FILTER MEDIATED DESIGN

Generating Coherence in (collaborative) Design

by

John R. Haymaker

B.S. Architecture
University of Michigan, Ann Arbor, 1990

Master of Architecture
U. of Illinois - Chicago, 1995

SUBMITTED TO THE DEPARTMENT OF ARCHITECTURE
IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE IN ARCHITECTURE STUDIES
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 1999

Signature of Author: ______________________
John R. Haymaker
Department of Architecture
May 20, 1999

Certified By: ______________________
Edith K. Ackermann
Visiting Professor
Dept. of Architecture
Thesis Co-Advisor

Certified By: ______________________
William J. Mitchell
Dean / Professor
Dept. of Architecture
Thesis Co-Advisor

Accepted By: ______________________
Roy Strickland
Associate Professor
Dept. of Architecture
Chairman
Comm. for Grad Students

Thesis Reader

William L. Porter
Professor
Deptartment of Architecture

Thesis Reader:

Patrick H. Winston
Professor
Artificial Intelligence Laboratory
Dept. of Electrical Engineering
and Computer Science

FILTER MEDIATED DESIGN

Generating Coherence in (collaborative) Design

by

John R. Haymaker

SUBMITTED TO THE DEPARTMENT OF ARCHITECTURE
IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE IN ARCHITECTURE STUDIES
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 1999

ABSTRACT

Architectural design involves the integration of diverse, sometimes conflicting, concepts and requirements into a coherent single composition. This paper proposes a method for negotiating architectural design across domains, by examining issues of ontology, perception, generation and evaluation, and detailing a prototype in which these mechanisms are augmented using computational agents for achieving coherence and innovation in remote collaborative design. The paper proposes a common geometric and topological database, from which multiple semantic models are constructed. Filter Mediated Design is intended to explore the processes and strategies of constructing intelligent designs and design intelligence.

Thesis Co-Advisors:

Edith K. Ackermann
Visiting Professor, Dept. of Architecture

William J. Mitchell
Dean / Professor, Dept. of Architecture

ACKNOWLEDGEMENTS

I would like to thank the following people, who have helped make my experience at MIT such a rewarding one.

Mitsubishi Electric Research Laboratory under the leadership of Dr. Richard Waters, provided generous support for the research, and Open Community, which served as the platform and some of the inspiration for work. Without the incredible technical, intellectual, and motivational support of Bill Lambert, Barry Perlman, Derek Schwenke, and Evan Suits, of the affiliated Horizon Systems Laboratory, the implementation would not have been possible.

Edith Ackermann's insights from the developmental psychology perspective were fundamental to the direction of the work. I enjoyed her spirit, intellect, and sense of humor immensely.

The mezzanine gang: Cindy, Neidre, Mine, Rashmi, Michael, Singh, Haldane, Jose, Birgul, Twigs, for the good times, and the discussions, I learned a great deal from you all.

My SmarchS colleagues: for the valuable discussions in and out of thesis meetings.

The faculty at MIT has been remarkable. I would like to especially thank William Porter for his kind support of the research, and for his insights in the realm of design inquiry. I would also like to thank William Mitchell for his support, and direction. Patrick Winston is a incredible educator, and would like to thank him for sharing so much of his valuable time and knowledge with me.

Paul Keel has been my friend, guide, mentor and colleague. Our many discussions at the whiteboard formed the foundation for this thesis, and were an experience I will always remember.

Finally, I want to thank my family, for the loving support I've received from all of them.

TABLE OF CONTENTS

Section	Section Title	Page
1.0	**Introduction - a Framework**	**13**
1.1	The Changing Nature of Collaborative Design	14
1.2	An Opportunity for New Means of Constructing Coherence and Innovation	16
1.3	Making Explicit the Mechanisms of Design	17
1.4	Filters: Addressing the Issue of Representation, Meaning and Communication	18
1.5	Constructors: Addressing the Issue of Memory and Information Processing	23
1.6	Managers: Addressing the Issue of Coordination	26
1.7	Filter Mediated Design is a Tool for Design And for Thinking about Design	27
2.0	**Ontology and Meaning**	**31**
2.1	Design as Database Structure and Access	31
2.2	On the Structure of Data	32
2.3	Existing Paradigm of Data Structure	34
2.4	Existing Paradigm of Data Access	35
2.5	Filters Equipe Computers with the Power of Perception	36

Section	Section Title	Page
3.0	**Implementation**	**39**
3.1	Interacting with the Implementation	40
3.1.1	Filters	41
3.1.2	Constructors	43
3.1.3	Mediators	44
3.2	Open Community	48
3.3	The Program - System Architecture	49
3.3.1	filter.space()	50
3.3.2	constructor.space()	51
3.3.3	filter.enclosure()	52
3.3.4	constructor.enclosure()	52
3.3.5	filter.light()	53
3.3.6	constructor.light()	53
3.3.7	filter.structure()	54
3.3.8	constructor.structure()	54
3.3.9	filter.chair()	55
3.3.10	constructor.chair()	55
4.0	**Future Work and Implications**	**57**
4.0.1	Future Work on Ontology	57
4.0.2	Future Work on Perception	58
4.0.3	Future Work on Communication	60
4.1	Conclusion	62
	References	65
	Appendix A : Related Bibliography	67
	Appendix B:The Code	71
	Appendix C: Images	117

1.0 Introduction

Information growth in recent decades has drastically altered the structure and location of the knowledge one uses to design. Individuals increasingly specialize, relegating entire domains of knowledge to other specialists, computational agents and reference materials. The field of collaborative design has evolved to make these different domains and agendas explicit, and to bring them back into dialogue. The goal is to increase the capability to bring about innovation and cohesion from the field of possibilities and constraints by communicating concerns and possibilities across disciplines.

Different domains have evolved their own languages and representations, making communication between disciplines difficult. The challenge is to develop new methods for negotiating across different domains. This paper considers a new approach to collaborative design and describes opportunities for new means of generating coherence and innovation by reformulating the construction and flow of information. In section 1, the work proposes some fundamental processes at play in design - perception, generation and evaluation - and details a prototype in which these mechanisms are augmented using agents for achieving coherence and innovation in remote collaborative design. Section 2 explains the theory of ontology that forms the basis for 'filter mediated design'. The paper proposes a common geometric and topological database, from which multiple semantic models are constructed. These semantic models then negotiate, either through the common geometric database, or through specific languages which could evolve between semantic models. Section 3 details the prototype which implements the mechanisms described in Section 1 on top of the ambiguous database explained in section 2. Section 4 examines implications and future directions for the work. 'Filter mediated design' is intended as a tool for design, and for thinking about design.

1.1 The Changing Nature of Collaborative Design

In recent decades, the designer's work environment has changed. Information growth and advancements in information technology have transformed the location, structure, and flow of the knowledge one uses to design. A number of phenomena mark this transformation:

- **Specialization** – humans confine the scope of their knowl edge in order to go in depth on a particular subject.
- **Delegation** – humans increasingly rely on other specialists, computational agents, and reference materials to augment their cognitive abilities.[1]
- **Cooperation** – an increasing number of people become involved in any one project, often working in parallel.
- **Distribution** – Increasingly people are working remotely.

Design, as a process of generating coherence and innovation, must operate in this altered information landscape. Some of the things designers do in the process of generating coherence and innovation, according to Nigel Cross [2], involve search for emergence, imposition of additional constraints, use of solution conjectures, and change of solution goals. New mediating techniques which support these design processes are needed to bring the different disciplines back into dialogue and allow cohesion to emerge in the changing information landscape. Negotiations, which traditionally could occur internally (and often implicitly) in the designer's mind are projected outward, and thus are made explicit.

Traditional collaborative design relies on verbal communication and artifacts representing aspects of the design to share and negotiate this knowledge. However, the forces of expansion, specialization and distribution of knowledge often make traditional modes of collaboration difficult. Some of the issues traditional collaborative design faces in a world of increasing specialization and delegation are:

- **Communication** – Spatial and temporal constraints of verbal communication inhibit the transfer of knowledge.
- **Representation and meaning** – Different disciplines evolve different representations and languages for their domains.
- **Memory and information processing** – Individuals and collaborators struggle with large, combinatorial tasks.
- **Coordination** – Distribution of Intelligence generates extensive difficulties in the task of coordination.

One of the major issues in developing tools for remote collaborative design is how to structure, access, manipulate, communicate and coordinate data. The general approach has been to try to explicitly represent intent by imbedding semantics into the database. Designers access the data by searching for semantic labels [3 4 5]. Manipulating the database then involves changing the geometry, and updating all the semantic consequences of the change [6 7], a difficult prosepect for an individual designer who understands all the semantic consequences. When the task is distributed among disciplines, the maintenance of consistency becomes daunting. These approaches are useful for later stages, when most semantic decisions have been determined, but are counter-intuitive for schematic design.

