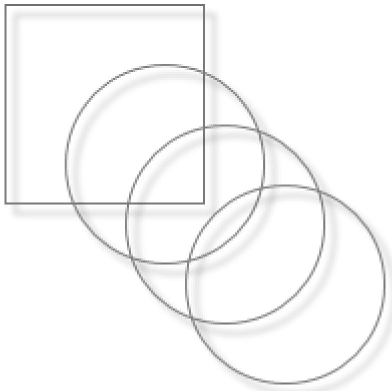
#### *1.17.4.14. Quadrado + dois circulo*

#### *1.17.4.15. User Interface*



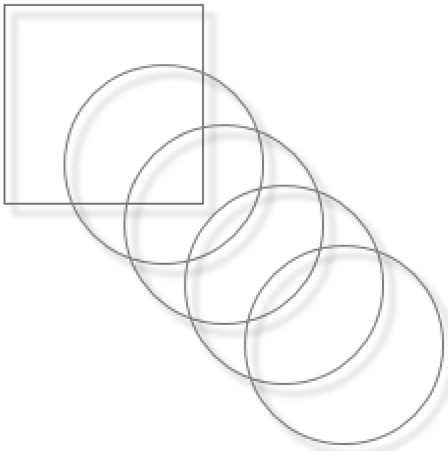
#### *1.17.4.16. Quadrado + três circulo*

#### *1.17.4.17. User Interface*



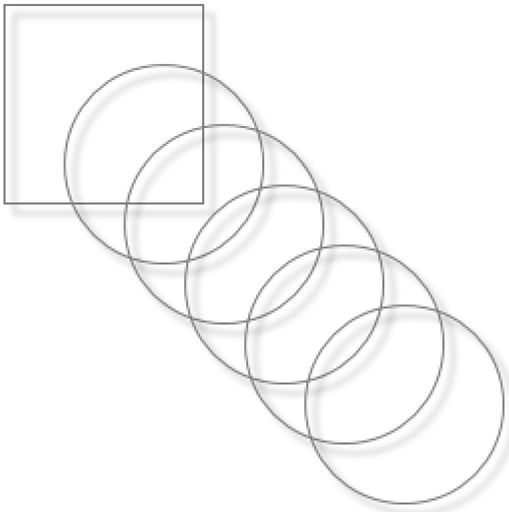
### 1.17.4.18. Quadrado + quatro circulo

### 1.17.4.19. User Interface



### 1.17.4.20. Quadrado + cinco circulos

### 1.17.4.21. User Interface



### 1.17.5. Telinha #1

#### *1.17.5.1. Quadradinho*

#### *1.17.5.2. User Interface*

### 1.17.5.3. Limpo

### 1.17.5.4. User Interface

### 1.17.5.5. Circulinho

### 1.17.5.6. User Interface

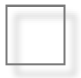
### *1.17.5.7. Triangulinho*

### *1.17.5.8. User Interface*

### 1.17.6. Telinha #2

#### *1.17.6.1. Quadrado*

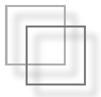
#### *1.17.6.2. User Interface*



#### *1.17.6.3. Vazio*

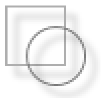
#### *1.17.6.4. Quadrado + Quadrado*

#### *1.17.6.5. User Interface*



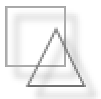
#### *1.17.6.6. Quadrado + Circulo*

#### *1.17.6.7. User Interface*



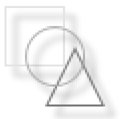
#### *1.17.6.8. Quadrado + Triangulo*

#### *1.17.6.9. User Interface*



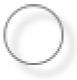
#### *1.17.6.10. Quadrado + Circulo + Triangulo*

