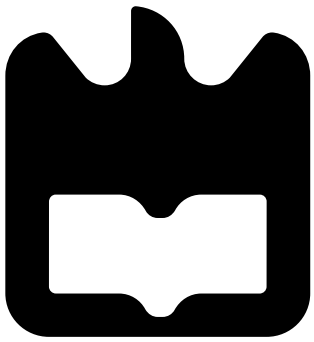
Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2014

Michael André Pinto Domingues

Plataforma de Modelação e Controlo para Impressoras 3D
Modelling and Control Platform for 3D printers



Michael André Pinto Domingues

Plataforma de Modelação e Controlo para Impressoras 3D
Modelling and Control Platform for 3D printers

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Joaquim João Estrela Ribeiro Silvestre Madeira do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri

Presidente	**Paulo Miguel de Jesus Dias** Professor Auxiliar, Universidade de Aveiro
Arguente Principal	**Frutuoso Gomes Mendes da Silva** Professor Auxiliar, Dep.Informática da Universidade da Beira Interior
Orientador	**Joaquim João Estrela Ribeiro Silvestre Madeira** Professor Auxiliar, Universidade de Aveiro

Dedication

To my beautiful bride Diana, because everything is so meaningful with her...
I love you so much!

Acknowledgements

During the preparation of this dissertation were involved directly and indirectly various people. I express my gratitude and thanks to all of them. However I thank in particular:

the founders and partners of *BEEVERYCREATIVE* for the opportunity they gave me to accomplish this dissertation.

Professor Joaquim Madeira, for his splendid personality and human characteristics that provided an excellent working environment, but also for the exemplary scientific and technique guidance that has shown over time. It was a great pleasure to work together.

Professor Beatriz Sousa Santos, for simplicity and kindness, by the constant availability and the tremendous help in achieving the usability testing. The words of support and knowledge transmitted allowed reaching of one of the objectives of the dissertation.

Alexander Torres, my colleague and friend for the proof-reading of the entire work. His kindness, great willingness and desire to help me, was a support in order to move forward knowing I had his help. Glad to meet and work with you mate.

my parents, for always being there when I needed, by inexhaustible affection, the strength to believe in the success and rewards of trying to be better every day. Their presence in the most difficult moments and their words of tenderness, meant a lot to me. You made me see what is really important in life. Many many thanks, mom and dad.

to my dear brother, the constant motivation, the affection in those big hugs that gave me so much confidence, by grinning with delight when I went over obstacles. Thanks for being who you are, for being an example of who I want to be, the wonderful, cheerful person and tireless fighter in the struggle for the broad objectives of life and a better person tomorrow. I will always be with you big brother!

to my beloved bride, for all that she means to me and the tremendous desire of spending the rest of my life with her. Thank you for your warm presence, your affection so special, the safety and welfare of your hugs, your strong words of confidence, your hand in mine and you always beside me.

my great and everlasting friend Bernardo, for your presence when I needed, for your capacity to amaze me by your words and surprises, for your constant concern. Thank you for having contributed every single day with moments of laughter, motivation and confidence against the obstacles, for having been you tirelessly everyday, on the best and worst moments. You will be forever in my hearth!

to all who were involved in this dissertation

Thanks a lot

palavras chave: impressão 3D, processamento 3D, malha, FFF, CAM

Resumo

As impressoras 3D estão cada vez mais baratas sendo usadas em variados nichos de mercado como na arquitectura, design industrial, engenharia automóvel e aeroespacial, fornecendo uma forma fácil e barata de produzir partes e maquetes.

Embora já existam aplicações de controlo de impressoras 3D, apresentam uma interface pobre e são difíceis de usar ou perceber devido ao conhecimento necessário. A maior parte das impressoras 3D permitem importar e processar um modelo 3D e imprimir com recurso a configurações avançadas. Tendo isto em consideração pretende-se desenvolver e avaliar uma aplicação constituida por funcionalidades básicas e avançadas com uma interface de utilizador estável e simples de usar.

A aplicação (*BEESOFT*) foi projectada e desenvolvida tendo em consideração os objectivos acima referidos. E potencia o modo como os utilizadores interagem com a impressora *BEETHEFIRST*, fornecendo um ambiente integrado onde cada utilizador pode, de forma orientada, usar as funcionalidades para rapidamente imprimir. A interface de utilizador simples de usar, os assistentes e auto-explicativos para ajudar a configurar rapidamente a impressora 3D, a robustez, o controlo de erros, a estabilidade da interface de utilizador, o motor 3D e o protocolo de comunicação, destacam esta aplicação de outras existentes.

Esta nova aplicação foi avaliada através de um plano de testes de usabilidade e de uma análise de avaliação heurística. Os resultados confirmam que o *BEESOFT* é uma aplicaçao fácil de usar e que permite aos utilizadores, com variados níveis de experiência, imprimir facilmente um modelo 3D, mas também reveleram alguns problemas de usabilidade no motor 3D aquando de operações de modelação. A reacção positiva e os resultados mostraram que a interface de utilizador do *BEESOFT* é simples, funcional, com uma linguagem acessível e com perfis de impressão optimizados, possibilitando o seu uso em qualquer sector da indústria, educação, etc.

keywords: 3D printing, 3D rendering, mesh, FFF, CAM

Abstract

3D printers are becoming cheaper and widely used in various market segments as architecture, industrial design, automotive and aerospace engineering, providing an easy and cheaper way to produce parts and mockups.

Although applications for controlling 3D printers exist, they are still poor in graphical design and difficult to use or understand due the advanced knowledge required. Most 3D printing applications allow importing and rendering a model with a 3D engine, and printing it with advanced configurations. Taking this into consideration it is intended to design and evaluate an application that presents common and advanced funcionalities through a simple, stable and easy to use user interface.

An application (*BEESOFT*) was designed and developed taking the above objectives into consideration. It enhances the way users interact with the *BEETHEFIRST* printer, by providing an integrated environment where each user can, in a coordinated way, use the functionalities to quickly print. The simple user interface, the guided and self-explanatory wizards to help configure quickly the 3D printer and the robustness, error control and stability of the user interface, 3D engine and the communication protocol, distinguishes it from the other applications.

This new application was evaluated using an usability testing plan and a heuristic evaluation analysis. Results confirm *BEESOFT* is an easy to use application that allows users with differents degrees of expertise to easily print a 3D model, but they also revealed some usability problems in the 3D canvas when modeling. The positive reaction and the results show *BEESOFT* simple, user spoken language and functional user interface with optimized printing profiles can be used in any sector of industry, education, etc.

Contents

List of Figures

List of Tables

List of Acronyms

ABS Acrylonitrile Butadine Styrene.

ACK Acknowledge.

AMF Additive Manufacturing File Format.

CAD Computer-Aided Design.

CAM Computer-Aided Manufacturing.

CNC Computer Numerical Control.

FFF Fused Filament fabrication.

OS Operating System.

PLA Polylactic Acid.

STL STereoLithography.

URBs USB Request Blocks.

USB Universal Serial Bus.

Chapter 1

Introduction

This thesis presents the main ideas and basic stages of design and development of a 3D printing application called *BEESOFT* by *BEEVERYCREATIVE*, which is a company located in Ílhavo (Aveiro, Portugal) that produced the first portuguese 3D printer, called *BEETHEFIRST*, with the goal of challenging the status quo of 3D printing as we know it.

As part of the work done in the company *BEEVERYCREATIVE* is being developed an application to control their 3D printer. *BEESOFT* allows the control of *BEETHEFIRST* and is an application that manages the rendering of an imported model, the 3D printer calibration and the printing of the model chosen.

Concepts and also a global understanding of what a platform of this kind can be and what functionalities it should have to fulfill the basic requirements, will be addressed introducing marks on the development process as well as the mockups, the committed final user interface and a user testing plan. Along with thesis, a global, technical and explanatory view of what is 3D printing is provided, how can we develop an application of this kind and how it works.

1.1 Motivation

3D printing has been widely used in the last few years with many developed projects for home or industrial solutions with results depending on the fabrication method and the consumables. The growth of this technology, both in hardware and software, has lead to an increased opportunity for companies to develop new consumables and new gadgets to take advantage of the enhancements and therefore explore the mechanical limitations. This opportunity enabled the emergence of numerous companies and distinct printers with different prices and target segments.

The evolution of hardware components and the market absence for new solutions and funcionalities in the market required that computer-aided manufacturing software support that new hardware improvements. Independently of price, each user expects the system (printer and the controlling software) to be as stable and reliable as possible with advanced settings available to control the printer, but technology and frameworks have been limiting the development of an easy to use system due to processing time, processing complexity and integration of modules responsible for the most important tasks. 3D printing applications have many issues and are the weaker component of a 3D printing system due to Operating System (OS) and frameworks compatibility and limitations, reducing user options and application functionalities.

The ability to apply modeling operations to an imported virtual object and transform it according to printer specifications is achieved by a 3D rendering framework. This framework renders all the transformations on a display window so the user can easily control them, through easy input methods such as using a mouse and keyboard.

Handling this user interaction with the printer, as well as processing and transmitting the scene data containing the virtual object, is a responsability of the communication protocol. It has to ensure a stable, reliable and error-free communication channel, so all information exchanged is delivered and recovered correctly to ensure no data is lost. These frameworks (3D engine and communication protocol) and their integration within the software modules are part of the complexity this kind of application has to handle, and which funcionalities it can provide.

There is a constant interaction between modules and they need to be robust enough to ensure there are no exceptions untreated. The integration of new hardware functionalities, modules responsible to operate under several operating systems with varied processing complexities and the global system behavior can worsen user experience. The software limitations are therefore the successive limitations each module has and the system responsiveness and ease of use can be bad if the interface isn't easy to use and self explanatory enough.

These aspects and the specifications of the model imported into the application, such as file size and detail level, affect the application performance limiting the capability it has to satisfy the user needs. To get to know what are those limitations and other problems, applications can be submitted to evaluation tests for further improvement. They evaluate important modules and funcionalities using tests to discover the largest number of problems possible contributing to several cycles of re-evaluation and re-deployment.

To test the capabilities, the needs an application of a 3D printing like this will be identified as well a description of the architecture to the achieve the correct integration of the various modules. This analysis will produce a mockup of a user interface for *BEESOFT*, as well as a proposed implementation that will take into consideration some usability practices. After an implementation of the necessary user interface elements, as well as all the functionalities to make the most of *BEETHEFIRST*, tests will be done to adequately demonstrate the proper functioning of the 3D printing software and hardware controllers.

The development of *BEESOFT* will help to understand what functionalities a 3D printer controling application needs and what problems arise from the integration of several key modules. Furthermore, will also be addressed the limitations associated with the technology and how usability testing can help improve the usability and performance as well as interoperability with other devices from other technologies.

1.2 Goals

The work in this thesis aims primarily to focus on the development of a 3D printing application to control *BEETHEFIRST* printer. As secondary objectives it has the intention to explain how a 3D printing platform should be, how it can communicate with the 3D printer and how it handles the conversion and manipulation of a virtual model to the printed object.

From a top down approach and to achieve the goals, it will be necessary to go through the following steps:

- familiarity with the concepts of technology, model design, consumables, file formats and capabilities

- familiarity with some 3D printing hardware and software controllers and how they work
- understand the slicer engine and its purposes
- awareness of the tecnhonology deficiencies
- understand user interface design patterns
- develop a user interface
- develop a 3D engine
- develop a communication protocol
- perform a usability testing plan
- demonstrate applicability of the application to other 3D services

To better understand the developed application — *BEESOFT* — it is necessary to understand the technology concepts and methodologies as well as the global operation scheme of this architecture.

1.3 Architecture

The 3D printer platform is comprised of three main components: hardware, software and communication protocol (figure 1.1) that work together — *BEESOFT* graphical user interface encapsulates a 3D engine and a model interpretation agent (slicing engine for mesh model processing) that generates tasks to the *BEETHEFIRST* 3D printer. This communication is done using data streams that deliver the information to the firmware hosted in the printer, in the electronic board, that receives the data from the jobs into mechanical commands — the way the parsing is done will be explained later.

The software module defines the interface look and feel and the funcionalities responsible for the major operations using feedback to help the user during the processes. There are two sub-modules responsible for major operations: the 3D engine that renders a virtual model, represented in an STL file format, into a polygonal mesh model that is subsequently processed by the slicer engine, to generate a queue of commands to be sent to the 3D printer. This communication is ensured by a communication protocol that allows USB connection between the electronic board and the software (figure 1.1).

The hardware module boots the firmware when *BEETHEFIRST* is switched on and processes the commands received in the data streams using a parser and interruption handlers. This allows receiving, processing and sending information anytime to the application or to the board ports to perform the mechanical commands and control actions.

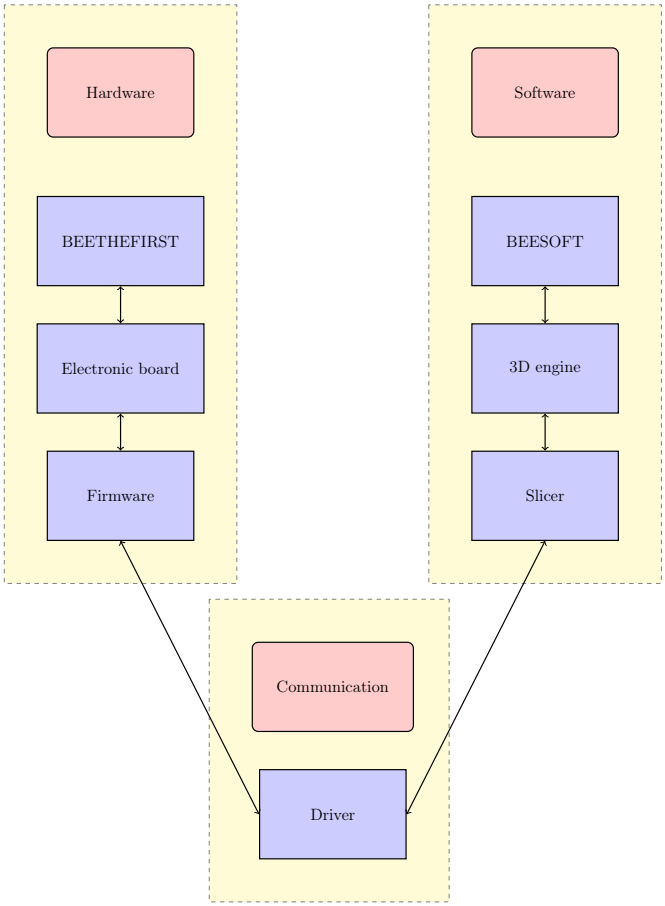


Figure 1.1: Operation scheme: the *BEESOFT* user interface and the *BEETHEFIRST* printer

1.4 Overview

This thesis is divided in six chapters:

- Introduction — the motivation for this thesis is presented
- 3D Printing — the history and terminology along with model design and rendering aspects are addressed
- User Interface — good patterns for a 3D application as the mockups over these good pratices are addressed and explained. Based on these pratices and the mockups the implementation of the final user interface is presented;
- Implementation details — a global and short explanation of the basic concepts behind application modules with a technical approach how they were developed within the user interface
- User testing and results — an analysis of the user testing plan used in *BEESOFT* to validate the usability and the proper functioning. Results are also shown.

- Conclusions and future work — conclusions about the work done are addressed as well as future work that can be done in order to improve the application based on the results of the user testing plan.

Chapter 2

3D Printing

This chapter introduces the basic concepts of 3D printing technology, from history to functionalities. It will be explained how this technology appeared and how the first 3D printing community contributed in that way. Although current hardware drivers and software are constantly evolving, the printer and its software have several weaknesses such as the complexity of the 3D rendering and the manipulation of models. This rendering is driven by existing file formats, allowing the graphical representation of the models, that are interpreted by a mesh interpretation algorithm and then printed. This processing defines the model in a sequence of movement commands and consumables extrusion layer by layer. The next sections will explain each 3D printing module's role and details and how they make part of a functional system with integrated functionalities, identifying their dependencies and limitations.

2.1 3D Printing

3D printing emerged in 1984 when Chuck Hull of 3D Systems Corp created the first 3D printer [1]. From a digital model made by any available Computer-Aided Design (CAD) application, it is possible to create a three dimensional object by an additive process where successive layers are laid down in different ways according to an algorithm (slicer) which converts the model in printing commands that are interpreted and executed. This is a unique process when compared with other techniques based on Computer Numerical Control (CNC) drilling and cutting, where a raw material is cut into the desired shape and size by a controlled removal process.

2.1.1 Evolution

Improvements in hardware and software have enabled the emergence of 3D models galleries and communities like the RepRap project who founded the first and largest 3D community [2]. The appearance and growth of several 3D printing communities, led to open source software and patches for specific printers and drivers.

One of the first surveys on 3D printing commmunities revealed an increasing number of 3D printing services and growing economical interest due to new manufacturer start-ups and Kickstarter-driven 3D printers. It also emphasized the interest of new target segments instead of being only the guru community the major consumer. This shows how culture, interest and technology products adoption boosted new segments and new target communities, so-

called early adopters (figure 2.1). This survey also revealed Makerbot and RepRap printers were the systems most used by those surveyed (figure 2.2) and recent funded companies, as Stratasys and 3DSystems, are looking to expand and conquer consumer market more rapidly than expected.

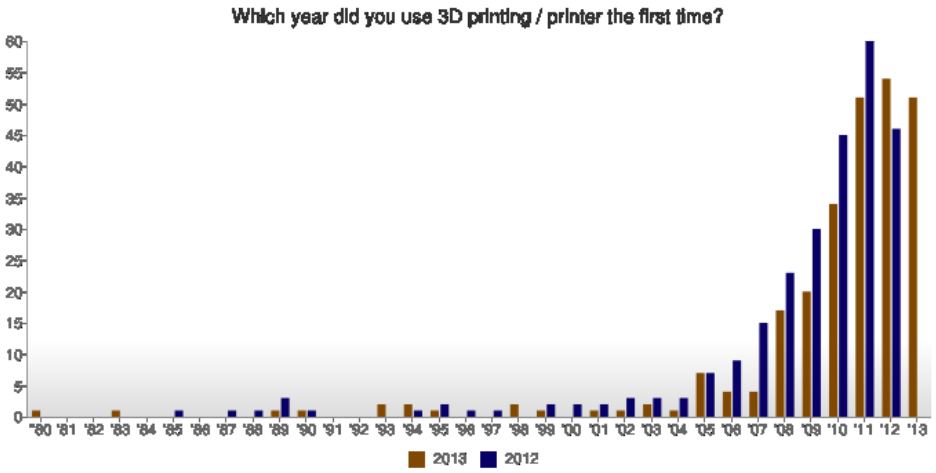


Figure 2.1: Inquiry results about the first 3D experience [3]

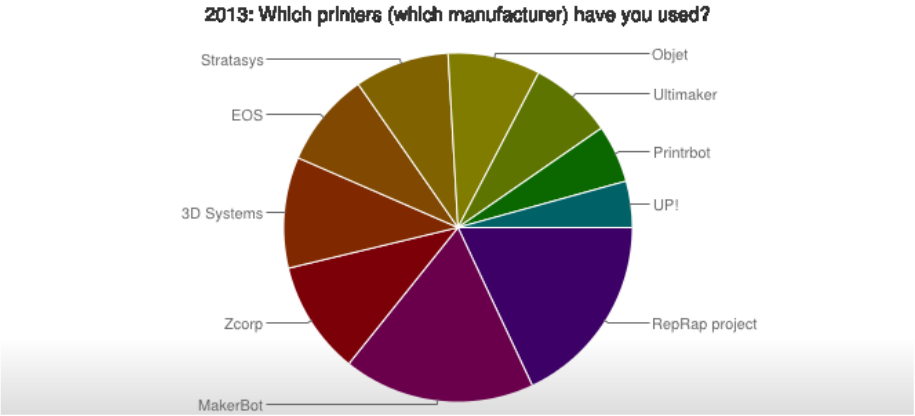


Figure 2.2: Printers used by focus group [3]

A functional and cost fair 3D printer is the main goal of the actual industry where domestic and commercial use is becoming the major investment, thanks to the RepRap project. Until a short time ago the industrial sector dominated the mass production and technological advances, but new trends and demands brought smaller printer dimensions with reduced build volume and energy consumption. This is what last decades research is bringing to the community and local stores, small domestic printers with huge printing capabilities and set up options allowing prototyping any possible 3D virtual model.

The major companies leading this technology are MakerBot, Ultimaker, Stratasys, ZCorp, Cubify, PP3DP, 3DSystems and Sculpteo (figure 2.3), having their own segment of printers and distinct services for desktop and industrial usage and even web applets providing online

Computer-Aided Manufacturing (CAM) and printing. These services make up part of a gradual research and development investment centralizing technology to build customer loyalty and improve user 3D experience.

[1]Source: `http://www.3ders.org/pricecompare/3dprinters/`

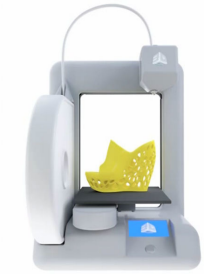
(a) RepRap

(b) Makerbot

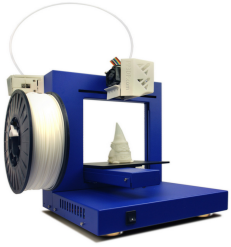
(c) Ultimaker

(d) Stratasys

(e) ZCorp



(f) Cubify



(g) PP3DP

(h) 3D Systems

Figure 2.3: 3D printers from major companies worlwide [1]

2.2 Technology

The printing process consists in obtaining a physical object from a virtual model (shape) represented by a polygonal mesh whose complexity is affected by the number of vertices and faces. An increase in representation level-of-detail means an increase in the number of polygons of the polygonal mesh(figure 2.4).

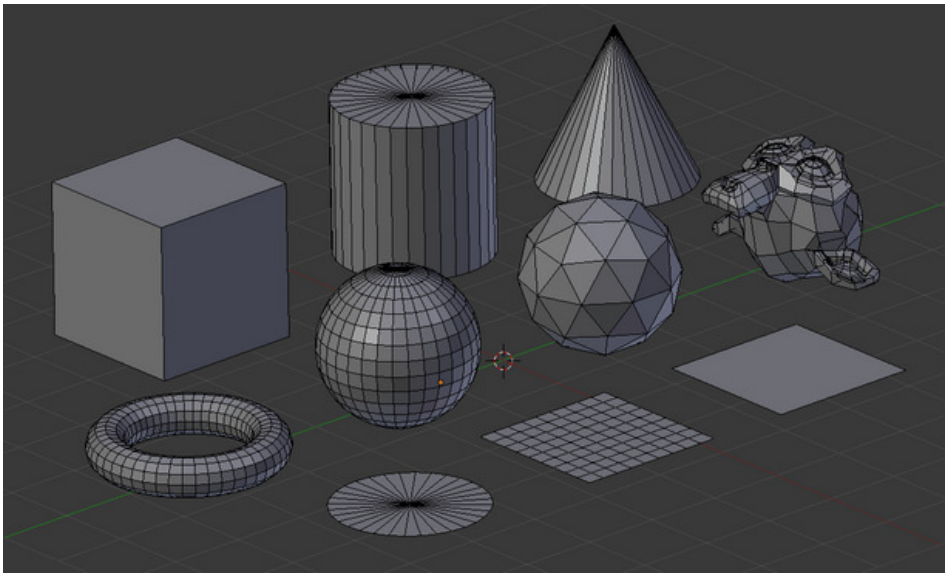


Figure 2.4: Polygonal mesh models [4]

The modeling of those meshes can be done using CAD software for further manufacturing with CAM applications, that are responsible to create detailed instructions for CNC printers. Recently some CAD software (AutoCad) embedded exclusive drivers and new user interfaces for 3D printers allowing printing directly without printer calibration and setup. This linkage between CAD and CAM software is achieved by a standard data file format (STL) that represents the surface of a virtual model as a set of small triangles (facets). Facets are important for the printing slicer with useful information about the paths to be scanned, the conditions for the transition between these paths and the planning of nozzle extrusion movements layer by layer.

When printing, the software reads the STL file and processes the data into machine commands in order to lay down successive layers of fused material from a series of cross sections. The quality of the printed model depends mainly on the material used to fill each layer (thermoplastics, metal alloys, plaster and photopolymer, see next section) and on the quality of the CAD model and the slicer efficiency.

This type of fabrication, also called rapid manufacturing, differs from traditional machining techniques, as we will see next, in the way the material is removed with the possibility to be an additive or subtractive process. 3D printing or additive manufacturing is faster, more flexible and less expensive when producing fewer and small parts.

2.2.1 Printing processes

Since the late 1970s, several different printing processes have appeared which differ in the way layers are deposited to create parts and in the materials that can be used. There are several technologies [1]:

- Selective laser melting (SLM)
- Direct metal sintering (DMLS)
- Selective layer sintering (SLS)
- Fused deposition modeling (FDM) or Fused filament fabrication (FFF)
- Stereolithography (SLA)
- Laminated object manufacturing (LOM)

Each of these processes uses raw materials with different composition and physical properties. Thermoplastics like Acrylonitrile Butadine Styrene (ABS) and Polylactic Acid (PLA) are used for FDM technology, metal alloy in DMLS and SLM, paper, metal and plastic film in LOM and photopolymer for SLA.

Table 2.1 presents these printing processes and their materials.

Printing process	**Material**
SLM	Most metal alloys
DMLS	Most metal alloys
SLS	Thermoplastics, metal powders, ceramic powders
FFF	Thermoplastics, eutectic metals, plasticine, silicone, porcelain, etc
SLA	Photopolymer
LOM	Photopolymer

Table 2.1: Printing processes and materials

ABS and PLA thermoplastics have distinct physical properties that determine their use (e.g, temperature) and the conjugation of printer hardware components with model physical properties determine the optimal cost value for printed pieces. ABS is a light, durable and strong thermoplastic being perfect for tools and utensils with the possibility of being used for hot liquid containers. This plastic has a high point of fusion and it must be extruded at 260 degrees Celsius and printed on a heated print bed [5]. PLA thermoplastic, Polylactic Acid, is derived from renewable sources as sugar cane, tapioca and corn. It can be extruded at lower temperatures than ABS, due to its low glass transition temperatures, but for excessive ones it is a disadvantage. PLA has better cost-effective benefits for 3D printing and it is more ecological since it is biodegradable and recyclable but it is less stronger and durable than ABS [5].

2.2.2 3D rendering

CAD applications allow viewing a 3D model from any angle with the push of a button and to zoom in or out, for close-ups and long-distance views. In addition the computer keeps

track of model design dependencies so that when a functionality feature changes, all other that depend on it are automatically affected accordingly. These models are represented in a strict and well-formed defined language or data structure that contains geometry, viewpoint, texture, lightning and shading information, as a virtual description of a scene. Let's consider a hand, a common shape, but also a very complex one due to rounded edges and concave parts, as an example how the render engine works (figure 2.5).

(a) 3D hand shape

(b) 3D hand with mesh structure (wireframe)

Figure 2.5: Mesh structure [6]

Each 3D model is structurally defined by a collection of vertices, edges and faces that defines the shape of any polyhedral object. This result is a visual representation of a 3D model with vertices connected together — using straight lines — where the vertices of each triangle are indexed and contain additional information such as rendering properties (figure 2.6b).

This kind of representation cannot contain information of the internal characteristics of an object since it is only concerned with the spatial distribution of the vertices of the object, and not the mapping characteristics as texture, lightining and others [6] — also makes difficult the interpretation of the model by the slicer algorithm (see later) which converts the virtual model into a series of printing commands to be processed by the printer. Figure 2.6 shows how the rendering properties of an object like this are represented over a mesh structure.

(a) Face model with texture

(b) Face model with mesh structure

Figure 2.6: Rendering properties over a mesh structure [6]

Using these properties in a data structure implies a file format capable of being transposed to a CAM application and to be understood in terms of syntax. In the following section some features of STereoLithography (STL), Additive Manufacturing File Format (AMF) and other file formats will be described.

2.3 File formats

2.3.1 STereoLithography (STL)

Design applications are endowed with tremendous capabilities. Compatibility between CAD and 3D printing CAM platforms is possible using a file format like STL. STL, STereoLithography and Standard Tessellation Language, is a file format native to Stereolithography CAD software created by 3D Systems [7]. It is used for rapid prototyping and computer-aided manufacturing (CAM) defining the surface geometry of a 3D model without any representation of color, texture or other attributes. Mainly it has a description of raw unstructured triangles, by the unit normal and vertices coordinates in a Cartesian coordinate system [7]. A STL file can be defined with ASCII or Binary encoding, but the last one is more common, safe and practical since it is a more compact file representation.

Figure 2.7: 3D printing chain

Each model is defined by polygons described by their vertices and unit normal vector. The normal vector and each vertex are described by floating-point values in sign-mantissa 'e'-sign-exponent format [7]:

```
facet normal ni nj nk
    outer loop
        vertex v1x v1y v1z
        vertex v2x v2y v2z
        vertex v3x v3y v3z
    endloop
endfacet
```

There is no restriction to the minimum number of vertices defining the outer loops of a polygon but for CAD/CAM applications, facets are usually considered as simple triangles.

Lets take for example this STL content from a simple 3D model:

```
solid Default
  facet normal -5.572511e-16 -8.419192e-15 1.000000e+00
    outer loop
      vertex 9.617157e-01 5.867651e+00 1.500001e+00
      vertex 9.617348e-01 3.057211e+00 1.500001e+00
      vertex 9.617271e-01 4.394419e+00 1.500001e+00
    endloop
```

```
  endfacet
  facet normal 1.071325e-01 -9.940594e-01 1.919783e-02
    outer loop
      vertex 5.546177e+00 -2.562629e+01 0.000000e+00
      vertex 5.772125e+00 -2.559646e+01 0.000000e+00
      vertex 5.772934e+00 -2.559599e+01 2.253218e-01
    endloop
  endfacet
  facet normal 2.221350e-01 8.954742e-01 3.857228e-01
    outer loop
      vertex 7.361565e+00 4.819891e+00 1.658161e+00
      vertex 7.361588e+00 4.822271e+00 1.659211e+00
      vertex 7.361565e+00 4.889780e+00 1.500001e+00
    endloop
  endfacet
endsolid Default
```

STL files can be represented with binary data mainly because file size can grow very quickly in more complex models. Such model data has basically a sub-divided bytes section containing a header, a variable indicating the number of triangular facets, data describing each triangle and a final attribute byte count variable [7].

```
UINT8[80] - Header
UINT32 - Number of triangles

foreach triangle
REAL32[3] - Normal vector
REAL32[3] - Vertex 1
REAL32[3] - Vertex 2
REAL32[3] - Vertex 3
UINT16 - Attribute byte count
end
```

STL isn't the only file format but it is the most used and reliable when comparing common formats used by 3D printing companies — other are the OBJ and DAE formats. The STL file format has some limitations as we have seen before and this led to the development of a new format capable of color representation, among other features.

Independently from the input file format, the model data structure is exported in machine code — GCode — to be correctly interpreted by the printer hardware executing those commands over the printer tools.

2.3.2 Additive Manufacturing File Format (AMF)

Additive manufacturing processes read data from input files describing objects and essentially their structural definition. AMF is a recent release that boosts STL capabilities. It stands for AMF and is an open standard with native support for color, material, lattice and constellations attributes [8].

It is an XML file for universal compatibility with CAM applications, to help design and render 3D objects using a basic file structure with five top level elements:

- Object - Allows the definition of a volume or volumes of material with associated IDs for printing
- Material - Defines one or more materials for printing with an associated material ID
- Texture - Defines one or more texture images with an associated ID. The texture can be specified for the volume or for each triangle in the mesh
- Constellation - Combines objects and other constellations in a relative printing pattern
- Metadata - Defines additional information about the objects and elements

The necessity of better compatibility, and taking advantage of the slicing algorithms and code optimizations on the data mesh, were two of the reasons for the creation of this format. Each model contains a single mesh, vertices list and one or more volume elements. *Vertices* is a list of all the object vertices with the assigned vertex information disposed in the order of declaration and containing the coordinates of each vertex in 3D space. Finally each volume is characterized by triangles that map the surface of the volume into a list of 3 vertices from the set of vertices.

```
<?xml version="1.0" encoding="UTF-8"?>
<amf unit="millimeter">
  <object id="0">
    <mesh>
      <vertices>
        <vertex>
          <coordinates>
            <x>0</x>
            <y>1.32</y>
            <z>3.715</z>
          </coordinates>
        </vertex>
        ...
      </vertices>
      <volume>
        <triangle>
          <v1>0</v1>
          <v2>1</v2>
          <v3>3</v3>
        </triangle>
        ...
      </volume>
    </mesh>
  </object>
</amf
```

Each triangle is defined by its vertices according to the right-hand rule — vertices are listed in counter-clockwise order as viewed from the outside. To improve accuracy, AMF format allows curving the triangle patches reducing the number of mesh elements that otherwise would be necessary to describe a curved surface.

```
<vertices>
  <vertex>
    <coordinates >
      ...
    </coordinates >
    <normal>
      <nx>0</nx>
      <ny>0.707</ny>
      <nz>0.707</nz>
    </normal>
  </vertex>
  ...
  <edge>
    <v1>0</v1>
    <dx1>0.577</dx1>
    <dy1>0.577</dy1>
    <dz1>0.577</dz1>
    <v2>1</v2>
    <dx2>0.707</dx2>
    <dy2>0</dy2>
    <dz2>0.707</dz2>
  </edge>
  ...
</vertices>
```

Besides model geometry, these files can represent texture, material and color attributes. Color is a (R,G,B,A) attribute element that can be inserted or associated with the volume, object, material, vertex and triangle, as the snippet below shows.

```
<volume materialid="1">
  <color>
    <r>0.9</r>
    <g>0.9</g>
    <b>0.2</b>
    <a>0.8</a>
  </color>
  ...
```

Assigning a color or material to a surface or volume is possible using texture maps representing the data in a 2D or 3D array.

```
<triangle>
  <v1>0</v1>
```

```
  <v2>1</v2>
  <v3>3</v3>
  <color>
    <r>tex(1,x,y,z)</r>
    <g>tex(2,x,y,z)</g>
    <b>tex(3,x,y,z)</b>
    <a>0.1</a>
  </color>
</triangle>
```

2.3.3 3D Manufacturing Format (3MF)

The AMF file format was conceived to cover the existent weaknesses in the STL format, but due to proprietary market growth, 3MF format was released. 3MF is a Microsoft file format released to meet the goals for 3D printing support in Windows. It is an XML-based data format that includes definitions and third party extensibility for custom data related to 3D manufacturing in Windows operating systems, becoming possible to integrate printing support [9]. With a similar behavior as a driver of the operating system, where from a list of available 3D printers one is selected, the 3D model is converted to 3MF format and encapsulated in an OpenXPS document package.

In the printer, data is unpackaged, converted and queued to the print agent.

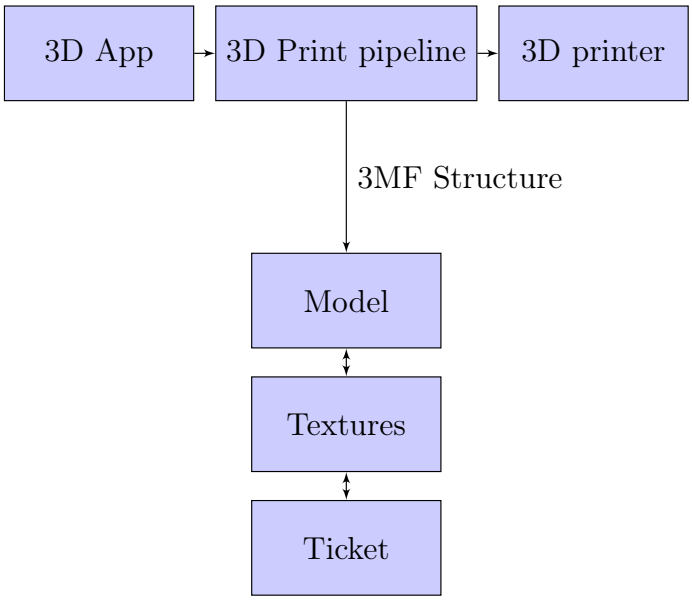


Figure 2.8: Microsoft 3MF structure

2.3.4 GCode

GCode is a numerical control (NC) programming language and is quite a simple language queuing actions to the printer [10]. The communication established defines where the printer (print bed and the nozzle) must move, by what cross section and how fast, responding to a

well-defined command syntax.

Briefly, from 1970s to 2010s, CNC machine builders faced compatibility difficulties because extensions and variations were added independently, by machine tool manufacturers, and understanding these disparities was rarely possible [11]. In 2010 GCode syntax was defective and limited without any kind of constructs, variables with natural-word names and conditional operators, but latest syntax brings an improved and rich language that allows machine tool operations including:

- Quick movements
- Controlled feed movement in a straight line or arc
- Controlled feed movements for highly sensitive turns and filament fusion accuracy
- Switching coordinates system
- Setting machine tool information

Gcode has an alphabet where each variable is a letter (from A to Z) with an associated interface description defining a list of arguments and a carriage return value. For example, some representative codes are:

- 'F' defines the feed rate, i.e., the velocity of the tool
- 'G' defines a motion movement
- 'X' represents absolute or incremental position on the X axis
- 'Y' represents absolute or incremental position on the Y axis
- 'Z' represents absolute or incremental position on the Z axis
- 'G1 F1000 X10Y10Z10' performs a rapid positioning motion by an incremental value of 10 units (usually inches or millimeters) in each of the represented axes with a speed of 1000 units (commonly, inches per minute and millimeters per minute)
- 'M' code represents requests to obtain tool values, either real-time or established previously, as temperature or home position coordinates

Let's take for example this GCode produced in the Slic3r application (2.5.3):

```
M109 S200
G90
G92 E0
M82
G1 F1800.000 E-1.00000
G92 E0
G1 Z0.500 F7800.000
G1 X74.505 Y74.418
G1 F1800.000 E1.00000
G1 X75.495 Y73.428 F540.000 E1.08841
G1 X76.325 Y72.668 E1.15947
G1 X76.885 Y72.208 E1.20523
```

This code snippet has three series of singular commands —- block command, configuration parameter commands and a motion command. In detail, M109 heats the machine nozzle until temperature is about 200 degrees Celsius, G90, G92 and M82 commands affect printing variables by telling the firmware to use absolute coordinates and absolute distances for extrusion and G1 motion command to move the tools (nozzle and print bed) to the specified singular point represented by the X, Y and Z absolute position.

With proprietary solutions, at a hardware level or even a software level, technology advances are promoting research and development departments to optimize machine code and create specialized code.

2.4 Printer functionalities

The current market is segmented into various types of printers, with different features, from office to industrial solutions. This implies technologies that allow large printing volumes, extrusion of multiple colors, fast algorithms for virtual model interpretation, automatic calibration, etc. The integration of these features requires a high cost of research and development but all applications have common functionalities that are extremely useful for 3D printing: raft, support, resolution and print density.

It is possible to design small and large-scale objects that, depending on model's complexity, can be glued or fit together with different colors. This exploits several opportunities, allowing a lot of operations for better resolution, density and reduced time cost. The user can manipulate the resolution and density values and also select raft (figure 2.11) and support (figure 2.12) options for additional results.

Resolution represents the quality of the printed object, where slower thermoplastic extrusion, in less quantity and with controlled speed movements, provides better results (better layer deposition). Nozzle diameter and layer height, affect the resolution inasmuch a bigger nozzle allows printing a slightly higher layer height but it doesn't really constrain that much when we want to decrease it. A bigger nozzle can't extrude material with a larger height than its diameter because layer fusion will be poor (figure 2.9).