Filter mediated design differs from similar research by focusing on the early stages of design when ambiguity serves as a powerful heuristic for creative design. This work explores a database design with as little semantic content as possible to allow for multiple readings. Semantic data is actively constructed from geometric and topological data rather than being read from labels on database entities. To maintain consistency, this research proposes the imposition of external constraints to allow for a more flexible structuring of the data. In design, it is often desirable to dynamically relax, remove and impose constraints. Proposed changes to the common database by individual users can first be negotiated between the semantic models.

This research is also concerned about design interaction, the protocols of cooperation, and conflict management. Some research [8] involves multi-agent design processes, in which specialized agents focus on domain specific aspects of the design to construct a collective intelligence. The design world consists of components with inputs and outputs that are connected, sized and evaluated. 'Filter mediated design' expands on the idea of multi-agent distributed design, considering schematic design in the architectural domain, where issues of multiple readings and criteria are negotiated.Other work [9] fosters collaborative design activity by blurring authorship throughout the design process by encouraging users to exchange their models iteratively. 'Filter mediated design' seeks to further blur authorship by immersing users in a distributed virtual environment where users can contribute simultaneously. This thesis explores a design architecture for user/user, user/agent, and agent/agent interaction.

1.2 An Opportunity for New Means of Con structing Coherence and Innovation.

This paper views the distribution of knowledge and the mediation of constraints and possibilities through digital technology as an opportunity, not to facilitate traditional means of collaboration, but to construct new means by which cohesion and innovation can be achieved in collaborative endeavors far more complex than those of the past. Figure 1.1 shows a schematic diagram of traditional collaborative design compared to 'filter mediated design'. In traditional collaborative design, communication of possibilities and constraints occurs through conversations with others, with representations of the project, and with others through representations of the project. Coherence emerges in this formulation when multiple points of view find satisfaction in the artifact. 'Filter mediated design' proposes implementing 'filters', which serve to reformulate the construction and flow of information in collaborative design. Users view and modify objects through interconnected computational devices that automatically exchange and display information about user contributions. The intent is to supplant some of the verbal communication to improve efficiency and allow for more people to participate in the design process.

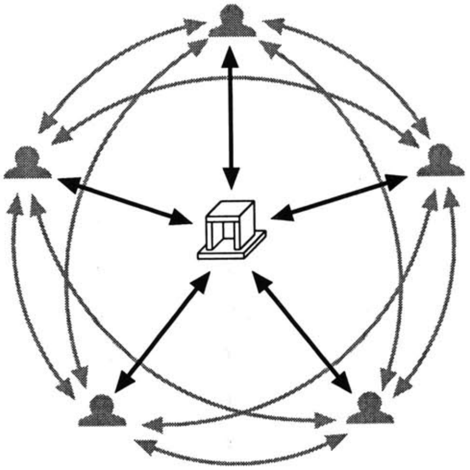


Figure 1.1a: traditional collaborative design

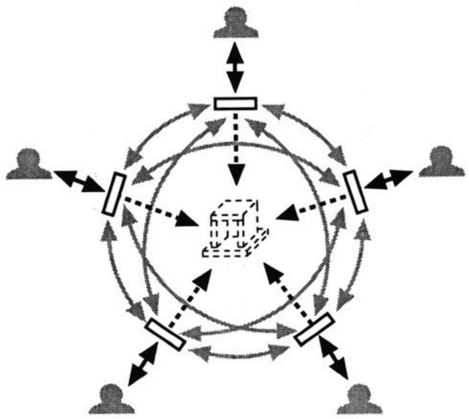


Figure 1.1b: filter mediated design

When knowledge becomes distributed, the challenge becomes how to support and develop the ability to integrate diverse perspectives in new and interesting ways and to begin exploring new means of supporting and generating design intelligence. This paper addresses this challenge through a multi-disciplinary exploration of the design process, considering a number of issues from the domains of Design Inquiry, Developmental Psychology, Artificial Intelligence, and Communications Technology. Several fundamental mechanisms are suggested for an individual to construct design coherence in the world: Perception, Evaluation, and Generation. This section discusses how these mechanisms can then be employed to construct coherence and innovation across individuals in the field of design. This approach does not privilege particular design strategies; rather, it proposes a mediating mechanism that can support various strategies. This reconstruction directly addresses the issues of communication, representation, memory and information processing, and coordination, with which traditional collaborative design struggles.

1.3 Making Explicit the Mechanisms of Design

Collaborative Design is a Constructive Dialogic process. This means that coherence evolves through multiple cycles of concept generation and testing in "feedback loops". The feedback allows for the generation and test processes to be reconsidered in light of each other. In an attempt to model this process, this project proposes a dialogue between three mechanisms:

- **Filters** – the ability to sense attributes and construct readings in the world. 'Filters' also can communicate their readings to other 'filters'.
- **Constructors** - The ability to judge, and the ability to act.
- **Managers** – the ability to coordinate "constructors" and 'filters'.

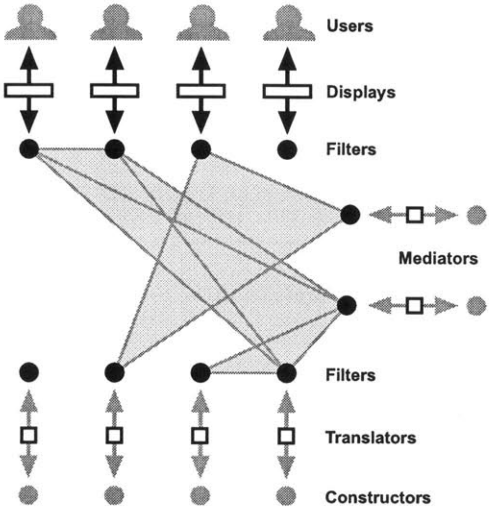


Figure 1.2: Filter Mediated Design

Figure 1.2 illustrates an overview of 'filter mediated design'. This conceptual overview proposes the integration of Users, computational 'constructor' agents and 'manager' agents in a collectively intelligent module, mediated by 'filter' agents. An overview is provided here to provide the reader with a sense of the issues addressed in this paper and explain the individual mechanisms in more detail below. Coherence in design, it is postulated, emerges from the dialogue between the mechanisms proposed.

Design is a dialogue, through the object, with the contexts for which the object is intended. During this process the designer adopts many perspectives. Each perspective focuses on different aspects of the project, and requires specialized representations suited for the task. When testing possible solutions, the designer simulates different aspects of the environment, imagining how the project will respond to issues like wind, light, and human use; when generating possibilities, the designer manipulates different aspects of the object, trying possible solutions. Coherence emerges from these transactions when the object satisfies multiple points of view. These ideas are explored from the perspective that generation of coherence and innovation within an individual could well be analogous to the generation of innovation and coherence across individuals. In a collaborative exercise, the individual designers distribute and specialize on the different perspectives of the project, but the fundamental dynamic of cycles between generation and evaluation is fundamentally unchanged, although the negotiation between perspectives is made more explicit.

1.4 FILTERS: Addressing the issue of repre sentation, meaning and communication

'Filters' are computational agents that provide customized views of the design world under consideration, and support non-verbal communication by relaying information to other 'filters' in the system. 'Filters' are expected to cultivate the range of design solutions by allowing individual views and modifications of the design world, while automatically managing the flow of ideas to and from a shared solution space.

Specific disciplines have evolved special means of representing, manipulating, and communicating about their specific concerns. That is to say they have developed their own language. For example, the word "window" for an architect carries different connotations and issues from those of the mechanical or structural engineer. Because of the different worlds that these three disciplines deal with, they perceive different realities when experiencing the same artifact. Figure 1.3 illustrates that architects might associate the word "window", with issues of view and light, the mechanical engineer is primarily concerned with opening size, and orientation, while the structural engineer is concerned with span and loading.