#### *1.17.6.11. User Interface*



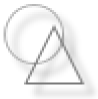
#### 1.17.6.12. Circulo

#### 1.17.6.13. User Interface



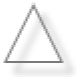
#### 1.17.6.14. Circulo + Triangulo

#### 1.17.6.15. User Interface



#### 1.17.6.16. Triangulo

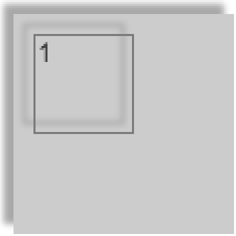
#### 1.17.6.17. User Interface

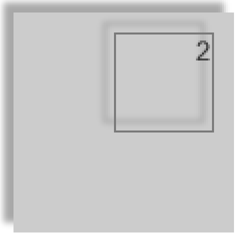


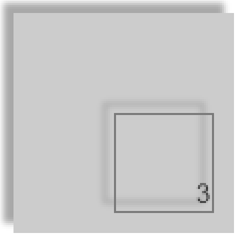
### 1.17.7. other_results

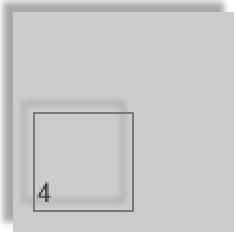
#### *1.17.7.1. Um quadrado*

#### *1.17.7.2. User Interface*





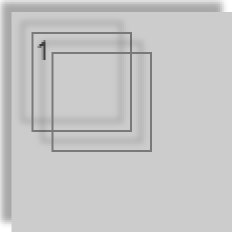


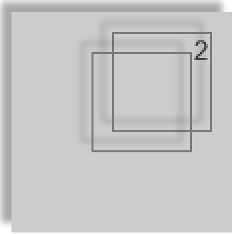


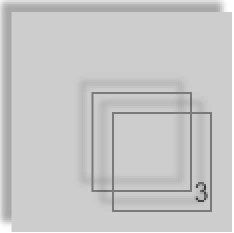
#### *1.17.7.3. Vazio*

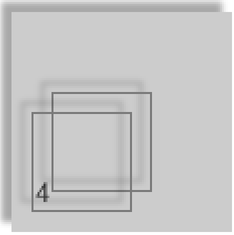
#### *1.17.7.4. Dois quadrados*

#### *1.17.7.5. User Interface*



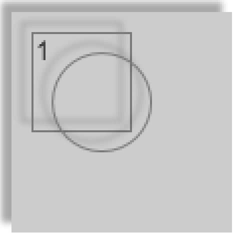


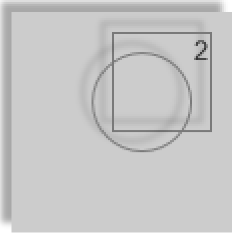


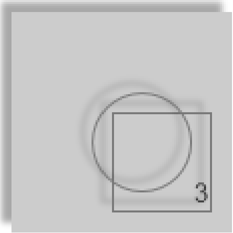


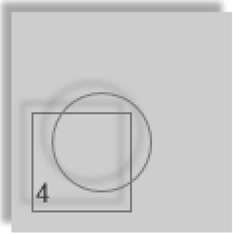
### *1.17.7.6. Quadrado + Circulo*

### *1.17.7.7. User Interface*



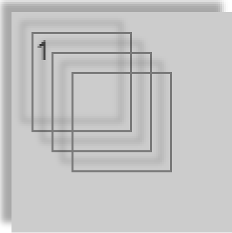


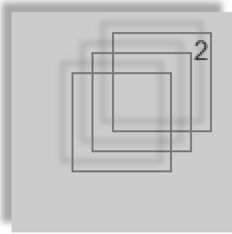


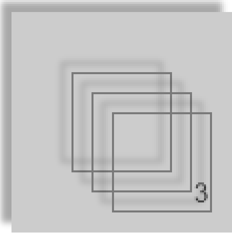


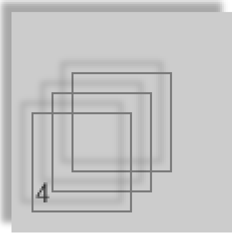
### 1.17.7.8. Três quadrados

### 1.17.7.9. User Interface



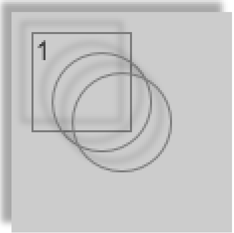


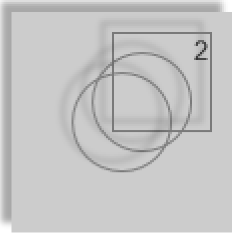