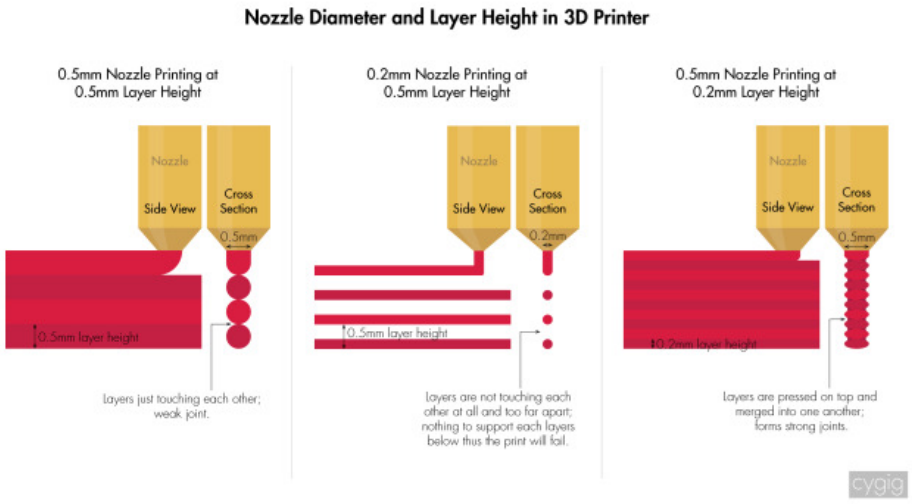


Figure 2.9: Nozzle diameter and PLA deposition [2]

Figure 2.9 shows layers may get leaned or flattened depending if we have equal layer and nozzle dimensions or if layer height is too small. In this case the next layer presses the previous one forming strong joints. Indeed, a smaller nozzle gives finer details in the horizontal planes because the fused plastic is deposited as a pancake around the nozzle output. The lower the layer height, the greater the "resolution" of the print. A 0.5 mm layer height will display visible layers, while with a 0.2 mm layer height, layers will look smoother.

Density represents the fill internal factor of each model making it less or more solid. This variable and the resolution are two major concerns of all 3D printing companies using Fused Filament fabrication (FFF), because algorithms have to handle the height and width of each layer, the resolution and the amount of inner material. The possibility of applying multi-color filament in multi-color print sessions is still a funciontality unavailable and impossible for many manufacturers (figure 2.10). It implies hardware equipped with a multi-extrusion head and an enhanced slicer, as well as a data file format to reproduce the color scale.

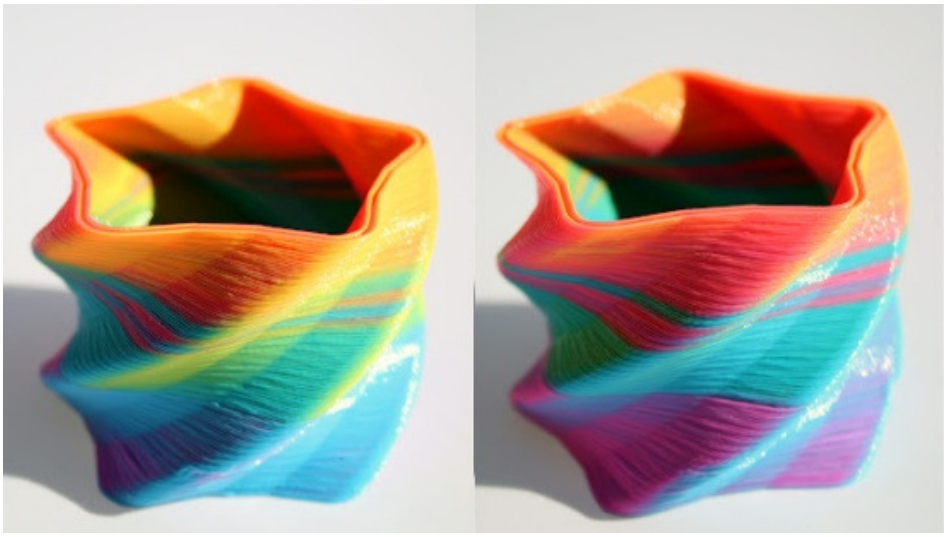
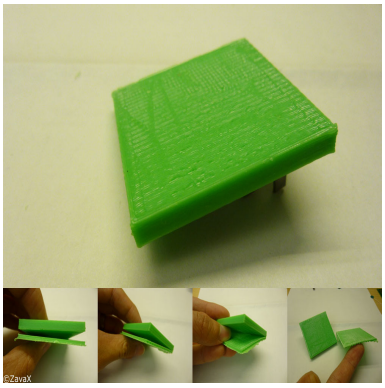


Figure 2.10: Multi-color print [3]

Raft is a feature that improves adhesion to the print bed when this is defective or when pieces are too big and start lifting at the edges (warping). Parts are built over disposable material (bigger than the part) instead of directly onto the build surface (figure 2.11).

[2]Source: `http://www.cygig.com/2013/08/review-portabee-3d-printer-most.html`
[3]Source: `http://richrap.blogspot.pt/2012/08/3-way-quick-fit-extruder-and-colour.html`

(a) Visible raft at printed model



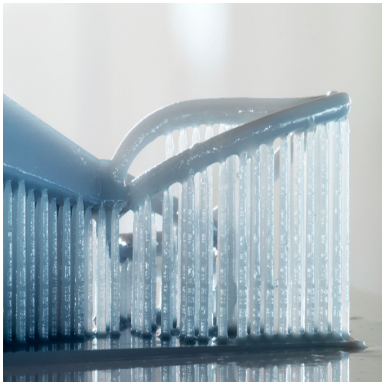
(b) Raft removal process

Figure 2.11: Raft results [4]

Support is also a funcionality that attempts to minimize structural issues in the model and, therefore, extra material is appended to the original model to ensure the printing of suspended sections. Otherwise those wouldn't hang (figure 2.12).

This represents the necessity of distinct materials in a single model: model material and support material. Model material constitutes the main model itself and support material is used as sacrificial material to provide structural support under model overhangs and in hollow areas. Once the printing is finished, support material is removed and becomes leftovers.

(a) Localized support From Web



(b) Long way support

Figure 2.12: Support results [5]

The printing setup process encompasses the configuration of these parameters so that the slicing engine parses each layer correctly in order to achieve structural safety and to easily to remove object.

As stated before, a model interpretation algorithm queues jobs to the printer which are parsed into mechanical commands controled by the hardware. The interaction between the

[4]Source: 2.11a) `http://store.3dprintingsystems.com/3d-printed-sample` 2.11b) `http://zavax.wordpress.com/2013/04/24/how-to-3d-print-peel-able-supportive-raft-base-platform-with-slic3r/`

[5]Source: 2.12a) `http://3dprintingsystems.com/products/up-plus-3d-printer/` 2.12b) `http://www.kickstarter.com/projects/formlabs/form-1-an-affordable-professional-3d-printer`

software and hardware controllers demands a strong and reliable architecture, developed and optimized for 3D printing.

2.5 Controllers

Current 3D printing systems can be considered as made up of independent subsystems that communicate via the hardware controller and software driver. In order for the user to use the system as a whole, it is important that the software subsystem can properly define the communication interface to the device driver level. Through a graphical interface, each user is allowed to perform CAM operations and printer maintenance assistants communicating with the 3D printer.

The software funcionalities include a 3D environment rendering agent that processes the virtual model in a set of motion and extrusion commands, allowing the printer to understand and print the model. This agent is located in the hardware subsystem within a kernel and execution environment code that listens to the communication incoming endpoints parsing the commands messages into mechanical commands.

These subsystems are in constant communication so that the user performs the printing process through the software manipulation and control interfaces and the execution on the hardware side.

2.5.1 Hardware Controller

An electronic board or chip controls the processing of the commands and the communication protocol management. The standards for communication depend on the technological advances or in proprietary implementations designed for a proper printer. Most common protocols used are USB, Serial (RS-232) and SDcard.

R2C2 (figure 2.13) is an electronic board that works for RepRap 3D printers like Mendal, Prusa and Huxley offering distinctive characteristics, such as heated bed support, micro SDcard for configurations storage, buzzer for alerts, plug and print, expansion header and a high speed 32 bits ARM microcontroller with fast USB controller [12]. It can also control any other 3D printer, laser or milling, representing a tremendous versatility for a wide range of final purposes.

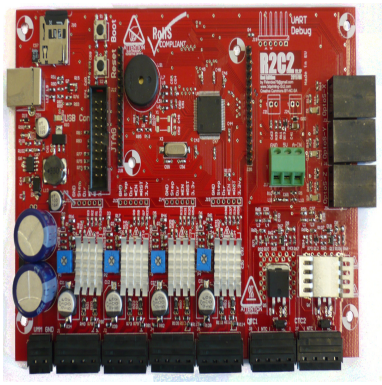


Figure 2.13: R2C2 board [6]

R2C2 characteristics show how important a board in a 3D printing system is, and the software on it — bootloader and firmware. Bootloader is a program that loads the main operating system or runtime environment in order to load a more complex kernel [13]. Generally speaking, the bootloader has to push the kernel into memory, provide it with the correct information to work properly and effectively support the switch to a new-loaded environment, dealing with transferring the control to the kernel. The newly-loaded environment is called firmware. Firmware is the combination of persistent memory and program code with data stored in it [14]. Basically this low-level software allows the control of the device and may not be changed in device economic lifetime but when it happens, bug fix and new features are common reasons for the update.

USB and Serial are two current protocols used for command processing but each of them describes an architecture that considers different IEEE standard interfaces. USB is a high speed upward extensible fully standardized interface between one computing device using a single port and N peripherals using one port each. All control is accomplished by signals within the data stream. It is difficult to find low level interfaces but those that exist (available in Serial) — called Simple interfaces — hide a very large degree of complexity.

RS-232 was developed to be one-to-one relatively low-speed semi-standardized interface between one computing device and one peripheral per port with the hardware control as an integral part of operation.

2.5.2 Software Controller

Similar to hardware controllers, 3D printing software is prototyped and built specifically for the printer. It is responsible for establishing and maintaining communication and managing main phases: modeling, calibration and print updating, as well as the printer configuration values and operational status.

At the beginning, 3D printing community projects were very active, with online RepRap open source communities as one of the largest, with many others appearing later. But even being easier and cheaper to maintain community code and functionalities, companies started to develop their own feature-bundling design, communication and slicing. A look through existing software shows these differences, where some are bundled with features such as the 3D and slicing engine. For example, ReplicatorG is 3D printing software (figure 2.14) that has been widely used thanks to existing multi-drivers pack for compatibility with Makerbot's Replicator, Thing-O-Matic, CupCake CNC and RepRap [15].

Let us consider ReplicatorG as the reference software for interface and functionalities analysis, since it assembles standard and advanced options, also including a wide list of drivers for cross-printer controllers (figure 2.15).

The main window displays 6 main panels:

- Bar buttons for printer interaction — the user can connect and disconnect the printer, build the model, reset configurations or even access the advanced control panel
- Machine state bar — indicates the current state of the printer and which driver is selected
- Model state bar — indicates what model is loaded and the visualization panorama selected

[6]Source: `http://www.3dprinting-r2c2.com/?q=book/export/html/1`

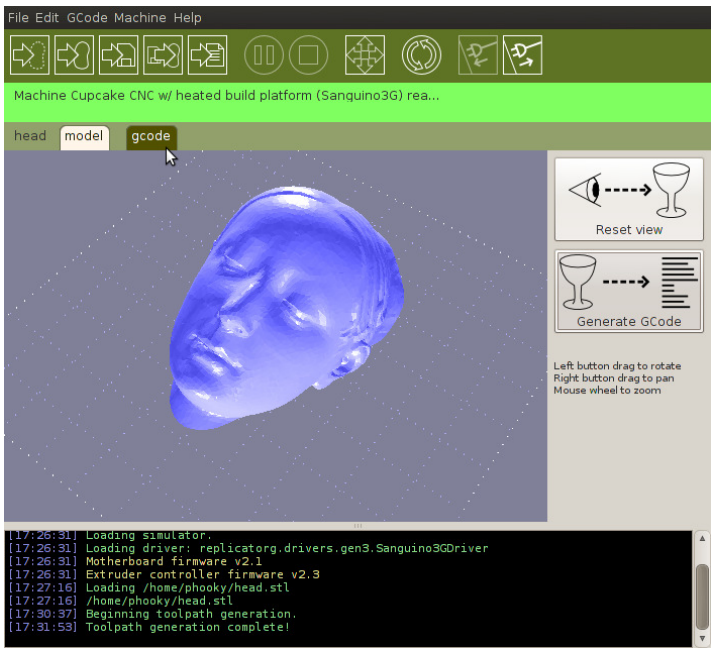


Figure 2.14: ReplicatorG software [15]

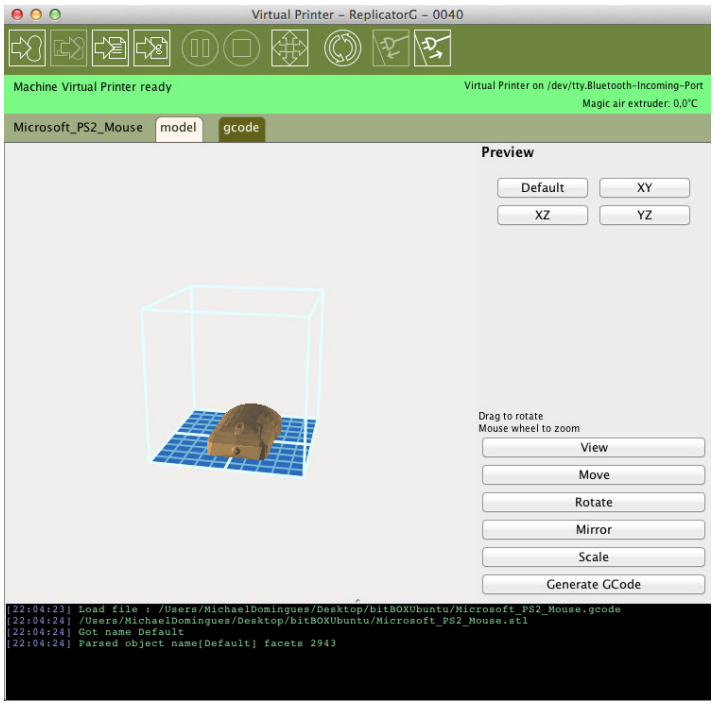


Figure 2.15: ReplicatorG software with enabled state

- Universe canvas — 3D modeling environment displaying the model and the building envelope size
- Modeling action buttons — operations to Move, Rotate, Mirror and Scale the model
- Console — output informations panel for enhanced actions feedback

Figure 2.15 allows us to see, on this screen, the user does mainly all the important and most frequent actions for the printing process. The application menu buttons support several other entries with menus necessary to correctly configure the printer (figure 2.16).

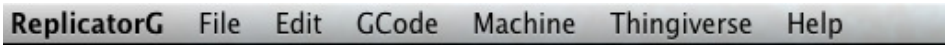


Figure 2.16: Application menu bar

- File — load and save model operations
- Edit — text edit and undo/redo operations
- GCode — slicer interface to estimate, simulate, generate and build the canvas with the possibility to pause or stop a printing session or even edit the profiles used to print
- Machine — driver and printer selection and also access to the printer information, advanced control panel and firmware update
- Thingiverse — access to online gallery
- Help — general help options

The control panel present in the *Machine* menu entry allows fine tuning, while the printer is operating, adjusting x, y and z axis movements, feedrate, motor speed and temperature (figure 2.17).

ReplicatorG's major funcionalities, including the printing itself, are gathered on the screen of figure 2.17. Building a model doesn't require other windows and all the important information regarding printing variables is displayed in the screen above in the specific status bars (figure 2.15). The possibility of editing a slicing print profile is a major funcionality, not very frequent among 3D printing software, because it requires technical knowledge to set the correct input variables for an optimal slicing as we will understand on the next section.

ReplicatorG is a simple, maintenance wizard free, pratical and multi-printers software with advanced funcionalities for advanced users that allows each user to control the printer in a small set of screens.

2.5.3 Slicer

A 3D printer doesn't process directly files from a CAD program, those files need to be processed before they become printable. This process is called slicing — the conversion of a 3D mesh into a set of toolpaths for the extruder, in the form of GCode. A slicer commonly uses STL files to create GCode files which contain instructions for the 3D printer on where, when, and how fast the movements will be executed. However, GCode programming is not suitable for CAD and 3D design. This is where the slicer and the STL file comes into use.

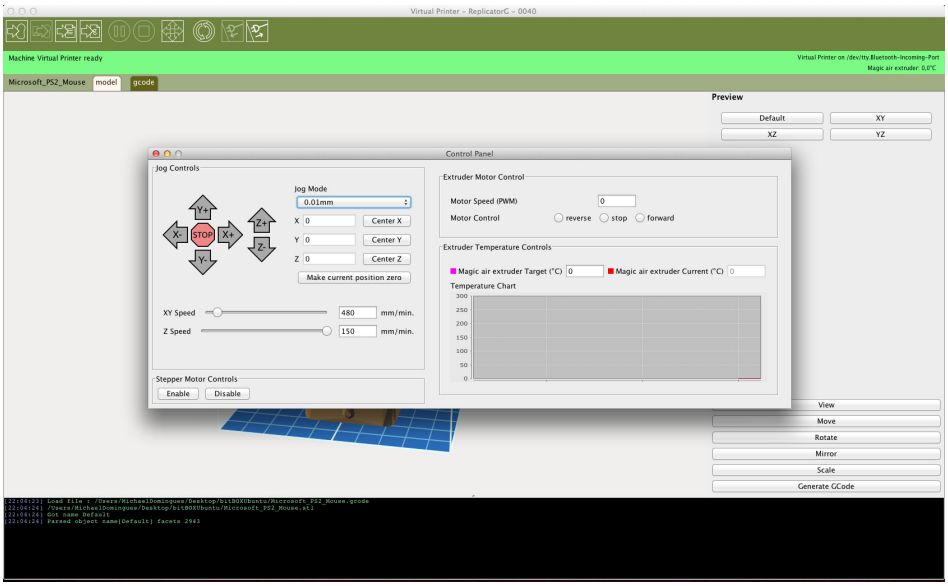


Figure 2.17: ReplicatorG advanced control panel

The STL file is a 3D model file that can be exported by all common CAD and 3D modeling software. The slicer then slices the STL 3D model into layers and print paths to create a 3D printable object defined in a GCode file format.

A slicer takes a 3D model and translates it into individual layers to generate the machine code the commands agent in firmware will parse. Generation time completion is important and the speed, layer infill, structural support and cross section courses optimization between layers are prominent factors that influence the algorithm (figure 2.18).

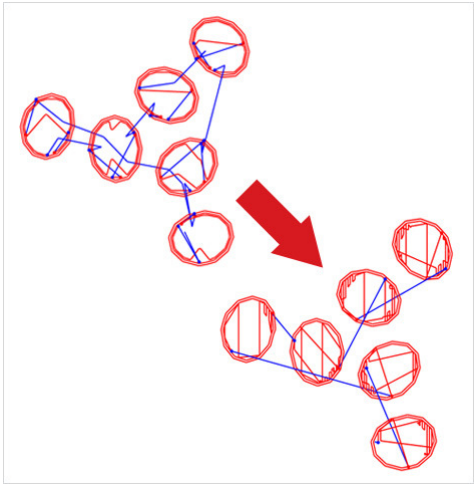


Figure 2.18: Makerbot simplified slicing algorithm [7]

The behavior of the movements during the print of each layer depends on the geometrical interpretation by the slicer algorithm as shown in the previous figure where the deposition

[7]Source: `http://raphaeljimmy469.wordpress.com`

of the material may be made on nearby spatial points if considered as so. This leads to an optimization of the algorithm with faster travel movements of the entire layer reducing the duration of the print and the amount of free movements, i.e., movements without extrusion that only reposition the axes for the next extrusion. This improvement has brought more efficiency in the printing of each layer and in the transition between layers minimizing the error or imperfections in the final object.

Some of these engines combine the functionality of slicer and interface/control software sending machine code to the 3D printer and, at run-time, control (move the print head manually around the x/y/z axis) and change some parameters (e.g. speed, flow and temperature).

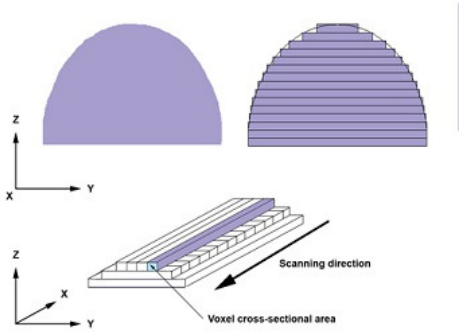


Figure 2.19: Rapid prototyping slicing [8]

Figure 2.19 illustrates a typical behavior for a rapid prototyping slicer, scanning from one voxel (volumetric pixel) cross-section area to another by x, y-axis and layer by layer (z-axis). This scanning system is mapped in a set of procedures which is entirely present on the slicing engine of Skeinforge:

- Slice Layer
- Preface Layer
- Inset Layer
- Fill Layer
- Raft Layer
- Jitter Layer
- Clip Layer
- Comb Layer
- Cool Layer

[8]Source: `http://commons.wikimedia.org/wiki/File:Rapid_prototyping_slicing.jpg`

In general, each layer is scanned and structured for the choosen values of the resolution, infill, raft (first layer only) and support with additional layer-over-layer intelligence. They adjust for example, where the z-height increments when completing a layer (Jitter), when to stop extruding to prevent loop plastic build up at the end of a loop (Clip) and the hold time or the print speed to jump to next layer when previous is still soft (Cool) [16].

Processing one of the models with Skeinforge (comparison slicer) to test the slicer (figure 2.20d), produces the following output with several optimizations and calculations made through each layer:

```
File 3DModels/Dificult - Carboy.STL is being chain exported.
Slice layer count 78...
Inset layer count 78...
Fill layer count 15 of 354...
Raft layer count 15 of 354...
Jitter layer count 15 of 354...
Clip layer count 15...
Comb layer count 15...
Cool layer count 15...
```

At the end, Skeinforge outputs the procedure results:

```
Preface procedure took 12 seconds.
Inset procedure took 12 seconds.
Fill procedure took 30 seconds.
Speed procedure took 1 second.
Temperature procedure took 0 seconds.
Raft procedure took 4 seconds.
Jitter procedure took 1 second.
Clip procedure took 12 seconds.
Comb procedure took 5 seconds.
Cool procedure took 3 seconds.
Dimension procedure took 2 seconds.
Bookend procedure took 0 seconds.
```

These procedures are called plugins and can vary between slicers, but the layer scanning and processing, have similar ways of working. Categorizing, we have pure slicers and pure control software. Pure slicers are the ones responsible only for slicing the model into layers and producing the respective machine code, as for example, Skeinforge, Cura, MiracleGlue and Sclic3r. Pure control software embeds the control feature referred previously and that way we have some engines as Repetier-Host and ReplicatorG. Combining both features we have Netfabb's software [17].

Let's see 4 slicers that have boosted the 3D printing industry and then, we will compare their efficiency on some models, each with distinct characteristics.

Cura

Cura is developed by Ultimaker and besides the compatibility with other 3D printers, thanks to being open-source, is considered to be easy and streamlined as possible. This led

today's companies to develop their own frameworks rather using open source solutions like Skeinforge, that is typically used by communities. The latest version of Cura includes major releases and updates due to new hardware specifications with code refactored, to support multi-head extruder and to slice even faster, reducing print sessions duration to minutes instead of hours. It has also a printing algorithm that, according to Ultimaker users, reduces printing time significantly [18]. Usually 3D printers cut across each space between the objects on each and every layer slowing printing process. CuraEngine made it time-cost effective printing each object in the 3D canvas in sequence and not separately.

As one of the fastest slicers on the market, Cura has a free open source command line integration that is used by some 3D printing open-sources projects such as *OctoPrint*. Unlike Ultimaker, Markerbot decided to close their slicer, the Miracle Glue.

Miracle Glue

This engine is an improved solution from Makerbot for their CAM software, MakerWare, leaving behind the old and tremendously used slicer — Skeinforge. It is faster and was built from scratch to allow printing bed optimizations and model corrections, with fewer retractions (stops and starts) and faster and consistent prints.

As open source slicers there are two slicers with command-line integration that have been used for several purposes on the 3D printing industry and nowadays are following opposite paths — Slic3r and Skeinforge.

Slic3r

Slic3r is an open source tool that enables the user to convert a 3D model in GCode and like Cura, it is compatible with a lot of 3D printers. It divides the model into horizontal layers and generates optimized code or in some situations, better than other algorithms. Movements are made layer-by-layer, traversing space between objects and the interface configuration parameters are easy to customize.

Cura slicer was initially based on a slicer commonly used by the 3D printing community called Skeinforge that was also the predecessor of Miracle Glue.

Skeinforge

Skeinforge is open source, similar to Slic3r but less efficient and is compatible with several 3D printers, being embedded in their software. It is one of the slicing engines used in Makerbot software and in ReplicatorG. The printing intelligence is similar to Slic3r with layer-by-layer movements but less effective and hard to customize.

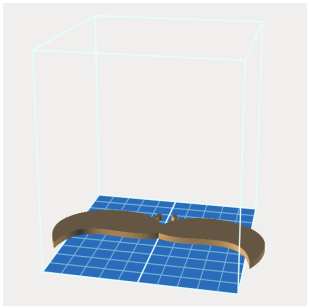
These four widely-used slicers have distinct characteristics and therefore it is relevant to compare their GCode generation speed efficiency over some models.

Benchmark Results

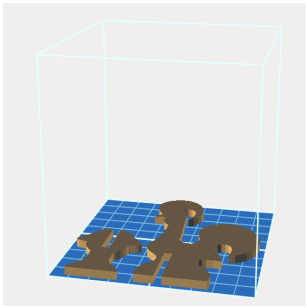
These four slicers are designated *pure slicers* and their work around presents different complexities, inputs, user interfaces and algorithms. Let's compare their efficiency with an input of five 3D models in STL format, with increasing complexity. The sample is a set of five models graded with printing difficulty based on three meaningful parameters: number of layers, number of facets and number of vertices (figure 2.20) (table 2.2). Flatter and less

curved models are easier to be printed but rather, curved models present a higher number of flat surfaces for each segment meaning more details.

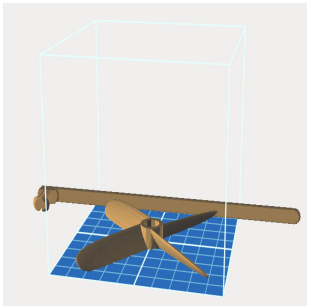
These five models are good examples to perform the slicer analysis (table 2.3).



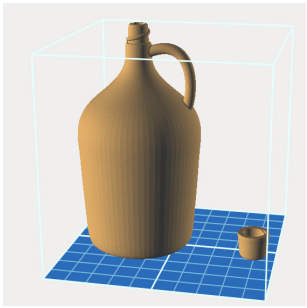
(a) Very Simple model



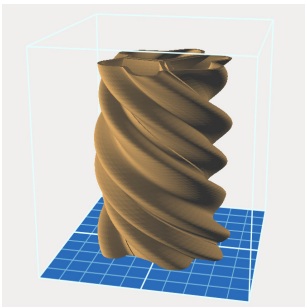
(b) Simple model



(c) Medium model



(d) Difficult model



(e) Very Difficult model

Figure 2.20: Models sample

	Layer count	**Triangles count**	**Vertex count**
Very simple model	15	21380	64140
Simple model	20	900	2700
Medium model	907	2816	8448
Difficult model	532	43520	130560
Very difficult model	1448	83306	249918

Table 2.2: Models complexity

Table 2.2 shows cleary the difference between complex models, such as the Difficult (figure 2.20d) and Very Difficult (figure 2.20e) and the other three. These two models are larger but even so, their curved shape with highly inclined angles result in more triangles and consequently more vertices to complete each flat plane. The next table (table 2.3) illustrates the performance results obtained when processing the five models, given the characteristics

present in table 2.2.

	Resolution	Cura	Miracle Glue Slice	Slic3r Time (seconds)	Skeinforge
	High	1,22	10		
Very simple model	Medium	0,64	10	12	27
	Low	0,36	7		
Simple model	High	1,14	18		
	Medium	0,76	9	20	54
	Low	0,38	6		
Medium model	High	7,08	43		
	Medium	4,05	12	27	136
	Low	1,94	12		
Difficult model	High	17,41	160		
	Medium	10,33	70	61	397
	Low	4,98	48		
Very difficult model	High	76,41	615		
	Medium	46,81	260	82	1169
	Low	15,41	235		

Table 2.3: Slicers generation time

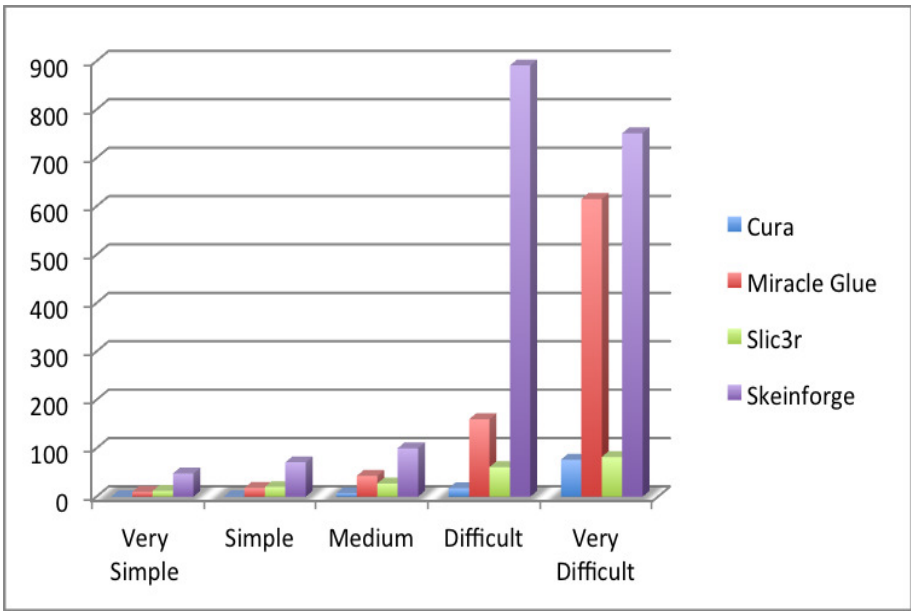


Figure 2.21: Benchmark results for high resolution

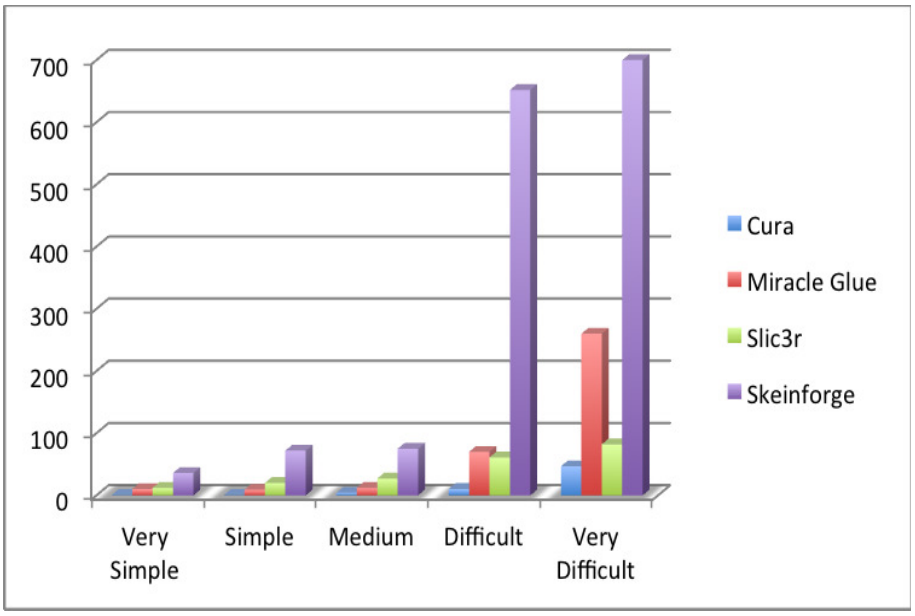


Figure 2.22: Benchmark results for medium resolution

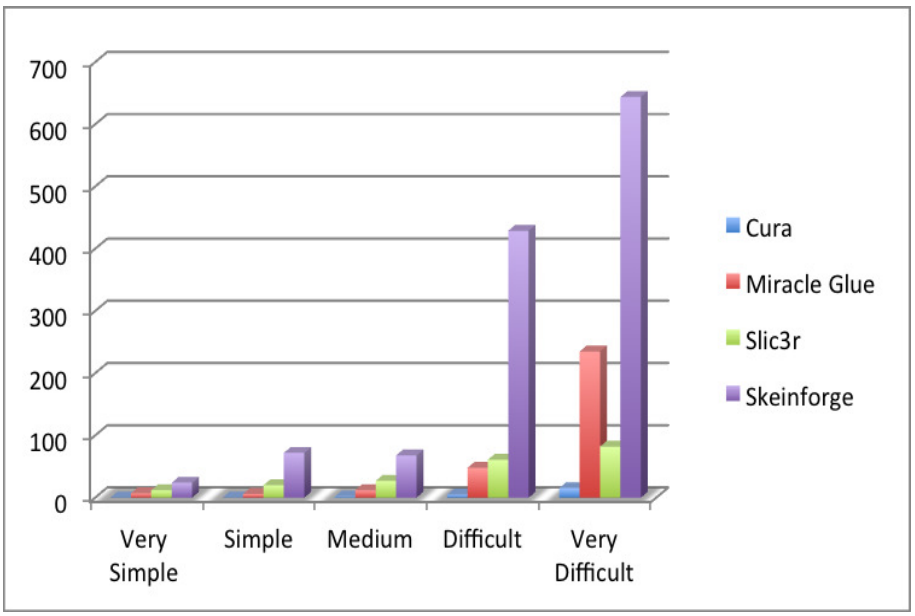


Figure 2.23: Benchmark results for low resolution

Graphically, the difference of performance between the four slicers is evident, in particular for models printed with high resolution profiles (figure 2.21). We need to note that Slic3r results are equal independent of the resolution because the native application doesn't support enough input parameters to simulate this analysis, just close approximations. Comparing figure 2.21 and figure 2.23 results, we see Cura has a powerful slicing algorithm with a generation time approximately 8 times less than Miracle Glue for the Very Difficult model.

The speed of the slicer algorithm is one of the 3D printing software modules companies

are trying to improve and to integrate over other technologies. The need for a 3D printing application that is reliable, effective, robust and user-friendly led to the emergence of various frameworks for integration with 3D printing software but still, with several issues.

2.6 Technology issues

In the past couple of years, 3D printing has grown with notorious improvements in the quality of the objects, but although models are looking better, there is no guarantee they are structurally safe. The delicate of the printing process and the model characteristics when handling, can damage the printed object. There is a study analysis that makes it possible to detect and correct problematic structural cases by combining three approaches: hollowing, thickening and strut insertion [19].

Printers are getting more affordable and lightweight but there is a cost — build volume. Today's companies compete over several segments (industrial and desktop solutions) for better consumables, ideal printer size and for the best fully functional printer with the lowest cost. Reducing the build volume decreases printer cost but also limits the dimensions of the 3D printed models when importing to the software. Unlike the previous issue there are already developments with a framework called Chopper that decomposes a large 3D object into smaller parts so that each part fits into the printing volume. Later these parts can be assembled to form the original object [20] [21]. This framework formulates criteria for this decomposition that include assemblability, unobtrusiveness of the seams and structural soundness.

Shape rendering along with operations of manipulation of the model worsen the software performance and the user interface complexity, requiring the maximum of virtual memory to support the render engine. This engine constitutes another issue of a 3D platform and depending on the programming language or the virtual machine characteristics, it can become a deficiency. Common software presents a 3D environment of proprietary or third party libraries with functionalities such as importing a model or scaling, rotating and moving, as heavier operations affect performance. Cumulatively, some software such as Cura from Ultimaker, performs instantaneous slicing for each transformation applied, happening very quickly thanks to an optimized library and frameworks. In the worst case, with a heavier model and successive operations, the rendering engine won't handle it.

We have seen problems regarding structurally unsafe STLs, oversized models and virtual memory limitations, but we must also consider the printing cable connection. Depending on computer performance and machine code build time, printing sessions may take some time requiring constant pyshical connection between the computer and the printer. This concern ceases if it is possible to transfer the GCode — called autonomous mode.

2.7 Conclusions

In this chapter we reviewed the main functionalities of 3D printing and the model fabrication process, with the import of a model and the transformation of the virtual object. We have also seen current technology used in a 3D printing CAM application to control the printer and to generate the necessary data to be processed and represented in a GCode file.

The data are a set of vertices and triangles that defines the shape of the polyhedral object (after the interpretation of the model loaded into the application) using a format widely used in 3D printing — STL. Importing the model creates a mesh represented in a 3D environment

on which it is possible to apply various transformations that produce a virtual object to be printed and whose rendering environment takes care of rendering those transformations into a 3D view.

We also saw that it was possible in some 3D printing applications to have access to printer maintenance tools to ensure the distance from the nozzle is calibrated as well as print bed level, on which the PLA is deposited layer by layer. The deposition movements at an ideal melting temperature depend on the nature of the planning algorithm called slicer. The various existing algorithms have different characteristics, some with proprietary enhancements to the company printer. Is important to take into account the optimization of the courses of printing, the ability to handle multiple extruders, the management of consumables, the transition between each layer to ensure the best quality possible and the ability to integrate funcionalities such as extra material in the model (raft and support), to ensure the fabrication process is capable of solving problems or minimize models issues and technology issues.

The manufacturing process as a whole is still an evolving process with ongoing issues such as multi-color printing, material cost, integration with services that provide the correction of defective models and the clarity, efficiency and robustness of the application, so that the user can easily use its funcionalities.

So far we have learned what 3D printing is, its specifications and deficiencies. New file formats, algorithms to correct objects represented in an STL file format, autonomous mode and multi-color extrusion heads, reveal the emphasis to build an enhanced closed system that tries to be simple, effective, powerful and error free. Next chapters will explain how we can develop a platform able to print a 3D object and what features it should have for those objectives.

Chapter 3

User Interface

The architecture of *BEESOFT* needs a user interface that enables data input and integration with important modules of any 3D printing application — communication and 3D engine. *BEESOFT* interface is comprised of screen menus, icons, shortcuts, mouse gesture movements, online help and many other elements to allow easily each user to configure and control the printer.

3.1 Introduction

The analysis previously made explained the existing debilities in 3D printing software and the capabilities within as the slicing algorithms and functional features.

Next, *BEESOFT* interface details will be explained, communication set up and 3D engine, that fulfil the minimum requirements based on the mockup, other CAM software and good practices regarding interface design and usability. This chapter will address several modules, such as the interface, communication protocol, maintenance wizards, print wizard and 3D engine module, in order to explain the implementation and some decisions, to achieve a usable, functional and stable computer-aided manufacturing platform for 3D printers.

3.2 User Interface

To improve the usability of an application, it is important to have a well-designed interface, from the mockup screens and with several cycles of implementation and testing to achieve a usable, error-free and robust interface.