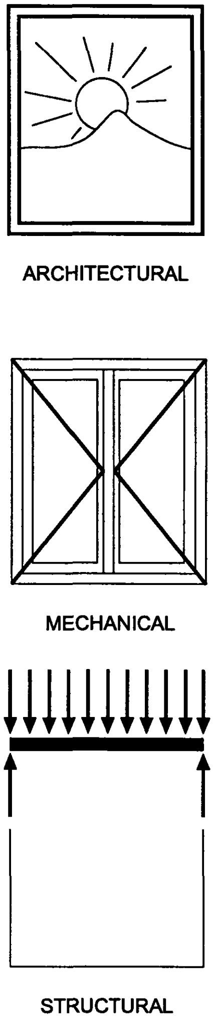
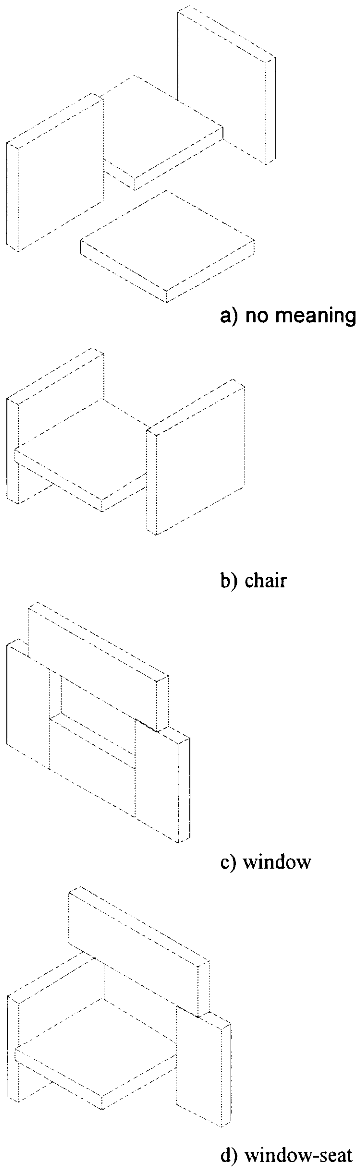


Fig. 1.3. Different Representations and Meaning for the same objects.

In nature, filtering has evolved to highlight relevant information. Jerry Lettvin [10] analyzed the signals which a frog's eye sends to the frog's brain. There are four separate operations in the frog's eye:

- Sustained contrast detection
- Net convexity detection
- Moving edge detection
- Net dimming detection

The frog's perception of the world is highly constrained. Before the brain ever receives information to process, the eye has produced information relevant to the frog's needs. From the available information the frog can determine the location of flies and the approach of large intruders. There is a whole array of other signals in the world that the frog simply ignores, but that other species have evolved to detect. Human designers also benefit from the mechanism of 'filters'. 'Filters' abstract and reconstruct the world into features that are relevant to the organism's success.



a) no meaning

b) chair

c) window

d) window-seat

Figure 1.4: Filters determine the semantic content of the world

In design the need to create selective representations of a greater complexity is also apparent. WIlliam Porter[11] suggests that at least three classes of description are necessary for the designer:

- Object descriprtion - elements of the design and location
- Formal description - edge or boundary
- Semantic description - ideas that reside outside

The filtering mechanism goes a long way towards problem framing. Omitting the importance of different representations for different tasks privileges one representation over others and greatly diminishes problem solving ability. Designers invest a lot of time producing different representations to solve different tasks. The important role of perception in an intelligent system is the issue. A paradigm often found in computation has been primarily one of direct encoding. A window is represented in the computer as a composite object with attributes and relations. The object is then labeled "Window". This paradigm ignores two aspects of visual perception:

- Perception selects, transforms, and reconstructs.
- Perception is informed and directed by "concepts" – one sees what one knows to look for.

A representation is needed which will allow the flexibility of perception to rapidly construct new readings. One way to satisfy this need is to try to anticipate the possible readings and provide a database full of constraints and relations that can address these readings. This work explores another solution where these relations might be imposed by perception, not in the structure of the database.

'Filter mediated design' takes the ontological position that geometric and topological data is the information available in the world, before filtering occurs. Any semantic information is generated in the process of filtering. This paradigm allows for multiple, overlapping and conflicting meaning structures to be read into the same design world. In the database, a window is described only in terms of the geometric and relational properties of its materials. It becomes a window only when a particular 'filter' views it as such. Figure 1.4 shows how a database of geometrical and topological data can be subject to multiple readings. The first image is read as nothing, The second image satisfies the "Chair" 'filter', the third image satisfies the "Window" 'filter', and the fourth image satisfies both the "Window" and the "Chair" 'filter'. These ideas are explored further in the Section 2 discussion on ontology.

Human perception is capable of rapidly switching focus. The ability to reframe a problem in a number of ways is a key to building coherence. George Stiny's [12] work on Shape Grammars represents a breakthrough from the computational paradigm of representing the world as finite components, asserting that designers' visual perception has the capacity to construct multiple readings from the same world. The ability to adopt and switch between different ways of looking at things is a powerful heuristic in the creative process. George Stiny's work states that the world is not an absolute from which one cuts out pieces, but that one constantly engages in reconstructing the world for different purposes, and that computational tools should, and can, work the same way. [a]

Stiny's work suggests that a component representation of data inhibits the designers use of ambiguity. Filter Mediated design shows that in the realm of architectural design a great deal of ambiguity can occur even if the world is represented as components, and that semantic representations of data is as great an inhibition to multiple readings.

The concept of 'Filter' contains not just the ability to select but also the ability to transform. The prototype described in Section 3, for example, constructs a notion of space from a database that contains no references to space. 'Filters' are the changing interfaces in constant negotiation between the environment and the organism that the 'filter' is serving[b]. Marvin Minsky concurs with the importance of 'filters' in the construction of intelligent agencies. "Each type of knowledge needs some form of representation and a body of skills to use that style of representation. Once that investment has been made, it is relatively easy for a specialist to accumulate further knowledge, provided the additional expertise is uniform enough to suit the same style of representation."[13] In design a 'filter' serves to select, focus attention on, and reconstruct information in the project.

[a] Nelson Goodman elaborates on this subject by proposing five ways in which we construct our world for our own ends:

a. **Composition and Decomposition** – World-making consists of taking apart and putting together.
b. **Weighting** –Ratings of relevance, importance, utility, and value often yield hierarchies rather than dichotomies.
c. **Ordering** – Nothing is primitive or is derivationally prior to anything apart from a constructional system.
d. **Deletion and Supplementation** – We find what we are prepared to find.
e. **Deformation** – The physicist smoothes out the simplest rough curve that fits all his data.

Goodman, N. Ways of Worldmaking, Chapter 1, Hackett Pub. Co, Indianapolis, (c1978).

[B] The question of what kinds of reconstruction are appropriate should be addressed. Piaget speaks of assimilation and accommodation as two facets of the adaptive process. These are also at play in the construction and maintenance of filters. Assimilation has to do with altering the perception of the world to make things fit what we already know. The complementary function, Accommodation, alters the filter when the experience of the world no longer supports the current representation Piaget, J. The Origins of Intelligence in Children pp 352-54; International Universities Press New York, (c1952)

‘Filters’ also can be used to address the issues of communication in collaborative design. When the number of participants in a design task increases due to increased distribution of knowledge, when the sharing of knowledge becomes increasingly mediated by communication technology, and when the methods of representation between disciplines become too highly specialized, traditional collaborative methods break down. Verbal communication, as a spatially and temporally constrained medium of exchange, can no longer serve as the preferred medium to connect this vastly distributed network because only limited numbers of people can talk at once. In addition, the mediation of face to face communication tends to diminish spontaneous interaction, as phone messages and e-mails disrupt the flow and opportunistic nature of discussion and discovery. In ‘filter mediated design’, much of the communication that is traditionally expressed verbally could be transmitted through ‘filters’. Also, by means of the ‘filters’, some of what is traditionally expressed silently, through the experience of the object, can be focused and sharpened. Communication through ‘filters’ could alleviate some of difficulties which different languages evolved in different disciplines place on understanding and generation of coherence.

Detecting both conflict and agreement is critical to the success of collaborative design. Figure 1.5 shows one application of using a ‘filter’ for communication: that of displaying agreement. The ‘filter’s’ interface could be similar to an overlay. In this scenario, the ‘Filter’ has determined through the actions on these elements of other ‘constructors’ and users where agreement is being generated. Many users have deleted the overhang, proposed by one of the ‘constructors’, so the overhang is fading out, and its declining value could be signaled with a black outline. As more users have accepted the parapet, it becomes more solid, and a white outline denotes gaining acceptance.

Other filtered views could communicate specific concerns like structural instability. A structural engineer's display could express an architect's desire to have the view unobstructed, warning the engineer whenever a move violates this principle, or an individual designer could alternate between different ways of considering the design under consideration. The 'filter' will consult with other 'filters' to analyze all user modifications and negotiate what is being displayed on the individual's view, such as highlighting the most important changes under consideration or abstracting extraneous information to the task at hand.