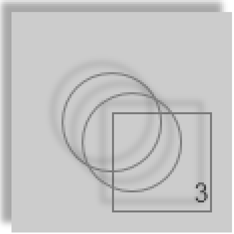


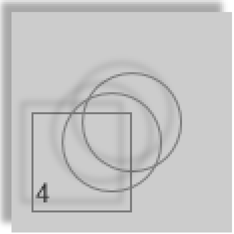
### 1.17.7.10. Quadrado + dois circulos

### 1.17.7.11. User Interface



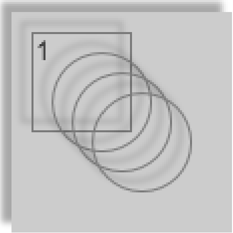


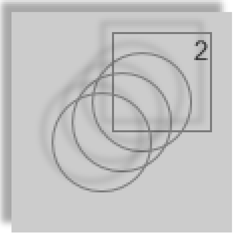


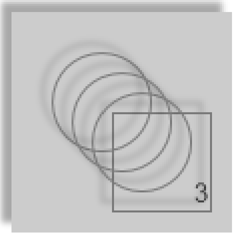


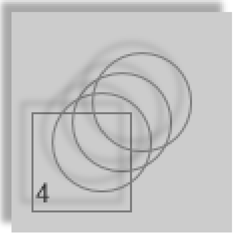
### *1.17.7.12. Quadrado + tres circulos*

### *1.17.7.13. User Interface*



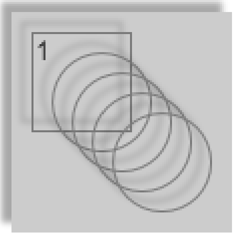


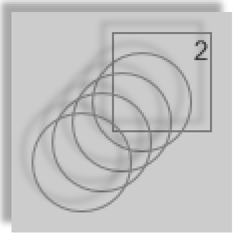


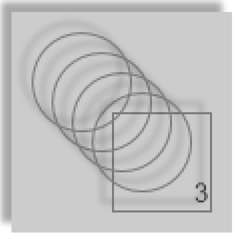


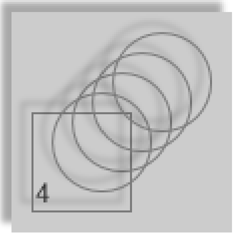
### *1.17.7.14. Quadrado + quatro circulos*

### *1.17.7.15. User Interface*



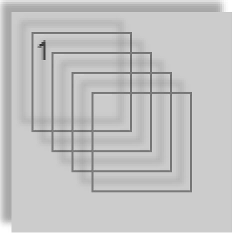


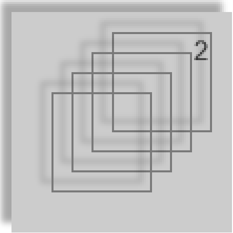


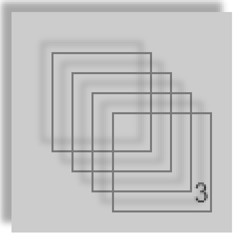


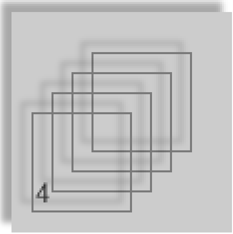
### 1.17.7.16. Quatro quadrados

### 1.17.7.17. User Interface



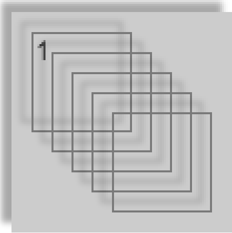


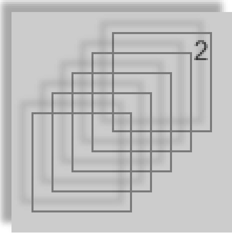


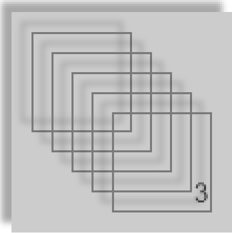


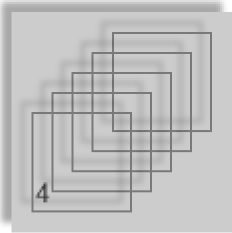
### 1.17.7.18. Cinco quadrados

### 1.17.7.19. User Interface









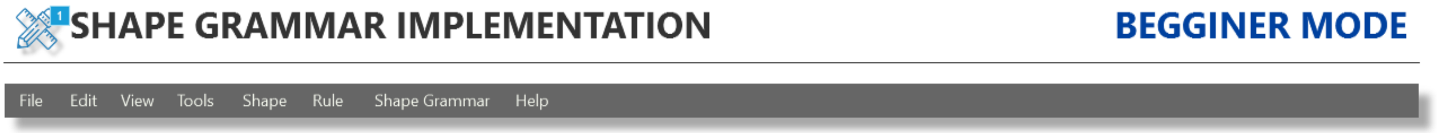
# 2. Masters

## 2.1. Master List

Topo + menu

## 2.2. Topo + menu

### 2.2.1. User Interface



### 2.2.2. Widget Table

| Footnote | Interactions |
|---|---|
| 1 | OnClick:<br>Case 1:<br>Open Home in Current Window |