3.2.1 Guidelines

The design of graphical user interfaces means building a visual design with the integration of graphical elements, able to support interaction between users and printers [22]. Users demand a software that is reliable, good-looking and easy-to-use, otherwise it will be considered bad. Therefore, there are patterns which describe the practices within a given design domain to understand what a user sees and what a user expects.

Creating something, conversing with other people, controlling or monitoring a printer, tells that an application must fulfil the requirementes to a varied set of users with different experiences. Every user is unique and so it is important to specify and become aware of [23]:

- What are the goals of each user using the software
- The tasks they undertake in the pursuit of those goals
- The language used to describe to others what they are doing
- What are the user skills using similar software
- User attitude towards what is designed and how different designs affect it

We need to understand patterns and their influence in design choices, to produce an interface that helps users to achieve goals. Most used patterns are based on user observations where the best cognitive way to define the interface implementation is to watch people doing things understanding how they think about what they do. The most important patterns, regarded as suitable for an application of 3D printing, are [23] [24]:

- Safe exploration: The freedom of exploring an interface without suffering direct consequences contributes in helping the user to learn more quickly. This way, an interface should provide exploration "paths" for users to experiment rollbacking easily and without consequences.
- Instant gratification: Instant gratification implies design division, because it is all about immediate success of the user experience. Each user must be capable of reaching the goal in fewer steps, with the interface supplying visual components to instruct him about important information regarding the current task, and how far he is from the end.
- Satisficing: The ease with which users understand the interface and move along it, represents a rational behavior called satisficing. Short labels, plain words that are quick to read, an expressive and communicative interface background and multiple paths with escape that hatches impose a low cognitive cost on new users.
- Changes in midstream: The start of a process, stopping in the middle and resuming it later is essential and users can't be constrained into a choice-poor environment. Each application needs to have a global navigation tree.
- Differed choices: It is not correct to overload user with too many and unecessary questions while he is trying to get something done. Each user only needs to care about the most important steps considering the other optional ones because it avoids too many upfront choices — instant gratification.
- Incremental construction: Achieving the goal is more likely by successive or incremental operations with possible rollbacks for adjustments. An interface needs to support this style of work, quick fixes during an action to help it finish correctly without undesirable and unpredictable errors.
- Habituation: Habituation improves efficiency, but it can also trap the user. Repetead actions can induce that they will work as intended everywhere else, but they may not. Consistency within an application is important as much as using controls where they are supposed to facilitate each user task and not used everywhere.

- Spatial memory: Positioning components, as action buttons, always on the same place independently of the screen is advisable to avoid spatial memory disruption. Rearranging buttons screen to screen doesn't help memorizing relative positions.

- Prospective memory: Despite being difficult to implement, there should exist a support for passive prospective remembering. Users may find it useful to note down some temporary information before proceeding with the current operation.

- Streamlined repetition: The easier a task can be, the better, reducing the number of clicks or the number of repetitions. Actions need to be adaptive to support repetitions with more intelligence, avoiding redundancies.

- Keyboard only: Input devices such as keyboards and mice represent important and fundamental peripherals and their use can't be restricted to just one. If the combination of both improves user experience than it is recommended the application supports both in an appropriate way.

- Other people advices: The advice from of peers influence, direct or indirect, peoples's choices being relevant to consider interface re-design if option A is preferred most of the time to option B. Not all applications can accomodate a social component but if we are considering having more than a few dozen users, it is better taking this seriously.

The next section will explain how these patterns were respected and how they contributed to the development of a mockup interface.

3.2.2 Mockup

To show the end user what the software will look like without underlying funcionalities, we can use simple screen layouts. These screens are used to acquire feedback from the users to test one part of the system, enabling the evaluation of the user interface without system dependencies, e.g., hardware for complex computation. They are essential to describe and understand the behavioral requirements to define how the screens structure and how they will behave between them. [25].

Taking into consideration the design and features of *ReplicatorG* software (2.14), described in section 2.5.2, use-case diagrams were developed to reflect the requirements for *BEESOFT*. This helped reducing goal-level confusion, becoming more intuitive to explain what was intended in each screen. These use case diagrams intend to detail features, from the basic to the more complex, that a CAM platform should have.

BEESOFT contains several screens (figure 3.1):

- Main window — Allows main access to the major actions, such as printer maintenance and print. It also contains the 3D environment and the model control panels (appendix A.1).

- Models operation center — Provides modeling operations in the 3D environment (appendix A.2).

- Scene details — Details important information about the scene under manipulation (appendix A.3).

- Filament wizard — Controls the load or unload of filament in the printer (appendix A.4).
- Nozzle wizard — Allows cleaning the nozzle before calibrating (appendix A.5).
- Calibration wizard — Allows calibrating the printer to ensure bed height and level for future print sessions (appendix A.6).
- Preferences — Allows editing software preferences such as language and modeling settings (appendix A.7).
- Print wizard — Controls print sessions using the input parameters for the session (appendix A.8).
- Buttons bar — Quick access to major and important features (appendix A.9).
- Help — Provides support for important help topics regarding troubleshooting and FAQ issues (appendix A.10).
- Update — Allows software updates from repository (appendix A.11).
- Gallery — Allows scene and model edition and file browsing to help import files to the main window (appendix A.12).

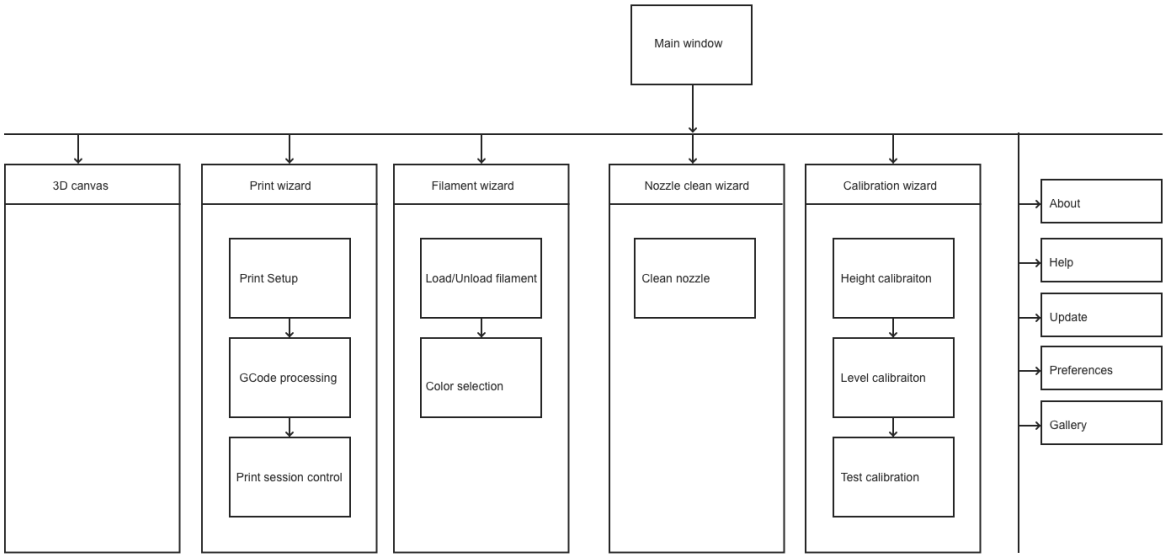


Figure 3.1: *BEESOFT* screens

Appendix A includes diagrams and respective mockup interfaces for the all the above items. First we will see what functionalities were considered in the main window and how they were set in order to obtain a usable, friendly and understandable interface.

Main window

Quick access to the main operations was taken in consideration in order to easily load a model, generate the code to be processed in *BEETHEFIRST* and print the scene (see figure A.2). The buttons position provide a good arrangement of the functionalities of each container in terms of modeling buttons, application buttons, main operation buttons, etc. A visible and easy to understand separation of these containers avoids extra elements such as text tips to explain their purpose (figure 3.2).

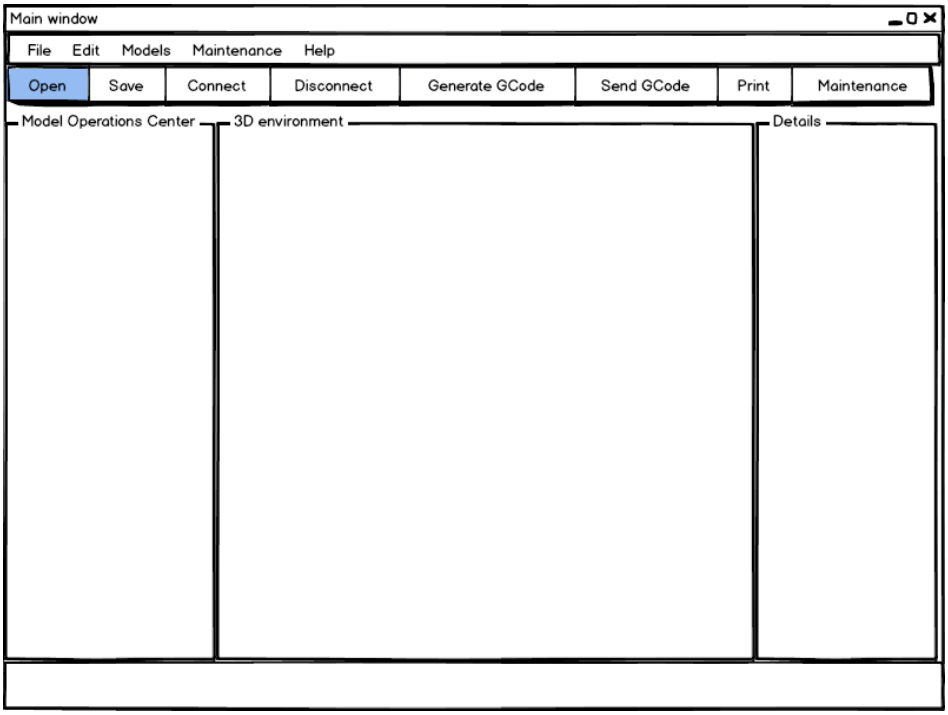


Figure 3.2: Main window screen mockup

On this screen, we also have the 3D environment at a central stage to be more appealing and representing one of the major modules of this platform. The tools (move, rotate, scale and mirror) and extra information were placed on both sides to allow simultaneously the control of the models (see use case diagram A.11) and the visualization of the information associated with the scene (see use case diagram A.5). It has also a file menu bar with important entries (see use case diagram B.21) and whose appeareance was considered as seen in figure 3.3.

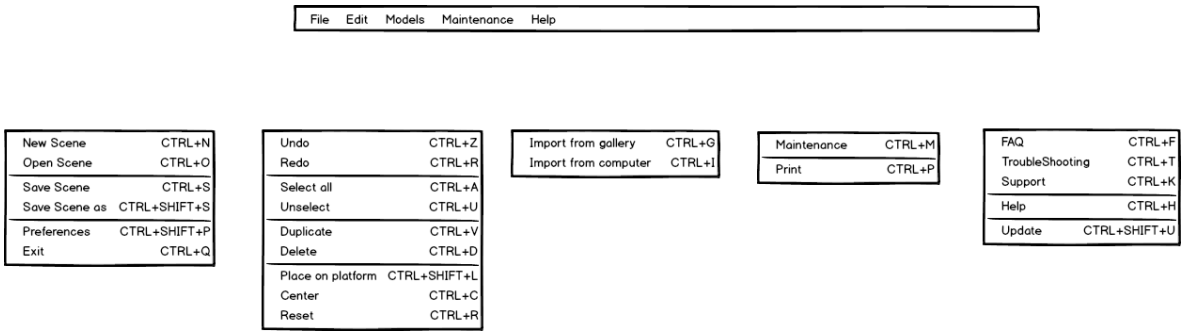


Figure 3.3: Buttons bar interface mockup

Another extremely important module for the print process is the maintenance operations module. No other available software has a maintenance center or even complete and well explained wizards to help the user configure his printer without several mechanical interactions.

ReplicatorG interface (figure 2.15 and figure 2.14), analysed previously (section 2.5.2), does not contain such functionalities. Ultimaker (figure 2.3c), PP3DP (figure 2.3g), Makerbot (figure 2.3b) and Cubify (figure 2.3f) software does not contain it either. Wizards guide users to achieve a goal through a controlled road map where complex and difficult tasks need to be splitted into a sequence of iterations [26]. Each user can deal with each task more easily than if they were all together because, with a wizard, per major task, operations are simpler as the information helping him, and are more focused on avoiding unnecessary errors.

Filament wizard

The filament wizard (figure A.8) allows the load or unload of the filament from the printer at an established controlled temperature to print with different color materials. This is a flow that is interruptible at any moment and is confined to two direct user interventions: load or unload and color selection (see use case diagram A.7). This process may cause the drying of small bits of remaining filament in the nozzle, which prevents correct calibration, if made afterwards. To avoid it, the user can clean the nozzle through the wizard available by a simple button click.

Nozzle clean wizard

The nozzle clean wizard (figure A.10) is available through a button, which aids the user to unblock or clean the nozzle chamber without having to perform harmful operations to *BEETHEFIRST*. Once again, user direct interventions are short and oriented (see use case diagram A.9). The last wizard that can't be skipped from a 3D CAM platform is the calibration or at least the calibration functionalities set for the proprietary printer.

Calibration wizard

The market analysis made previously (section 2.1.1) revealed several printers whose calibration is ensured by a calibration levelling interface different from this one (figure A.12). The calibration points depend on mechanical aspects to level the bed relative to the nozzle (flat and height operations are not made by all available software) and to keep it flat with physical screws to help the adjustment. As seen until now, on previous wizards the user did

not need to interact more than pressing buttons, and this assistant aims to be similar (see use case diagram A.11). The *BEETHEFIRST* requires three calibration points, and to enhance experience and user proximity to technical details, the wizard contains a test calibration to validate the set-up before concluding the wizard (see use case diagram A.11).

These three wizards represent the maintenance operations center available from the main window, ensuring important usability conditions:

- Simplicity — low number of clicks and quick wizard conclusion
- The possibility to cancel anytime
- Real-time feedback
- Actions rollback
- Wizard validation before concluding
- Static interface to improve spatial memory and habituation
- Safe exploration

The second relevant module to be explained contains the model operations center and scene details center (figure A.4 and figure A.6), available from the main window. As we have seen, all important and desirable functionalities are accessible from this screen and the easy access to these containers, their information and operations, allows the user to preprare printing with a single screen.

Models operation center

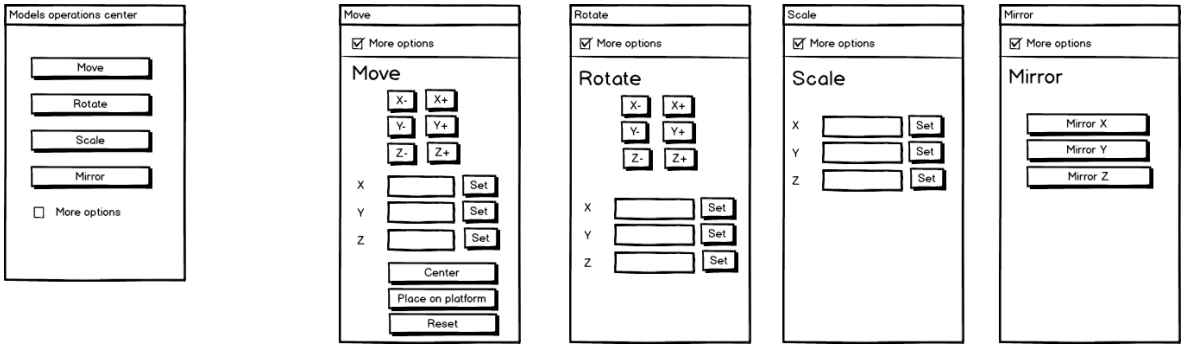


Figure 3.4: Models operation center interface mockup

The models operations center lists the four global modeling operations — Move, Rotate, Scale and Mirror — in all CAM software, with the possibility to apply static values in all of the 3 axis (positive and negative). Besides pre-defined values, the user must be able to enter values as illustrated in figure 3.4. In particular, the Mirror option doesn't have these kind of funcionalities, just the x, y and z axis mirror button. However this interface has a center, place on platform and reset position button which reduces the cost to prepare the print, enhancing user experience and reducing printing time — better gratification (see use case diagram A.3). The transformations in this container are applied on the 3D canvas with the scene details center showing information regarding the scene and the models, after each operation.

Scene details

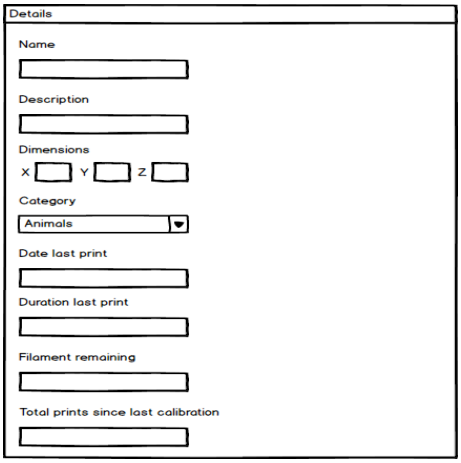


Figure 3.5: Scene details interface mockup

The scene details container (figure 3.5) allows setting several fields, such as the scene name and description, and to consult information such as the duration of the last print and filament remaining (see use case diagram A.5).

Both of the last two containers are always visible on the main window screen and can be accessible directly without other panels for processing and viewing data. Finally, the print wizard assembles all the work done, as explained previously.

Print wizard

This is a wizard each user expects to be quick and short since other ones that prepare the print are more costly. In these containers, the user selects the resolution, density and other print options (section 2.4) (see use case diagram A.15). These print choices and the complexity of the printing process justify extra screens for GCode generation and the control of the print session, to ensure they happen as expected (figure 3.6).

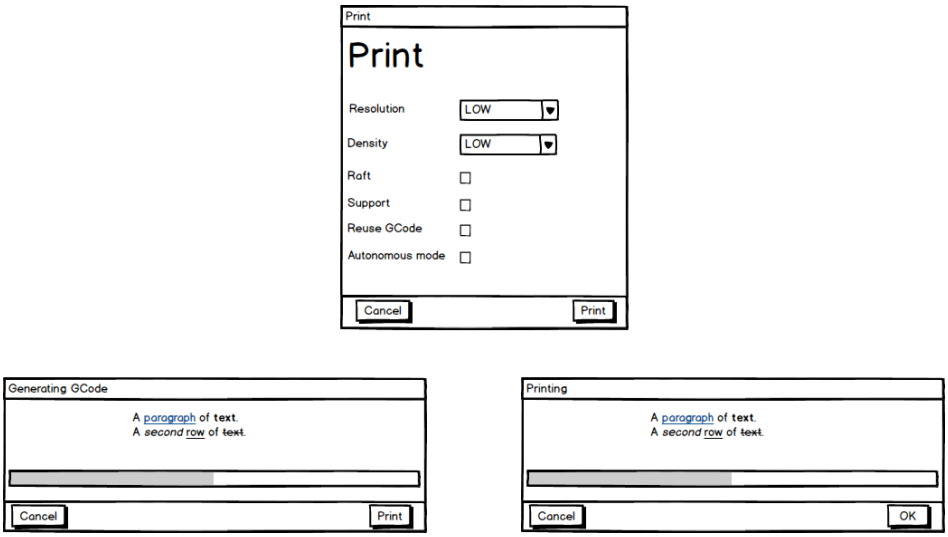


Figure 3.6: Print interface mockup

A platform that has many features and screens of varying degrees of complexity originates behaviors that may seem unfamiliar to the user. This event forces novice users to seek assistance on the application or software updates that may correct the unwanted behavior.

Help and Update

With less impact in the interface, but also relevant and mandatory for the correction of problems and usage clarity, there is the Help (figure A.20) and the Update screen (figure A.22). Both help user improving skills with guidelines and troubleshooting (see use case diagrams: A.19 and A.21 respectively).

We have seen up to now major and important screens such as the main window and the wizards, but also other screens with secondary functionalities for this kind of application — help and update screen. With secondary purposes, there is also the platform preferences and the gallery for model management.

Preferences

Every software allows setting their preferences for additional control or functionalities, avoiding creating extra icons or menu entries, which confuse users. Given the funcionalities for *BEESOFT*, the preferences interface (figure A.14) enables the user to select the application language, measurements, model color and two extra important options. Locking the vertical axis is one important option or a must have functionality to prevent unwanted transformations in the z axis, when we just want to apply operations in 2D. Resetting configurations can help debug and solve some system problems, changing the variables to factory values (see use case diagram A.13).

Gallery

Importing from the gallery implies having a screen that lists all available models with common operations, such as delete, add to 3D environment, rename characteristics (see use case diagram A.23), etc. The goal is to have a simplified and intuitive screen where model or

scene properties can be set and categories can be added/renamed/deleted, with a list view of all models and actions, applying properties changes or importing selected objects (figure 3.7).

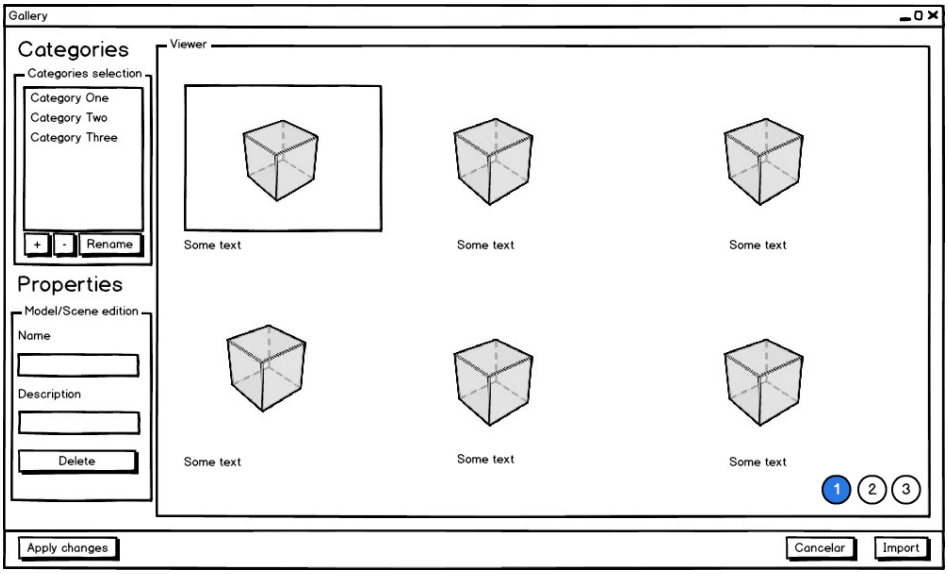


Figure 3.7: Gallery interface mockup

The next section will describe the implementation for these mockup screens as well as some relevant observations that help taking great advantages from *BEESOFT*.

3.2.3 Implementation

The figures in appendix B are the result of usability analysis over the mockups (appendix A) implemented in Java3D language. This interface improves *BEESOFT* major modules, such as print and maintenance, caring also about simplicity, user feedback and quick achievement.

It takes into consideration several aspects:

- *BEEVERYCREATIVE* branding
- Maintenance control panel
- Printer state information
- Operations feedback
- 1-2-3 methodology — few clicks to achieve goal
- Closed system — software works only with *BEETHEFIRST*
- Free advanced operations interface
- Easy to update
- Easy to recover from faults

The next sections will explain the User Interface components and the reasons behind the implementation, starting with the main window as one of the most important and complex screens. Appendix B includes screens for all the following interface screens.

Main window

In the main window (figure 3.8 and figure B.1) the containers take a prominent place as designed in the mockup (section 3.2.2 — figure 3.2): the model operations center on the left, the canvas in the middle and the scene details on the right. This order allows a better spatial localization and recognition, with the main focus in the middle and the operations and details on opposite corners — this behavior is common in CAD applications, with the toolbox located on one side and the informations displayed on the other.

Above these three containers we have the main screen buttons, with the most frequent and important operations. Compared to what is proposed in the mockup 3.2, the quick buttons for GCode generation and send were removed to be part of a more specific container (figure 3.14) because the user has to choose the print parameters (useful for the slicer), as we will see later on. This is a behavior present in the remaining screens of *BEESOFT*, where if an operation needs user input and can't be simplified down to be single button, it needs its own container in a new screen. It does not help the user to achieve the goal quickly as explained in section 3.2.1 (instant gratification), but it is balanced with user's reduced memory load, remembering choices and making it all simpler and faster [27].

The connection buttons were also removed due to the USB communication protocol ensuring plug and play connectivity. This eliminates the need to select a communication port and a printer driver as the serial communication needs to do.

On the application bar, there are the application menus as shown in figure 3.3.

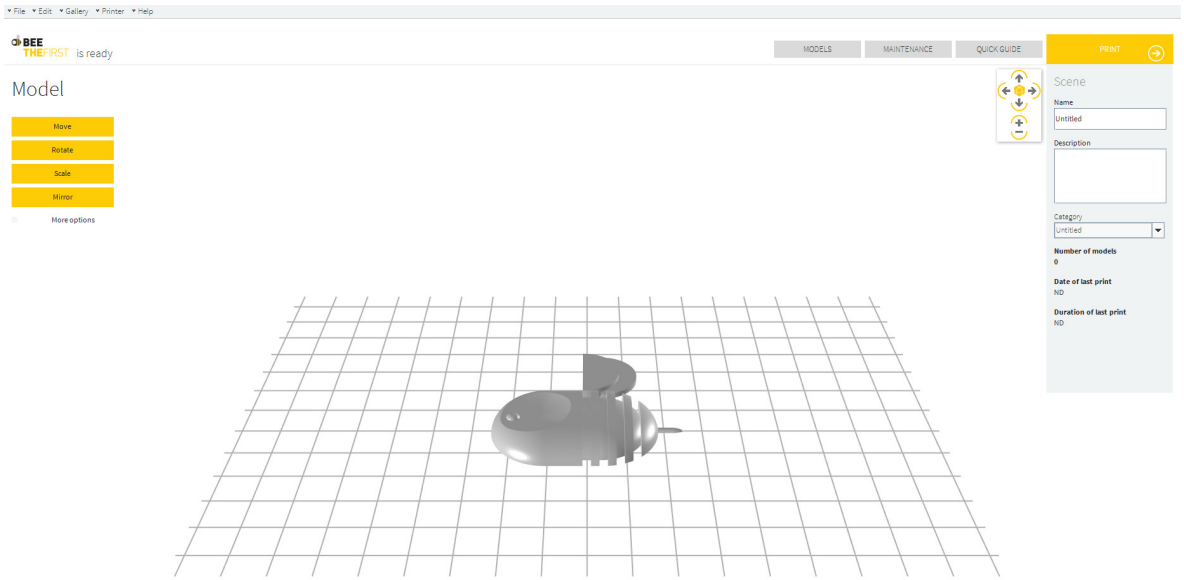


Figure 3.8: Main window interface

The quick access to the maintenance panel improves accessbility to configure correctly *BEETHEFIRST* before printing. The wizards available on it split each task into a subsequence of small tasks with information to aid the user through each action, sparing his effort figuring out how to reach the goal and simplifying the task. The graphical layout

organizes wizards access in equal containers with a delimiter line to identify the groups and respective information [28].

Next, we will see each of those wizards, their interface and *modus operandi*.

Filament wizard

The filament wizard (figure 3.9) remained structural and operationally equal to the mockup (figure A.8) but without the introduction and finish screens to grant simpler and incremental operations.

It has 3 screens:

- Heat (figure B.8) — heats nozzle to the correct temperature to load/unload filament
- Load/Unload (figure B.9) — loads or unloads filament
- Filament selection (figure B.10) — sets new color if a filament is loaded

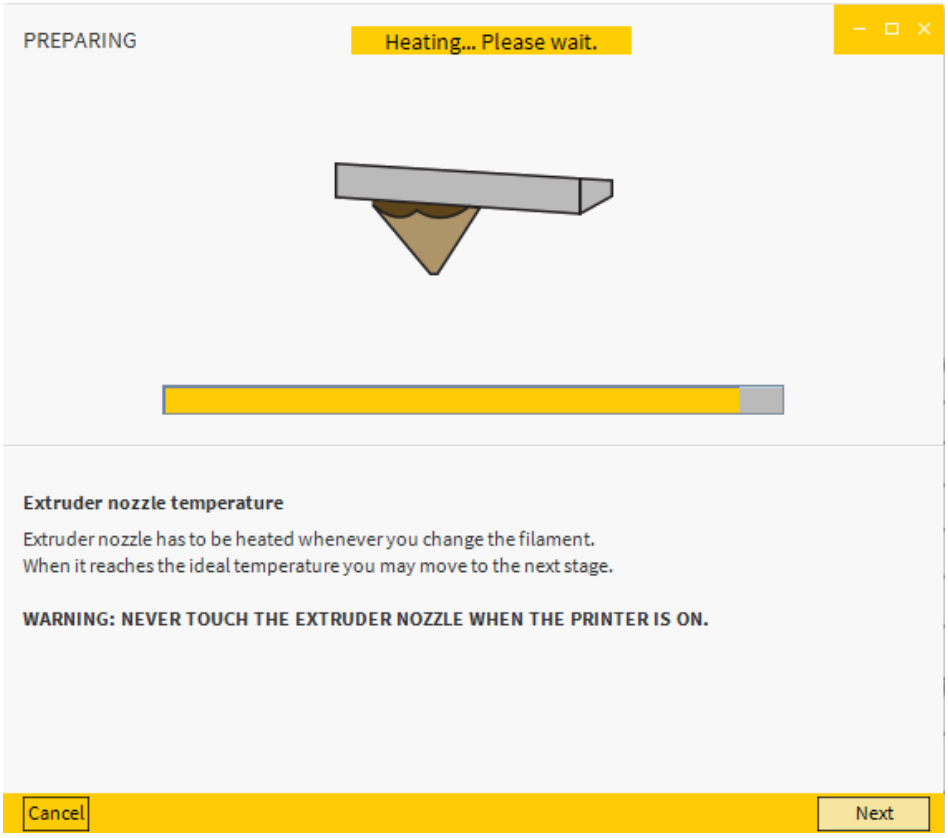


Figure 3.9: Filament wizard interface — Load/Unload

The load/unload operations can leave unnecessary leftovers in the nozzle, which are prejudicial to a correct calibration and a clean print.

Nozzle clean wizard

The nozzle wizard (figure 3.10) is equal to the mockup (figure A.10) and with similar methodology to filament wizard — without the introduction and finish screens, for the same reason. It has one screen that heats the nozzle to the clean temperature.

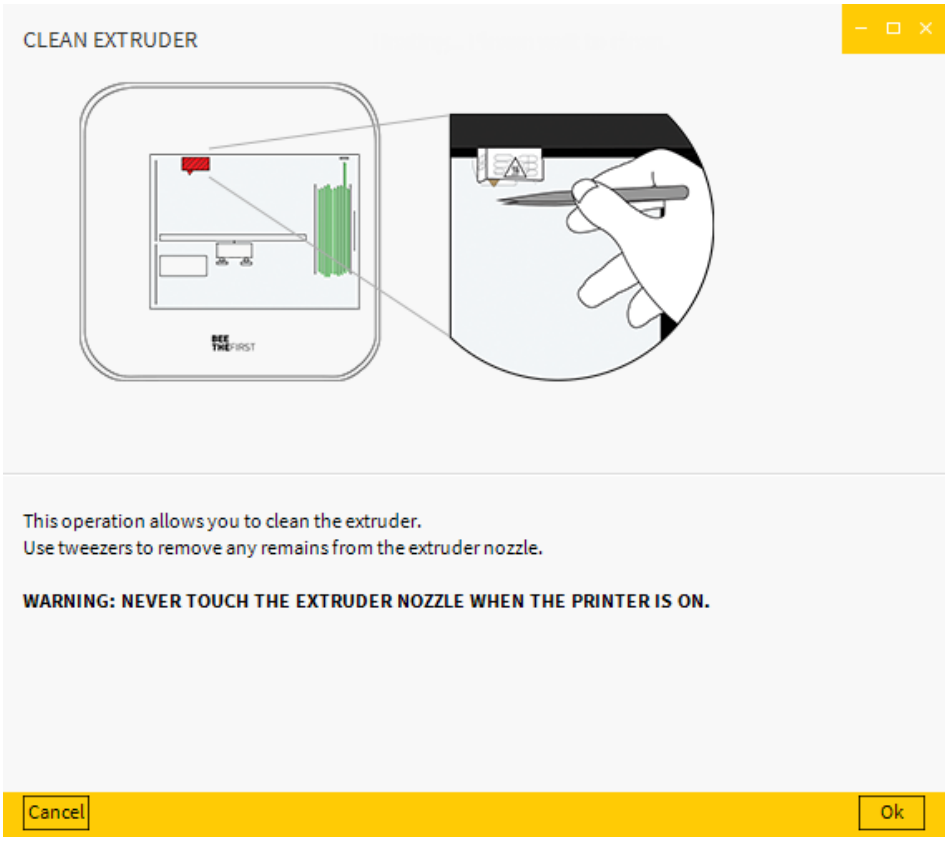


Figure 3.10: Nozzle wizard interface

It is important to run this wizard before calibrating to help remove any remains from the extruder nozzle for a correct and remains-free clearance distance between the print bed and the nozzle.

Calibration wizard

The calibration wizard (figure 3.11), like the previous wizard, is also structural and operationally equal to the mockup (figure A.12), without the introduction and finish screens for quicker goal achievement.

It has 5 screens:

- Table height (figure B.12) — calibrates the table height relative to the nozzle position
- Table level, phase 1 (figure B.13) — calibrates the table level at one of the 3 calibration points of *BEETHEFIRST*
- Table level, phase 2 (figure B.14) — calibrates table level at one of the 3 calibration points of *BEETHEFIRST*

- Test calibration (figure B.15) — allows to test the calibration
- Validate calibration (figure B.16) — allows to validate the calibration with comparative results

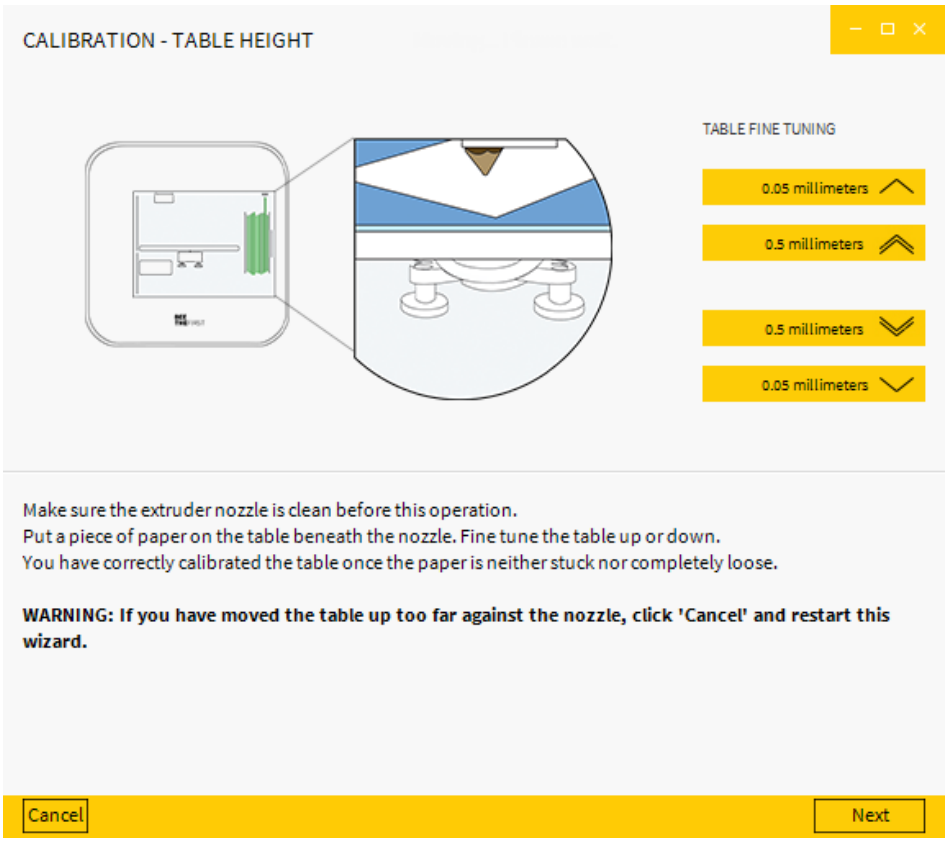


Figure 3.11: Calibration wizard interface — table height

The wizards available are meant to be simple and goal-oriented with the possibility of error correction. The fact of having a static layout makes them very consistent. The buttons position and text intend to be evident, helping the user to avoid any effort understanding and thinking about them.

The maintenance represents a complex module where it wasn't possible to get rid of complementary text or some elements to help the user understand what is happening in background. Not knowing what is happening, what to do or if the interface is non-responsive, could leave the user frustated and disoriented. According to Steve Krug ([26]) interfaces need to be self-evident and self-explanatory and that way, it was important to insert images and text in all steps.

Information is truly important to the users, enabling them to perform almost all of their tasks, but only the most important and indispensable information should appear in the main screens. Secondary textual details are relegated to secondary or subsequent screens [28]. Although textual information should be concise, plainspoken and speak user's language, their existence in available wizards demand to be explanatory and very simple at the same time. As they use the maintenance wizards, the users will have to gain and improve their skills to

correctly calibrate *BEETHEFIRST*, due the technical details transmitted over each screen. The amount of information available on each screen should be reduced and clear (rule *"Less is more"*) to clarify about the maintenance processes. This is difficult because of various risks associated with the handling of *BEETHEFIRST* [28].

The information *BEESOFT* provides tries to be clear, focused and task-oriented on completing each operation. This makes sense and helps users to realize how their actions are or will be reflected in changes.

On the main screen, the models operation center has actions that are processed and converted into visual information, either in the 3D environment or on the information panel of the scene.

Models operations center

The models operation center (figure 3.12) is simpler than the one of the mockup (A.4). It had buttons to transform the models with pre-defined values, and textfields for data submission by the users but, to improve the interaction with peripherals, the 3D environment supports those operations as we will see later in the 3D section.

There are 5 screens (equivalent to 5 states) available:

- Idle (figure B.1) — the model is not selected
- Move (figure B.2) — the model can be moved to a relative position using the text fields for each axis
- Rotate (figure B.3) — the model can be rotated by degrees values using the text fields for each axis or using the arrows. The arrows improve a better rotation accuracy when it is intended to rotate the model onto a flat face
- Scale (figure B.4) — the model can be scaled by absolute values using the text fields for each axis
- Mirror (figure B.5) — the model can be mirrored using the buttons for each axis

The 3D environment renders the transformations made on this panel to inform about the success or failure of the operation. But there is some information that was considered in the mockup and which is transmitted by the scene details screen.

Scene details

The scene details screen (figure 3.13) is important to help distinguish some details of the loaded scene as the name and category or even global details, such as the duration of last print. The additional details in the mockup (figure A.6), like the remaining filament and the number of prints since last calibration, were placed in the maintenance control panel so that their access is intuitive. When a model is selected, this component is updated to display model details (figure B.7). Since the canvas takes control of the front plane and at the left are operations buttons, the operations in the *Edit* application menu entry (place on platform, center and reset position) are duplicated to accommodate the user to more operations over the model.