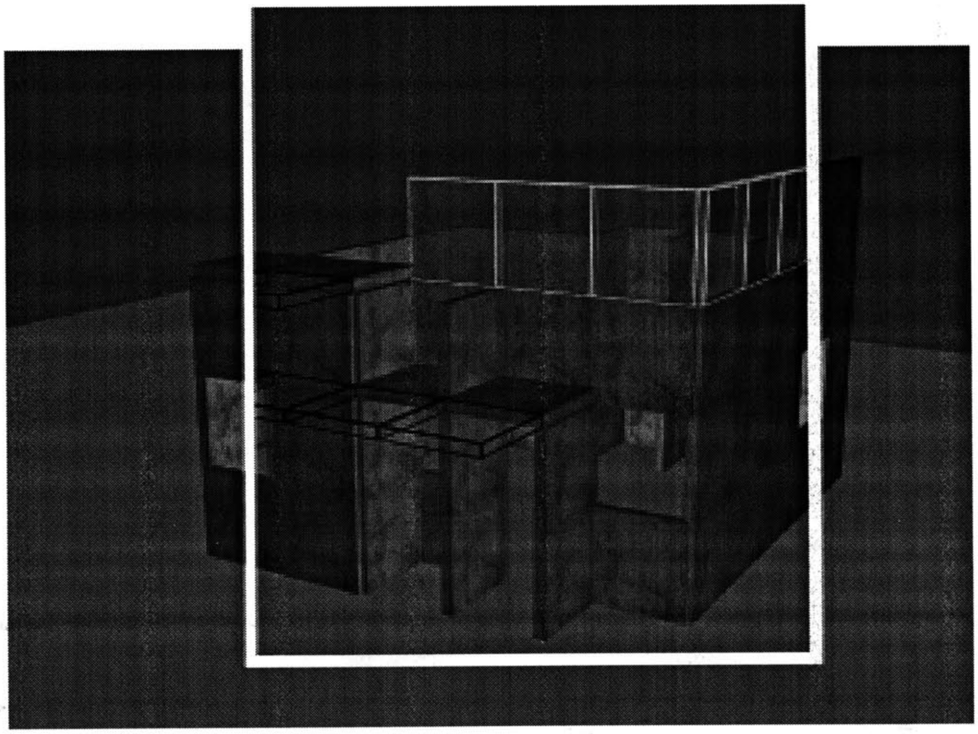


Figure 1.5: A filtered view as seen by user

1.5 CONSTRUCTOR – addressing the issue of memory and information processing

This research is concerned with how to merge human users and computers into a collective intelligence. Many combinatorial strategies place unbearable strains on people when too many constraints must be tracked. In addition, Humans often fail to determine clever strategies, due to immense or unperceived solution spaces. Finally, while the human perceptual system can switch between representations with remarkable flexibility, our ability to consider more than one representation at any time is constrained.[14] In a world of specialization and distribution of knowledge, the role of computational agents can become greater in addressing some of the issues described above. 'Constructors' become external memory and information processing aids when the knowledge becomes too extensive and distributed for the unaided mind to process.

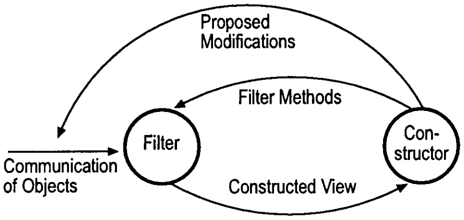


Figure 1.6: Interdependent, Evolving relationship between Filter and Constructor

In our conceptualization of 'filter mediated design', 'constructors' are analogous to Marvin Minsky's agents [13], and 'filters' are the lenses through which they perceive the world. 'Filters' and 'constructors' are in constant dialogue. If a given filter/'constructor' pair is not performing well two solutions are possible. One can either alter the 'filter', or alter the 'constructor'. In our formulation, the 'constructor' evaluates a view provided by the 'filter' and proposes a modification to the world. Just as there is a recursive, interdependent relationship between 'constructor' and 'filter', there is also an interdependent relationship contained within the 'constructor', between the evaluative and generative functions. Evaluative and generative strategies evolve in light of each other.

A great deal of the richness in design arises from just this interdependence. Design is a creative and open-ended process. Generative and evaluative strategies are in constant dialogue. This tension in turn informs the way one 'filters' the world, and preferred 'filters' settle in over time. As Piaget states "behavior becomes intelligent as pathways between the subject and the environment cease to be simple, rigid and unidirectional, but become more reversible and mobile, as the scope of the interaction goes beyond immediate and momentary contacts to achieve far reaching and stable relations." [15]

Each 'constructor' requires a specialized representation of the world, provided by a customized 'filter'. From this representation, 'constructors' can test for satisfaction, and generate to improve satisfaction. Figure 1.7 shows schematic representations for four 'constructors'. Figure 1.7a shows a 'light constructor', whose algorithm analyzes the amount of light cast on the floor from any south facing windows. Figure 1.7b helps the 'structural constructor' assure that any spanning planes are appropriately supported. Figure 1.7c is for the 'space constructor', which is interested in determining the amount of usable floor area. Figure 1.7d is used by the 'weather constructor' to confirm that a roof covers all floor area.

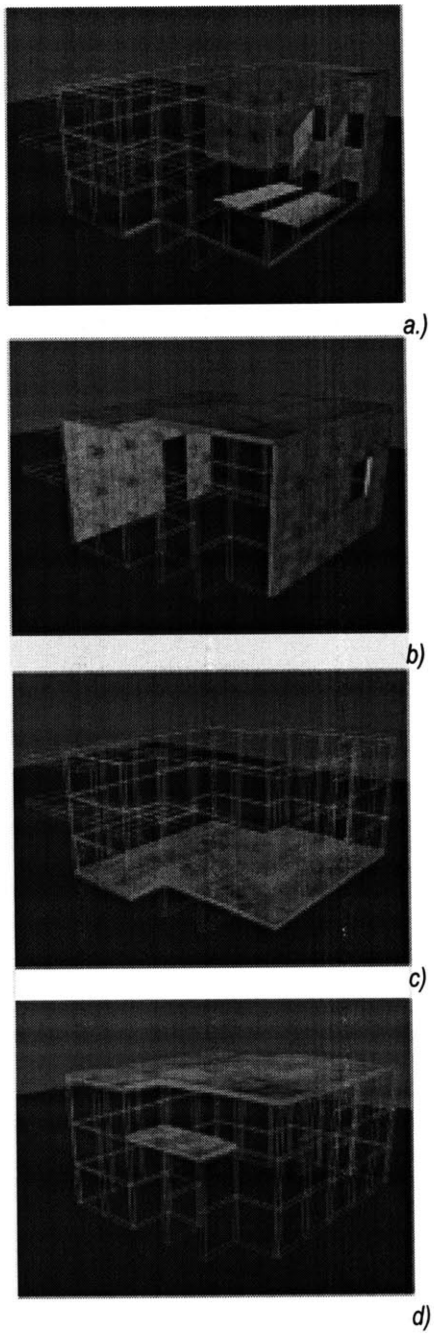


Figure 1.7: Filtered views for Constructors

Figure 1.8 shows a schematic scenario of how coherence might evolve from specialized 'constructors'. Interactions of 'space', 'economy', 'light', 'structure', and 'enclosure' 'constructors' collaborate in 12 steps to construct a somewhat coherent structure, addressing the concerns of each. First an empty site is presented to all 'constructors' (1). Then, the 'space constructor' provides rooms to it's liking (2), after which the 'enclosure constructor' covers the spaces (3). The 'structure constructor' recognizes unsupported roof area, and provides support (roof not shown - 4). The 'light constructor' eliminates the North, East, and West walls to allow for more light (5), and the 'economy constructor' eliminates any roof not covering its conception of space (6). The 'structure constructor' shores up the roof (7), while the 'space constructor' tries to reclaim space and to assure that all rooms are connected (8). The 'enclosure constructor' covers the new space (9) which causes a lighting conflict. Because of this, the 'light constructor' raises the roof (10), and the 'structure constructor' supports the raised roof (11). Finally the 'enclosure constructor' fills in exterior walls with glass (12).