# Inquérito de Usabilidade (SUS- System Usability Scale)

*Workshop – Gramáticas de Forma na Arquitetura – Interfaces Computacionais*

Inquérito de Usabilidade (SUS- System Usability Scale) requests your help. Please complete the following Customer Satisfaction Survey based on the project we recently completed for your organization. Thank you for your time.

***1.* Considero que gostaria de usar este sistema com frequência**

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

***2.* Considero o sistema desnecessariamente complexo**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

***3.* Considero o sistema fácil de usar**

□ Discordo Totalmente □ Discordo □ Concordo ☒ Concordo Totalmente

***4.* Considero que necessitaria de apoio de um técnico para ser capaz de usar o sistema**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

***5.* Considero ~~o sistema desnecessariamente complexo~~** q as várias funções deste sistema estão bem integradas.

□ Discordo Totalmente □ Discordo □ Concordo ☒ Concordo Totalmente

***6.* Considero que o sistema apresenta muitas inconsistências**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

V.S.F.F.

***7. Considero que a maioria das pessoas aprenderia a usar este sistema rapidamente***

□ Discordo Totalmente □ Discordo □ Concordo ☒ Concordo Totalmente

***8. Considero o sistema complicado de usar***

☒ Discordo Totalmente □ Discordo □ Concordo □ Concordo Totalmente

***9. Senti-me muito confiante ao usar este sistema***

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

***10. Considero que necessitaria de aprender muitas coisas antes de conseguir usar o sistema***

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

Nome: João Ventura Salvador Lopes

Email: joint_ventura@hotmail.com

Curso e Ano: 3º Doutoramento

Data: 06/12/2016

Muito Obrigada pela vossa colaboração

Joana Tching,

ISCTE-IUL Dez 2016

ISTAR-IUL
INFORMATION SCIENCES, TECHNOLOGIES
AND ARCHITECTURE RESEARCH CENTER
UNIVERSITY INSTITUTE OF LISBON

ISCTE-IUL
Instituto Universitário de Lisboa

## Declaração de Consentimento

Eu, João Ventura Salvador Lopes, pessoa singular titular de dados pessoais declaro para os devidos efeitos autorizar por minha livre, específica e informada vontade, a captação, tratamento e respectiva difusão da imagem própria e os dados pessoais inerentemente a esta associados para investigação no âmbito do doutoramento ***IM-SGI: interface model for shape grammar implementations*** apenas para os fins anteriormente indicados e durante o período de tempo estritamente necessário à prossecução dessas mesmas finalidades.

Lisboa, 06 Dezembro 2016

João Ventura Salvador Lopes

ISTAR-IUL
INFORMATION SCIENCES, TECHNOLOGIES
AND ARCHITECTURE RESEARCH CENTER
UNIVERSITY INSTITUTE OF LISBON

ISCTE-IUL
Instituto Universitário de Lisboa

## Declaração de Consentimento

Eu, Guida de Jesus Macedo Ramos,
pessoa singular titular de dados pessoais declaro para os devidos efeitos autorizar por minha livre, específica e informada vontade, a captação, tratamento e respectiva difusão da imagem própria e os dados pessoais inerentemente a esta associados para investigação no âmbito do doutoramento ***IM-SGI: interface model for shape grammar implementations*** apenas para os fins anteriormente indicados e durante o período de tempo estritamente necessário à prossecução dessas mesmas finalidades.

Lisboa, 06 Dezembro 2016

Guida M. Ramos

# Inquérito de Usabilidade (SUS- System Usability Scale)

*Workshop – Gramáticas de Forma na Arquitetura – Interfaces Computacionais*

Inquérito de Usabilidade (SUS- System Usability Scale) requests your help. Please complete the following Customer Satisfaction Survey based on the project we recently completed for your organization. Thank you for your time.