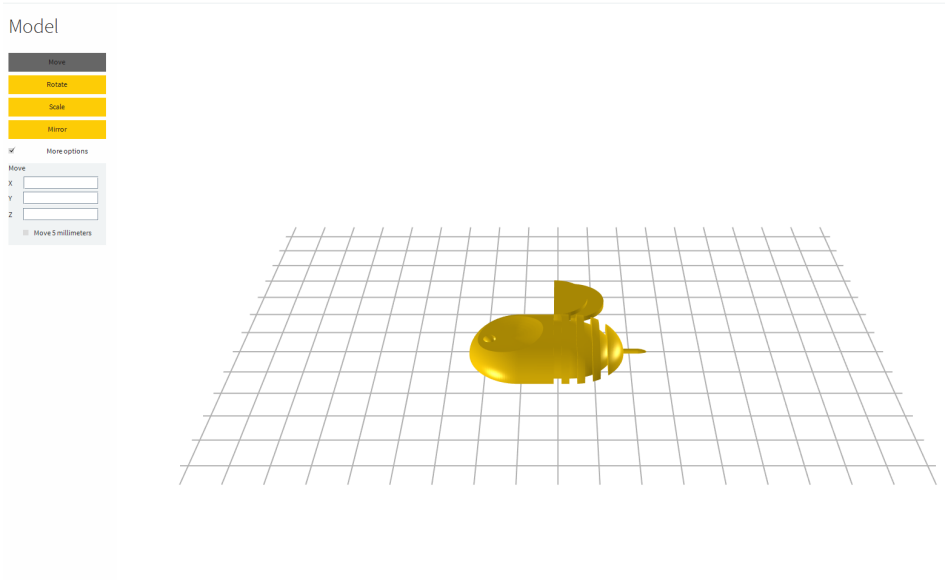


Figure 3.12: Models operation center interface — Move

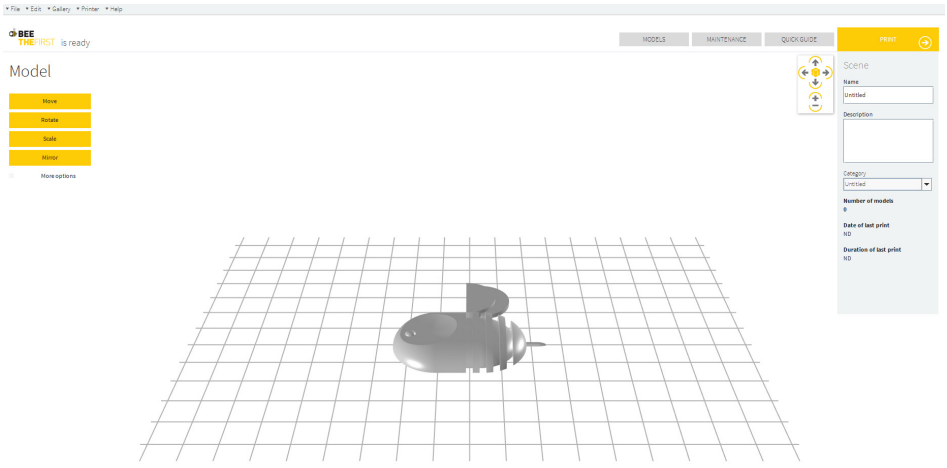


Figure 3.13: Scene details interface

We have seen until now what panels are responsible for the scene edition and rendering. The processing of the scene by the slicer algorithm is part of the printing process that is included in the printing wizard. This wizard deals with the GCode generation and parameterization of print profiles.

Print wizard

As stated in the main window section (section 3.2.3), the GCode generate and send buttons were removed due to dependencies related with the slicer. The printer wizard (figure 3.14) allows setting parameters for each print session as designed in the mockup (figure A.16), with track bar elements to preserve the visual appearance of the interface and of the information.

Further it will be explained how *BEESOFT* works in major operations to help the user remember about his last settings, e.g., warnings that GCode must be re-generated. The GCode generation (figure B.19) and print itself (figure B.20) are post-parameters selection screens with work progress on progress bars, with the percentage elapsed and the percentage to go.

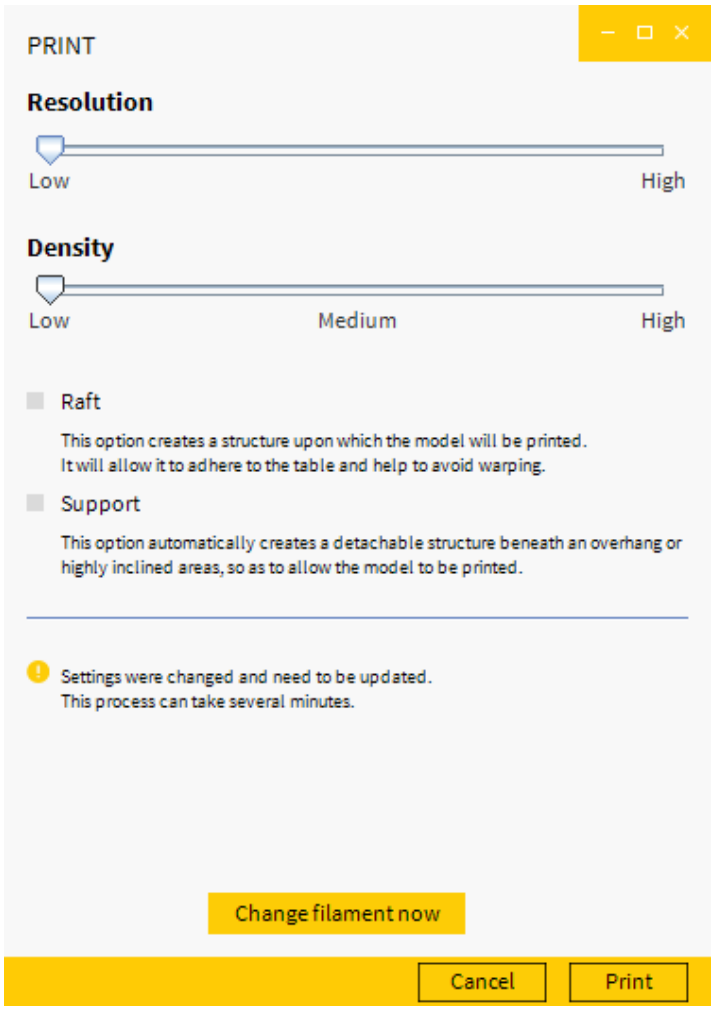


Figure 3.14: Print interface — choice of parameters

The complexity of this wizard, and the occurrence of problems or inconsistencies in the behavior of the interface may lead users to seek help or software updates. It is important that these screens are presented to the user in a quick, easy, simple and effective way, as mentioned in the explanation of the main screen interface.

Help and Update

The Help (figure 3.15 and figure B.22) and Update (figure B.23) screens are composed of most of the components from the mockup (figure A.20 and A.22 respectively). A list of topics for help could not satisfy user expectations, and because *BEESOFT* does not require constant internet connection, this component was refactored.

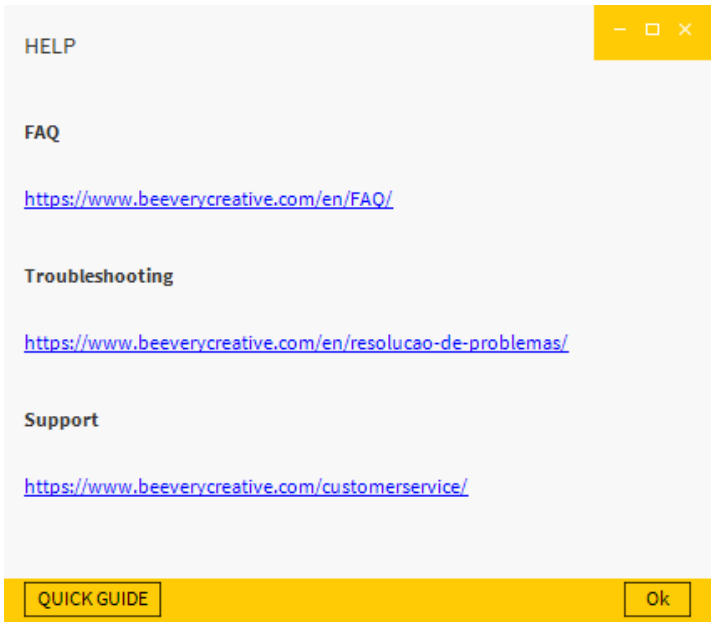


Figure 3.15: Help interface use

Important as these screens are, there are others which allow us to change the settings of the application.

Preferences

A CAM application for 3D printers can have a standard or advanced interface with basic or advanced preferences, depending on the user experience. This platform contains a simple preferences screen (figure B.17) where friendly user settings represent all the customization available. The information about the model color was removed from the mockup (figure A.14) and placed under the *About* menu allowing the user to obtain general settings just from one location.

The information of the scenes and models are on the main screen as explained previously and its edition is also allowed in an operations center, designated gallery, that allows the user to manipulate files and change the properties. They are shown in the model details and scene details panel.

Gallery

The Gallery (figure 3.16 and figure B.24) is equal to the mockup (figure A.24). It is divided in a browsing container of existing models and scenes, and a left container to edit their properties. This screen follows the current *BEESOFT* methodology: major or group operations are focused in a central access point to avoid confusing the user's spatial memory.

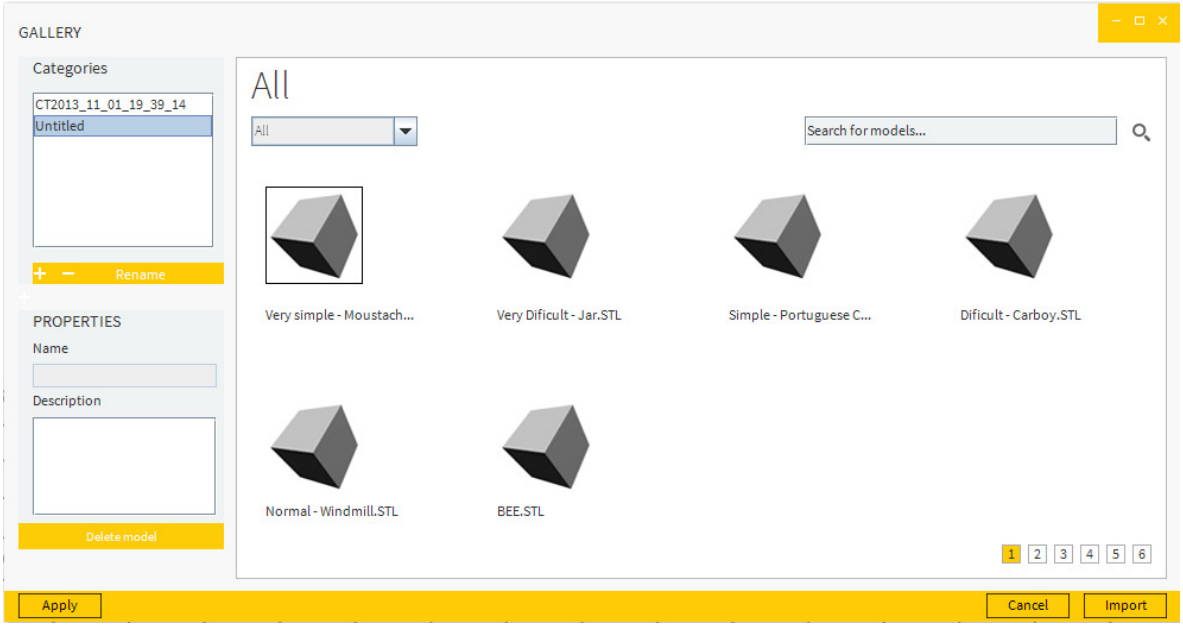


Figure 3.16: Gallery interface

With the purpose of importing and transforming a model, *BEESOFT* needed an interface to import the models. The absence of an internal library with all files and the need to get quick access to them imply that navigation on *BEESOFT* was not effective — that is the purpose of the gallery.

According to Steve Krug ([26]), navigation through a system intends to show where the user is and to get what he wants ([26]), with integrated components as a search bar. This screen layout was designed to provide an overview of which models we have and their global properties, to allow comparison and distinguish between ambiguous file names and a search bar to help the search. Because users are accustomed to search ([26]), search bar is straight forward to what it should be, including a text tip saying *Search* to clarify its functionality and an identifier element to help understand what it is for with no direct confusing reference to its scope.

3D

In section 3.2.3, we highlighted the central container as the most important and center container of *BEESOFT*. The 3D canvas renders the transformations applied on the model operations center (section 3.2.3), in the Scene details container (section 3.2.3) and in the Gallery. These transformations can only be applied to a selected model (figure B.26) even with multiple models on scene (figure 3.17). Similar to the rest of *BEESOFT* user interface, this container provides feedback for each operation becoming more cohesive.

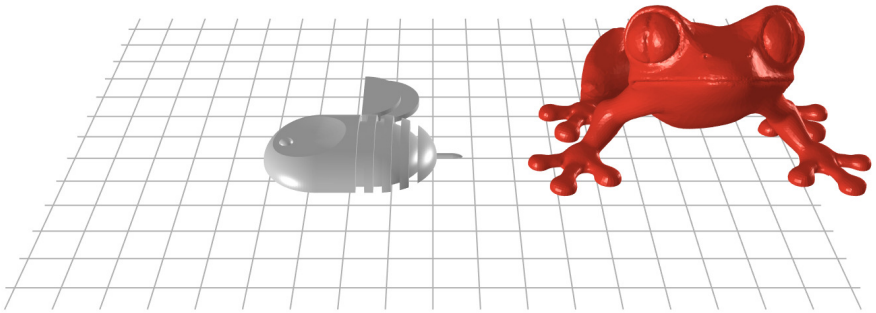


Figure 3.17: 3D interface — two models on scene with transformation

The model operations center affects the canvas directly, triggering internal listeners to handle the event (we will see this later on section B.13). This event causes the container to refresh and to apply the changes. Accordingly to Jenifer Tidwell ([23]), input devices are important and a fundamental interaction, and their use can't be restricted to just one. *BEESOFT* handles mouse and keyboard listeners that provide equal transformation operations in the model operation and scene details center. Therefore these listeners provide the user two ways to transform the scene with feedback over the 3D canvas. This structured scheme reduces complexity and improves engine usability, with familiar OS shortcuts and I/O good pratices.

Besides the model operations center, a model, when clicked, causes the refresh of the panel at the right — the scene details panel. If the model is unselected, the selection will redraw the main window with new model details and the opposite if already selected. A quick refresh could lead users to get disoriented not understanding what happened, and if it occurred too slowly, they get frustrated [29]. This rearrangement of elements occurs always in the same context of actions and the redraw is noticeable and user friendly, improving user habituation and learning [24].

Previously, we have seen how and why the screens of *BEESOFT* have this look. In all of them, contents were organized in small areas to help distinguish visual priority, making obvious what is clickable and reducing background noise. This focuses the user in what is most important [26].

Despite *BEESOFT*'s numerous text entries, they are short, making the content more prominent and more detachable, emphasizing basically the essential. The UI elements are just visible locally at each container and their prominence on-screen is achieved by the position and not by the color scheme. This scheme has two main hues, gray and yellow for branding purposes, to produce subtler color combinations that work very well in brightness levels. Using one to three main hues, we can get a greater depth and dimension, calling user attention without causing saturation and a brightness impact [26]. Gray provides a more recede interface with the possibility to combine colors to make it cooler or warmer. In this case, the yellow and gray for a simple and warm look, with white as background in the main screen — main window. This way, colors aren't used to contrast visual elements, but to provide a better feedback over the buttons and to promote a harmonious, aesthetic and effective interface.

3.3 Feedback

Every application deals with data input and validation with constant thread operations. These operations change the global behavior and the printer state while new operations get available.

These transitions and successive manipulations demand an application robust enough to let the user understand what is happening and what he can do based on his action history.

Using feedback in *BEESOFT* is beneficial and helps the user interface to get more understandable if it applies [24]:

- Notification elements
- Buttons
- Images
- Colors
- Message alerts
- Sound alerts

To enhance usability and inform about small details or operations that were not completed or started, *BEESOFT* uses notification elements in distinct ways.

3.3.1 Notification artifacts

In the printing panel (figure B.18), the user is warned if there is no coil filament remaining (figure 3.18) or if the GCode must be recalculated. This improves the sensorial perpception that for each print these are mid conditions that need to be taken upfront.

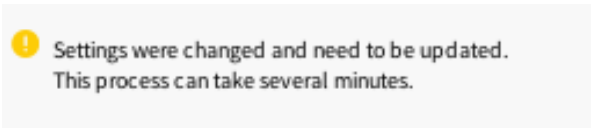


Figure 3.18: Artifact warning about the need to recalculate GCode

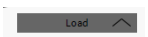
3.3.2 Buttons

Buttons are responsible for triggering the listeners of the current screen through their behavior: blocking further actions through a pressed state (figure 3.19d) while some are being executed, giving visual response with enabled (figure 3.19a) and disabled state (figure 3.19b) and mouse hover for the availability (figure 3.19c). Figure 3.19 shows the possible cases.

(a) Button state — enabled

(b) Button state — disabled

(c) Button state — hover



(d) Button state — pressed

Figure 3.19: Buttons behaviors — example from Filament wizard

3.3.3 Images

Wizards need to inform the user about their purpose and instructions helping him reach the goal. Images (for example figure B.13) are one of the best feedback components to supply such information as what the screen is about, how to operate *BEETHEFIRST* and also to compare the results obtained in relation what was expected. When they are not clear enough or are insufficent, there are messages and sound alerts.

3.3.4 Colors

Moving, rotating, scaling and mirroring models can place them in an invalid position and therefore, it can ruin the print. The 3D canvas gives feedback through a pallete of colors for a model: idle and valid (grey color - figure B.25), model picked and valid (yellow - figure B.26) and an invalid model (red - figure B.27). The red color means a model is out of bounds, i.e., the dimensions were exceeded or the model is not in the platform.

3.3.5 Message alerts

Status bar

The status bar (figure 3.20) is located at the bottom of the main window and has temporary visibility. It is intended to inform the user about all the main operations made in this screen, such as modeling, importing models, saving or opening scenes or even errors that may occur.

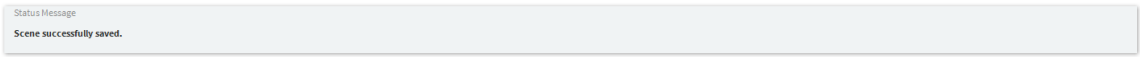


Figure 3.20: Status bar for Main window operations

There are many situations that can fire an alert condition and so it's important that they transmit helpful and task-oriented information [29] instead of an error code. *BEESOFT* has three kind of set of operations — internal, I/O and system calls — being important to distinguish each execution and error handling to provide an accurate information how to get over it. This bar describes the problem in terms related to the task the user was trying to do, with enough information for him correct the problem. Also there is the top printer status bar for printer connectivity with the *BEESOFT*.

Printer status bar

BEETHEFIRST has enabled (figure 3.21a) and disabled (figure 3.21b) states showing if the printer is switched on or off and plugged in or not. This restricts what a user can do within the application affecting all UI components. The status bar informs about this state to help the user understand if he can operate with the printer.

(a) *BEETHEFIRST* ready

(b) *BEETHEFIRST* not ready

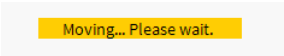
Figure 3.21: *BEETHEFIRST* states

Similar to the main screen the wizards provide feedback about the printer status to show what it is doing and to control the behavior of the UI elements.

Moving/Heating status bar

The operations available in the wizards imply mostly heating or a movement. This causes the necessity to inform the user of what the printer is doing and if he can go forward with the maintenance. The status bar (figure 3.22) lets him know if *BEETHEFIRST* is under operation and when it finishes it. It shows a heating status (figure 3.22a) and a moving status (figure 3.22b).

(a) *BEETHEFIRST* heating



(b) *BEETHEFIRST* moving

Figure 3.22: *BEETHEFIRST* states

This feedback can be enhanced with an audible alert to confirm effectively that the operation ended and the UI elements are available.

3.3.6 Sound alerts

Wizards with more delicate operations and the progress save points, like saving scene before printing or GCode generation completion, can be enriched with a beep providing more information that suppresses the failures of visual representation. According to Alan Dix ([30]) the addition of audio confirmation for keyclicks or events, reduces user mistakes through a second representation of action feedback granting a confirmation. It supplies redundant supporting information through an abstract generated sound that distinguishes from background sounds — auditory adaptation.

3.4 Conclusions

As functional and communicative as they may be, these elements don't comprise all the necessary situations of feedback that can happen. More complex operations or more detailed models may cause the platform to get disfunctional with low responsiveness, compromising user satisfaction.

Performance and responsiveness are quite different: how quickly the computer executes the task and when the computing freezes the interface due to tasks not optimized priorities or bad I/O requests management, is a responsiveness problem [29]. These two sympthoms are considered UI designer's responsibility, because response times are not taken into consideration well enough when designing the application. They may depend on what response times are achievable for a particular operation in a given environment [29].

In *BEESOFT*, we have two main distinctive situations where we can evaluate responsiveness: operation running in background with an interface feedback; and importing and transforming a model not optimized for *BEESOFT*. Saving/Loading a scene is a top level operation that requires disk access to Write/Read, and when the process is started, the interface is inoperable without feedback. When done, the user understands if it was successful thanks to the message that appears in the bottom status bar in main window — feedback bar. The existing loading bars intend to inform about the current progress but the absence of the expected completion time and of some details about the operation running in background, affect user awareness regarding how long is going to take and if it is really doing something [29]. The mechanical commands of *BEETHEFIRST* while the interface is updating minimizes this problem. Applying transformations in a model outside the specifications of *BEESOFT* causes the interface to get low responsiveness without any visual artifact or indicator about what is happening or the possibility to cancel — detailed and more complex scenes lead to low responsiveness.

According to Robertson and Nielsen's time constraints table ([29]) regarding temporal goals that sofware must meet in order to be perceived as responsive, 0.1 second is the time for a feedback of a button that has been clicked or displaying frames and indicators; 1 second, is the time for progress indicators or opening a dialog box or autosaving; 10 seconds is the approximate unit of time into which people abort their planning and execution of larger tasks — it is more like the time people look up from their tasks and reassess their task status. *BEESOFT* can have a better and responsive feedback for each operation with reduced times helping to be better perceived and preventing the user from getting exhausted and frustrated.

Every single screen is located and accessible evenly in each *BEESOFT* start-up and interface elements, such as textfields, buttons and drop-lists have the same behavior and data every access. Actions and response behaviors respect this consistency. According to Jakob Nielsen ([28]) the same information should be present in the same location on all screens and formatted in the same way. Actions should have the same effect, constantly encouraging the user to try exploratory learning strategies.

Wendy A. Kellogg ([31],[32]) refers consistency is an aspect more important than screen design, using the considerations of the task and functional structure of the system as an example of consistency. Interface functionalities should be provided under the same scenario over and over, ensuring the process is familiar to the user, with minor variations possible. *BEESOFT* builds each scenario in the same way, with elements and actions at static locations, with action-oriented behaviors and without unpredictable or random background external services execution that could start unfamiliar screens.

A control application of 3D printing has to ensure the correct data processes and stability in the communication channel protecting the system printing failures. The model imported into the CAM application is transformed into triangular meshes rendered on a 3D container that will display the appeareance (shape) of the printed model. With this container there are other major application modules that will work with the data processed by it — module communication and control module of the machine (UI).

The integration of these modules in a system architecture with structured operational flows and robustness and stability in data processing, allows the application to have a simple and functional interface. Oriented towards simplicity, functionality and accessibility, *BEESOFT* applies the recommended good usability rules, such as error control, static interface and easy to get used to features, allowing the user to operate in a transparent and solid way over the 3D environment and using the communication protocol so that the control of the *BEETHEFIRST* is easy to use.

Chapter 4

Implementation details

This chapter introduces fundamental concepts and also explains important implementation details regarding model rendering and the communication protocol in *BEESOFT*, and how they are integrated with implementing the interface.

4.1 3D rendering

A virtual representation of the printing bed and the models to be printed is an advantage for the control of a 3D printer. It allows users to change the position of a model and get feedback in a more interactive, simple and intuitive way than with an interface similar to an advanced form of control buttons.

The 3D rendering window takes a prominent position in all CAM applications, occupying large panels and with the main features of 3D printing directly integrated. Its availability in the main screen and easy of use enables the user to prepare most of the printing sessions in a simple way.

4.1.1 Introduction

3D printing an exported model from CAD software may imply repositioning model parts. *BEESOFT* embeds a 3D engine (figure 4.1) to manipulate models (or parts) with the transformation operations explained before (see section 3.2.2).

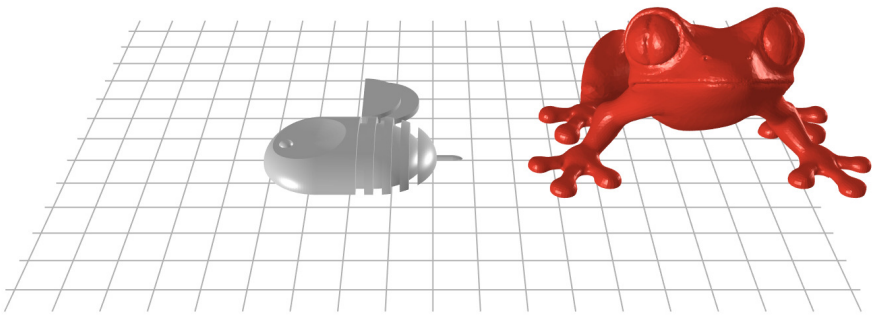


Figure 4.1: Two models rendered by the 3D engine

It is important to find the best balance between printing bed calibration, print parameters and model position on the bed, based on those transformations, in order to achieve the best print results possible.

The 3D engine needs to:

- Load an STL file
- Assemble virtual models from STL files
- Render a virtual world to represent the models
- Import/export models to/from a scene
- Allow selecting a model
- Allow applying operation
- Apply each operation
- Render operations giving feedback

Once an STL model is represented as 3D triangular mesh it has properties such as material, light, color and texture which are rendered in the scene. Transformations represent examples of operations to be applied to models, which are then rendered for better user feedback (figure 4.2).

Figure 4.2 illustrates the main 3D rendering engine operations through a simple flow chart represents in a temporal sequence, the rendering engine. The user imports a model to a 3D world similar to a workbench where the data in the imported file is represented as a 3D mesh. Later, all the applied transformations, such as move, rotate, scale and mirror operations, are processed and the scene is redrawn, validating those changes by ensuring that there are no models outside of the printing area or volume, or that no error occured during file reading or mesh processing procedure.

4.1.2 Details

A 3D rendering engine applies computer graphics, geometry and mathematics algorithms and uses data structures to represent the collection of vertices, edges and faces that define the surface of a polyhedral model. The available resources of the computer running the application and rendering the scene can degrade the scene graph management performance, and consequently, the user experience — the increased complexity of the models (more triangles imply more detail) and the scene rendering, slow down the application responsiveness because of the I/O interactions [33].

3D printing comprise vertices, faces, textures, camera, lighting and environment effects (behaviors), but as mentioned before, as the scene becomes more complex, performance tends to worsen. Therefore, arranging objects in a scene graph with a hierarchy tree of all 3D objects improves efficiency [34]. The scene graph consists of various modules, each responsible for handling a particular type of object in the virtual world or a particular process, where more realistics graphics need more complex processes to decide which triangles are sent to be renderered. The scene graph contains all the geometry of a particular scene useful for representing translations, rotations, mirrors and scalings of objects.

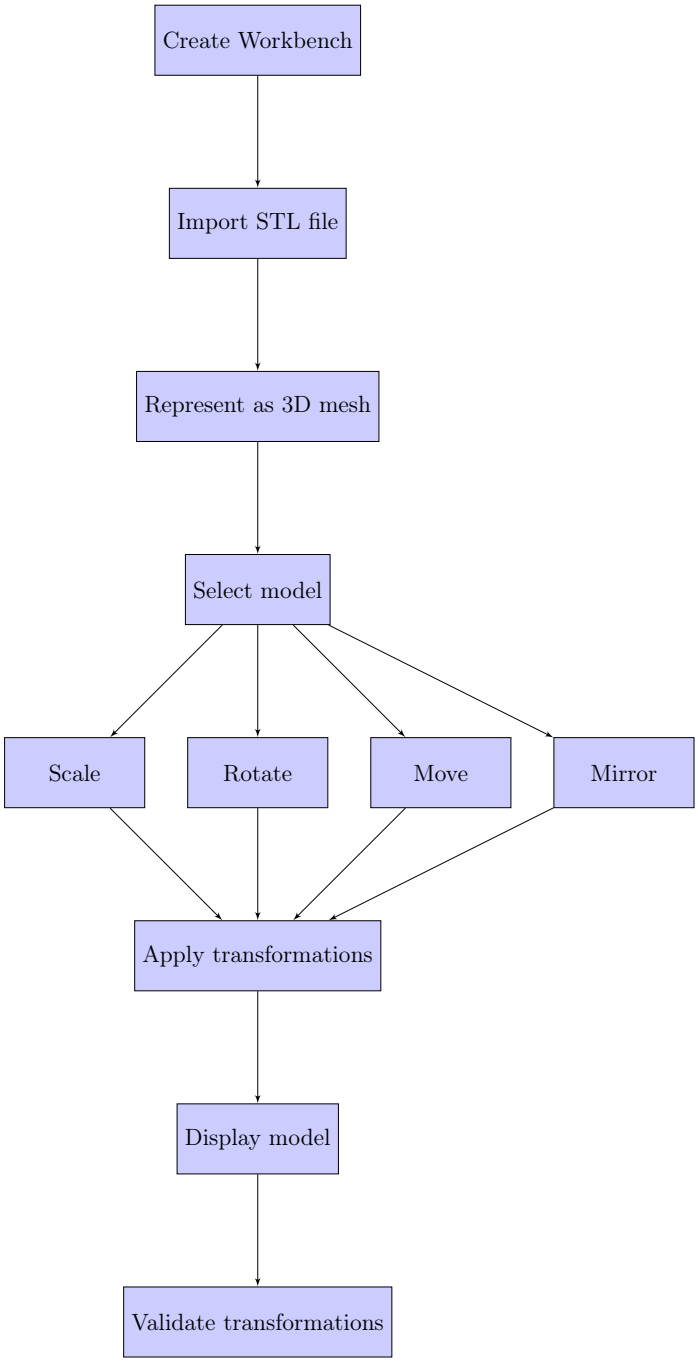


Figure 4.2: Main 3D rendering engine operations

All the elements and properties that make up a scene are encapsulated in a virtual universe organized hierarchically in nodes and groups (collection of node objects) within a virtual geographical region defined by the Locale [6]. More specifically, the 3D scenegraph, has a view mode, responsible for rendering the nodes position and the orientation information, and also the scene branch, containing the scenegraph elements (figure 4.3). The view mode, as shown in figure 4.3, has a parent node that defines scaling, rotation and translation information for the view, called 3D canvas which is visible and directly modified by the user.

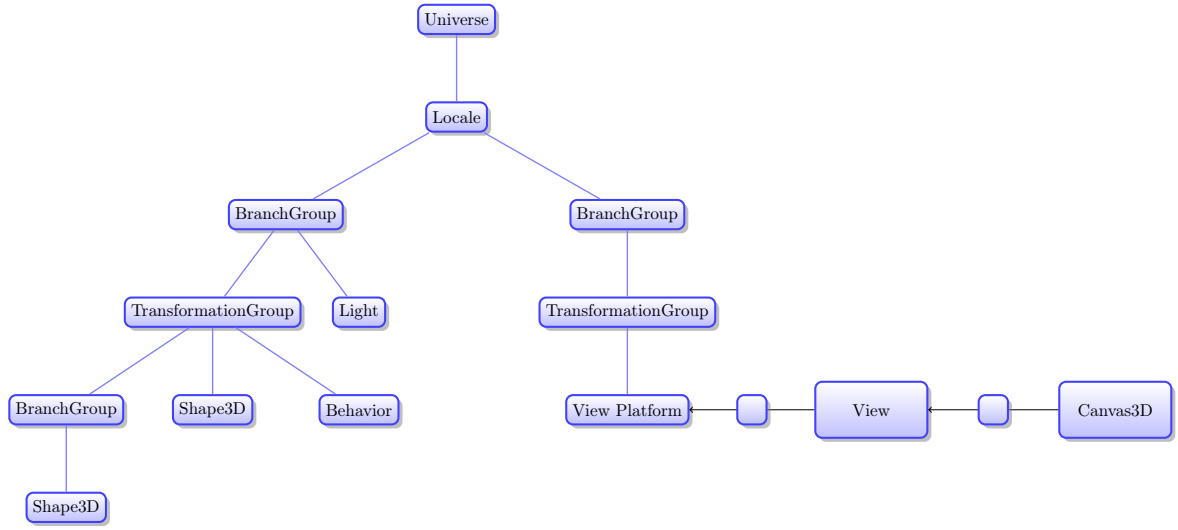


Figure 4.3: 3D scenegraph

The view node compiles these elements into a user interface 3D component called canvases — 3D Canvas. Each view can be rendered into multiple canvas leading to an increase of complexity. But to manage, it is necessary that each scenegraph level be well-defined with all the information set per Node [6]. The 3D canvas is therefore a group of Nodes, each containing the virtual object information and the modeling transformations applied, stored in a file that will be processed and sent to the lowest communication protocol layer — the driver.

4.2 Communication protocol

An application that interacts with hardware components needs a stable, robust and secure communication channel to ensure data delivery. 3D printing applications use mostly the RS-232 (serial) protocol over Universal Serial Bus (USB), but it is a non-reliable standard with hard to recover transmission failures. *BEESOFT* uses the USB protocol so that all control operations are effectively delivered and data transmission during a printing session has error control with a re-send packets system and communication recovery for a stable connection.

4.2.1 Introduction

The USB was aimed to replace specialized interfaces and simplify the configuration of communication devices [35]. Nowadays, it is an extremely used protocol for device commu-

nication, as printers, facilitating channel configuration and sending data packets controled by an error control algorithm.

4.2.2 Details

USB is reliable, fast, versatile, power-conserving, inexpensive and supported by all operating systems, which makes this interface suitable for large-scale use. Thanks to long years of research and upgrades, it has become an easy-to-use interface, satisfying and justifying developers choices with clear benefits [36]:

- Single interface for all devices
- Automatic configuration by OS detecting device and loading appropriate software driver
- Easy to connect
- Convenient cables with small connector and cable length restrictions for reliability
- Plug and play with OS detection
- No user selectable settings before using
- No power supply is needed because it is included in the interface

These advantages also make it easier for developers to ignore cable characteristics or error control software. Besides that, USB has multiple speeds and transfer types (four each) making the interface usable for many types of peripherals. Each transfer type can exchange both large and small blocks of data, with and without time constraints guaranteeing bandwidth. For *BEESOFT* we want a communication diagram as illustrated in figure 4.4. In the next sections we will see what this implies.

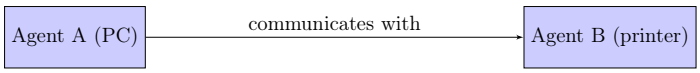


Figure 4.4: USB protocol agents

4.2.3 Characteristics

To establish the figure 4.4 communication, software must have an initialization module that:

- Scans OS USB devices for the printer
- Once found, creates the data channels (pipes) and blocks of data (IRP)
- Sends the printer initialization code
- Initializes software variables

And a control module that:

- Constantly evaluates the communication line state
- Sends data to printer when requested
- Processes data from printer

Initialization module

Scanning USB devices means relying on the operating system to determine the bus speed, assigning an address and additional information (figure 4.5). For the hardware to be recognized it is necessary to install the printer firmware to respond to standard USB requests from the host and other events on the bus. The requests ask for a series of descriptors, which are data structures that describe the device's USB capabilities [36]. Additionally, operating systems as Windows need a driver information file — INF — that names the driver the device will use on the host computer (3D application).

Once found, the printer is queried with an initialization code requiring the existence of communication pipes — one for input and the other for output. Pipes are associations between a device's endpoint through an endpoint address and the host controllers. Endpoints are buffers that consist of blocks of data memory or registers in the device-controlled chip. USB communications use IRPs that contain structures called USB Request Blocks (URBs) enabling the driver to configure devices and sending data. Drivers communicate with the bus drivers that handle USB communications through pipes (figure 4.5).

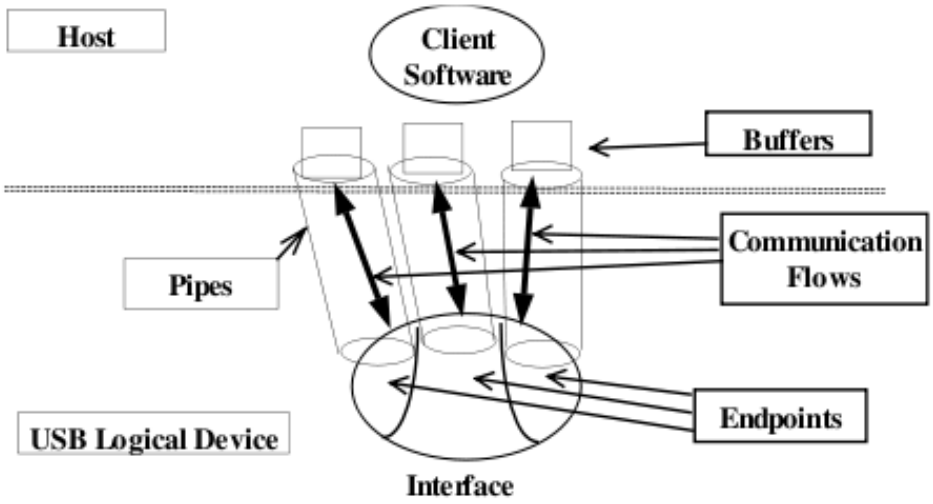


Figure 4.5: USB data flow [35]

If a user detaches a device from the bus, the host requests new pipes or removes unneeded pipes by using control transfers to request an alternate configuration or interface for a device. Every device has a default control pipe that uses endpoint zero [36].

Once the initialization code is sent, the agent listens to the pipe for an answer to enable the printer operable state — ready state. This communication between the host and the device is kept alive with an end-to-end packets transfer system operating over several layers that control individually distinct characteristics of the control modules. The whole process is transparent to the user.

Control module

Maintaining an up-line state requires a well-formed printer driver, correct initialization, well-formed sent/received messages and a pipes-testing mechanism. Functionalities embedded in the plaftorm communication layer(s), as the send and receive message methods, have to per-operation, deal with pipes-testing, creation or opening, connection recovery and driver state management.