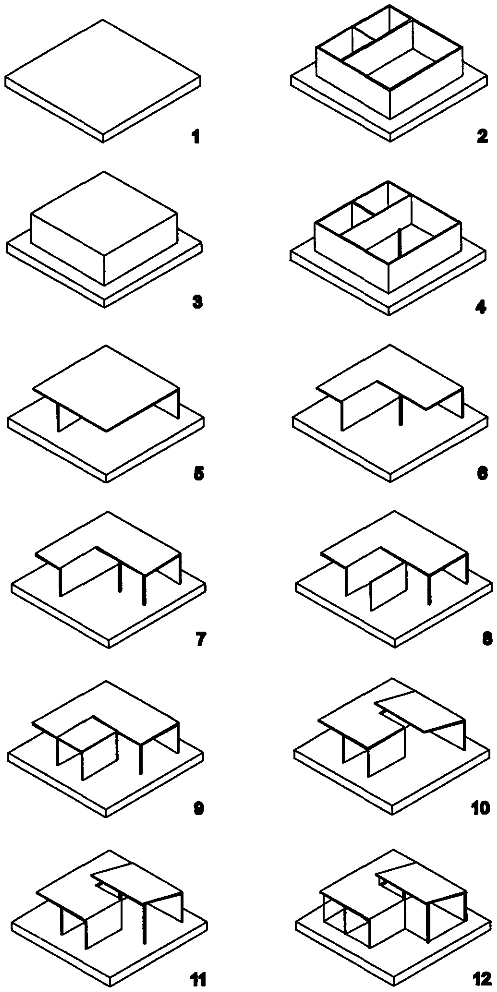


Figure 1.8: Emergent Coherence from Constructors

Finding ways to integrate human and computational designers is a major focus of the work. The 'filter' mechanism provides a useful interface to allow multiple design strategies and representations to coexist cohesively. At times, users may want to use their ability to appreciate, allowing computational agents to generate possible solutions. At other times, the users may want to freely generate, allowing the computational agents to keep track of the various constraints on the project and advise users of particular transgressions. The application proposed should ultimately allow users to 'switch gears' between different dialogic modes depending on how much control they wish to keep or delegate.

1.6 MANAGER – addressing the issue of coordination

In 'filter mediated design', 'managers' are behind the scene agents that facilitate the construction of coherence out of a plurality of views, by organizing the communication flow. Figure 1.8 suggests that coherence could be generated with a minimum of hierarchical control. When a particular 'constructor' becomes notably interested, it produces a proposal. Other 'constructors' are then left to consider this production in terms of their own evaluative function. Marvin Minsky proposes the need for a heterarchy, or dynamically changing hierarchy, in the construction of intelligence. [13] Herbert Simon states that this administrative power can come at a cost not incurred in decentralized systems, and concludes that different kinds of organization suit different tasks.[1] Benefits and costs are inherent in any kind of organizational system.

When information becomes specialized and increasingly distributed, what organizational strategies best serve our desire to construct coherence and innovation? The 'managers' are intended to explore this idea. Therefore any system constructed should be capable of implementing both hierarchical and decentralized structures and, most importantly, hybrids that contain both. Which aspects of coordination are best left to humans and which aspects could be handled through computation is a subject 'filter mediated design' is intended to explore. 'Managers' can be equipped with 'filters', so that they can alter their perception of the world as well. What they are interested in observing, however, is less characteristics of the object and more characteristics of the dialog between 'constructors', Users, and other 'managers'. What they evaluate and manipulate is not the object but the communication flow.

A 'manager' could, for example, observe the activities of 'filters' and group those that successfully match proposed modifications multiple times. The 'manager' would then ensure that the selected 'filters' would then propose their modifications primarily to their group of 'filters' and secondarily, after agreement has been determined within the group, to other groups of 'filters'. This procedure would allow users and 'constructors' with common interests to focus on a specific issue for a certain period of time without interference from those with other interests. 'Managers' might also notice dynamics of system communication and try to divert the flow when progress is not being made.

1.7 Filter Mediated Design is a tool for design, and for thinking about design.

As a Tool for Design

This work is interested in improving the communication required to generate innovation and coherence in design. When multiple participants and computational agents come together to evolve a design, the exchange of meaningful information becomes the primary concern. Generating coherence and innovation is the goal which, when left to an individual relying on verbal communication to understand the concerns of others, can be difficult. This work proposes that communication and generation of coherence could be mediated with 'filters'.

Among the things 'filter mediated design' will allow Users to:

- Adjust the 'filters' of various 'constructors', obliging them to focus on alternate aspects of the project,
- Alter the 'constructors' themselves, obliging them to adhere to differing evaluative and generative strategies.
- Manage the 'managers', privileging different 'constructors' or negotiation strategies over others.
- Share 'filters', adopting the point of view of other Users of ' constructors'

This work describes 'filter mediated design' in the field of collaborative architectural design. However, if one were to replace architecturally specific 'constructors' and 'filters', the underlying function of negotiation over data based on the specific points of view of individual 'constructors', 'managers', and Users could serve other fields where coordination of multiple perspectives is a concern.

As a Tool for Research

In addition to studying the processes and interactions at play in design, 'filter mediated design' will provide a useful platform for exploring ideas ranging from notions of self, to the location of meaning, to the understanding of 'complex systems'. The generation of coherence from distributed processes draws from the work of Marvin Minsky who proposes that concepts of Self emerge from the interactions of many individual processes. When a design is collaboratively generated, issues of self and authorship are called into question.

Generation of Coherence is a process of building shared Meaning. Meaning does not exist in the world, but rather is constructed by individuals and groups. Privileging perception, as a process instrumental in the construction of meaning, may help to understand the origins and maintenance of meaning. A computational understanding of the concept of emergence can begin to be explored.

Finally the mechanism, with the introduction of 'managers', is an ideal module in which to study the interactions of 'complex adaptive systems'. This paper stated that 'filters' and 'constructors' evolve in relation to each other and the world in which they operate. The method of coordinating these processes and the interactions between agents constitutes a complex system. 'Filter mediated design's' modular nature should facilitate the study of organizational systems capable of generating coherence and innovation from a distributed system.

2.0 Ontology and Meaning

"we take a handful of sand from the endless landscape of awareness around us and call that handful of sand the world....Once we have that handful of sand, the world of which we are conscious, a process of discrimination goes to work on it. This is the knife. We divide the sand into parts. This and that, here and there. Black and white. Now and then. The discrimination is the division of the conscious universe into parts."

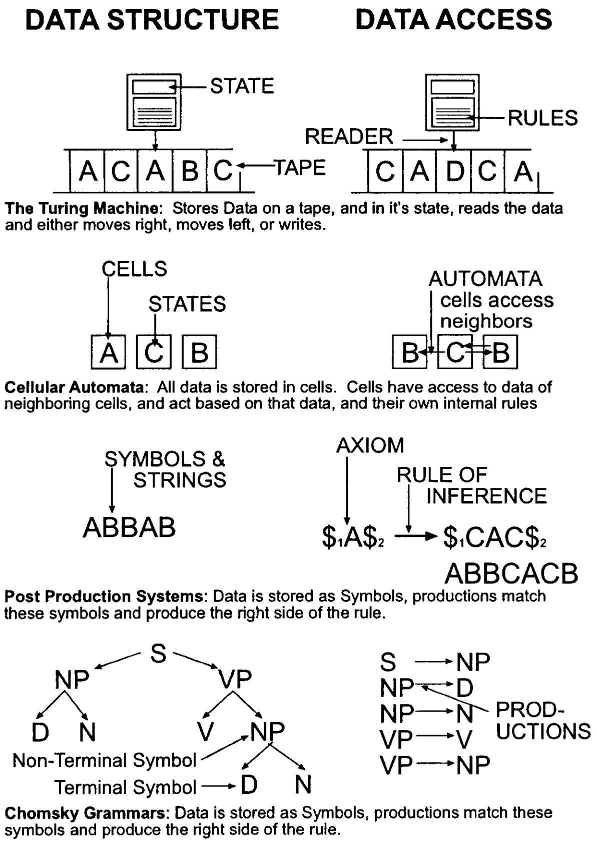
Robert Persig, Zen and the Art of Motorcycle Maintenance [17]

The previous section examined design as a relationship between processes and claimed that multiple readings and ambiguity are vital to the practice of design. This section examines design as a process of computation on a database, and illustrates that choices made in the structure and access of the database are critical when producing multiple semantic readings in a computational system is desired. First, a view of computation is defined, and this definition is applied to the domain of design. Next is an examination of some existing paradigms in representing design data, showing how these paradigms are limited in allowing flexibility and ambiguity. Finally this section explains how the database and access strategies implemented in 'filter mediated design' structure the design world in a way which is sensitive to the processes of design, allowing multiple semantic readings.

2.1 Design as Database Structure and Access

Computation could be viewed as a problem of two interrelated issues. The first issue is the represention of the data on which to perform the computation. The second issue is what are the strategies for accessing and manipulating this data. Of interest in this paper is the treatment of semantic data. The existing paradigm is to try to anticipate potential semantic relationships, and represent and label these relationships explicitly in the data structure. Data access simply searches for these labels to construct views. To count the number of chairs in the database, search for the number of labels on objects stating 'chair'.