*1.* **Considero que gostaria de usar este sistema com frequência**

☐ Discordo Totalmente ☒ Discordo ☐ Concordo ☐ Concordo Totalmente

*2.* **Considero o sistema desnecessariamente complexo**

☒ Discordo Totalmente ☐ Discordo ☐ Concordo ☐ Concordo Totalmente

*3.* **Considero o sistema fácil de usar**

☐ Discordo Totalmente ☐ Discordo ☐ Concordo ☒ Concordo Totalmente

*4.* **Considero que necessitaria de apoio de um técnico para ser capaz de usar o sistema**

☐ Discordo Totalmente ☐ Discordo ☐ Concordo ☒ Concordo Totalmente

*5.* **Considero ~~o sistema desnecessariamente complexo~~** que as várias funções estavam bem integradas

☐ Discordo Totalmente ☐ Discordo ☐ Concordo ☒ Concordo Totalmente

*6.* **Considero que o sistema apresenta muitas inconsistências**

☒ Discordo Totalmente ☐ Discordo ☐ Concordo ☐ Concordo Totalmente

V.S.F.F.

*7.* ***Considero que a maioria das pessoas aprenderia a usar este sistema rapidamente***

□ Discordo Totalmente □ Discordo □ Concordo ☒ Concordo Totalmente

*8.* ***Considero o sistema complicado de usar***

☒ Discordo Totalmente □ Discordo □ Concordo □ Concordo Totalmente

*9.* ***Senti-me muito confiante ao usar este sistema***

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

*10.* ***Considero que necessitaria de aprender muitas coisas antes de conseguir usar o sistema***

☒ Discordo Totalmente □ Discordo □ Concordo □ Concordo Totalmente

Nome: Guido Ramos

Email: gjmrs@iscte-iul.pt

Curso e Ano: Mestrado Integrado em Arquitectura 5º Ano

Data: 6/12/2016

Muito Obrigada pela vossa colaboração

Joana Tching,

ISCTE-IUL Dez 2016

ISTAR IUL
INFORMATION SCIENCES, TECHNOLOGIES
AND ARCHITECTURE RESEARCH CENTER
UNIVERSITY INSTITUTE OF LISBON

ISCTE IUL
Instituto Universitário de Lisboa

## Declaração de Consentimento

Eu, SOFIA PIMENTEL SEBASTIÃO,

pessoa singular titular de dados pessoais declaro para os devidos efeitos autorizar por minha livre, específica e informada vontade, a captação, tratamento e respectiva difusão da imagem própria e os dados pessoais inerentemente a esta associados para investigação no âmbito do doutoramento ***IM-SGI: interface model for shape grammar implementations*** apenas para os fins anteriormente indicados e durante o período de tempo estritamente necessário à prossecução dessas mesmas finalidades.

Lisboa, 06 Dezembro 2016

Sofia Sebastião

# Inquérito de Usabilidade (SUS- System Usability Scale)

*Workshop – Gramáticas de Forma na Arquitetura – Interfaces Computacionais*

Inquérito de Usabilidade (SUS- System Usability Scale) requests your help. Please complete the following Customer Satisfaction Survey based on the project we recently completed for your organization. Thank you for your time.

*1.* **Considero que gostaria de usar este sistema com frequência**

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

*2.* **Considero o sistema desnecessariamente complexo**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

*3.* **Considero o sistema fácil de usar**

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

*4.* **Considero que necessitaria de apoio de um técnico para ser capaz de usar o sistema**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

*5.* **Considero ~~o sistema desnecessariamente complexo~~** QUE AS VÁRIAS FUNÇÕES NESTE SISTEMA FORAM BEM INTEGRADAS

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

*6.* **Considero que o sistema apresenta muitas inconsistências**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

V.S.F.F.