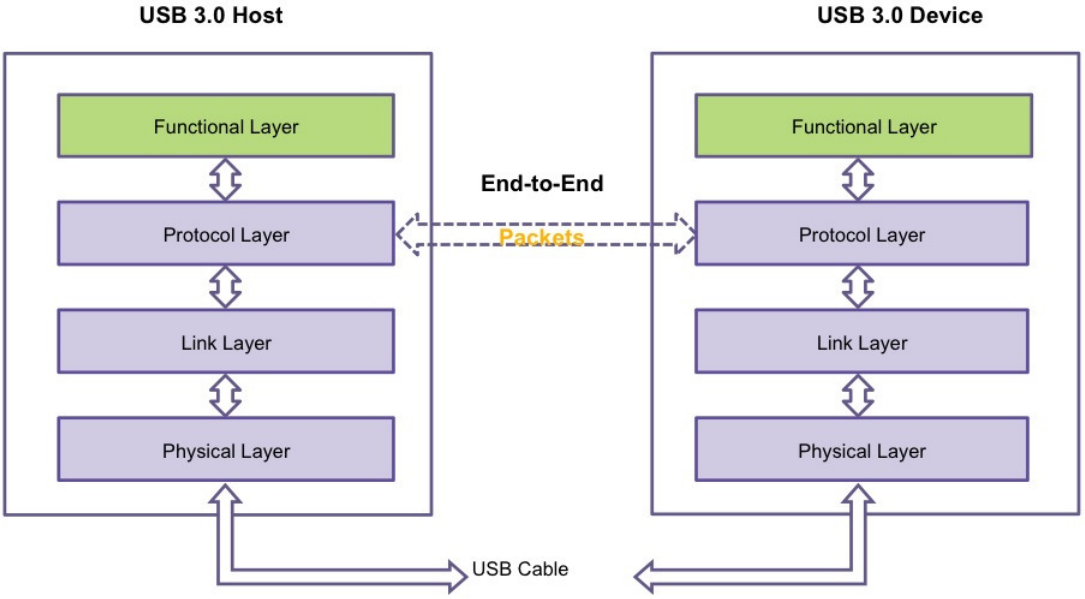


Figure 4.6: USB host and device layers [37]

As shown in figure 4.6, a host and device packets exchange in an end-to-end system is ensured by the Protocol layer; the link layer is the lowest layer that transfers data between nodes, providing functional and procedural means to transfer data with the possibility of detecting and correcting errors [38]. Below, the physical layer consists of the hardware technologies defining the means of transmitting raw bits rather than logical data packets over a physical link [39]. In a 3D print platform, a communication protocol with error detection is extremely important, since all data sent to the printer is content formed by sequential GCode commands and a single loss is visible on the printed object.

Software communication needs to be structurally strong and aware of all possible faults to respect the diagram of the figure 4.7 and figure 4.8. It is important to maintain an up-line state through a well-formed printer driver meaning it has to specify clearly, technical values such as mechanical limits, motor steps for millimeters, rates and OKs after resend. With a device

driver at the OS level (if needed) and in software, we can establish a communication respecting a high-level flow, as described in figure 4.7, where the initialization is made sequentially. The initialization process requires scanning the printer, grabbing it (virtual USB device) and opening the pipes after configuring read-and-write pipes for the communication flows (see figure 4.5).

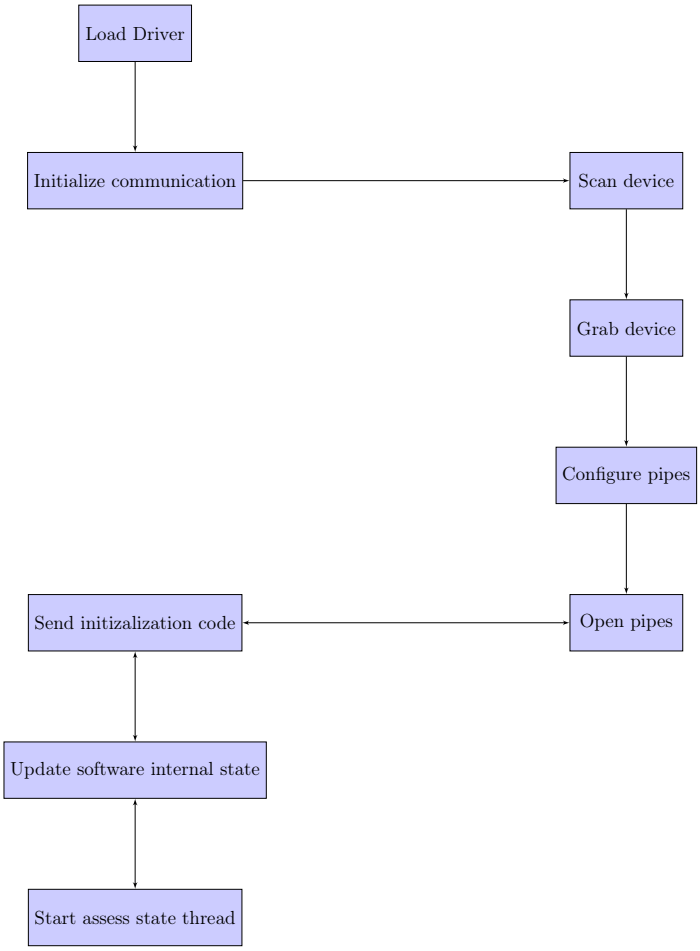


Figure 4.7: Software initialization flow

Once initialized, behavior can differ between proprietary printers, but mainly, all of them load the firmware, load important configuration values that keep the machine in a correct stable behavior and execute movements on all axes to place the print bed and the nozzle in a safe position. When ready to operate, the application needs to update its internal state starting an assessment state thread that will be running during application execution. This agent evaluates the current communication state and, if it modifies, changes the internal value depending on the driver state (figure 4.7).

No less important is how the host and the printer will see each other and deal with GCode commands. Figure 4.8 demonstrates how an active thread can send a message and get the response, explaining also how the commands are processed and submitted through the driver layers. In section 2.3.4 was explained what GCode format is and its importance in a 3D

printing application. Feed rate and motion movements commands (and also machine control commands) need to be interpreted by the firmware, implying the existence of proper agents at both sides that are responsible for decapsulating the packets (figure 4.8).

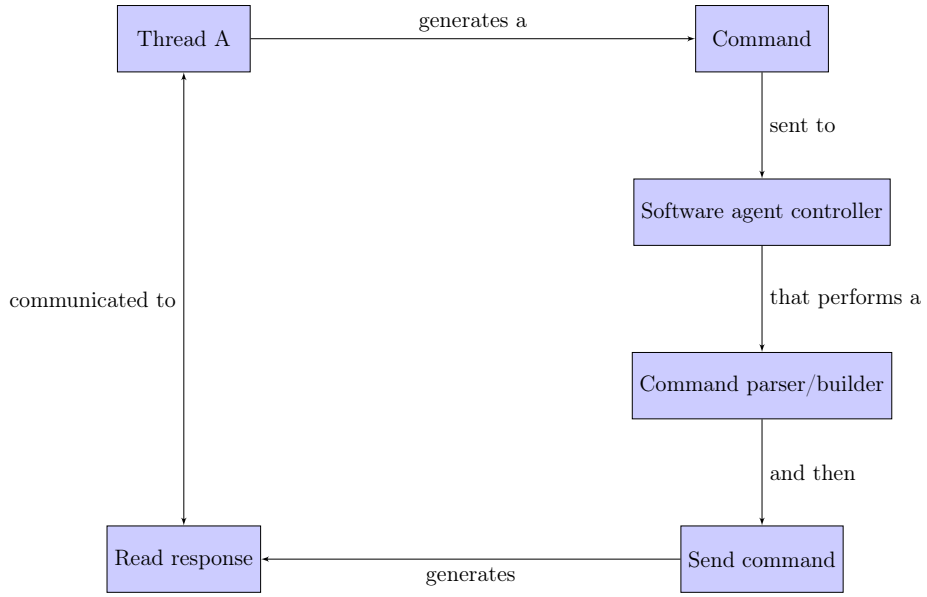


Figure 4.8: Software communication flow

A request is fundamentally building a command that requests the movement of the 3D printer motors, for example, with a speed of 100 millimeters per second. The software, as a pull-and-push agent, must ensure a correct and valid communication protocol, valid syntax and be understandable by the firmware — e.g. a motion command is made up of five parameters: **G1 X- Y- Z- F- E-**.

Firmware is controlled by an agent that parses the command, shipping it to the driver high -evel layer to be sent and executed by the printer (figure 4.9). Subsequent commands are sent only when an Acknowledge (ACK) is received, granting sequential execution and error control. The ACK may contain the response for the request — temperature request for example.

A 'G' GCode commmand is more complex than an 'M' commmand, because most of the time, it implies parsing axis values, printer envelop size and distance unitary system to evaluate correctly which is the absolute or incremental position for the desire request. Comparing with an 'M' command, the parse and handling is more complex but both need to be splitted in parameters and respective sub-commands, to be queued in the software queue of commands (figure 4.9).

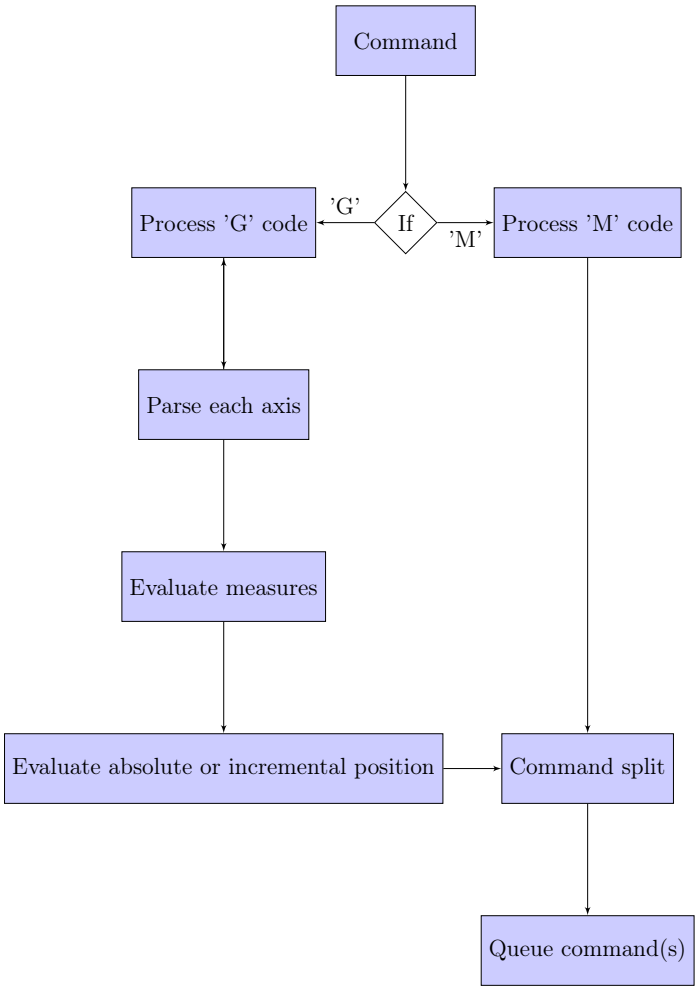


Figure 4.9: Flow of the software command parser

At firmware these commands are parsed and splitted by another agent which dispatches the orders to the hardware components. Written mostly in C programming language, firmware parser is known in 3D printing and by the 3D printing community as an interpreter and a CNC stepper motor controller [40] [41]. It is called *grbl*, and is a high-performance and a low-cost alternative to parallel-port-based motion control for CNC milling without extruder control. It has full acceleration-management with look-ahead planner and is able to maintain more than 30 kHz step rate delivering a clean, jitter-free stream of control pulses [41].

BEESOFT embeds a *grbl* interpreter due to its simplicity, efficency and community support. *Grbl* processes GCode blocks sent by the host and followed by a carriage return, ignoring comments in GCode and deleting blocking characters. For each command it returns an **ok** or **error** message. This logic is ensured by two internal threads running concurrently with the main thread (Thread A — figure 4.10) reading the port for GCode commands, parsing and sending them to the planner. The planner is responsible for the inclusion of the 3D printing hardware elements velocity (feedrate) and acceleration values completing the parse and command split, placing the job into a ring buffer. Thread B (figure 4.10) is interrupt-driven and is always working on the background, listening to this buffer and sequentially processing the

ring buffer events controlling the step motors, sending step pulses and direction bits to the stepper driver pins [40].

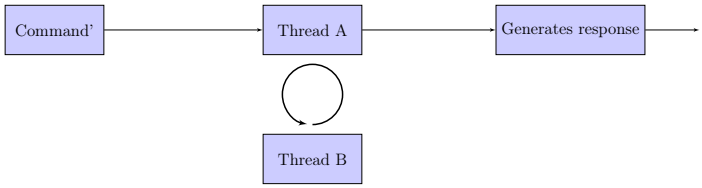


Figure 4.10: Flow of the firmware command parser

Chapter 5

User testing and results

5.1 Introduction

In order to assess the design and test the system to ensure it behaves as expected, we need to evaluate *BEESOFT*. Evaluation is a development life cycle process and must be done according to principles and prototyping techniques because it is easier to change a design in the early stages of development than in later stages. This evaluation can be split in evaluation by the designer or an usability expert (without final user interaction) and by an user who tests the actual use of the system [30].

One of the best and most popular ways to gather the maximum information is to observe users interacting with the system and analyse the environment variables such as user's facial expressions, their comments or screen footage. To provide useful insight into problems within an interface and if it meets user's requirements through their decision processes and attitude, we can use a Think Aloud method. This method collects the users description about what they are doing, what they believe is happening, the reason behind an action of a particular task and what they are trying to do [30]. This strengthens the usability tests providing large ammounts of information from many sources that illustrate well the system debilities.

Usability testing is conducted to find usability problems through the observation of users while they are using the software. These tests can be categorized by the development phase at which they are conducted — before development, during and after — and the formality of the testing method [29]. Tests can be done with mockups and story-boards, to test the design and simulating the use on paper, or they can be done with prototyping software to create virtual mockups of the application. After the development and in a head stage, tests are conducted over the product in a head development stage near the release. Basically, it is all about getting the results to replan iteratively the current design and methodology, solving design issues or to learn the weaknesses for future releases [29].

Formality in testing is the type of methodology where users are interviewed about the software, how they use it, how they like it and which kind of software they use. Usually this is done in front of a computer while users talk about what they are seeing and doing. The tester notes user reactions, comments and mistakes giving the participant progressively harder tasks over the course of the session. Other methodologies are task-oriented with a script the tester provides to compare alternative designs, or to determine an optimal value of a design parameter, or even to record user mistakes, and situations where they need help. These evaluations are mainly observational. According to Steve Krug ([26]) usability tests must be

done because testing is better than no testing and with an iterative testing process, a live audience analysis and a representative group of testers, we can find almost all the problems — Steve Krug also refers that it is more important having a small group of testers with several tasks iterations than a lot of testers and too few iterations. We get empirical evidence of what is working and what isn't, for a particular situation, understanding how it can become more intuitive and familiar, confronting our expectations and the difficulties that came across, allowing a map of changes and suggestions for new interface revisions [23].

5.1.1 Usability tests

The importance and the gains of user testing led to a testing and usability plan (appendix C.1) to be carried out by students in a Human-Computer Interface course at the University of Aveiro. This plan asks each user to complete a series of tasks while they are being observed by a tester who notes all important comments along the execution of the task, goal achievement, need for help, number of clicks, compliance time and ease of execution. After the session, each user also had to answer a questionnaire, rating his performance and *BEESOFT*.

Important and vital to the success of any user testing plan, users were selected with equal profile based on specific criteria to match the expected user population — similar age and level of education, experience with computers and technology domain [30].

When performing usability tests there are some problems that can be often found as users getting disoriented or confused about the user interface due to existing elements, the lack of them or if the funcionality is not clear. The excess of information can also degrade the understanding of the user interface [26].

Keeping this in mind, the usability tests were performed to understand what were the most fragile pieces of *BEESOFT*, especially where users failed to understand the purpose or to achieve the goal, what necessary elements were missing to enhance feedback and what errors they found. It was also pretended to gather the possible suggestions and observations about their experience during the test that could improve usability. The user script had major operations and screen understanding as the main focus, guiding users through various situations from importing a model and transformation operations to save, maintenance operations and print. Included in the various tasks were objectives such as understanding elements position, actions of buttons, screen navigation, resetting flows, the ability to learn quickly how *BEESOFT* globally displays, how it allows user freedom and how to access non-evident functionalities.

Tasks

The tasks to be carried out were relatively simple, to make it easier to analyse users performance. To test more thoroughly there were some difficult tasks involving modeling operations and screen navigation. Testing user adaptability to 3D concepts, their learning along the test realization and above all the clarity and simplicity of the interface of *BEESOFT*, was some of the parameters to evaluate. Each user had to complete 20 tasks within a given time window. The following are some of the tasks:

1. Load an STL file
2. Edit scene details
3. Save scene

4. Apply transformations to the models positioned on the bed
5. Undo actions
6. Set modeling parameters to complete a move operation
7. Use mouse control buttons to apply transformations in the same way as corresponding panels
8. Try to use disabled funcionalities to observe feedback given
9. Inspect the Preferences screen and understand its informations

This last task focus on evaluating the capacity of the user to extract qualitative information useful for modeling the object in the 3D canvas. This evaluation determines how successful the 6th task can be since it is necessary to unlock the height parameter to move the model to the coordinates asked with a Z positive position relative to the platform.

The first 5 tasks are simple and directed at evaluating some funcionalities, e.g. change model menus, enable/disable pickable state in the model to edit details on it or in the scene, understand how screen navigation works, how data is inserted and how it is processed, location of less used functionalities; the 7th task evaluates user understanding on how to move/rotate/scale/mirror each model using the mouse buttons without any visual information how to do it. This shows if it is clear the operation needs to be selected the same way by text fields transformations and if the buttons assigned make sense and are familiar to the common user. The 8th task allows the user to observe one kind of feedback given by *BEESOFT* when the 3D printer is disconnected, trying to understand if he/she is aware of the current printer state before clicking and the reason why it happens. It analyses the position and visibility of feedback elements avaliable in the main screen of *BEESOFT* — one for printer state and the other for common operations.

The tasks carried out by students in the HCI course were assigned to several pairs of users with a tester and a *BEESOFT* end-user. The results came from 30 students that formed groups of 2, i.e., 15 observers and 15 end-users and the choice was made randomly without any regard to the experience of each.

Collected data

During the session with the students, each observer had to register the user performance for each task:

- Number of clicks done
- Time spent perfoming the task
- Completion of the task — boolean value
- Mistakes made — quantitative by three ranges
- Got lost — boolean value
- Call for help — boolean value
- Difficulty felt by the user as judged by the observer

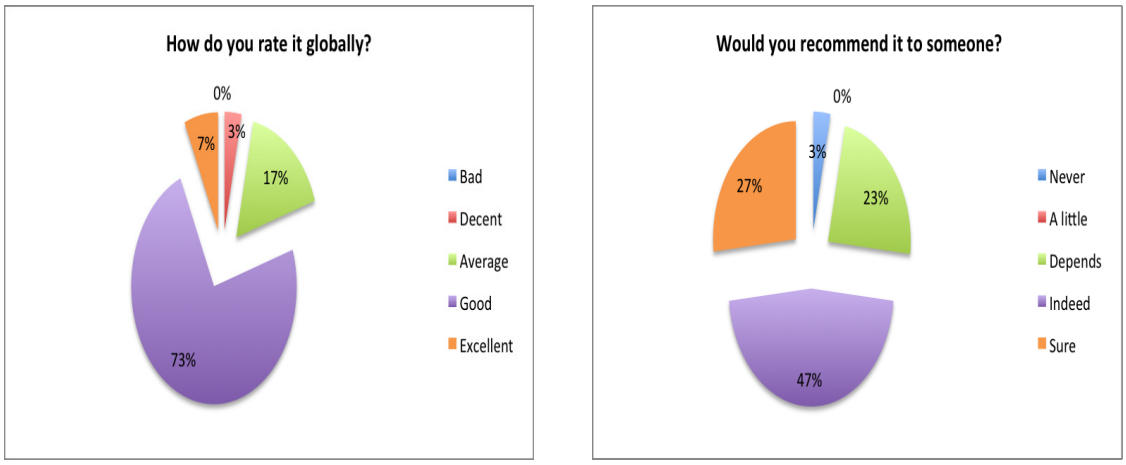
- Relevant observations

It was intended to gather relevant information about performance over each task, and if the users found each one easy and accessible or if they had problems with task completion. The need for help and disorientation with a probable high number of mistakes, tells if the task must be improved functionally or the interface elements that are used. Users were also asked to evaluate several functionalities, personal skills and interface characteristics (appendix C.1) of *BEESOFT* using a qualitative scale from 1 to 5 where 1 is a difficult task or not agreeing and 5 is an easy task or strongly aggreeing.

The results are illustrated by charts that use this scale.

Results

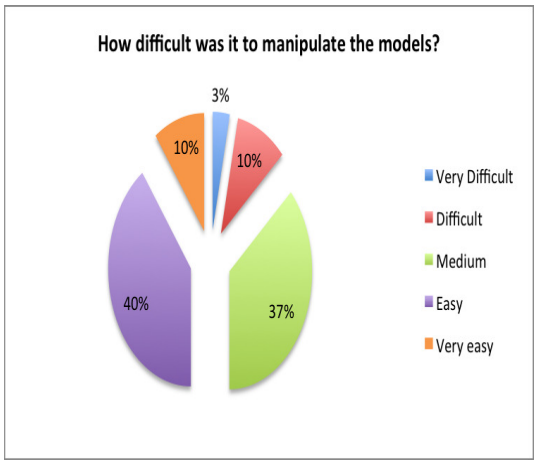
With the collaboration of 30 HCI students (25 male and 5 female) having little or no experience in 3D design, modeling or printing, the final questionnaire shows a positive reaction (figure 5.1).



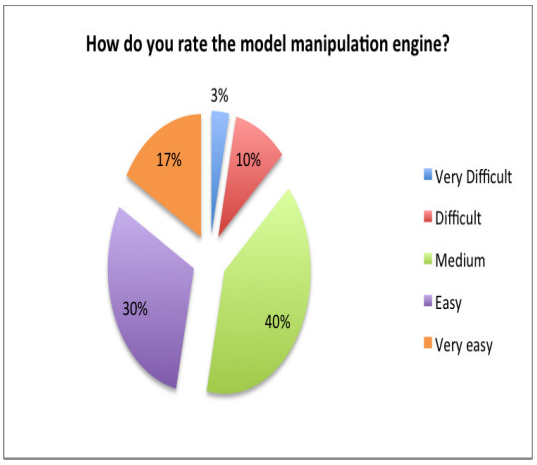
(a) *BEESOFT* overview (b) *BEESOFT* recommendation

Figure 5.1: *BEESOFT* evaluation

Even with a positive reaction some known difficulties regarding the hardest tasks were found and the simple and direct tasks did not cause any troubles in their completion. Users evaluate them as being really easy to accomplish, but the most complex revealed the expected results with common problems and behaviors between users (figure 5.2): lost for not knowing the behavior of some elements as introducing data in text fields for move and scale operations or when asked to scale with mouse buttons (figure 5.3a); making mistakes for not understanding what were the dimensions of the bed available on the 3D canvas. The coordinate system in use or the actual position of the model. Users sometimes assumed a incorrect behavior from their software experience preventing any attempt to try new and exploratory ways to conclude the task.



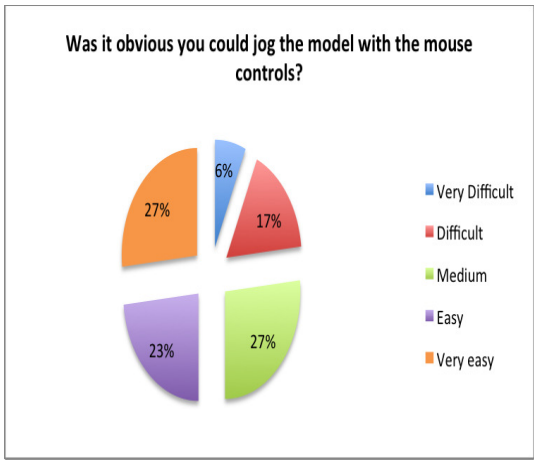
(a) Difficulty of modeling operations



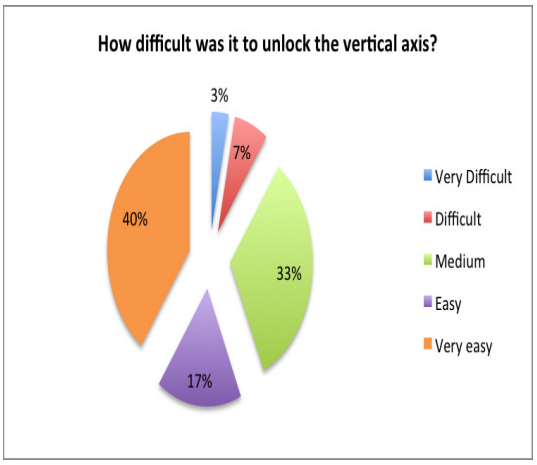
(b) Model manipulation engine overview

Figure 5.2: Model operation and engine evaluation

Although all tasks demanded work on the scene, results show users failed when trying to use previous knowledge to overcome barriers, such as moving a model to a Z blocked position (figure 5.3b), even when a previous task, introduced the corresponding screen for them to understand. This revealed lostness and tremendous disorientation, forcing them into a lot of mistakes in search of what could be wrong.



(a) Modeling with mouse controls



(b) Rate of the model manipulation engine

Figure 5.3: Difficulty of modeling auxiliars

The results of these particularly demanding tasks clearly show the difficulties they felt, meaning the user interface must be improved to enhance *BEESOFT*'s usability on the main screen — user and tester comments are useful in that way. Additional comments provided by the users and the testers reveal the need for elements that can improve experience in modeling, such as a locking artifact to apply transformations to all axes simultaneously, information about the actual position of the model, the existence of the lock height funcionality near the modeling panels for easier access and the existence of the coordinate axes in the 3D canvas.

Some behaviorial aspects were also considered problematic as the data submission in text fields where all modeling fields could have a submission button to apply transformations and thereby be consistent from tool to tool. The 3D canvas's lack of feedback when overlapping two models was another detail mentioned as the creation and rename of a category, that should be part of the scene details panel and not the gallery (chart 5.4).

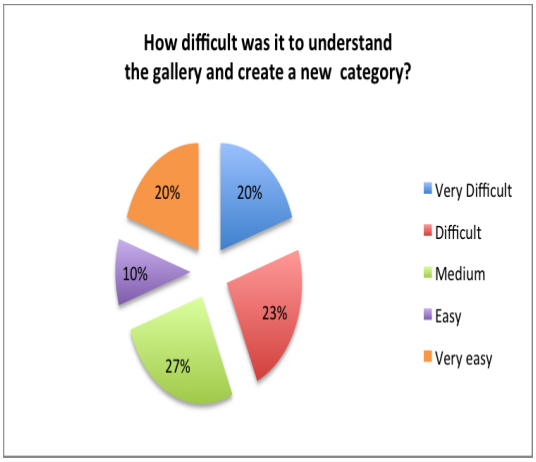


Figure 5.4: Gallery overview

In order to better understand these results, we will use a box-plot to assess the difficulties felt by the users in the most crucial tasks. This graphical representation of small data sets, that do not lend themselves to histograms, shows the data set's lowest value, highest value, median value and the size of the first and third quartile. It is a good option for allowing several simultaneous comparisons as in Figure 5.5, where we will analyse how tasks results are linked.

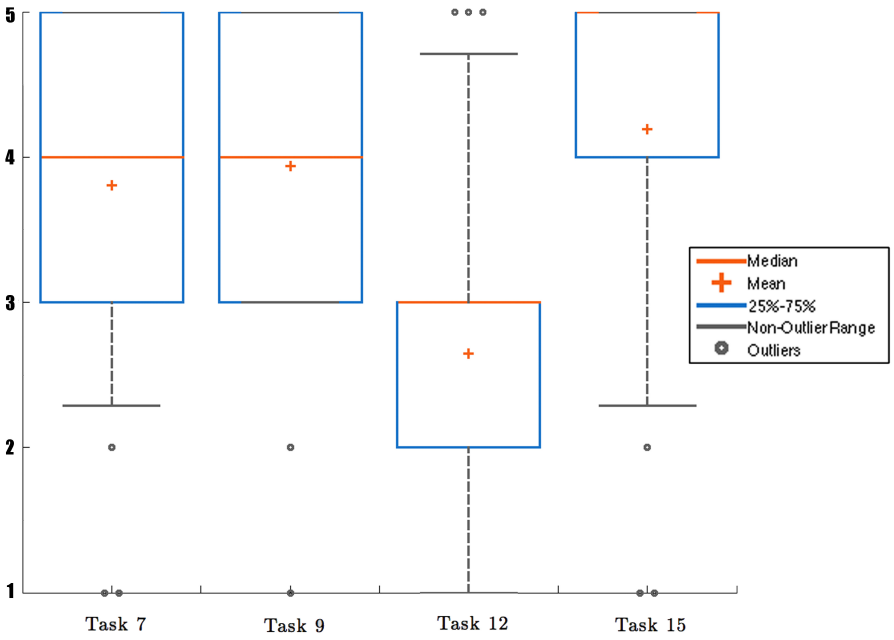


Figure 5.5: Modeling easiness

Figure 5.5 shows the box-plot for the qualitative assessment of the user performance when using modeling tools for tasks 7, 9, 12 and 15. This plot depicts the easiness of those tasks for move and scale operations with individual axis transformations (task 7 and 12) and transformations to the entire chosen model (task 9 and 15).

Observing figure 5.5, we notice task 12 had worse results with a lower median value which means users had difficulty to finish it, considering it a hard task. These results were expected because users needed to unlock the vertical axis in the Preferences screen (visited on task 6) and the performance over the other tasks, that were increasingly detailed, blocked the capacity to understand what was wrong and how to avoid the error.

The results discussed previsouly revealed the necessity to reposition this funcionality on the main screen for easy access and visibility. The median value of task 7 and the disparity of values show there were problems moving the model due a cause already explained above, the lack of information about the selected model coordinates. When scaling, in tasks 9 and 15, users felt the need of a lock transformation element to lock all axes to a single transformation operation. These two tasks are similar but led to curious results and wide easiness approval values: the median value from them differs significantly because users had more experience on task 9 (better performance) and therefore greater ease of use but on task 15 some users ran into more obstacles.

Figure 5.5 also shows that different tasks produced positive and negative results far from the median and range of values which could indicate some 3D experience or inability to perceive the request. The incapacity to understand how to apply what was proposed and the user interface issues damaged user experience causing several mistakes inhibiting the user to recover. It indicates that, at tasks 7, 9 and 15, some users felt more difficulties than the rest rating worse the modeling easiness — proved by the outliers indicating ratings out of the standard (e.g. 1 and 2 marks).

This difficulty and the complexity of the user interface revealed some inherent problems such as disorientation and the number of mistakes done, to complete the task. Figure 5.6 and Figure 5.7 show the percentage of users that found problems modeling and how confusing it was as the number of associated mistakes.

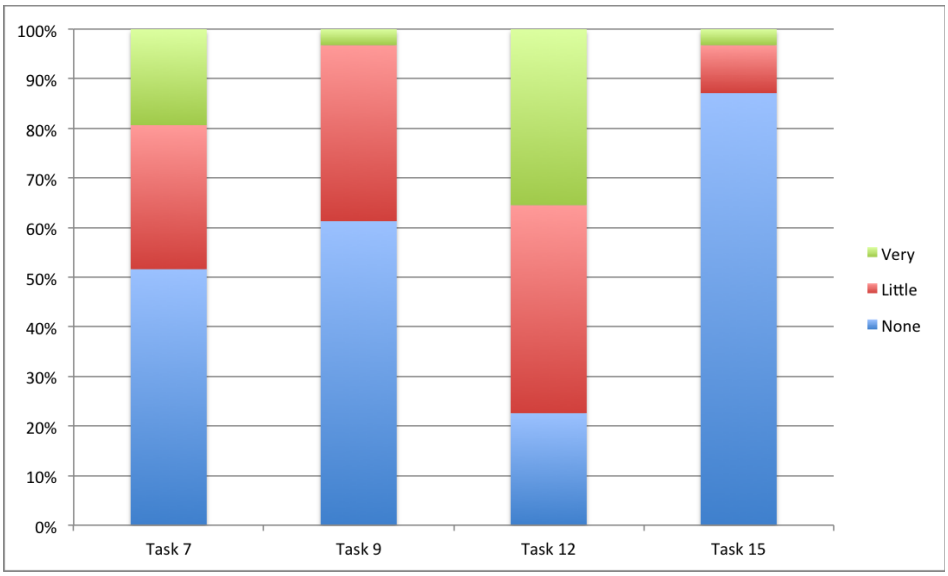


Figure 5.6: Disorientation in modeling

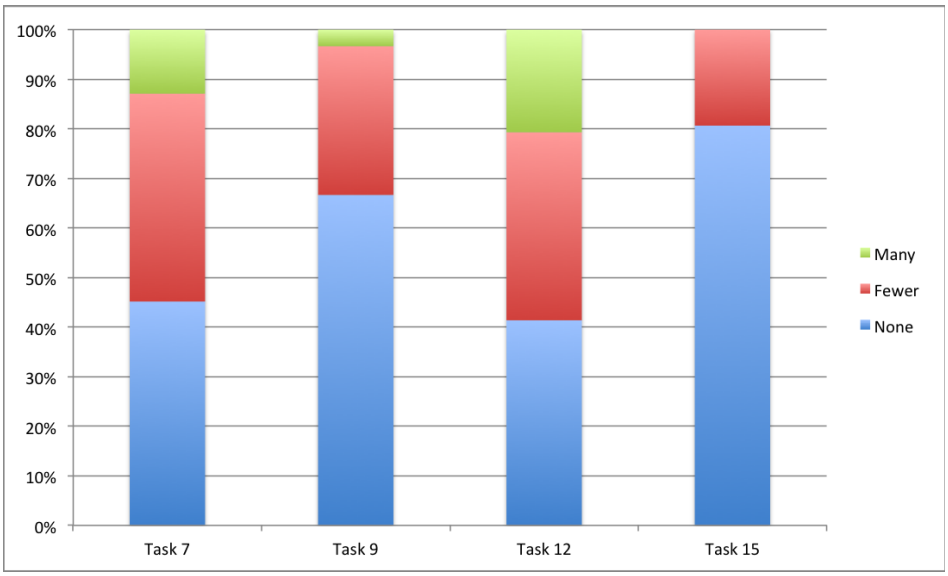
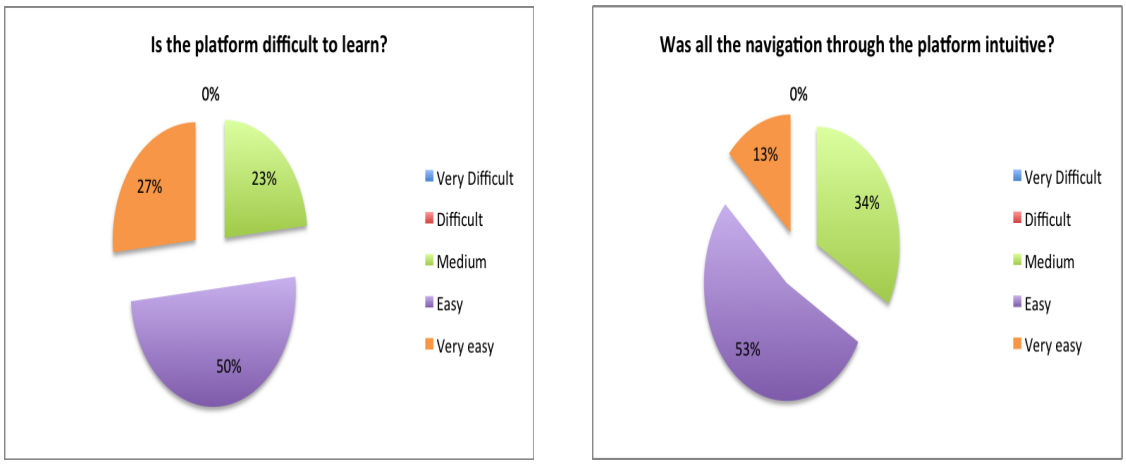


Figure 5.7: Mistakes in modeling

These figures prove task 7, 9 and 12 were the most problematic leading to more mistakes than task 15. The task that evidently caused most difficulty was task 12 with a significant percentage of users who committed several errors because they got lost. The location of the vertical height lock in Preferences and the lack of accessibility with the handling tools proved to be the reason for these difficulties — users could not turn around the problem.

Task 7 found other problems in the user interface. Most users were not able to understand which were the coordinates of the selected model and what was the direction of the coordinate axes in order to properly apply the transformation. Besides that, they felt the lack of an artifact to lock all the text-fields to apply an equal value on more than one field — this was noticeable as a disorienting factor causing some confusion. Task 15 revealed users were not fully able to understand the behavior of the text-fields and the same transformation in more than one axis in scale operations but it was not a big problem according to above figures.

These difficulties also became evident in the user's questionnaire at the end of the session (figure 5.8), showing a large percentage of users who considered the application to have some moderate weaknesses in terms of intuition and ease of use.



(a) Difficulty to learn

(b) Difficulty to navigate

Figure 5.8: Navigation intuitiveness

Usability tests are very helpful and contributed in realizing there was a need to correct some of *BEESOFT*'s functionalities as well the need for some minimum usability improvements. These have a low implementation cost but provide a much better experience of use. As previously mentioned, usability tests do not enable us to find all the problems which is why heuristic evaluations often take place to cover a greater number of encountered problems [42].

5.1.2 Heuristic evaluation

Usability tests are indeed important to find the major problems when the application is still under development or at a beta stage. Other methods such as heuristic evaluations can be done in the early stage and in an existing design so that detected problems can be fixed. Originally, heuristic evaluation was developed as a usability method for those who had some knowledge about usability principles not involving potential users in the tests([43]).

According to Jeffries ([44]) and Desurvire ([42]), expert heuristic evaluators find more problems than any other evaluation technique, including usability testing [45]. This proves that evaluators trained in usability issues and heuristics principles find more problems than inexperienced users but these evaluation techniques aren't accurate by themselves, and need to be used together: users's lack of experience and the naivete using a novel application proves to be better at finding unknown or minor problems besides the major ones.

A heuristic evaluation by trained evaluators was conducted to find potential problems in the *BEESOFT* interface, taking advantage of their skills and knowledge. This evaluation requires 3 to 5 evaluators to find nearly 75 percent of usability problems, where to each criterion is assigned a severity value for each problem encountered in the user interface, according to Nielsen and Mack's scale. Each evaluator goes through the user interface several times inspecting the various elements, comparing them with recognized usability principles ([46]):

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and standards
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help and documentation

Heuristic evaluation is not set to fix usability problems or to provide a way to assess the probable quality of any redesign but to explain each observed problem against the principles listed. Severity ratings can be used to provide an estimate of the need for additional usability efforts, using factors such as the frequency the problem occurs, its impact and persistence, but it is common to combine all of them into just one, to facilitate prioritizing and decision-making ([47]). Basically, it is a 4 level scale: 0 rating means the evaluator doesn't think there is any problem with the interface; 1 reflects a cosmetic problem; 2 a minor usability problem; 3 a major usability problem and 4 means a usability catastrophe.

The various problems encountered (see section C.1.2) aim to highlight the weaknesses, evidencing improvements to the current user interface to become more efficient and user-friendly. The improvements ensure a cohesion between them and an operational flow that allows the user to use the application as a whole. The revision cycle of the design and implementation of these improvements comes in several iterations so that it is properly resolved and the integration of new functionality or services may be performed in a modular way and on a stable application. Let's take a look at the results of the heuristic evaluation.

Results

With the collaboration of 6 HCI students (6 male) as evaluators, with the basic knowledge and academic training for the heuristcs evaluation, the results convey a positive reaction to *BEESOFT*, with few cosmetic and minor usability problems, but globally a problem-free user interface based on good principles.

Regarding the maintenance center and its wizards, the study revealed some light contradictory practices of good principles: cosmetic problems regarding user-control and freedom, consistency standards and help and documentation. Firstly, some of the elements available

should have more visibility, helping users to easily learn the elements position and where are the most important functionalities. Secondly, there is a screen where users actions can damage the 3D printer, but *BEESOFT* fails to avoid it — this summarizes an usability weakness that doesn't eliminate the error by checking for confirmation before committing. It is a violation of the error prevention principle. The remainig heuristic evaluation underlined faults that can improve *BEESOFT* if solved: restricting the user to a maximum scale limit (adjusted to build volume), a hover color on models and better detailed feedback message telling which model is outside the bed would prevent errors for cosmetic-related problems.