Filter mediated design shows that when data is stored ambiguously, and the process of data access assumes a more active role similar to perception in determining semantics, a richer design environment emerges. Figure 2.1 shows a few theories of computation seen from the point of view of data structure and access. The first three theories do not privilege the location of semantic data. Each of these could be constructed to either passively read semantic data from data storage, or actively construct semantics in the process of access. The fourth, Chomsky's grammar, begins to imply semantics by imposing a hierarchical structure on the data, privileging certain relationships between the elements.



The Turing Machine: Stores Data on a tape, and in it's state, reads the data and either moves right, moves left, or writes.

Cellular Automata: All data is stored in cells. Cells have access to data of neighboring cells, and act based on that data, and their own internal rules

Post Production Systems: Data is stored as Symbols, productions match these symbols and produce the right side of the rule.

Chomsky Grammars: Data is stored as Symbols, productions match these symbols and produce the right side of the rule.

Figure 2.1: Computational theories seen as storage and access of data

To frame this view of design as computation, consider the database as the information in the world, and perception as the means to read this database. Action manipulates this database. Our actions and results of our actions are immediately absorbed into the database to be re-perceived. In Section 1, design was described as communication with the object, with others, and with others through the object. This communication must first pass through the ambiguous database, to be re-perceived.

This database we call the world contains much more information than we could possibly process. Our perception evolves in this database, constructing meaning based on experience. We learn a number of strategies for interpreting the information in the world in ways that are meaningful to us. Design is a process that operates on the infinite array of information, adopting a number of strategies for perceiving needs, and for improving the database to address these needs. Every action on the database is mediated through the act of perception. Perception evolves in a complex world, creating different ways to understand the aspects of the complexity in reliable and meaningful ways. Ambiguity arises when a number of methods of perception read meaning onto the same data in the world. An interesting and important aspect of design is in finding solutions that satisfy or interest a number of different ways of looking at the world.

2.2 On the Structure of Data

Information technology and design databases have emerged as a powerful tool for helping us to represent the world in which we want to design. We simulate the world, and perform operations on that simulation, as if we were operating in the "real world". The dilemma arises of choosing a representation of the world which we, ourselves need so many representations to partially understand.

Simulation: the imitative representation of the functioning of one system or process by means of the functioning of another

Merriam-Websters Collegiate Dictionary
10th ed, Merriam-Webster Inc., Springfield, MA

A tradeoff between computational complexity and level of ambiguity presents itself. George Stiny's theory of shape grammar proposes a database that is highly ambiguous, by relaxing all data to a maximal line representation. In the chair example, an immense amount of perceptual power is needed to sort through these lines, to determine where the lines make blocks. Representing the world as blocks implies more structure, less ambiguity,but also less computational load to determine semantics. The issue is how much structure to represent .

As Christopher Alexander points out " if the world were totally regular and homogenous, there would be no forces, and no forms"[18]. The world is ambiguous, but there is enough structure for some shared meaning to be constructed. We develop a similar, communicable understanding of the world because there is enough structure in the world for our experience of that structure to be shared. A chair has certain geometric properties, and our shared experience allows us to identify these geometric properties as constituting a chair, and offer it to each other as a place to sit. The meaning of a chair, can in some sense be shared. That there is structure in the world, that can be perceived by more than one, is the basis of language and culture.

There is nothing innate in the structure of this object we have jointly called chair that makes it a "chair". It is only the experience of it as a conglomeration of structure in the world that has allowed us to attach any kind of semantic description. We all have different experiences in the world, so our concept of chair can differ to some degree. A large beanbag can be a chair to the teenager, but not to the grandparent. In addition, we have similar other semantic descriptions which also fit such a structure. We can hang a coat on some chairs, stand on a chair to reach something, fight off lions with a chair, or use it to help define spatial separation in a room. It is not a chair, until we perceive it as such. And our perception is constantly generating meaning in the world, based on context.

Existing database systems avoid perception by explicitly labeling semantics in the database, and accessing the data by reading these labels. This paradigm ignores the creative, flexible and ambiguous powers which perception has evolved to possess. This paper will now look at an existing paradigm for representing design data, and explain how this methodology limits the emergent, creative activities of perception in design.

2.3 Existing Paradigm of Data Structure

The general paradigm has been to avoid the ambiguity inherent in a partially understood world, by representing the world from one semantic perspective. Various researchers try to resolve these issues by constructing definitions of entities with attributes and by explicitly representing the relationships between entities.

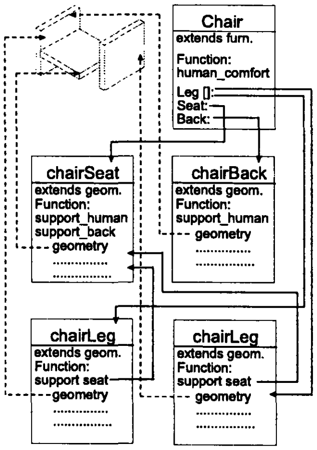


Figure 2.2: Representing a chair with semantics

Relationships between elements are often expressed through constraints contained within the object definition. What these models fail to provide is the benefits of an ambiguous database to designers. Figures 2.2 - 2.4 re-examine the issue of the window seat from the first section. This example shows the work that is required of a designer to represent that the window has also become a chair. If a designer does not complete this work, the design system is unable to recognize the structure as a chair. A simple geometric shift in the world can result in a dynamic semantic cascade. Expecting a designer to keep track of, notice and notate these changes would greatly restrict the opportunistic nature of design.

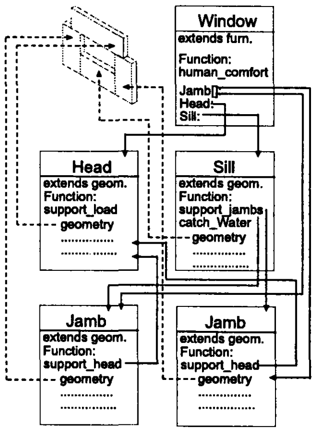


Figure 2.3: Representing a window with semantics

Figure 2.2 shows the hierarchical, object-oriented approach to defining a chair. Semantic Labels are attached to each of the objects (leg, seat. etc.) and these semantically defined objects are assigned to a higher level object 'chair'. Figure 2.3 shows the similar process applied to a window. Figure 2.4 shows in bold the work required of a designer who wants to create a window seat from a window.

'Filter mediated design' recognizes the need for a database structure which affords designers ambiguity and multiple readings. Size, and speed of access limit our databases. The more ambiguity present, the greater the search required to generate meaning from this database, so the challenge is to choose a representation that affords an acceptable level of ambiguity, while being mindful of the combinatorial explosion. Each addition to the database can generate additional computations at an exponential rate of the current size of the database, dependent on perceptual algorithms. The current implementation proposes a database which represents only geometric properties of blocks. Any semantic understanding of these blocks in generated actively by Filters.

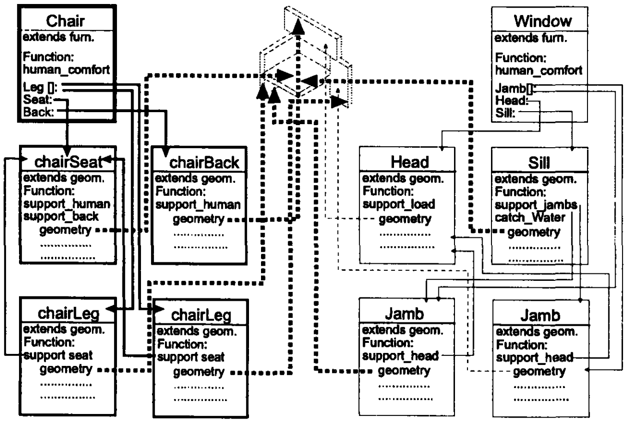


Figure 2.4: Modifying the window representation to also represent chair functionality

2.4 Existing Paradigm of Data Access

Representing the database semantically can cause an individual designer great difficulty when he wants to manipulate the data in ways other than how they were defined. When the database is to be shared, by other users, or computational agents, a semantically defined database has an even greater effect on the design process. One area of research involves the need for customized representations or 'views' of the data. A structural engineer needs a different representation of the project then the architect. The different views are achieved by labeling entities based on purpose (i.e. 'window') or function (i.e. 'provide_view'). The disciplines interested in these labels see the objects. When a new object is created, or an existing one is modified, the designer needs to be cognizant of who will be interestedin this design move. As shown in the window/chair example, however, a minor geometric change can have major semantic effects, causing different design disciplines to suddenly be interested in an aspect of the design that was previously not part of their domain. Figure 2.5 shows how a window and a chair definition would be labeled semantically to allow different disciplines to access data that is relevant to them. Different views then search the database for these labels, and read relevant objects.