*7.* ***Considero que a maioria das pessoas aprenderia a usar este sistema rapidamente***

☐ Discordo Totalmente ☒ Discordo ☐ Concordo ☐ Concordo Totalmente

*8.* ***Considero o sistema complicado de usar***

☐ Discordo Totalmente ☒ Discordo ☐ Concordo ☐ Concordo Totalmente

*9.* ***Senti-me muito confiante ao usar este sistema***

☐ Discordo Totalmente ☐ Discordo ☒ Concordo ☐ Concordo Totalmente

*10.* ***Considero que necessitaria de aprender muitas coisas antes de conseguir usar o sistema***

☐ Discordo Totalmente ☒ Discordo ☐ Concordo ☐ Concordo Totalmente

Nome: SOFIA PIMENTEL SEBASTIÃO

Email: SOPHIASEBASTIAO@HOTMAIL.COM

Curso e Ano: MESTRADO INTEGRADO EM ARQUITECTURA , 5º ANO

Data: 06-12-2016

Muito Obrigada pela vossa colaboração

Joana Tching,

ISCTE-IUL Dez 2016

ISTAR IUL
INFORMATION SCIENCES, TECHNOLOGIES
AND ARCHITECTURE RESEARCH CENTER
UNIVERSITY INSTITUTE OF LISBON

ISCTE IUL
Instituto Universitário de Lisboa

## Declaração de Consentimento

Eu, João Duarte Batalha Parcelas,

pessoa singular titular de dados pessoais declaro para os devidos efeitos autorizar por minha livre, específica e informada vontade, a captação, tratamento e respectiva difusão da imagem própria e os dados pessoais inerentemente a esta associados para investigação no âmbito do doutoramento ***IM-SGI: interface model for shape grammar implementations*** apenas para os fins anteriormente indicados e durante o período de tempo estritamente necessário à prossecução dessas mesmas finalidades.

Lisboa, 06 Dezembro 2016

João Duarte Batalha Parcelas

*7.* ***Considero que a maioria das pessoas aprenderia a usar este sistema rapidamente***

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

*8.* ***Considero o sistema complicado de usar***

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

*9.* ***Senti-me muito confiante ao usar este sistema***

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

*10.* ***Considero que necessitaria de aprender muitas coisas antes de conseguir usar o sistema***

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

Nome: João Duarte Botelho Parcelas

Email: joao.parcelas@gmail.com

Curso e Ano: 2°ano Licenciatura Arquitetura

Data: 6/12/2016

Muito Obrigada pela vossa colaboração

Joana Tching,

ISCTE-IUL Dez 2016

# Inquérito de Usabilidade (SUS- System Usability Scale)

*Workshop – Gramáticas de Forma na Arquitetura – Interfaces Computacionais*

---

Inquérito de Usabilidade (SUS- System Usability Scale) requests your help. Please complete the following Customer Satisfaction Survey based on the project we recently completed for your organization. Thank you for your time.

*1.* **Considero que gostaria de usar este sistema com frequência**

□ Discordo Totalmente □ Discordo □ Concordo □ Concordo Totalmente

*2.* **Considero o sistema desnecessariamente complexo**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

*3.* **Considero o sistema fácil de usar**

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

*4.* **Considero que necessitaria de apoio de um técnico para ser capaz de usar o sistema**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

*5.* **Considero o sistema desnecessariamente complexo** Que as várias funções neste sistema foram Bem integradas

□ Discordo Totalmente ■ Discordo ☒ Concordo □ Concordo Totalmente

*6.* **Considero que o sistema apresenta muitas inconsistências**

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

V.S.F.F.

ISTAR IUL
INFORMATION SCIENCES, TECHNOLOGIES
AND ARCHITECTURE RESEARCH CENTER
UNIVERSITY INSTITUTE OF LISBON

ISCTE IUL
Instituto Universitário de Lisboa

## Declaração de Consentimento

Eu, Tiago Gonçalves,

pessoa singular titular de dados pessoais declaro para os devidos efeitos autorizar por minha livre, específica e informada vontade, a captação, tratamento e respectiva difusão da imagem própria e os dados pessoais inerentemente a esta associados para investigação no âmbito do doutoramento ***IM-SGI: interface model for shape grammar implementations*** apenas para os fins anteriormente indicados e durante o período de tempo estritamente necessário à prossecução dessas mesmas finalidades.

Lisboa, 06 Dezembro 2016

Tiago Gonçalves

# Inquérito de Usabilidade (SUS- System Usability Scale)

*Workshop – Gramáticas de Forma na Arquitetura – Interfaces Computacionais*

Inquérito de Usabilidade (SUS- System Usability Scale) requests your help. Please complete the following Customer Satisfaction Survey based on the project we recently completed for your organization. Thank you for your time.