Other minor usability issues regarding save on quit, extra hotkeys and expanded clickable areas, can improve flexibility and efficiency of use. Quitting the program showed scene alterations weren't being saved and that is a serious usability problem — there is no alert, a dialog to save or a message causing the user certain doubts. Additional hotkeys to select model operations menus and more text elements as a clickable element were other minor usability issues the study revealed to be easily implemented, and improve flexibility and efficiency of use.

Final observations detailed two other major usability problems when applying transformations: only the last modification can be undone which is a problem for user-control and freedom. Undoing all changes solves this problem. The ability to undo all changes strengthens user confidence to go on editing all models on the bed. Scene edition is also constrained if we try to transform all picked models at the same time — results also show the impossibility to operate on all picked models, which is considered a consistency principle violation and a major usability problem.

5.2 3D services interoperability

Modern 3D printers are able to print colored 3D models at high resolutions in different types of material but the creation of a detailed, printable 3D model is a costly and skills demanding process requiring a significant amount of effort. All 3D printing applications as *BEESOFT* are responsible for loading that model, designed in a CAD application or produced by existing hardware equipment such as a 3D scanner, and printing the rendered virtual model.

A 3D scanner is a device that analyses a real-world environment to collect data to create its shape and if possible its color. This shape data is converted into a 3D model that can be loaded in *BEESOFT* or other similar software. Advancements in 3D imaging processing techonology, that captures extremely high resolution mesh and color information of objects, brought a lot of 3D scanners onto the market to be bundled with 3D printers. Adding this hardware module to the existing systems, created the need for an auxiliar software to aid the scannig process, enabling the export in a readable STL file format.

As an example, let's take a look to *MakerWare*, the *Makerbot* digitizer software that runs with their 3D scanner, to make a quick analysis of the functionalities it comprises (figure 5.9).

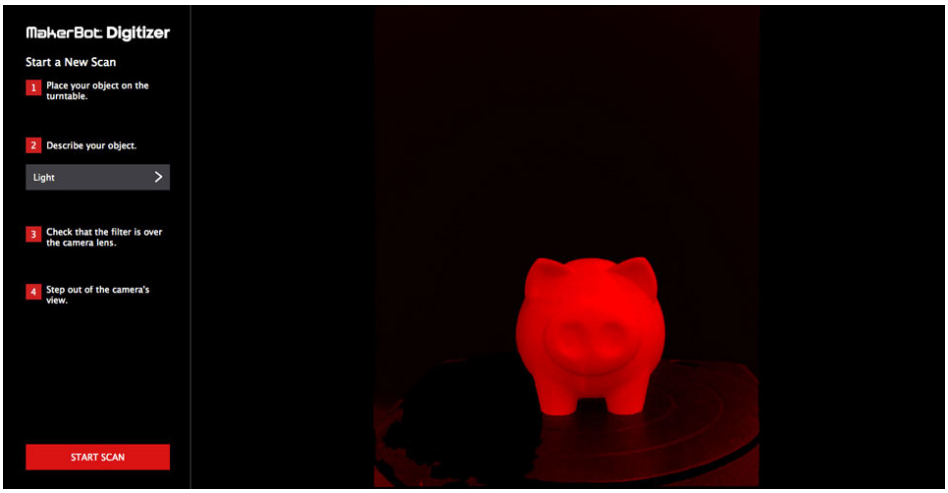


Figure 5.9: *Makerbot* 3D scan software [1]

This software allows setting the scan name, calibrating the camera, choosing a preset to set the model's level of detail, cropping the model after scan is complete to export the portion we want and an online sharing functionality. This solution is properly developed to this 3D scanner and any of the components, hardware and software, can be used outside.

This technology, despite the numerous existing solutions, rises many issues and barriers when attempting to 3D print directly which lays the software effort to fix or overcome those problems. The resulting models are not well suited for 3D printing as they are in general incomplete, not watertight, unsegmented and in many other possible aspects, not optimized for cost-effective fabrication. Therefore, after the scannig process, it is necessary to edit the model by mesh cleaning to actually get a model worth printing, eliminating excess points and meshes that are time-consuming trimming unnecessary areas. With this assembly completed, it may be necessary to correct errors the model has, fixing holes and non-manifold edges [48].

The need to apply model corrections after the scan limits how a system (3D printer and software) can operate and be upgradable with a 3D scanner unless it is developed by the same manufacter, as it happens with the software in figure 5.9. *BEESOFT*'s user interface addressed before, did not have a scanning functionality and support for a 3D scanner, but to evaluate the system performance, it was used a *GO!SCAN* and *SENSE* 3D scanner to generate a model to be printed in the *BEETHEFIRST* (see figure 5.10, 5.11 and 5.12).

[1] Source: `http://www.makerbot.com/makerware-for-digitizer/`

Figure 5.10: 3D printed model with low level-of-detail (right)

Figure 5.11: 3D printed model with high level-of-detail (left)

Figure 5.12: Comparison of the level-of-detail in both 3D printed models

Figure 5.10 and figure 5.11 shows a 3D person model, with different level of detail and with polygon errors — non-manifold. The file exported after the scan, besides the incorrect measures, cause *BEESOFT* some performance problems, when rendering after import and for each transformation on the 3D canvas, due the number of polygons to be redrawn. The printed results show this fabrication process (3D scan and 3D print) is still less accurate when capturing some human details as the eyes, hears or others that are barely perceptible. This can be a mesh problem that implies a third-party software to fix them or hardware limitations of the scanner as a inefficient imaging algorithm. A reduced number of polygons between both models mean printed pieces will have different layer textures and quality as it happens in figure 5.12. Even if the person body characteristics are well represented being possible to recognize who it is, layer quality is lower in the model at the right.

This test helped us understand there are several types of problems associated with the scanning process as the hardware limitations, scanning algoritms's low efficiency and mesh errors that need to be fixed by other software for mesh cleaning. The possibility to have a scan functionality in *BEESOFT* is then limited by these constraints and by the availability of a 3D scanner with a framework that could allow job requests between software and hardware within the *BEESOFT*'s user interface.

Chapter 6

Conclusions and future work

In this chapter, an overview of the work carried out is done, taking into account the objectives achieved. Some ideas for further work are also detailed. 3D printing emerged some years ago and has been used in several market segments with different solutions and fabrication processes, such as the Fused Filament Fabrication that mainly uses thermoplastics. A model is produced by extruding continuous beads of thermoplastic material layer by layer, that hardens immediately after the extrusion from the nozzle.

To control this extrusion and all of the print session, there are 3D printing applications that enable users to apply operations over a 3D model and print it. These applications help the user to connect and configure the 3D printer, and to print the imported model designed, most of the time, in some type of CAD software. Actual 3D printing software differs significantly from one company to another, offering the main functionalities and fewer new characteristics advantageous to this technology. Most of them have clunky, complex, non-explanatory and buggy user interfaces that have not been enhanced as much as the hardware, but with some improvements in the algorithms and edition of the printing profiles.

A 3D printing application called *BEESOFT* was developed to suppress all major problems in other applications through a simple, intuitive, responsive, self-explanatory and functional user interface to help each user to 3D print. *BEESOFT* was developed to guide each user to the most important task in a 3D printing application: printing. This goal needed detailed user interface planning in terms of design and functionalities, as well as in terms of being easy to use and self-explanatory. With the main common functionalities available, the problem was how to design a new user interface that would allow the user to configure and print in fewer clicks.

6.1 Conclusions

To quickly enable the user to print, the main screen was designed to provide access to the major necessary operations and to allow viewing the most recent printer and software informations. With the 3D canvas at the center of the screen and the modeling operations and information on both sides, the user can easily set up the print scene and access the print configuration through the available button. The reduction of screens and the centralization of the print set up operations, all in the main window, sets *BEESOFT* positively apart from other 3D printing applications, that are somewhat confusing. To achieve this simplicity, screens became complex, requiring all components, such as the 3D rendering engine, the

3D models processing engine and the communication protocol, to work in parallel with an advanced operations management so that the user will only see understandable feedback, a simple static design and functional interface. Since the user does not need to constantly configure the 3D printer, the configuration screens were transformed into wizards to facilitate goal achievement, within a controlled navigation, with self-explanatory information and user action control agents to avoid unnecessary and dangerous actions. This decision improved *BEESOFT* user interface's simplicity, separating the printer setup from the main flow of 3D printing, based on what the most frequent user actions are. This controlled environment implies a huge complexity on actions processing and user interface elements management to avoid deliberate or non-deliberate attempts to break the iterative flow of each of the wizards.

The option of reducing the visual complexity in terms of the user interface elements and the number of screens for the printer configuration became costly, due to the complexity behind each screen and the *BEESOFT* module that supports a full and functional integration of them all. Users like interfaces to be clear, intuitive and easy to use, not having to force them to consult documentation and allowing safe exploration and habituation to boost their experience. These characteristics make *BEESOFT* a single case in the 3D printing community, where none of the other applications achieve such a minimalist design and functionalities integration for the end user.

Once all the aspects related to the user interface were defined, it was necessary to plan and build the 3D engine, the 3D models processing engine and communication protocol to ensure the integration and the overall functioning of the application work cross-platform. This fundamental precondition demanded a market study of the current state of other 3D printers and frameworks documentation to perceive capabilities, debilities and implementation information. It is very important that the 3D engine and the 3D processing engine be cooperative and responsive, to rapidly apply the transformations, process models and start printing using the data streams channels of the communication protocol. The frameworks available, and those chosen, were not very complete and tested by the community, so it was a very difficult journey to get the *BEESOFT* final working environment because it was necessary to re-implement or implement from scratch the necessary functionalities and behaviors, especially in the 3D engine and the communication protocol. This provided better functional and robust modules to this application and the 3D printing community. The 3D models processing module integration was easier with simple changes and with little side effects on other modules — this integration embedded an existing and very used framework called *Skeinforge*.

To achieve the objectives and in order to improve the user interface to ensure the application fits the purpose for any 3D printing goal, we carried out some application evaluation tests that relied on different techniques to find the most problems possible and to get a global appreciation of the application. The usability tests were useful to realize how different types of users manage to use *BEESOFT*, with no 3D experience in modeling, and what difficulties they would feel in the several screens and their operations. This observation showed that the user interface, in some situations, was ambiguous and confusing when attempting to apply some transformations over the 3D canvas. The detailed and demanding tasks revealed also that some *BEESOFT* screens are not protected against some user actions and the performance of the overall system has issues causing a bad user experience after some time. These assessements were very useful to understand how modules's performance individually worsens the rest of the application. The heuristic evaluation helped to understand what were the usability problems, set against recognized usability principles, using a scale according to the severity of the problem — it helped us to see if the problem prevented the application from being

usable. The main benefit of applying these evaluations was to understand the fragile parts of the application, and to conceive a list of problems to be thoroughly analyzed, and whose problems are included in a plan to improve the application.

Besides fixing the problems found, to improve *BEESOFT* usability, it was really important to test the application with other 3D services, specially a 3D scanner, to assess the possibility to integrate them on the user interface. Using the 3D scanned results of a real person in *BEESOFT*, allowed us to understand the engine bear with difficulties rendering 3D models with mesh errors and with a lot of polygons, even after some mesh cleanning using third-party software. The printing process revealed also the slicing engine detects those errors, and how the algorithm tries to handle them to ensure a printable piece. We concluded a 3D scanning process, is a heavy fabrication process that requires cleanning and fixing model's mesh, and the integration of an existing scanner hardware would necessarily need auxiliar software for that. This undermines the ease and ability to integrate a service like this on the user interface of *BEESOFT*.

6.2 Further work

As future work, improvements to the printer and its software may be introduced, fixing the problems found with the tests and improving *BEESOFT* performance and usability.

The problems found (see C.1.2) with the usability tests revealed various sets of problems but none severe enough to tell *BEESOFT* was not usable. After an analysis it was possible to categorize the problems into two types of problems, one list representing those more crucial, with high impact in the *BEESOFT* robustness, and other that considers design and usability problems, with less impact in user experience. From all of the problems (C.1.2), we underlined the three most problematic of each list to help estimate the severity, for future work.

These are the two lists, with crucial and less crucial problems:

Crucial

1. Inability to undo more than one operation
2. Absence of model auto-resize when importing, if models exceed the maximum printing volume
3. Absence of a maximum limit when calibrating the *BEETHEFIRST* in height, to avoid colliding with the nozzle

Less crucial

1. Absence of confirmation dialogs when creating a new scene, deleting files in the gallery or quitting the application
2. Inability to apply the same transformation value to all three axes at the same time
3. Difficulty to understand if a model is selected in an invalid volume state

Crucial means the user is likely to commit errors, damaging user experience and ruining the ongoing progress, while less crucial means user interface needs elements that serve important information, elements or feedback during a task.

The application designed and developed in this thesis has numerous opportunities for future improvement, but some really depend on how we want to use *BEESOFT* and with what final purpose. Taking previous lists into consideration, firstly it would be necessary to fix the crucial problems due the severity and how they could affect user experience, preventing them from completing tasks. Consequently, the less crucial problems would come next, to improve *BEESOFT* usability, providing better and accurate feedback and better means to enhance user experience reducing the cost to perform each operation.

On a small scale, we have to fix previous problems and, at a large scale, *BEESOFT* improvement in terms of usability and functionalities and the integration with frameworks that can enhance 3D printing experience on the Web or with some hardware devices. Replacing the 3D models processing engine with a faster one that gives better results would definitely improve the speed with which the user achieves the print, the correction of a model's structural problems and extra functionalities, such as optimizing printing courses to print faster, achieving the same quality.

Another improvement would be the integration of *BEESOFT* to work with the new Microsoft file format (3MF), allowing each user to print via Microsoft services to *BEETHEFIRST*. Talking about services and Web capabilities, other possible *BEESOFT* gains would be a wireless support connection to the 3D printer, sending jobs via a cableless architecture. Since prints may take considerable time and users may need the computer to keep on working, implementing the autonomous functionality, with a module that supports GCode data transfer to the SDCard to print without cables, would be a major gain also.

BEESOFT can definitely be a better 3D printing application with optimized functionalities and architecture. If those goals are achieved, it will be a prominent application on the market.

Appendix A

User Interface Design

This appendix contains the draft for the mockup user interface and the corresponding use cases for the addressed screens.