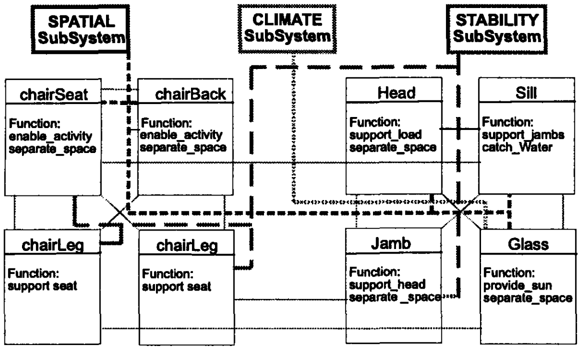


Figure 2.5: Creating views of the data based on labeled semantics

Another area of research is on the benefit of multiple versus centralized data models with focus on the maintenance of consistency between such models, detailing update rules that propagate changes between views. Figure 2.6 shows the window-seat constructed as two seperate, but related models. The 'same-as' constraint would change the window sill whenever the chair seat is changed. The 'constrained-by' constraint would adjust the dimensions or position of the window jamb when the window sill is manipulated.

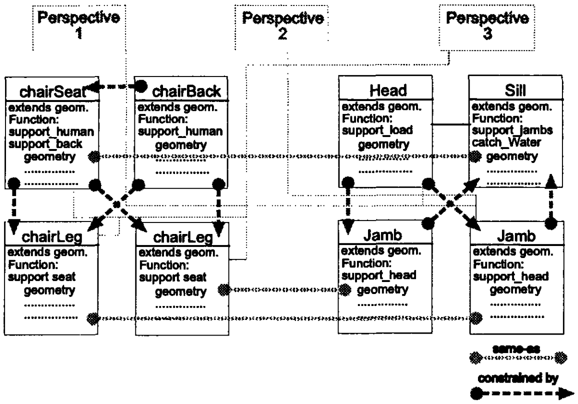


Figure 2.6: Multiple models require explicit relationships to maintain consistency

The work tries to simulate multiple points of view of the same object by constructing multiple objects, and defining relationships between the objects so the composite behaves as one. Maintaining these links, when the window becomes the chair would be even more difficult than in the example of Figures 2.2 - 2.4.

2.5 Filters Equipe Computers with the Power of Perception

Filter mediated design accesses the database through 'filters', the computational approximation of perception. These filters construct representations of the world which are suited for the task at hand. If it is a chair filter, the filter scans the database for elements which are approximately the appropriate height to be sat upon, with room for a body, and that looks like it is structurally supported. The window filter scans the same data looking for different criteria. Elements in the database are only interesting to the chair filter if they fit the filters criteria. A slight shift in geometry makes the difference, not a semantic label.

Persig notes that the world is infinitely complex, and that we are capable of only perceiving small pieces of it. As soon as we construct one understanding of the world, we privilege that understanding over others. There is no one representation that we can construct which can adequately address the reality in which we must navigate half blind. Our only recourse then is to develop a handful of representations, and learn which ones are useful in specific situations, and perhaps some relationships and consistencies between the representations. Design works in a similar way, as we sketch, build models, write poetry, generate spreadsheets, all in an effort to better understand the issues for which we design. Filter mediated design attempts to make this strategy of multiple partial representations from a vast, ambiguous database explicit.

Computational design environments are often used to simulate conditions and possibilities in the world. Filter Mediated Design takes the position that, while the information available in the world is infinite, it is not semantic, and that the act of perception constructs meaning. Therefore, the only data to be represented is that which resides in the world, no easy task when we are not entirely sure what the content of the database is, or what is relevant. Our only recourse is to carefully, sensitively, and playfully design a database that is useful and interesting to the task at hand. Simulation can use abstraction, we represent only the aspects that are relevant to the exploration. Ambiguity and multiple readings are relevant to the task of design.

"the machine that appears to be out there and the person that appears to be in here are not two separate things. They grow towards quality or fall away from quality together." [17]

Robert Persig, Zen and the Art of Motorcycle Maintenance

3.0 Implementation

This section details a prototype implementation of the mechanisms of design and ambiguous database detailed in the first two sections. The prototype implements five filters, four constructors and a simple mediator. Multiple users and computational agents can interact with a design simultaneously from separate work stations. This section will first describe how a user or users interact with the program, describing each of the features shown in Figure 3.2. Next, a technical view of the implementation will describe the program's structure. This technical discussion will first describe the Open Community platform developed by Mitsubishi Electric Research Laboratory in Cambridge, MA., which provides the platform for sharing large amounts of data over a network. Next, the technical discussion will explain the implementation program structure, which allows multiple points of view on the same data. Finally, the implementation of the filters, constructors and mediators is documented in detail.

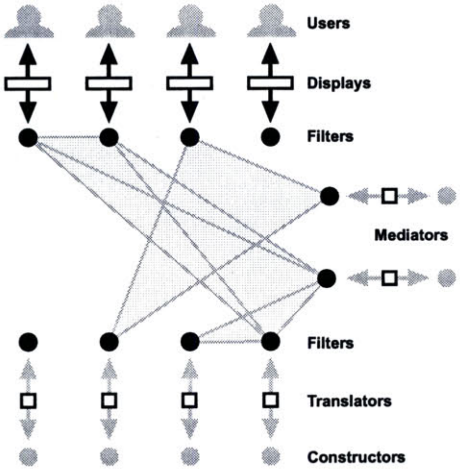


Figure 3.1 Overview of Filter Mediated Design

Figure 3.1 shows the overview of 'filter mediated design' as discussed in section 1. Figure 3.2 shows the menu for the implementation of the ideas expressed in Section 1. The implemented Filters, Constructors and Mediators are simple, addressing only first order approximations of the criteria. They were implemented to a level necessary to convey the idea of multiple criteria, but the content of these mechanisms is not the subject of the thesis. The prototype is intended to explore the processes of constructing meaning from a shared ambiguous database, and generating coherence between diverse, sometimes conflicting criteria.

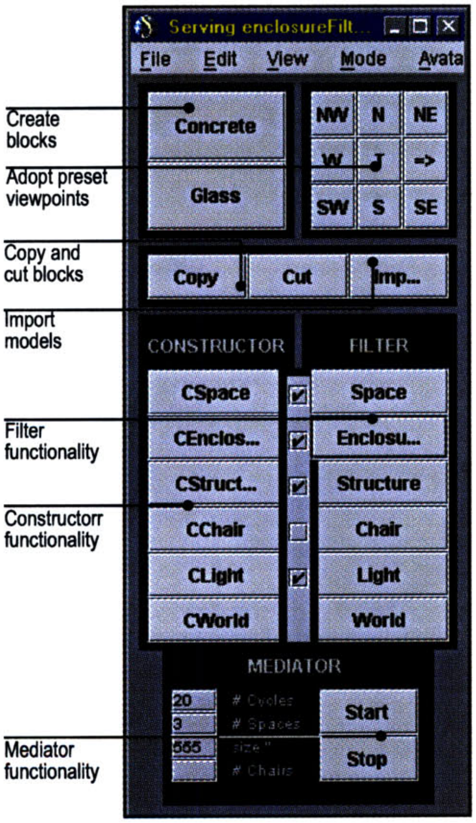


Figure 3.2: Menu for the Implementation

3.1 Interacting with the Implementation

The user starts up the program, and is presented the menu and rendering window shown Figure 3.3. The user can:

.begin constructing a new world
·join a world currently under construction by other users, or
·open a file containing a previously developed world.

For a particular session there is one 'worldLocale', which represents the information that is available to be negotiated. Any modifications are made in the 'worldLocale'. Multiple users can occupy this 'worldLocale', simultaneously viewing, adding, altering, and deleting data from multiple workstations. Only one user can manipulate the same object at any one time, but the object can be passed back and forth easily. Figure 3.3 shows the design environment.