**1. Considero que gostaria de usar este sistema com frequência**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

**2. Considero o sistema desnecessariamente complexo**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

**3. Considero o sistema fácil de usar**

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

**4. Considero que necessitaria de apoio de um técnico para ser capaz de usar o sistema**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

considero que varias funções foram bem integradas

**5. ~~Considero o sistema desnecessariamente complexo~~**

□ Discordo Totalmente ☒ Discordo ☒ Concordo □ Concordo Totalmente

**6. Considero que o sistema apresenta muitas inconsistências**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

V.S.F.F.

*7.* ***Considero que a maioria das pessoas aprenderia a usar este sistema rapidamente***

□ Discordo Totalmente ☒ Discordo ☒ Concordo □ Concordo Totalmente

*8.* ***Considero o sistema complicado de usar***

☒ Discordo Totalmente □ Discordo □ Concordo □ Concordo Totalmente

*9.* ***Senti-me muito confiante ao usar este sistema***

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

*10.* ***Considero que necessitaria de aprender muitas coisas antes de conseguir usar o sistema***

☒ Discordo Totalmente □ Discordo □ Concordo □ Concordo Totalmente

Nome: tiago Gonçalves

Email: tiago.goncalves~~@~~94@gmail.com

Curso e Ano: Arquitectura 3ano

Data: 06-12-2016

Muito Obrigada pela vossa colaboração

Joana Tching,

ISCTE-IUL Dez 2016

# Inquérito de Usabilidade (SUS- System Usability Scale)

*Workshop – Gramáticas de Forma na Arquitetura – Interfaces Computacionais*

Inquérito de Usabilidade (SUS- System Usability Scale) requests your help. Please complete the following Customer Satisfaction Survey based on the project we recently completed for your organization. Thank you for your time.

*1.* **Considero que gostaria de usar este sistema com frequência**

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

*2.* **Considero o sistema desnecessariamente complexo**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

*3.* **Considero o sistema fácil de usar**

□ Discordo Totalmente □ Discordo ☒ Concordo □ Concordo Totalmente

*4.* **Considero que necessitaria de apoio de um técnico para ser capaz de usar o sistema**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

que vária funções estavam bem integrados.

*5.* **Considero o sistema desnecessariamente complexo**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

*6.* **Considero que o sistema apresenta muitas inconsistências**

□ Discordo Totalmente ☒ Discordo □ Concordo □ Concordo Totalmente

V.S.F.F.

*7.* ***Considero que a maioria das pessoas aprenderia a usar este sistema rapidamente***

☐ Discordo Totalmente ☐ Discordo ☒ Concordo ☐ Concordo Totalmente

*8.* ***Considero o sistema complicado de usar***

☐ Discordo Totalmente ☒ Discordo ☐ Concordo ☐ Concordo Totalmente

*9.* ***Senti-me muito confiante ao usar este sistema***

☐ Discordo Totalmente ☐ Discordo ☒ Concordo ☐ Concordo Totalmente

*10.* ***Considero que necessitaria de aprender muitas coisas antes de conseguir usar o sistema***

☐ Discordo Totalmente ☒ Discordo ☐ Concordo ☐ Concordo Totalmente

Nome: Cheila Cordeiro Arruda

Email: arruda.cheila@gmail.com

Curso e Ano: Arq. 4º ano

Data: 6/12/16

Muito Obrigada pela vossa colaboração

Joana Tching,

ISCTE-IUL Dez 2016

ISTAR IUL
INFORMATION SCIENCES, TECHNOLOGIES
AND ARCHITECTURE RESEARCH CENTER
UNIVERSITY INSTITUTE OF LISBON

ISCTE IUL
Instituto Universitário de Lisboa

## Declaração de Consentimento

Eu, Cheila Cordeiro Arruda,

pessoa singular titular de dados pessoais declaro para os devidos efeitos autorizar por minha livre, específica e informada vontade, a captação, tratamento e respectiva difusão da imagem própria e os dados pessoais inerentemente a esta associados para investigação no âmbito do doutoramento ***IM-SGI: interface model for shape grammar implementations*** apenas para os fins anteriormente indicados e durante o período de tempo estritamente necessário à prossecução dessas mesmas finalidades.

Lisboa, 06 Dezembro 2016

Cheila Arruda