A.1 Main Window

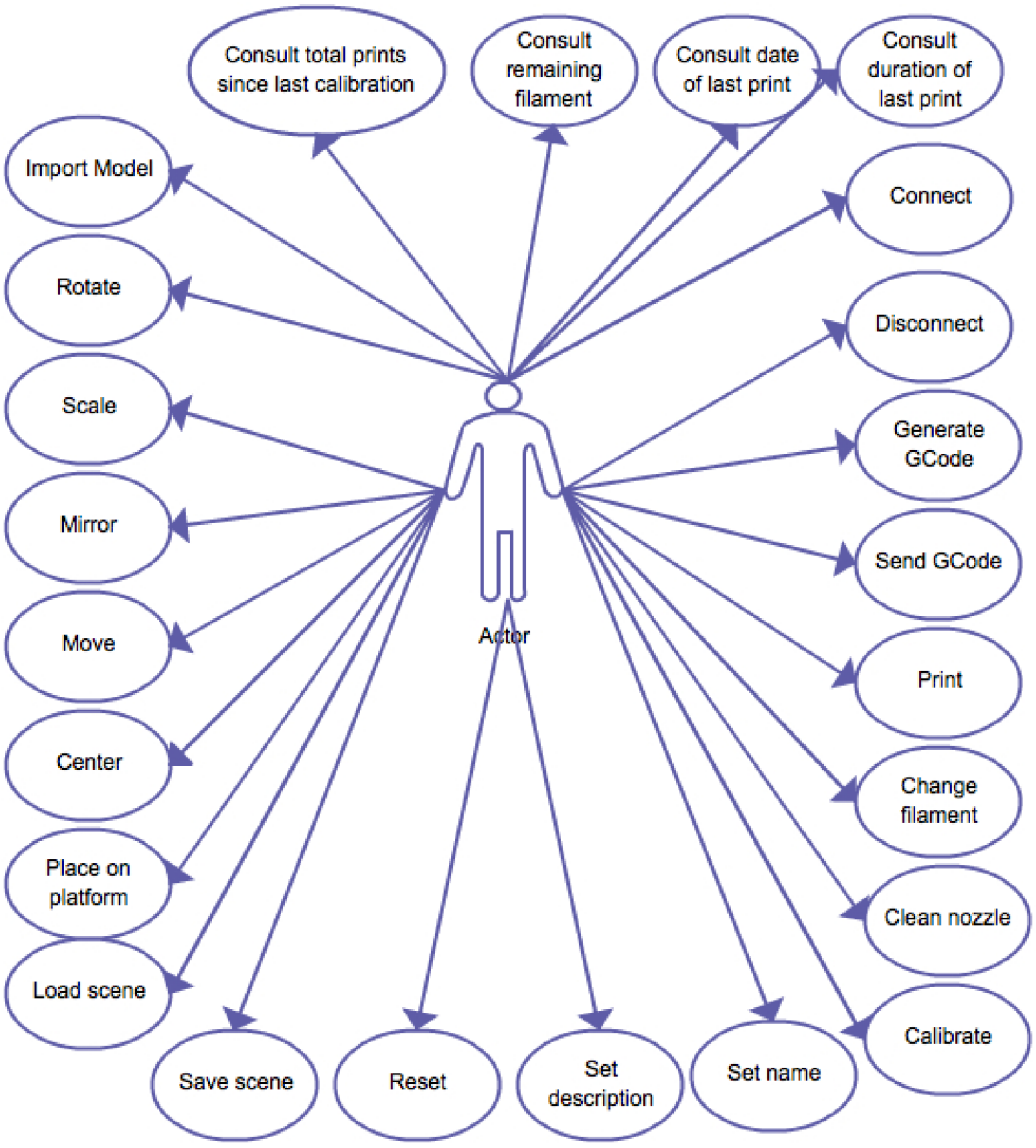


Figure A.1: Main window screen use case diagram

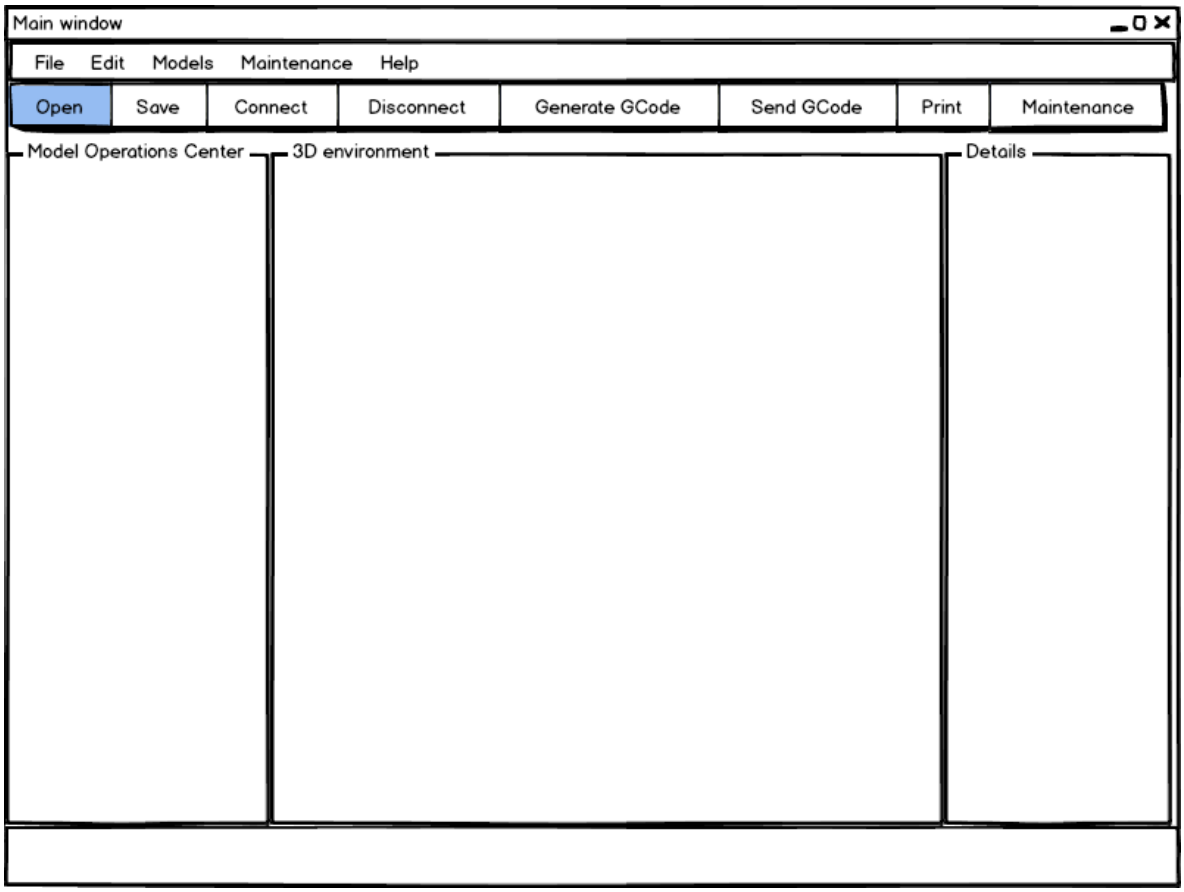


Figure A.2: Main window screen draft

A.2 Models operations center

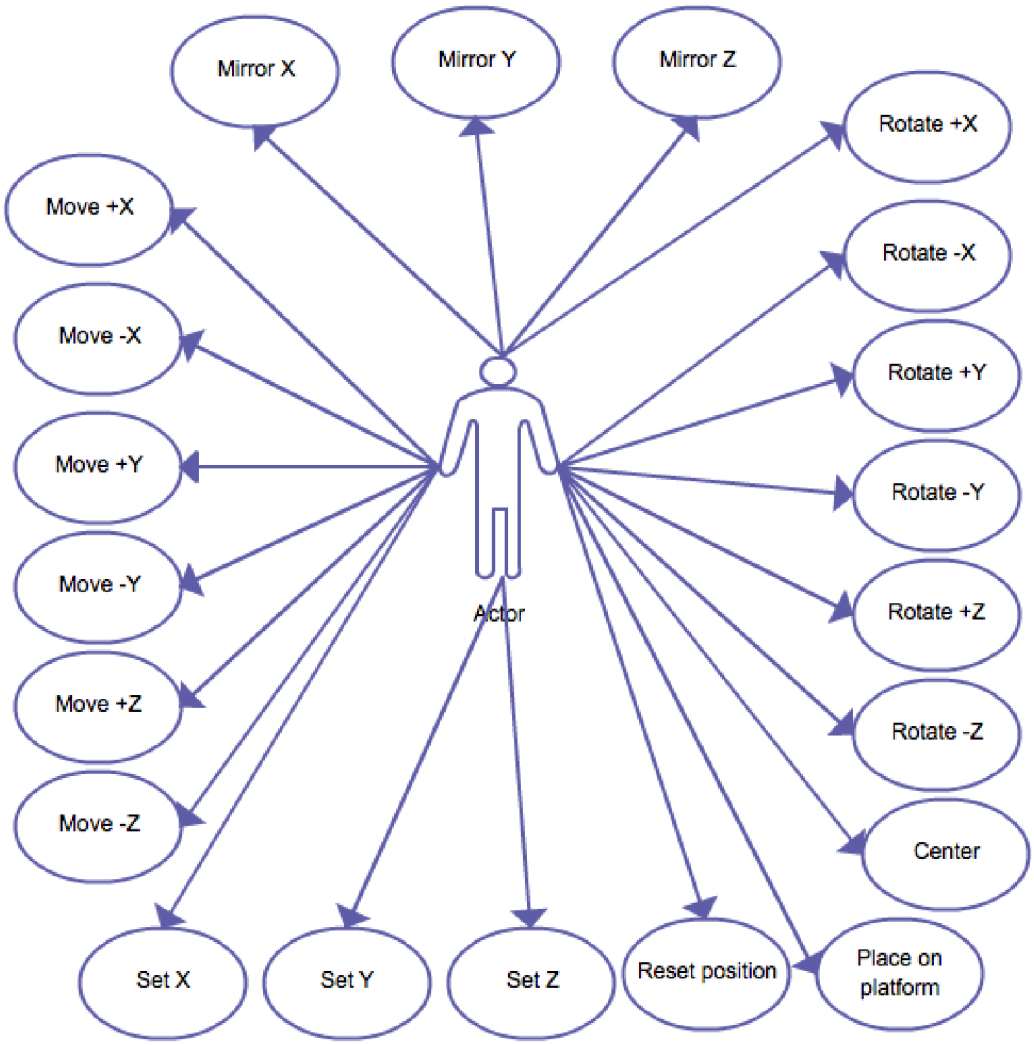


Figure A.3: Models operation center screen use case diagram

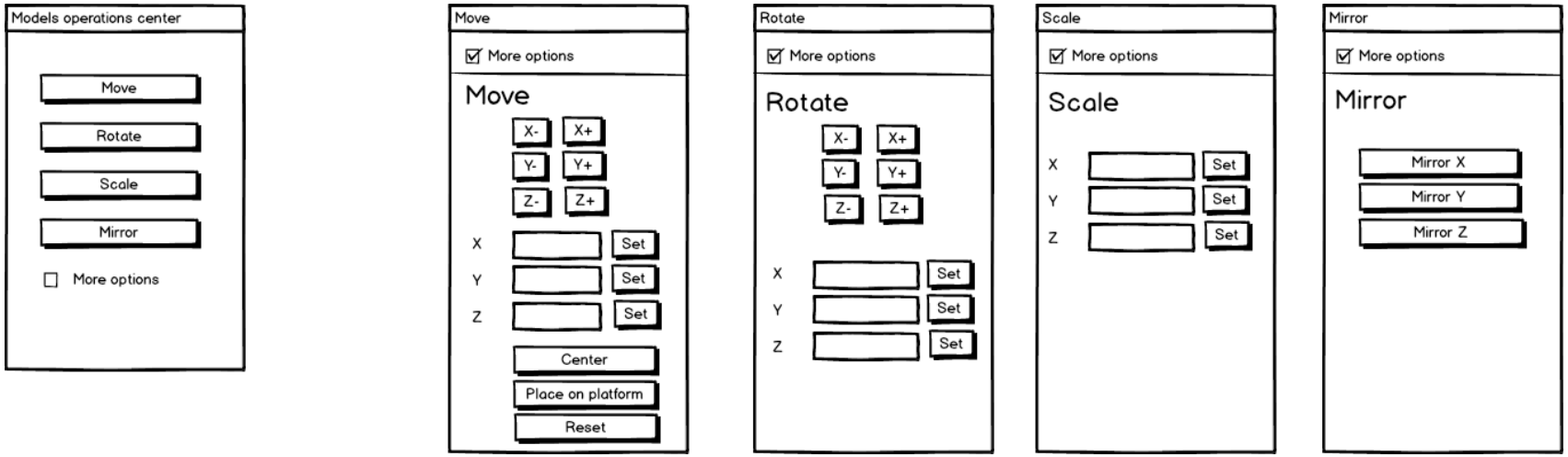


Figure A.4: Models operations center screen draft

A.3 Scene details

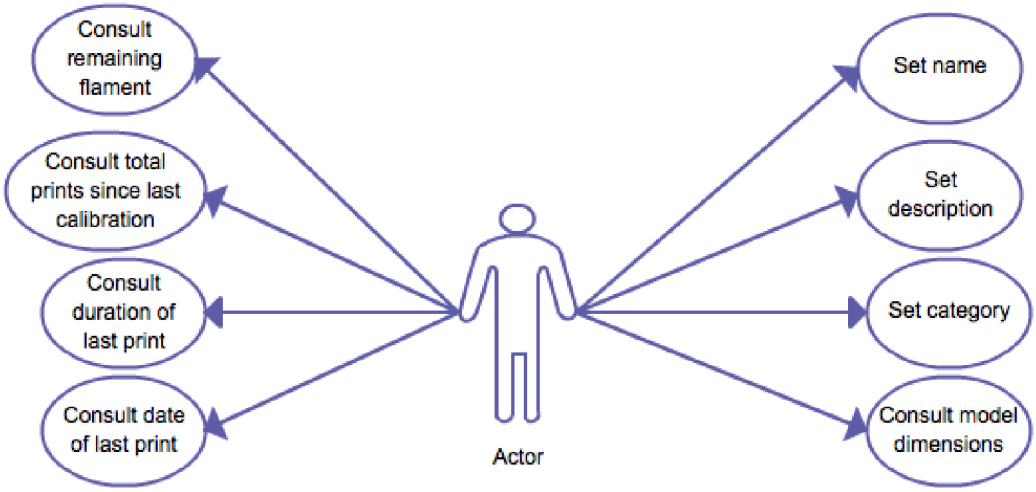


Figure A.5: Details screen use case diagram

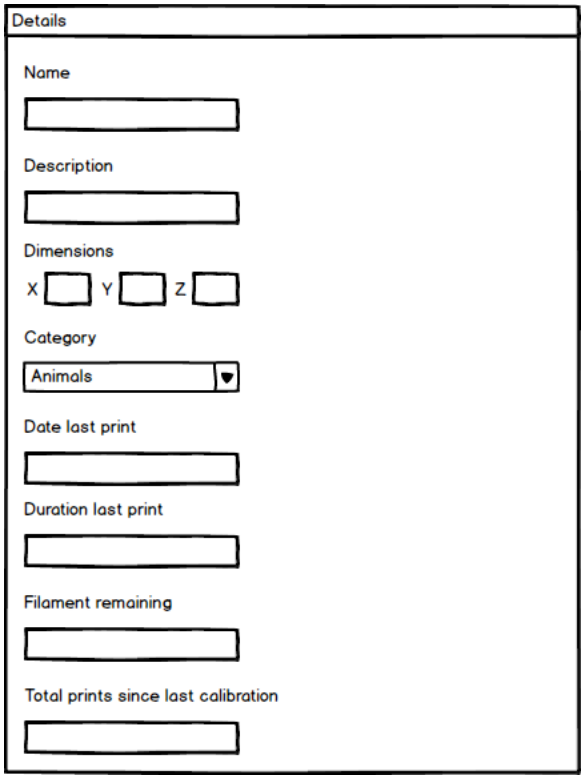


Figure A.6: Details screen draft

A.4 Filament wizard

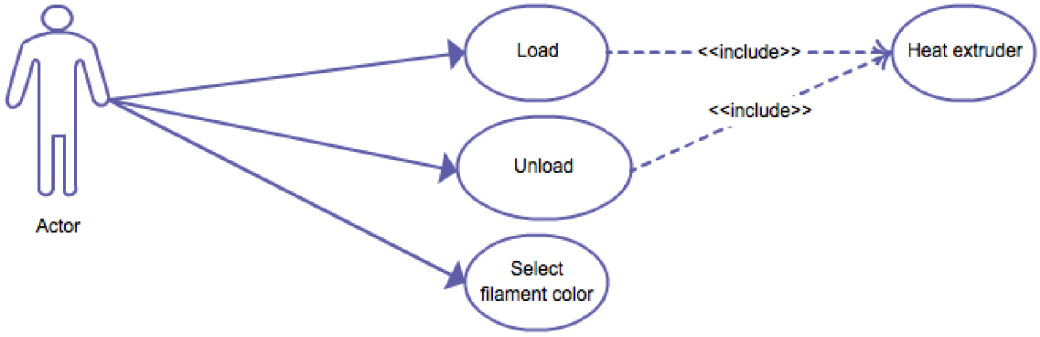


Figure A.7: Filament wizard screen use case diagram

Figure A.8: Filament wizard screen draft

A.5 Nozzle wizard

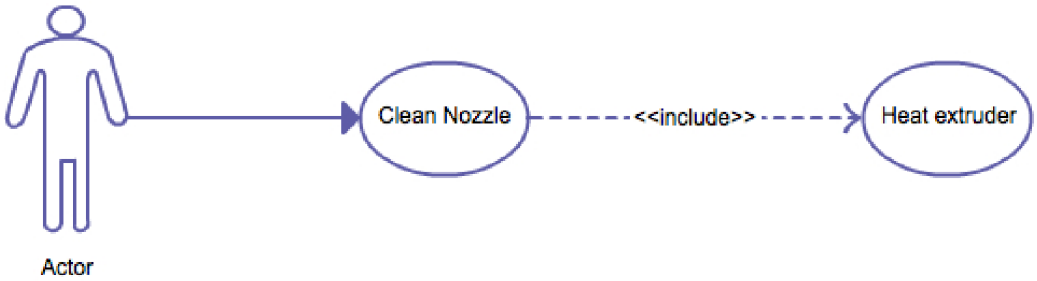


Figure A.9: Nozzle wizard screen use case diagram

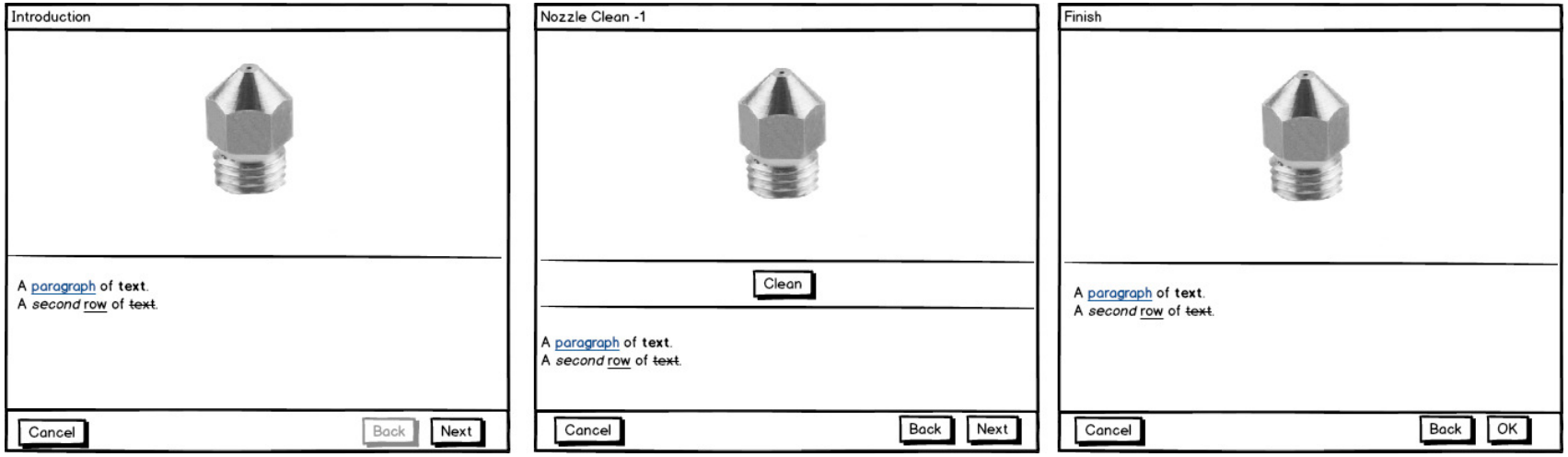


Figure A.10: Nozzle wizard screen draft

A.6 Calibration wizard

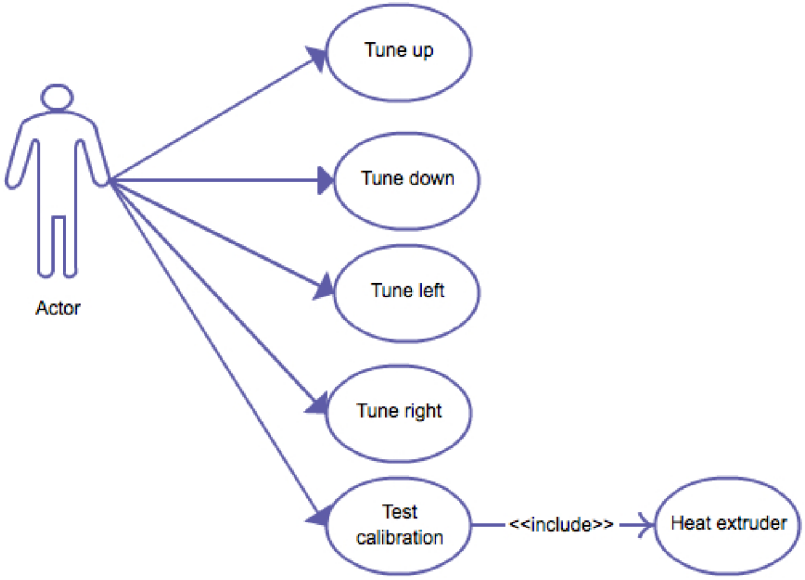


Figure A.11: Calibration wizard screen use case diagram

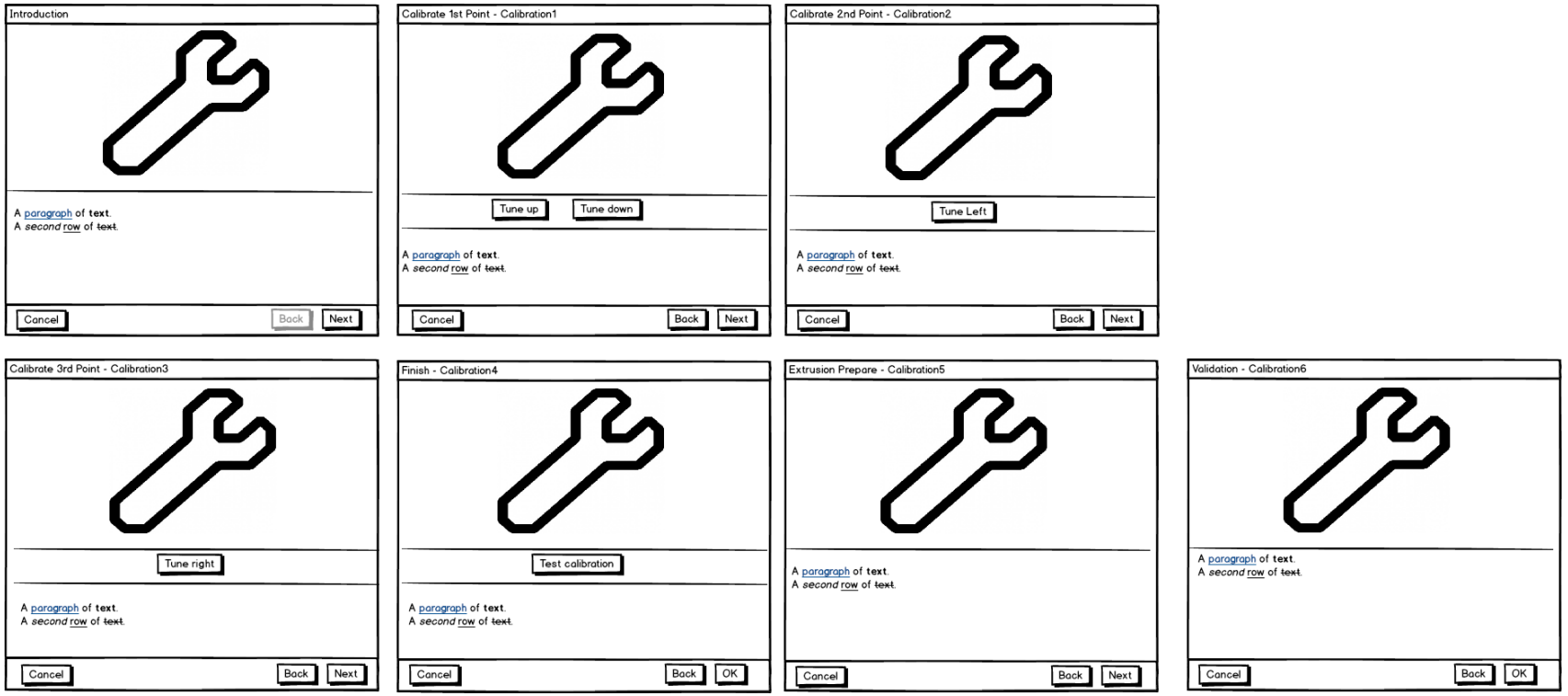


Figure A.12: Calibration wizard screen draft

A.7 Preferences

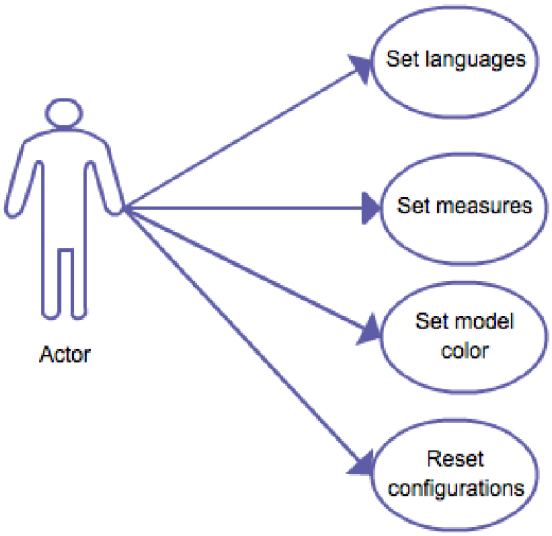


Figure A.13: Preferences screen use case diagram

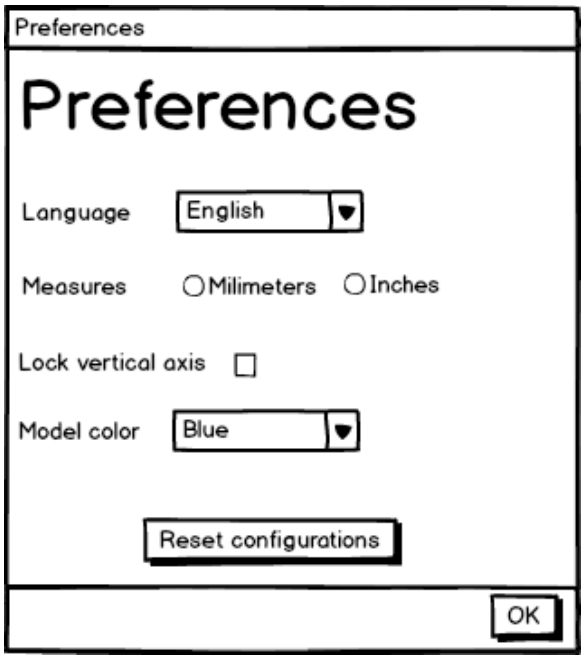


Figure A.14: Preferences screen draft

A.8 Print

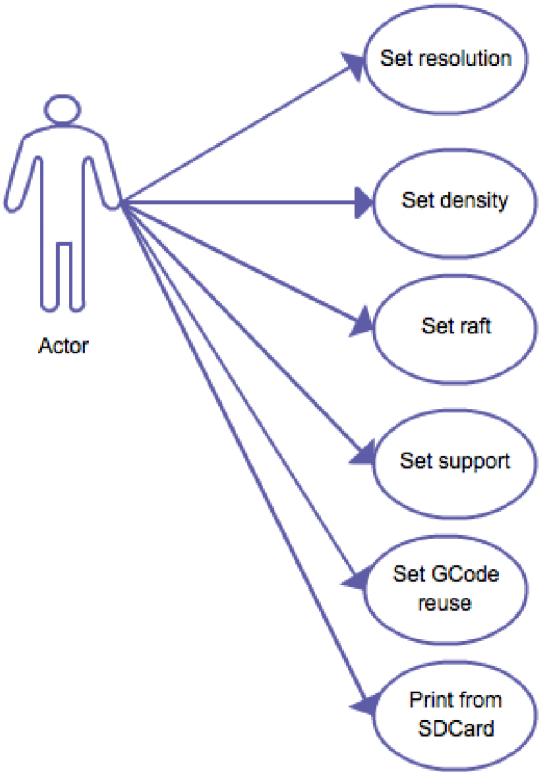


Figure A.15: Print screen use case diagram

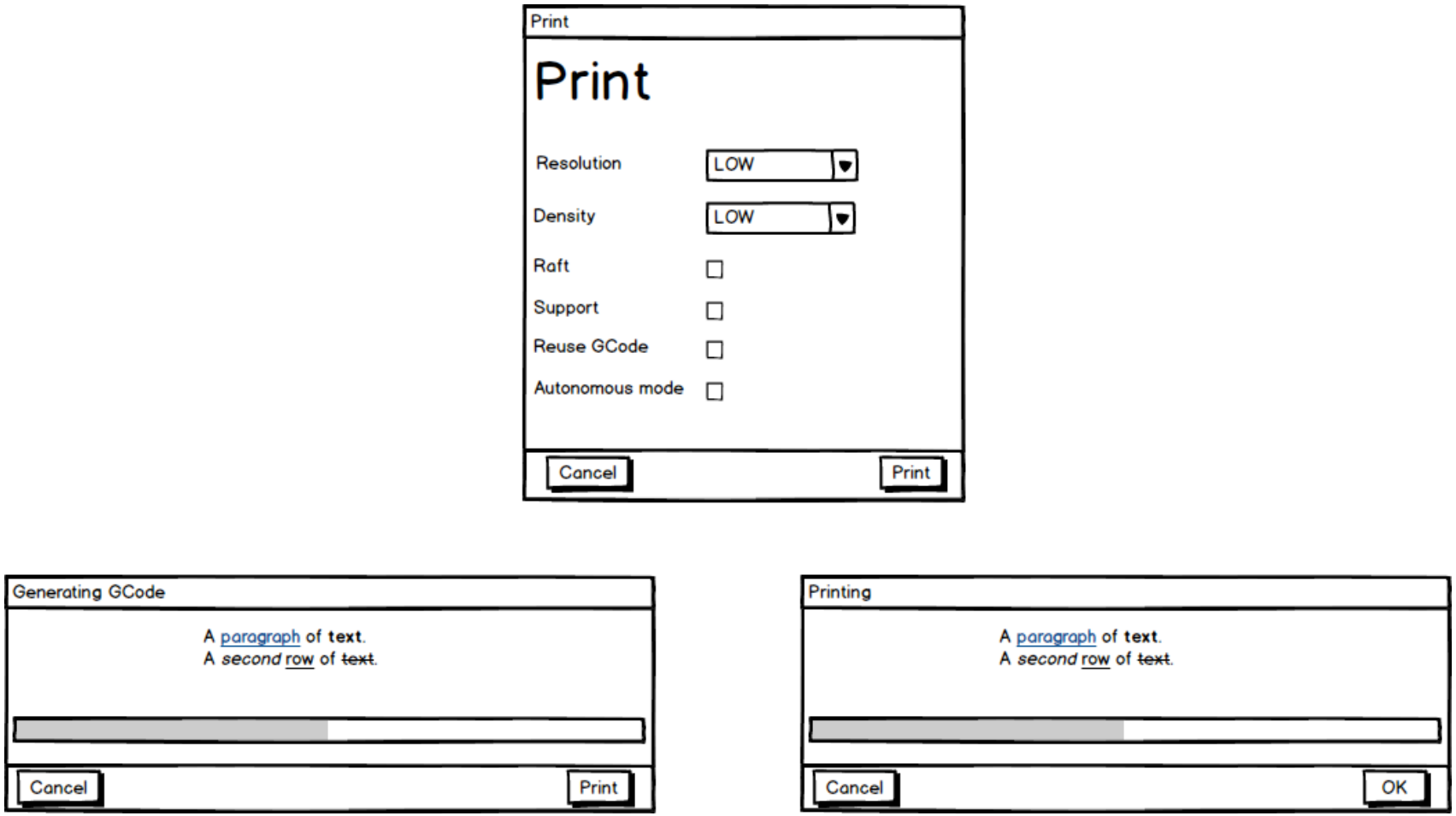


Figure A.16: Print screen draft

A.9 Buttons bar

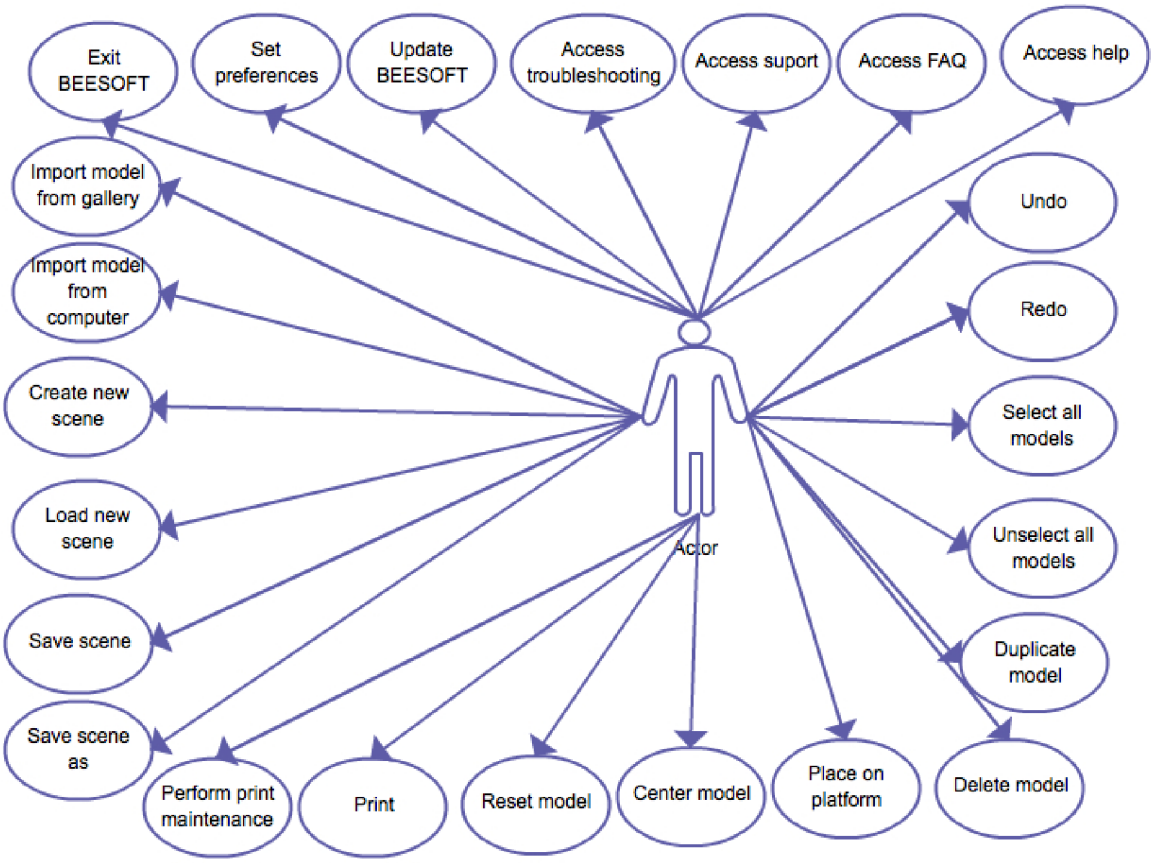


Figure A.17: Buttons bar screen use case diagram

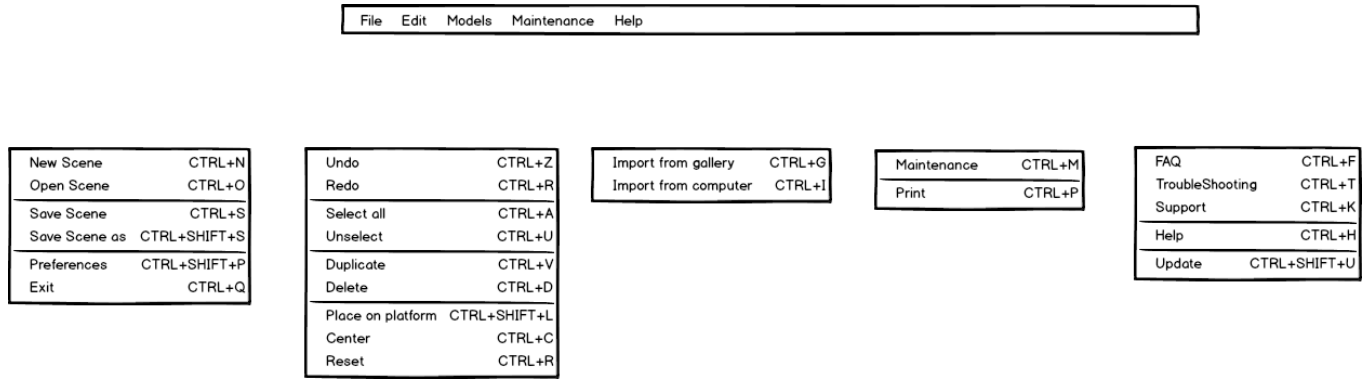


Figure A.18: Buttons bar screen draft

A.10 Help

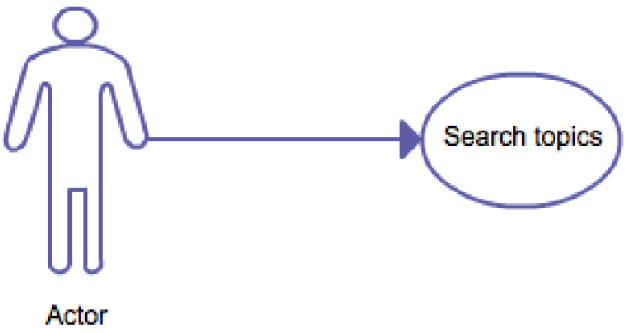


Figure A.19: Help screen use case diagram

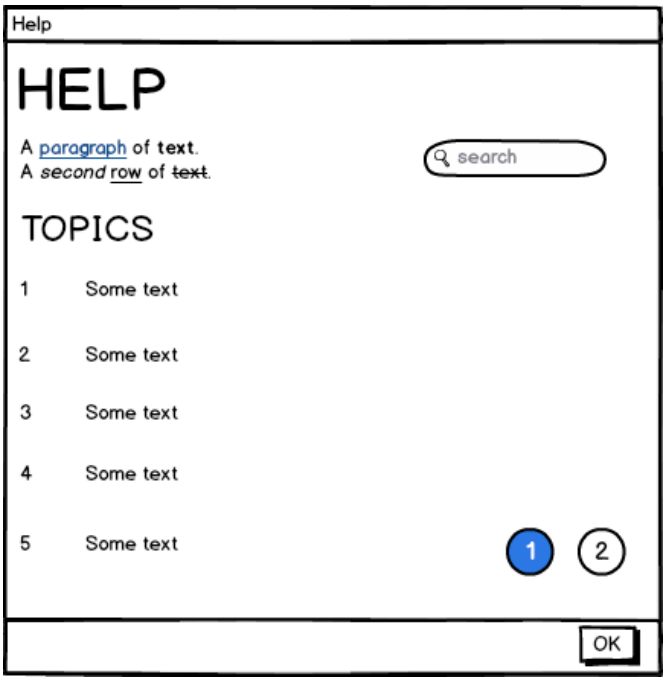


Figure A.20: Help screen draft

A.11 Update

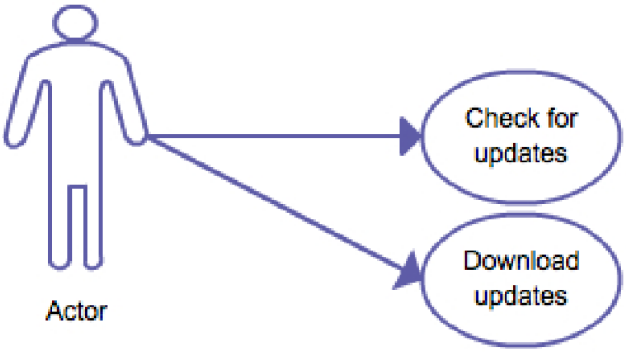


Figure A.21: Update screen use case diagram

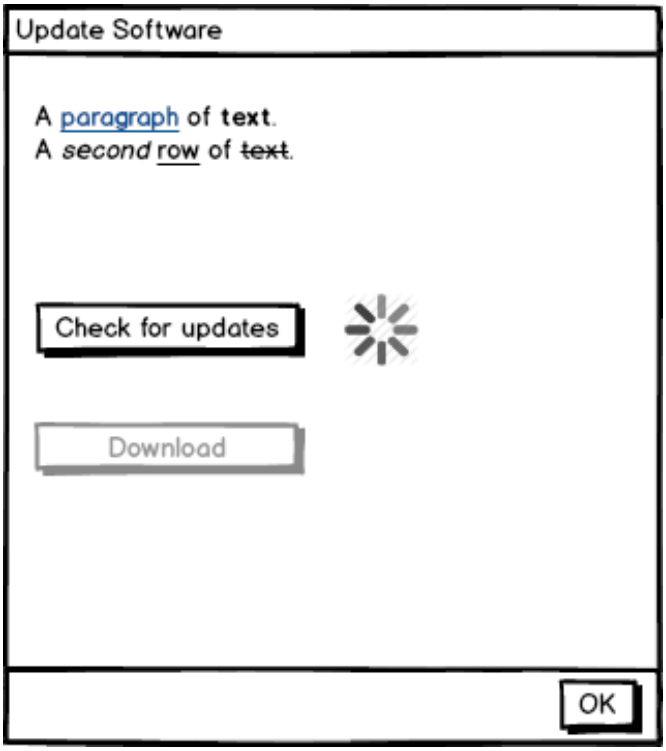


Figure A.22: Update screen draft

A.12 Gallery

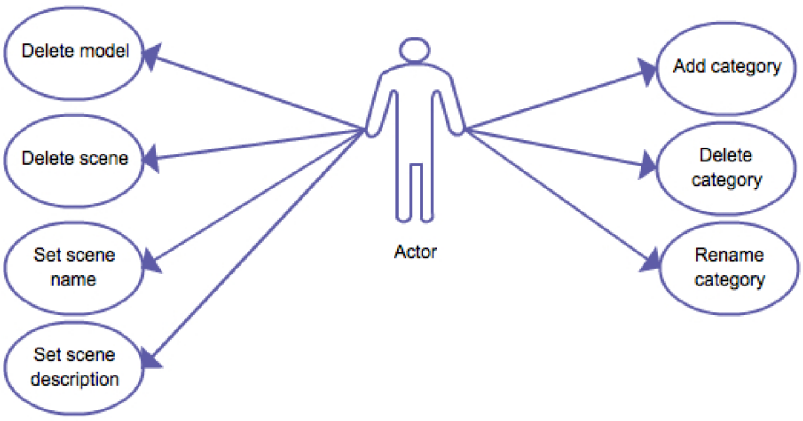


Figure A.23: Gallery screen use case diagram

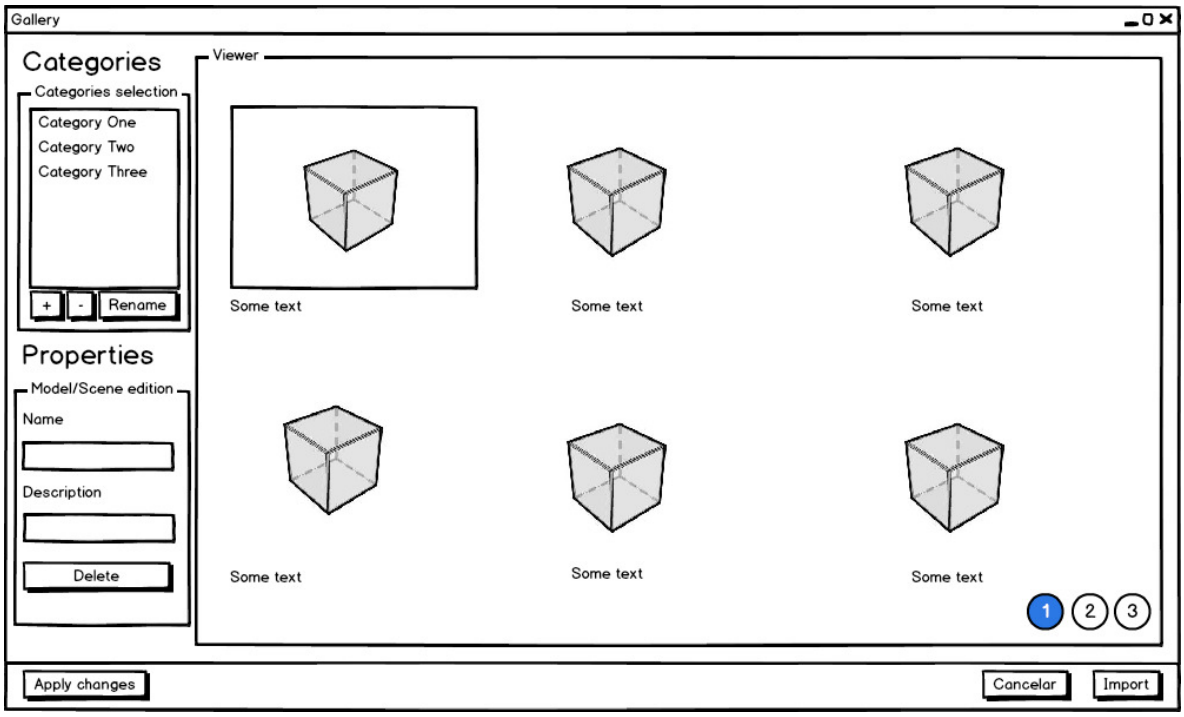


Figure A.24: Gallery screen draft

Appendix B

User Interface Implementation

This appendix contains the implementation results for the mockup user interface for the addressed screens.

B.1 MainWindow

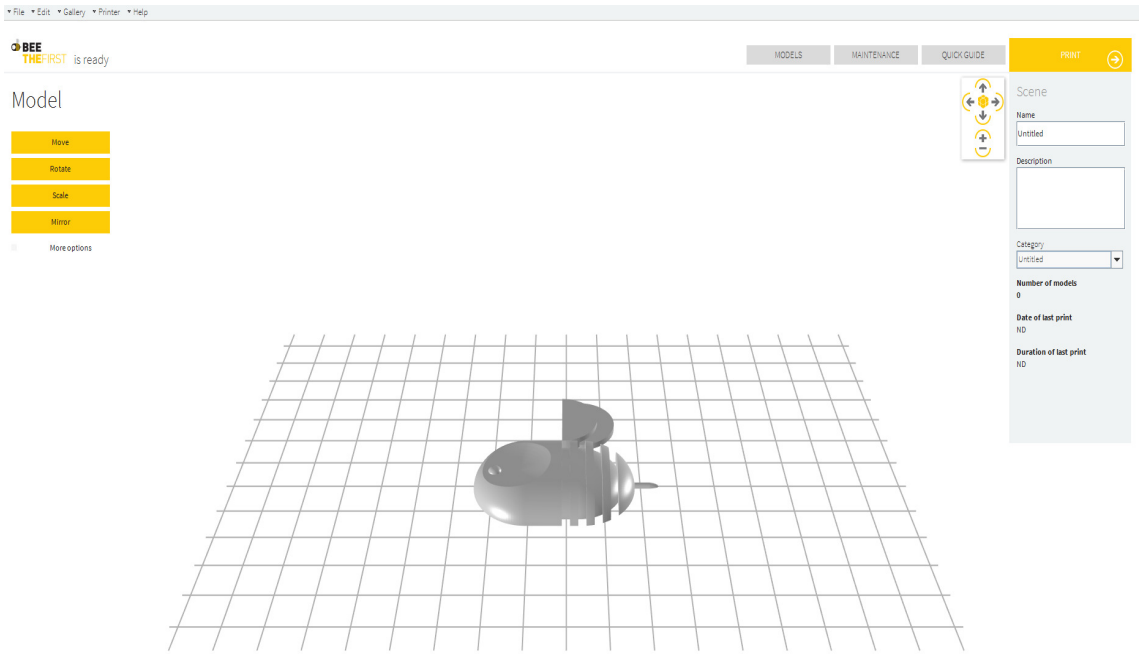


Figure B.1: Main window screen

B.2 Models operations center

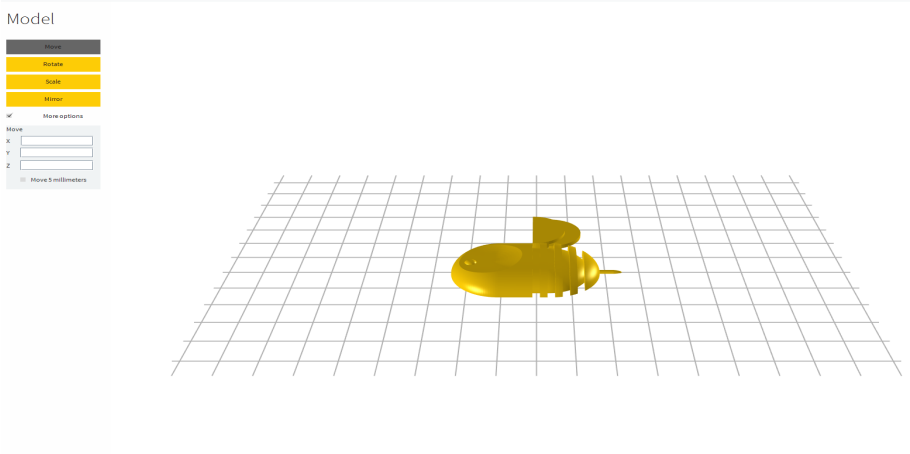


Figure B.2: Models operation center screen — Move

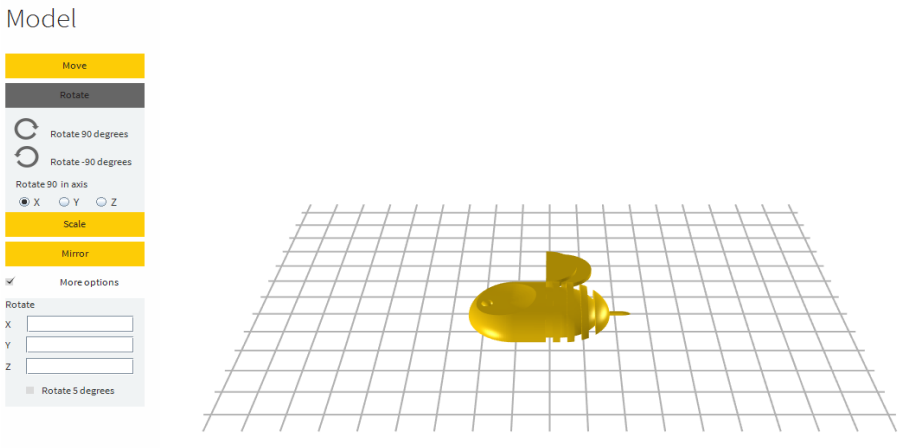


Figure B.3: Models operation center screen — Rotate

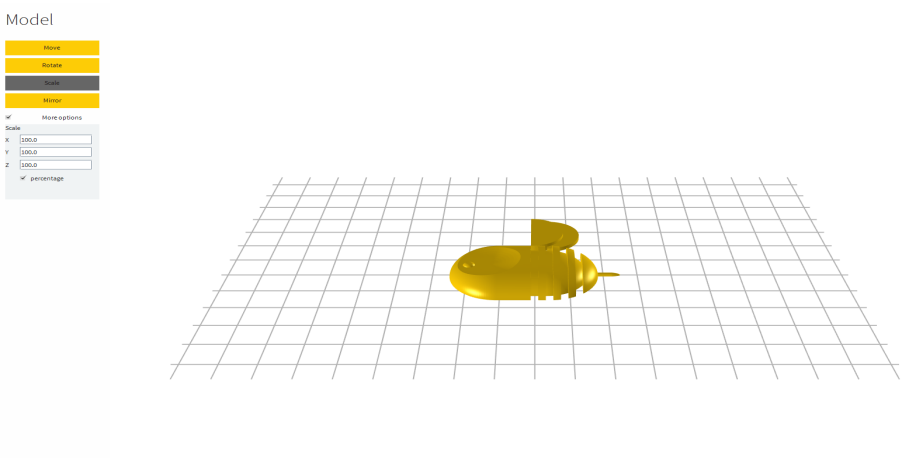


Figure B.4: Models operation center screen — Scale

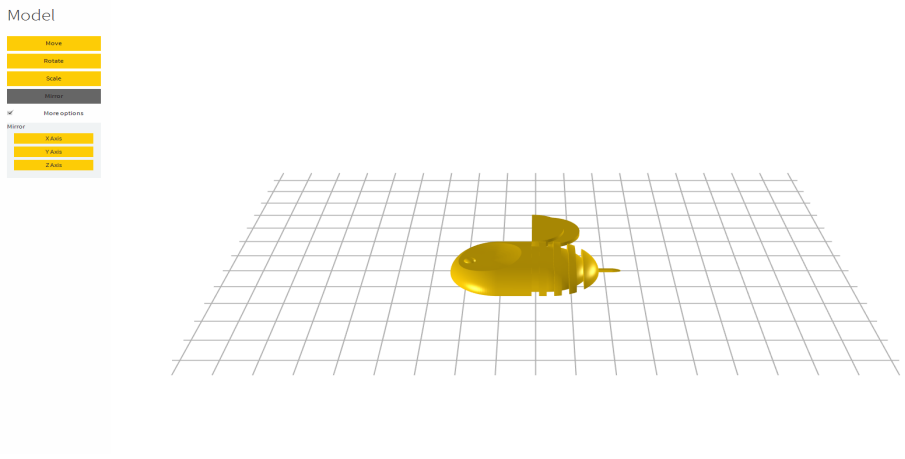


Figure B.5: Models operation center screen — Mirror

B.3 Scene details

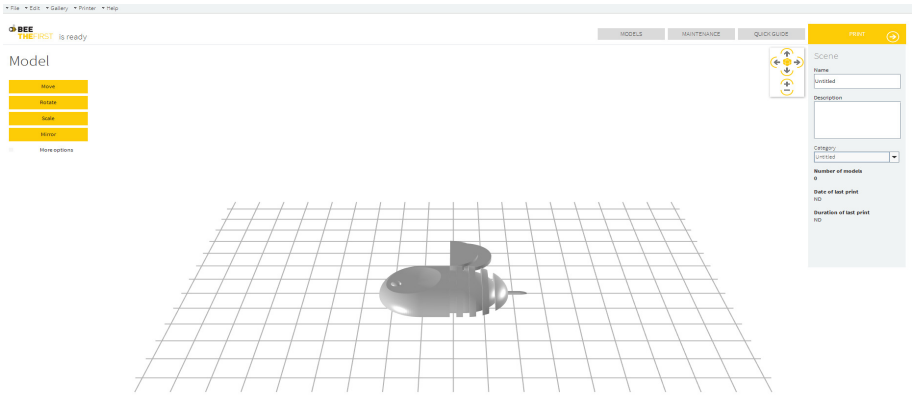


Figure B.6: Scene details screen — global

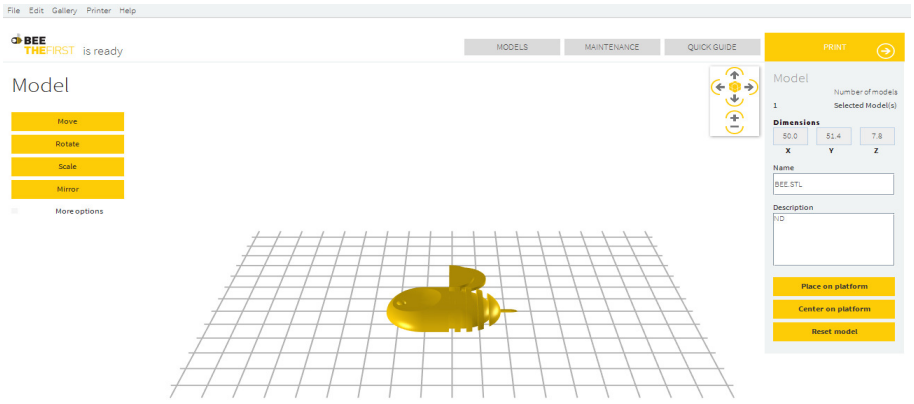


Figure B.7: Scene details screen — model oriented

B.4 Filament wizard

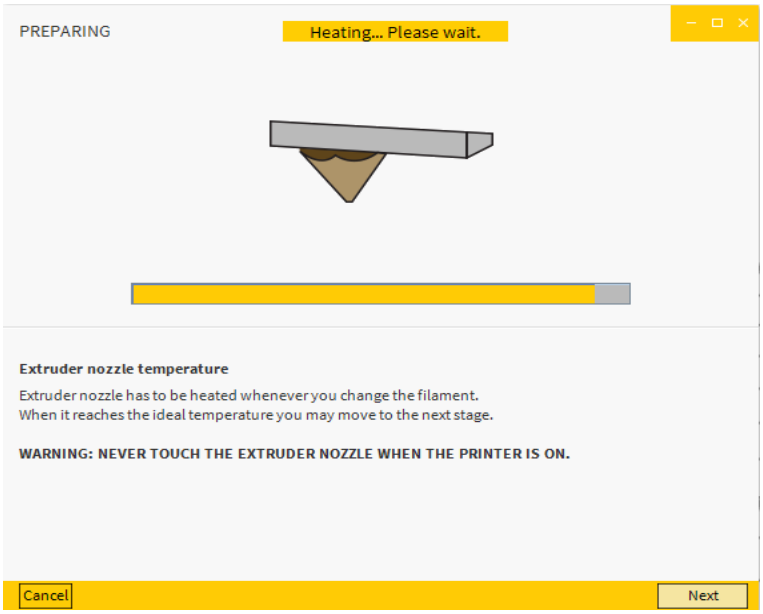


Figure B.8: Filament wizard screen — Heat

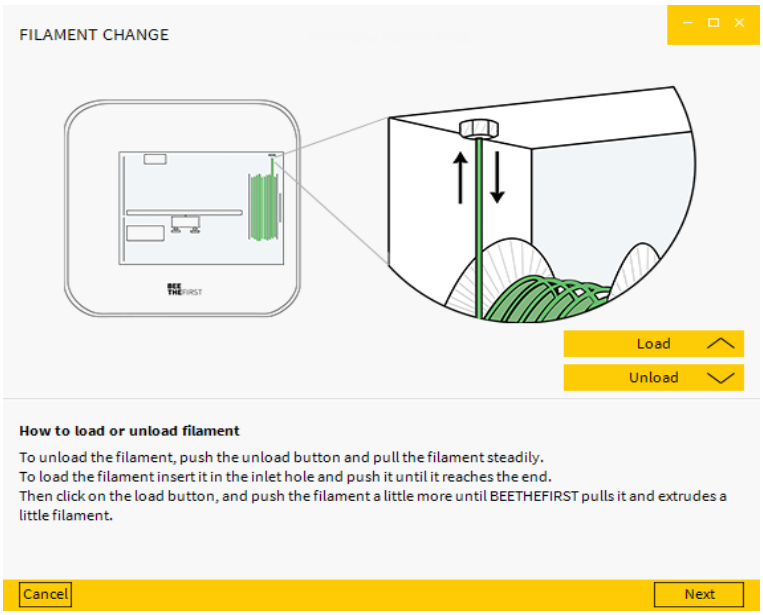


Figure B.9: Filament wizard screen — Load/Unload

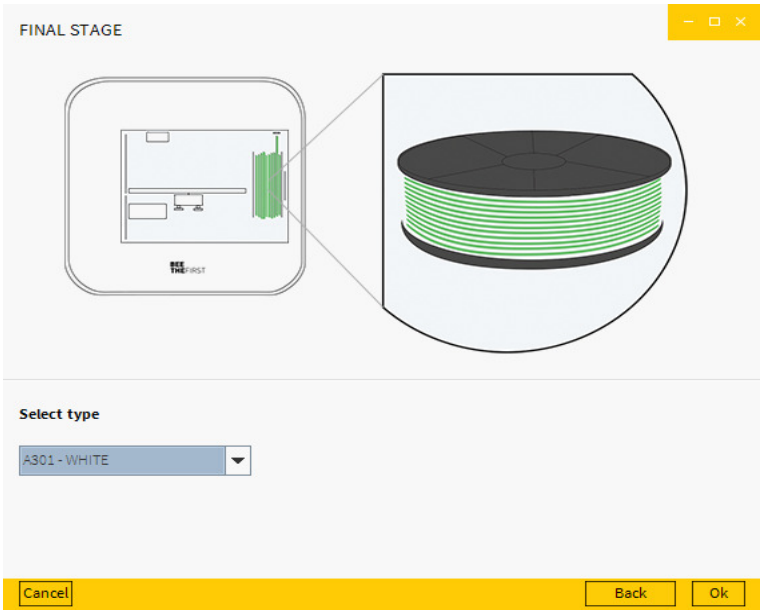


Figure B.10: Filament wizard screen — choice of the filament

B.5 Nozzle wizard

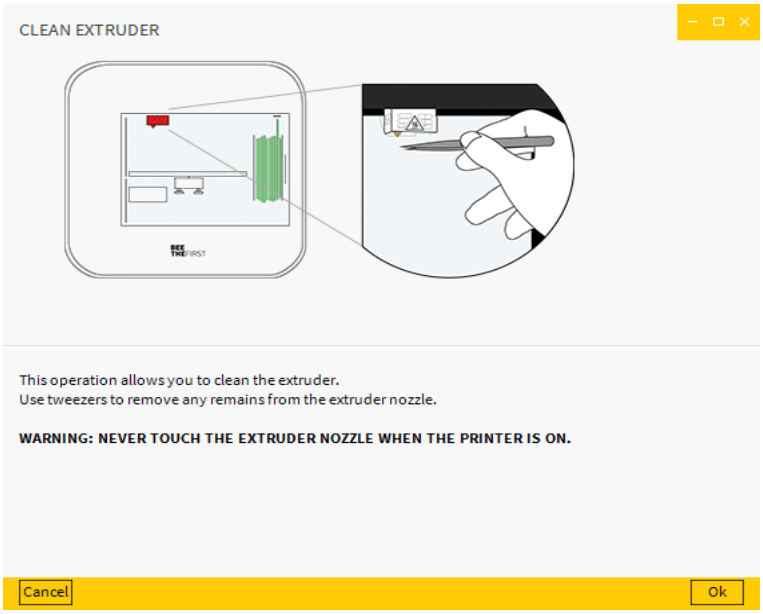


Figure B.11: Nozzle wizard screen

B.6 Calibration wizard

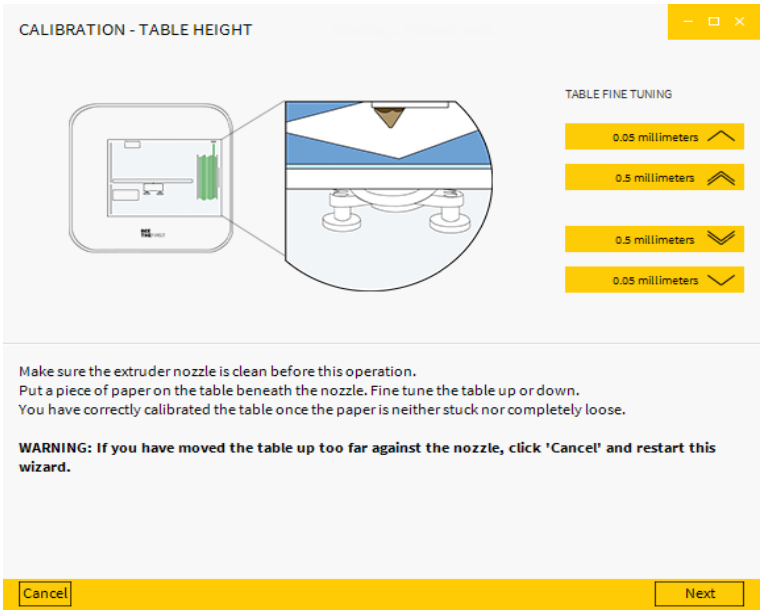


Figure B.12: Calibration wizard screen — table height

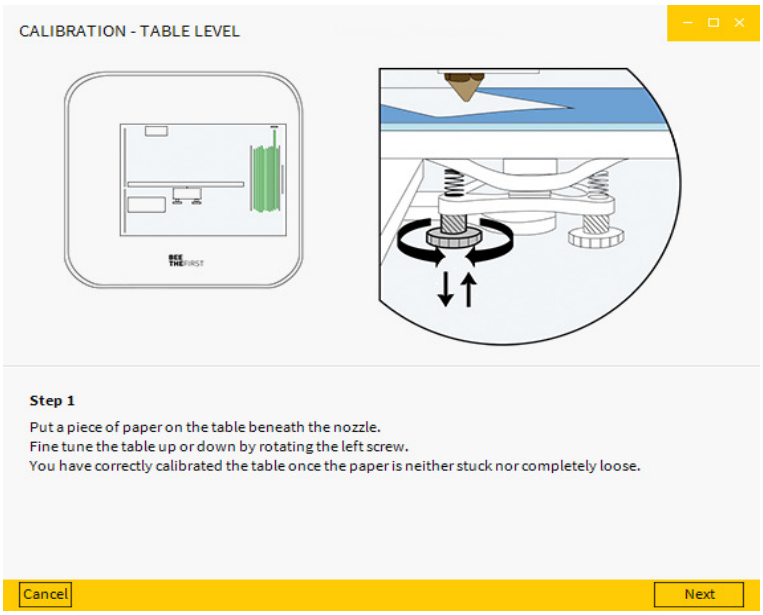


Figure B.13: Calibration wizard screen — table level (phase 1)

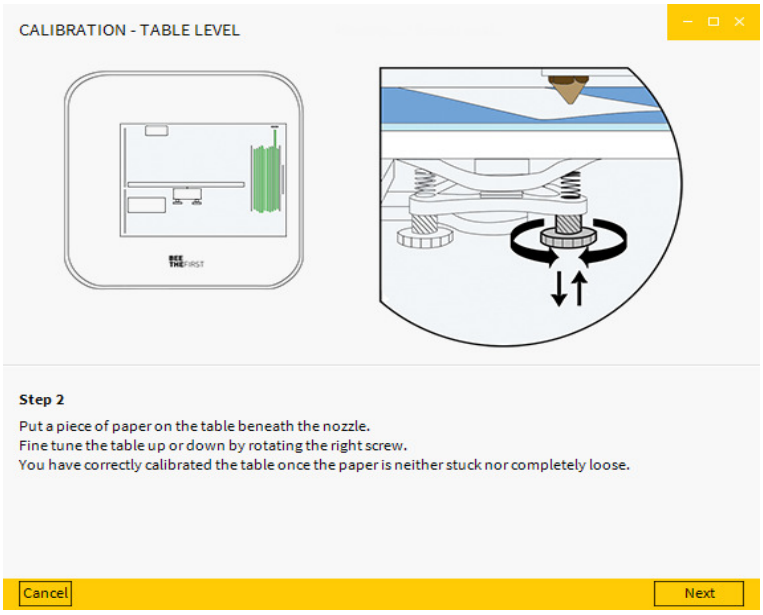


Figure B.14: Calibration wizard screen — table level (phase 2)

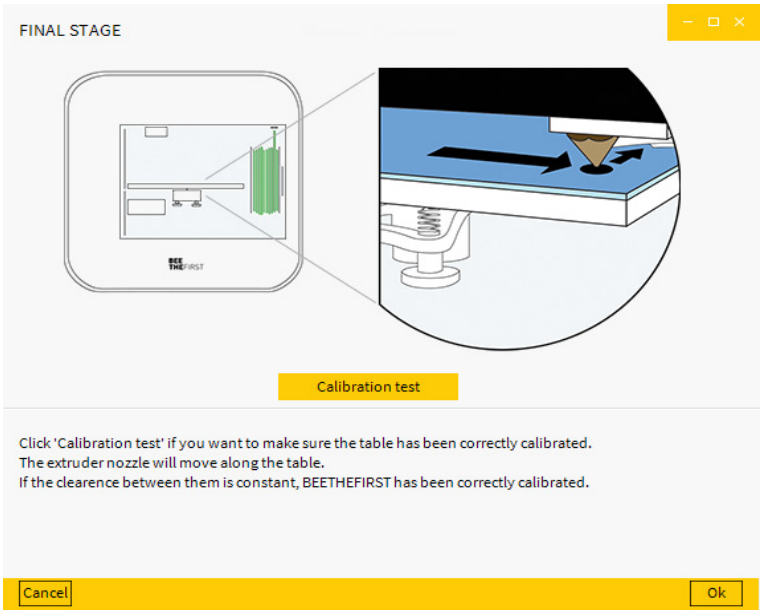


Figure B.15: Calibration wizard screen — finish

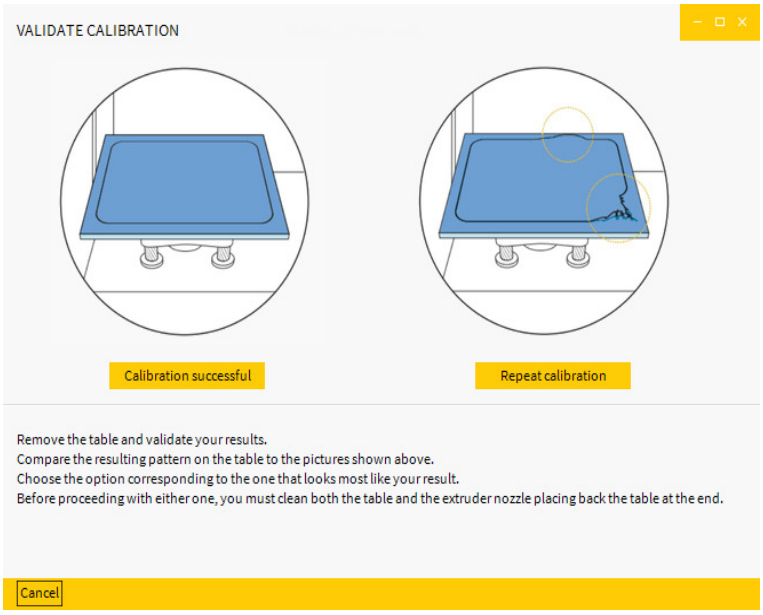


Figure B.16: Calibration wizard screen — validation

B.7 Preferences

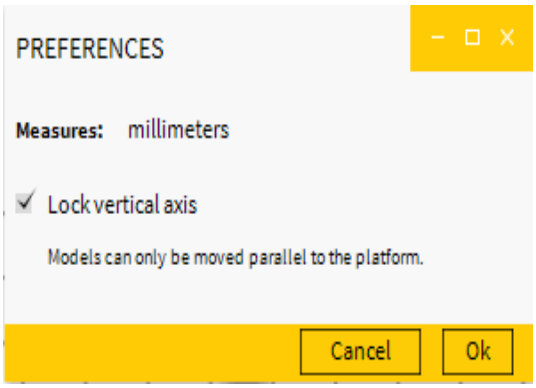


Figure B.17: Preferences screen

B.8 Print

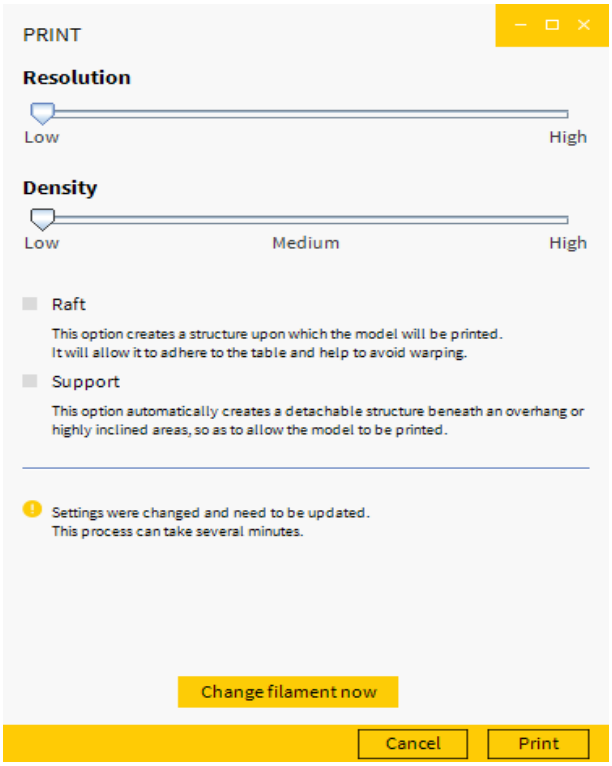


Figure B.18: Print screen — choice of parameters

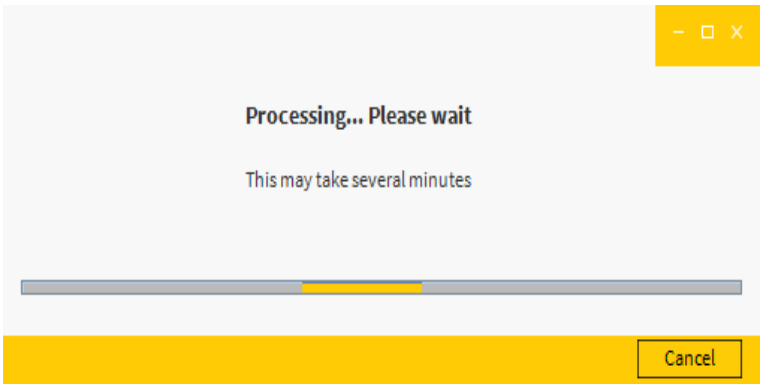


Figure B.19: Print screen — gcode generation

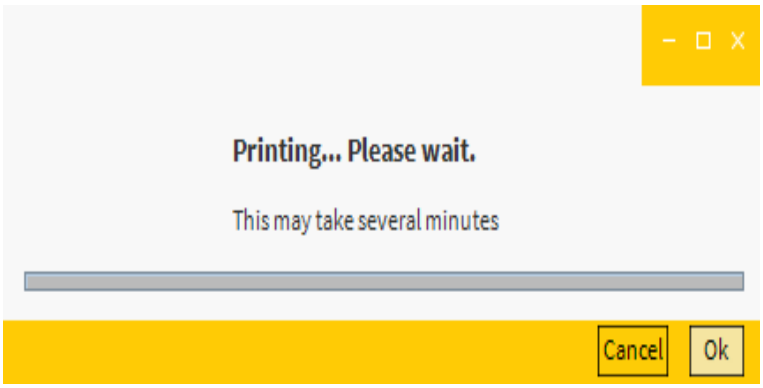


Figure B.20: Print screen — print

B.9 Buttons bar

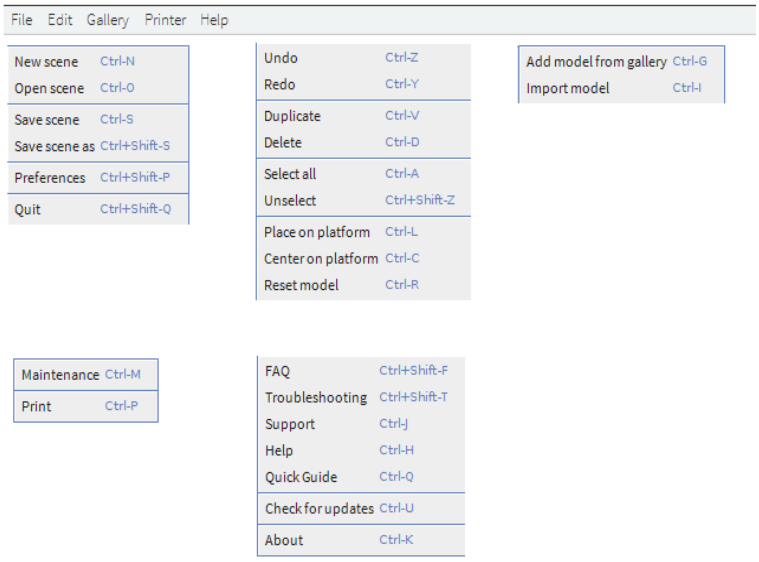


Figure B.21: Buttons bar screen diagram

B.10 Help

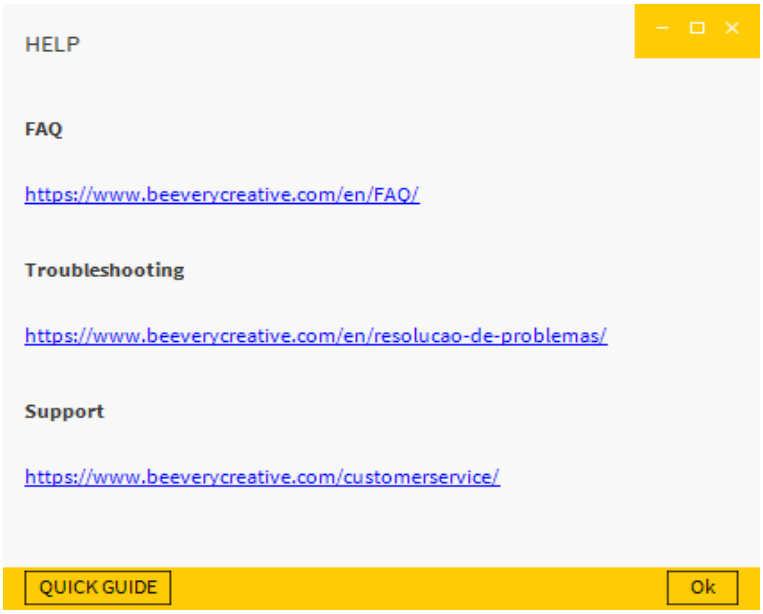


Figure B.22: Help screen

B.11 Update

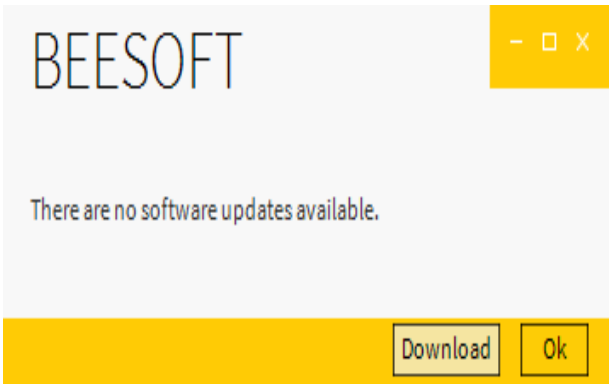


Figure B.23: Update screen

B.12 Gallery

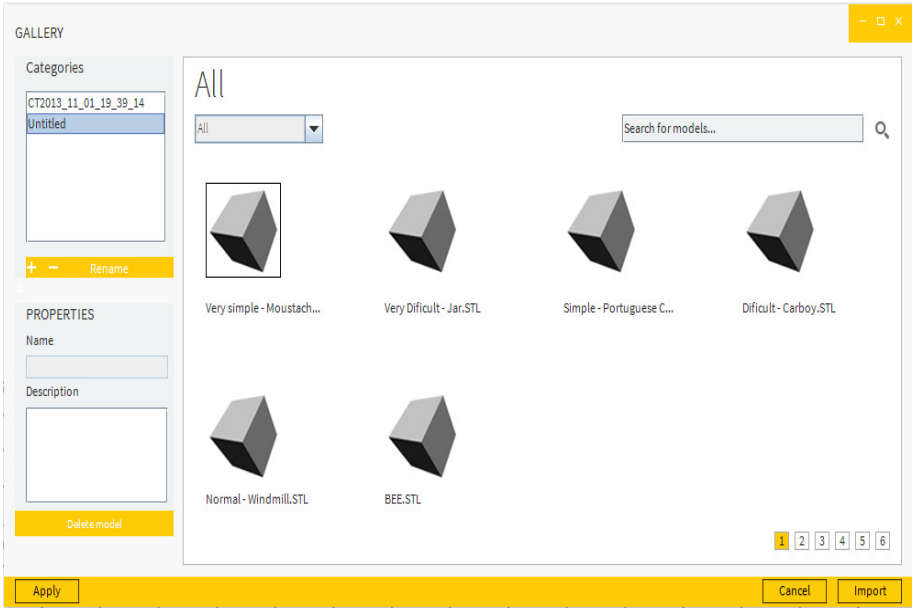


Figure B.24: Gallery screen

B.13 3D

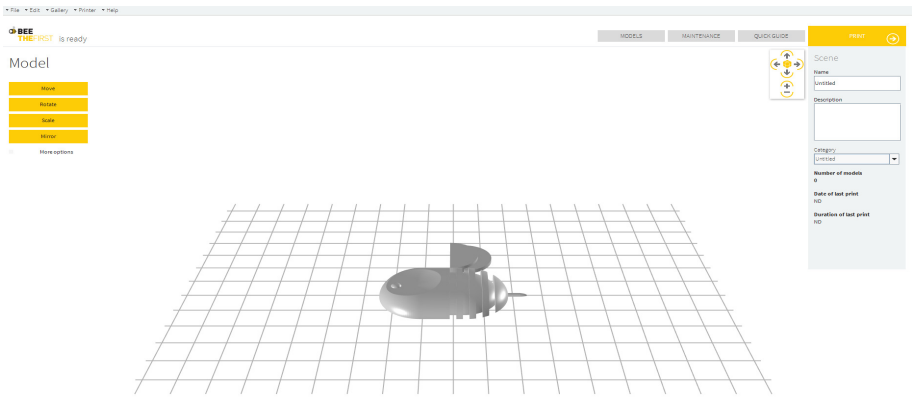


Figure B.25: 3D engine screen — model on scene

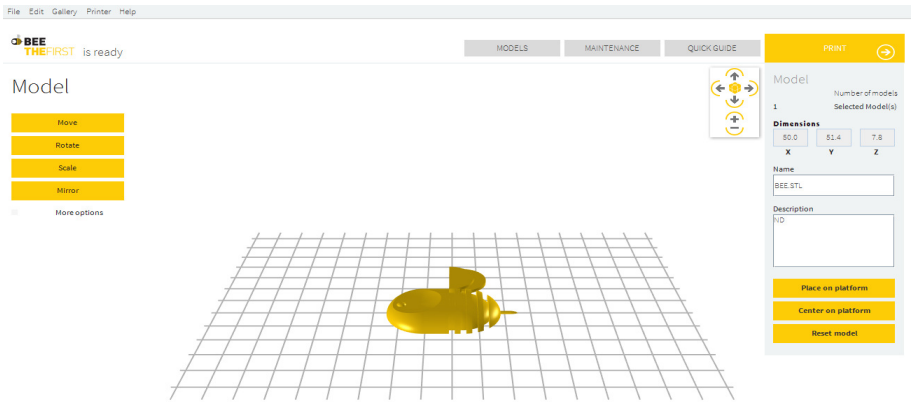


Figure B.26: 3D engine screen — model on scene selected

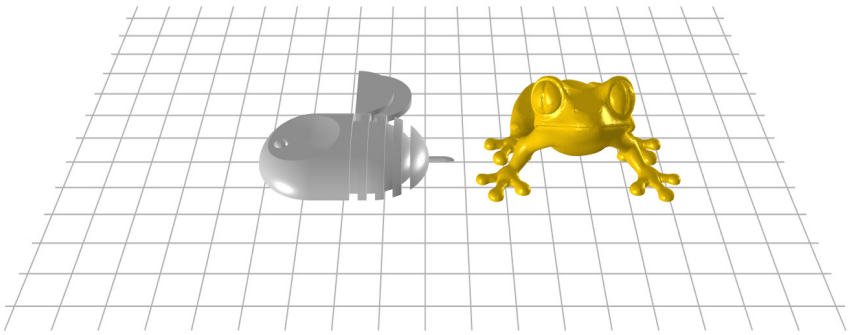


Figure B.27: 3D engine screen — two models on scene with selection

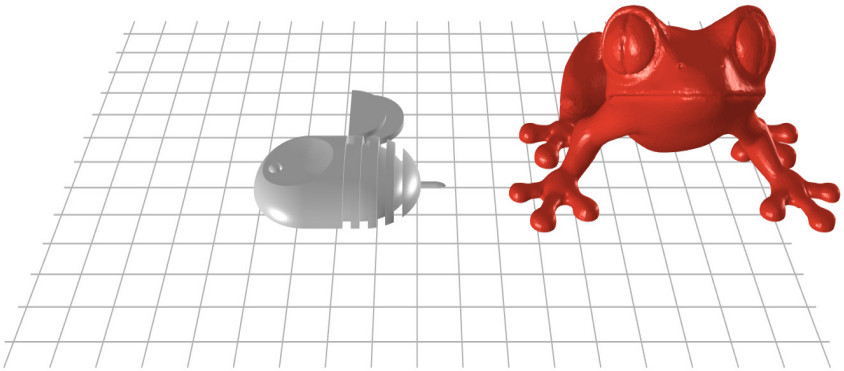


Figure B.28: 3D engine screen — two models on scene with transformation

Appendix C

User Interface Evaluation

This appendix contains all the documentation used to support the user interface usability evaluation, and the results obtained after a careful analysis.

C.1 Usability Test

C.1.1 Documents

Documents used in the usability tests.

Computer-aided manufacturing platform for 3D printers

Facilitators guide

Introduction and goals

The purpose of this round of testing involves obtaining feedback from the user to the proposed structure for the Master thesis in BEEVERYCREATIVE, identifying serious problems during a development phase.

Specific questions to be answered:

- The navigation conventions make sense?
- The navigation structure is easy and intuitive?
- The desired information is easy to find?
- User knows, every time, where and what he can do?

You as a facilitator have the main role to compose a structured and task oriented report of the user behavior while using the BEESOFT.

Facilitator profile

For this battery of tests you will have one user under your analysis. He/she should represent the characteristics of potential users of the 3D printing platform in terms of age, technical knowledge and technology fanaticism.

User characteristics

- Age group representative: participants above 20 years;
- It is not fundamental representative genus division.
- It may participate a person who has work at a similar company or developed a platform of the kind;
- Participants can in no way be connected to the development of the platform;
- Participants should be agile enough regarding computer usage;

Methodology

You will submit to each participant how the test will be executed telling them to speak out loud about everything they are seeing, doing and thinking during the session. If applicable, each participant will be called to fulfill a confidential form provided by the moderator. Participants will then have a number of tasks to perform. Each session will have an approximated duration of half hour.

Once all tasks done, it will be given to the user the possibility of placing questions. At last, it will be request each participant to fulfill a small inquiry.
The test is the following:

Test

- Welcome
- Presentation of the test (goals, framework, methodology…)
- Task lists:
  - **Task 1**
    - Open BEESOFT
  - **Task 2**
    - Create a new scene.
  - **Task 3**
    - Rename it to something you like and add a description.
  - **Task 4**
    - Import the model "BEE.STL".
  - **Task 5**
    - Name it "BEE1".
  - **Task 6**
    - Go to Preferences and understand available features.
  - **Task 7**
    - Move the model BEE1 15 millimeters up in Y-axis and 150 millimeters right in X-axis.
  - **Task 8**
    - Undo last action.
  - **Task 9**
    - Scale BEE1 to 150 percent using the text fields.
  - **Task 10**
    - Reset model BEE1.
  - **Task 11**
    - Scale it to a bigger size using the mouse buttons.
  - **Task 12**
    - Move BEE1 to the coordinates (100,100,10).
  - **Task 13**
    - Go to the Gallery and add a new category with the name Animals.
  - **Task 14**
    - Duplicate the model BEE1 and name the new one to BEE2.
  - **Task 15**
    - Scale the model BEE2 to 50% in Z-axis and then to 175% in all other axes.
  - **Task 16**
    - Assign new category to BEE1 and BEE2.
  - **Task 17**
    - Save the current scene with the following format: YourName_StudentID
  - **Task 18**
    - Try to print the current scene.
  - **Task 19**
    - Consult the version of the platform at About.
  - **Task 20**
    - Close BEESOFT.
- After test inquiry.

Facilitator role

You as a facilitator will just write a task oriented report taking into consideration extremely important observational patterns as:

- Does the user understand well the objective for each task and each panel?
- Did he/she find easily the interface components that fit the purpose? Which was the location of the components he/she thought?
- How did he interpret each task? This means what he thinks he has to do, where to click, what action is under the button event and what is going to happen in consequence.
- How long did the user take to complete each task? What were the main problems: the common errors, if there was any ambiguity in the interface, information or missing elements, …
- Did he cancel or rollback any operation by self-decision? Why and what was he doing?
- Did the user need help to perform some task? What was the task and why did he/she get stuck?
- Did the user gain easily sensibility and experience using the platform? Doing what?
- Did the user notice all the components displayed? Even the new ones after triggering an action?

Write down all important and relevant information you may retrieve from a careful observation not forgetting to cover the maximum possible points discussed above without disturbing the user nor interfering with the test. Read them very carefully to facilitate your notes.
Be detailed, specific, objective, observer and above all strict. Use the method you find suitable to monitor the tasks and taking notes at the same time.

Results

All possible records of footage and sound and all notes along the test will be processed. Observations and relevant findings must be summarized. Recommendations must be done.

Recommendations

Recommendations will be compiled in a report with an executive summary, test plan and tasks list, detailed discovers and recommendations lists. All the notes and comments from participants will be attached.

Report

Note: Use the table of evaluation provided and this space for additional notes or comments.

User ID: _____

Evaluation table of the facilitator

Task	Number of clicks	Did the user finish the Task?	Maximum time Observed time (mm:ss)	Did the user make mistakes?	Got lost?	Call for help?	How easily observed 1 – Not easy 5 – Extremely easy
1		no |__| yes |__|	1m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5
2		no |__| yes |__|	2m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5
3		no |__| yes |__|	2m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5
4		no |__| yes |__|	2m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5
5		no |__| yes |__|	2m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5
6		no |__| yes |__|	2m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5
7		no |__| yes |__|	2m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5
8		no |__| yes |__|	2m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5
9		no |__| yes |__|	2m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5
10		no |__| yes |__|	2m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5
11		no |__| yes |__|	2m ___:___	no |__| few |__| many |__|	no |__| little |__| very |__|	no |__| yes |__| which?	1 2 3 4 5

12		no __| yes __|	**2m** ____:____	no __| few __| many __|	no __| little __| very __|	no __| yes __| which?	1 2 3 4 5
13		no __| yes __|	**2m** ____:____	no __| few __| many __|	no __| little __| very __|	no __| yes __| which?	1 2 3 4 5
14		no __| yes __|	**2m** ____:____	no __| few __| many __|	no __| little __| very __|	no __| yes __| which?	1 2 3 4 5
15		no __| yes __|	**2m** ____:____	no __| few __| many __|	no __| little __| very __|	no __| yes __| which?	1 2 3 4 5
16		no __| yes __|	**2m** ____:____	no __| few __| many __|	no __| little __| very __|	no __| yes __| which?	1 2 3 4 5
17		no __| yes __|	**2m** ____:____	no __| few __| many __|	no __| little __| very __|	no __| yes __| which?	1 2 3 4 5
18		no __| yes __|	**2m** ____:____	no __| few __| many __|	no __| little __| very __|	no __| yes __| which?	1 2 3 4 5
19		no __| yes __|	**2m** ____:____	no __| few __| many __|	no __| little __| very __|	no __| yes __| which?	1 2 3 4 5
20		no __| yes __|	**1m** ____:____	no __| few __| many __|	no __| little __| very __|	no __| yes __| which?	1 2 3 4 5

Comments

_____

_____

_____

_____

_____

_____

Script of usability testing

Introduction and goals

Hi, my name is Michael Domingues and you will work with my project in today's session. I'd like to start presenting you briefly what we are going to do and what I am trying to achieve:

Today I will evaluate the structure of the platform for 3D printing. Your experience will help me assuring this platform is simpler, easy to use and intuitive. There will be a facilitator monitoring your session and observing you while you use the platform and taking notes too.

Please, remember I don't intend somewhat to evaluate you. I am just trying to understand how people use this kind of platform. Do your best but don't worry with the results. Any question or problem along the essay you must tell the facilitator, in order to develop a better product to be to everyone's liking.

This way you will have a list of tasks that must be done in a current and beta version of the platform, therefore pay attention for every detail.

I ask you that, at each moment and per task, to speak out loud about everything you see and think*, what you are looking for, what you are doing and what you expect to obtain.* You will be under supervision of a facilitator who will note down main aspects of your behavior using the system, you errors, tries and comments you gradually make. *The more information you provide the better.*

You can place any question but probably it won't be answered since it is important you explore the platform as if I wasn't here. At the end the facilitator will have free time to talk with you about your experience and then, answer all your questions.

Do you have any doubt? If not, lets start with the first task.

Tasks list

- **Task 1**
  - Open BEESOFT
- **Task 2**
  - Create a new scene.
- **Task 3**
  - Rename it to something you like and add a description.
- **Task 4**
  - Import the model "BEE.STL".
- **Task 5**
  - Name it "BEE1".
- **Task 6**
  - Go to Preferences and understand available features.
- **Task 7**
  - Move the model BEE1 15 millimeters up in Y-axis and 150 millimeters right in X-axis.
- **Task 8**
  - Undo last action.
- **Task 9**
  - Scale BEE1to 150 percent using the text fields.
- **Task 10**
  - Reset model BEE1.
- **Task 11**
  - Scale it to a bigger size using the mouse buttons.
- **Task 12**
  - Move BEE1 to the coordinates (100,100,10).
- **Task 13**
  - Go to the Gallery and add a new category with the name Animals.
- **Task 14**
  - Duplicate the model BEE1 and name the new one to BEE2.
- **Task 15**
  - Scale the model BEE2 to 50% in Z-axis and then to 175% in all other axes.
- **Task 16**
  - Assign new category to BEE1 and BEE2.
- **Task 17**
  - Save the current scene with the following format: YourName_StudentID
- **Task 18**
  - Try to print the current scene.
- **Task 19**
  - Consult the version of the platform at About.
- **Task 20**
  - Close BEESOFT.

User inquiry

Name	
Gender	
Occupation	
Hobby	
3D printing or 3D modeling experience	
Frequency using proprietary software	
Most used software	

All the supplied information will be for exclusive use of the author of this platform. It is fully confidential and it won't be distributed or used for other any kind of purposed besides this test.

After test inquiry

Please help me fulfilling this inquiry based in your experience during the test.

1. How do you rate your tasks during the test?	Hard	1	2	3	4	5	Easy
2. How do you rate the main design of the platform?	Confused	1	2	3	4	5	Evident
3. How do you rate the main screen components location?	Messy	1	2	3	4	5	Clean
4. Was it difficult to import a model?	Hard	1	2	3	4	5	Easy
5. How do you rate the model manipulation engine?	Hard	1	2	3	4	5	Easy
6. How difficult was it to manipulate the models?	Hard	1	2	3	4	5	Easy
7. How noticeable was it to edit model information?	Hard	1	2	3	4	5	Easy
8. How difficult was it to unlock the vertical axis?	Hard	1	2	3	4	5	Easy
9. Was it obvious you could jog the model with the mouse controls?	Hard	1	2	3	4	5	Easy
10. How difficult was it to understand the gallery and create a new category?	Hard	1	2	3	4	5	Easy
11. Was difficult was to find the About?	Hard	1	2	3	4	5	Easy
12. Was all the navigation through the platform intuitive?	Rarely	1	2	3	4	5	Always
13. Is the platform difficult to learn?	Hard	1	2	3	4	5	Easy
14. How do you rate it globally?	Bad	1	2	3	4	5	Excellent
15. Would you recommend it to someone?	Never	1	2	3	4	5	Sure

C.1.2 Main results of the Usability Test

Problems found in the usability tests:

1. Inability to apply the same transformation value to all 3 axis at the same time
2. Inability to unlock the vertical axis within the 3D engine or the modeling operation containers
3. Absence of model coordinates
4. Text should be clickable
5. Inability to undo more than one operation
6. Inability to manipulate several models at a time
7. Absence of a submit button on the modeling operations containers for the text fields
8. Incomplete set of keyboard shortcuts to help navigation on open screens
9. Difficulty to select and deselect a model
10. Difficulty to understand if a model is selected in an invalid volume state
11. Absence of confirmation dialogs when creating a new scene, deleting files in the gallery or quitting the application
12. Difficulty to create and manage categories
13. Difficulty to understand that it was possible to apply transformations with the mouse
14. Difficulty to understand that the text fields in the move operations container are to move the model with relative values and not absolute
15. Absence of collision detection between models in the 3D canvas
16. Absence of limit values when scaling, to avoid excessive scaling
17. Absence of model auto-resize when importing, if models exceed the maximum printing volume
18. Absence of a maximum limit when calibrating the *BEETHEFIRST* in height, to avoid collision up with the nozzle

Bibliography

[1] Mark Fleming. What is 3d printing? an overview. Onehossshay.wordpress.com, 2013. [Online; posted 2013].

[2] RepRap. Reprap. RepRap.org, 2013. [Online; posted 13-October-2013].

[3] Kyle Maxey. 3d printing trends, history. Enginnering.com, 2013. [Online; posted 18-September-2013].

[4] Blender. Meshes primitives. Wiki.blender.org. [Online].

[5] 3DPP. Choosing a filament: Abs vs pla. 3DPrinterprices.net, 2013. [Online; posted 22-March-2013].

[6] Daniel Selman. *Java 3D Programming*. MANNING, 2002.

[7] Ennex Corporation. The stl formatstandard data format for fabbers. Ennex.com, 1999. [Online; posted 1999].

[8] Hod Lipson. Additive manufacturing file format (amf) allows for volumetric specifications. Enginnering.com, 2012. [Online; posted 14-August-2012].

[9] 3ders. Microsoft explained 3d printing support in windows 8.1. 3ders.org, 2013. [Online; posted 23-August-2013].

[10] RepRap. G-code. RepRap.org, 2013. [Online; posted 2013].

[11] University at Buffalo. Computer numerical control (cnc). Position Paper from University of Buffalo.

[12] RepRap. R2c2 reprap electronics. RepRap.org, 2014. [Online; posted 15-February-2014].

[13] OSDEV. Bootloader. Wiki.osdev.org, 2012. [Online; posted 25-November-2012].

[14] Printrbot. *Printrbot Firmware Guide*, 2012.

[15] ReplicatorG. Replicatorg is a simple, open source 3d printing program. Replicat.org, 2012. [Online; posted 2012].

[16] Renosis. Skeinforge 41 quick start guide. renosis.net, 2011. [Online; posted 24-July-2011].

[17] Edutechwiki. Slicers and user interfaces for 3d printers. Edutechwiki.unige.ch, 2013. [Online; posted 18-November-2013].

[18] GENERAL FABB. Cura: An unusual slicer. Fabbaloo.com, 2013. [Online; posted 27-June-2013].

[19] Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. Stress relief: Improving structural strength of 3d printable objects. *ACM Trans. Graph.*, 31(4):48:1–48:11, July 2012.

[20] 3ders. 3d print a chair, piece by piece. 3ders.org, 2012. [Online; posted 3-December-2012].

[21] Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. Chopper: Partitioning models into 3d-printable parts. *ACM Trans. Graph.*, 31(6):129:1–129:9, November 2012.

[22] Klaus Hinum. Human centred design for graphical user interfaces. Master's thesis, Vienna University of Technology, 2004.

[23] Jenifer Tidwell. *Designing interfaces.* O'Reilly, Sebastopol, CA, 2011.

[24] Wilbert O. Galitz. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques.* John Wiley & Sons, Inc., New York, NY, USA, 1997.

[25] Alistair Cockburn. *Writing Effective Use Cases.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 2000.

[26] Steve Krug. *Don'T Make Me Think: A Common Sense Approach to the Web (2Nd Edition).* New Riders Publishing, Thousand Oaks, CA, USA, 2005.

[27] Donald A. Norman. *The Design of Everyday Things.* Basic Books, New York, reprint paperback edition, 2002.

[28] Jakob Nielsen. *Usability Engineering.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[29] Jeff Johnson. *GUI Bloopers 2.0: Common User Interface Design Don'Ts and Dos.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2007.

[30] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-computer Interaction.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[31] Wendy A. Kellogg. Coordinating user interfaces for consistency. chapter The Dimensions of Consistency, pages 9–20. Academic Press Professional, Inc., San Diego, CA, USA, 1989.

[32] W. A. Kellog. Conceptual consistency in the user interface: Effects on user performance. In H.-J. Bullinger and B. Shakel, editors, *Human-Computer Interaction: INTERACT'87*, pages 389–394. North-Holland, Amsterdam, 1987.

[33] David H. Eberly. *3D Game Engine Design, Second Edition: A Practical Approach to Real-Time Computer Graphics (The Morgan Kaufmann Series in Interactive 3D Technology).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[34] Adnan M. L. Karim, M. Sazzad Karim, Emdad Ahmed, and Dr. M. Rokonuzzaman. Scene graph management for opengl based 3d graphics engine. Position Paper from North South University.

[35] OSDev. Universal serial bus. Wiki.osdev.org, 2013. [Online; posted 15-August-2013].

[36] Jan Axelson. *USB complete: the developer's guide*. Lakeview Research, Chicago, IL, 2009.

[37] Luke Collins. Usb 3.0 protocol layer – part 1. Techdesignforums.com, 2013. [Online; posted 16-September-2013].

[38] University of California. Link layer link layer: Introduction. Position Paper from University of California.

[39] University of Wolverhampton. Osi physical layer. Position Paper from University of Wolverhampton.

[40] Sonny Jeon. Grbl: How it works and other thoughts.... Onehossshay.wordpress.com, 2011. [Online; posted 21-August-2011].

[41] RepRap. List of firmware. Onehossshay.wordpress.com, 2013. [Online; posted 2013].

[42] Heather Desurvire, Jim Kondziela, and Michael E. Atwood. What is gained and lost when using methods other than empirical testing. In *Posters and Short Talks of the 1992 SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 125–126, New York, NY, USA, 1992. ACM.

[43] Jakob Nielsen. Finding usability problems through heuristic evaluation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 373–380, New York, NY, USA, 1992. ACM.

[44] Robin Jeffries, James R. Miller, Cathleen Wharton, and Kathy Uyeda. User interface evaluation in the real world: A comparison of four techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 119–124, New York, NY, USA, 1991. ACM.

[45] Robin Jeffries and Heather Desurvire. Usability testing vs. heuristic evaluation: Was there a contest? *SIGCHI Bull.*, 24(4):39–41, October 1992.

[46] Jakob Nielsen. How to conduct a heuristic evaluation, 1994.

[47] J. Nielsen. Severity ratings for usability problems. Website, 1995.

[48] Jürgen Sturm, Erik Bylow, Fredrik Kahl, and Daniel Cremers. Copyme3d: Scanning and printing persons in 3d. In Joachim Weickert, Matthias Hein, and Bernt Schiele, editors, *Pattern Recognition*, volume 8142 of *Lecture Notes in Computer Science*, pages 405–414. Springer Berlin Heidelberg, 2013.