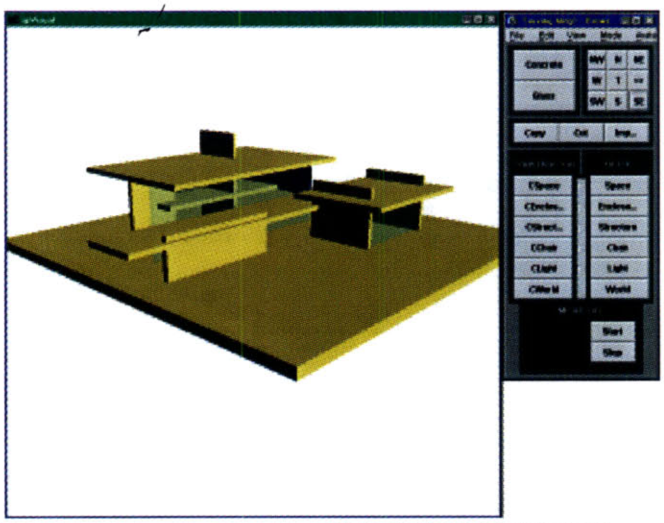
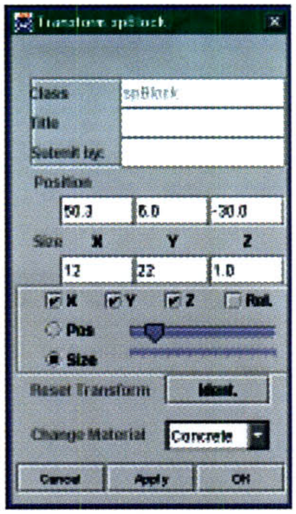


Figure 3.3 The design environment

The implementation creates a 'blocks world". The only elements in this world are concrete and glass blocks, which the program constrains to be stretched only in the X, Y, and Z-axis. Users can add new blocks, or cut and copy existing blocks. An existing block is manipulated by clicking on it, which makes that block the active block, and then either dragging the mouse to change it's X, Y, or Z position, or by typing in transform information which can alter any of the defining characteristics of the block (position, size, material). Figure 3.4 shows the transform type in dialogue, a window that 'pops-up' when a block is selected.



Transform spBlock

Class	spBlock
Title	
Submit by:	

Position

50.3	6.0	-30.0

Size	X	Y	Z
	12	22	1.0

☑ X ☑ Y ☑ Z ☐ Rot.

○ Pos

● Size

Reset Transform: Ident.

Change Material: Concrete

Cancel | Apply | OK

Figure 3.4 The transform type-in

Users can manipulate the position of their point of view by choosing one of nine preset locations around the object, or by selecting 'view mode' and navigating with their mouse or arrow keys to navigate through the space. Multiple users can select different points of view of the same 'worldLocale' simultaneously. Figure 3.5 shows multiple views of the same object from multiple workstations.

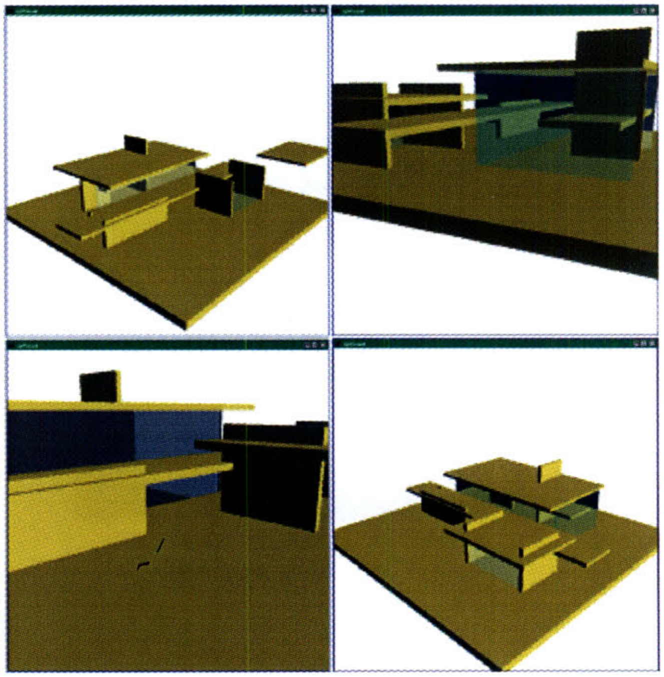


Figure 3.5: 4 simultaneous views

3.1.1 FILTERS

When in the 'worldLocale' a user can select a filter, which automatically constructs a view of the 'worldLocale' that is specific to that concern. This implementation constructs a new Locale, which is separate from the 'worldLocale', so that any manipulations done in the 'filterLocale' are not reflected in the 'worldLocale', although future implementations could explore other possibilities for filter / world or filter / filter interaction. For now, the filter is an opportunity to see the world from a particular set of criteria, imagine other possibilities, and refine concepts. The process of constructing a 'filterLocale' involves automated copying, omitting, transforming, and constructing new information from the information available in the 'worldLocale'. It is through this process that meaning is constructed from an otherwise ambiguous world. This process is analogous to perception. It is interesting that implementation of some filter views first requires a representation by the other filters. Concept formation seems to require input from other, previously defined concepts.

Space Filter

The space filter constructs spaces from the elements that it interprets as walls in the world. No spaces are defined in the 'worldLocale', rather the filter constructs them from it's concept of space. Spaces are presented as red translucent blocks of space. When two spaces overlap, the filter perceives them as being part of one larger space, and relates them as such. In the 'spaceFilter' the user is presented with a number of spaces, which can be manipulated, moved around, or deleted. Figure 3.6a shows a 'spaceFilter' view. Figure 3.6b illustrates that filter views can be navigated like any other view. See section 3.3.1 for more information about the implementation of the space filter.

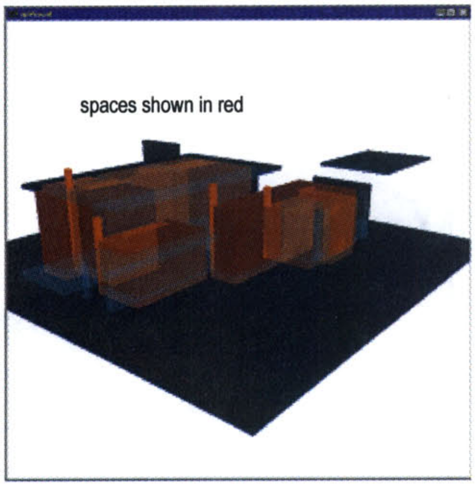


Figure 3.6a the spaceFilter

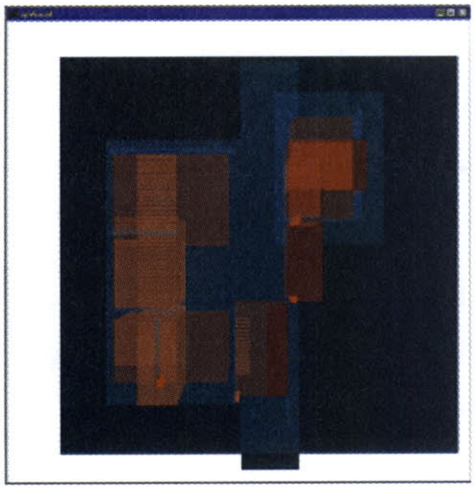


Figure 3.6b a plan View of the spaceFilter

Enclosure Filter

The enclosure filter takes the representation of spaces, and verifies which of these spaces are covered. Spaces that are entirely covered are rendered yellow, spaces that are not covered are rendered red. Possible roofs are rendered in green. The user can easily get a sense of where the enclosure issues are, and can see the potential roofs that might be used to cover these. Figure 3.7 shows a 'enclosureFilter' view. See section 3.3.3 for more information about the implementation of the enclosure filter.

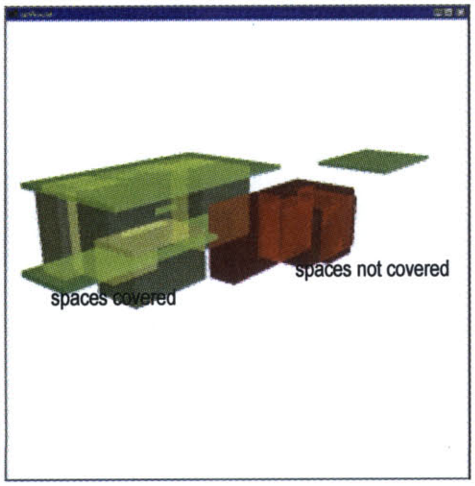


Figure 3.7 The enclosureFilter

Light Filter

The light filter takes the representation of spaces, and analyzes how much light can penetrate these spaces by checking what the material is of the walls that comprise these spaces. If a space is made of only concrete walls, the space is rendered red. If the space is made from a concrete wall and a glass wall, the space is rendered yellow. If the space is made from two glass walls, the space is also rendered as glass. The user can then easily get a sense of the quality of the spaces' light. Figure 3.11 shows a 'lightFilter' view. See section 3.3.5 for more information about the implementation of the light filter.

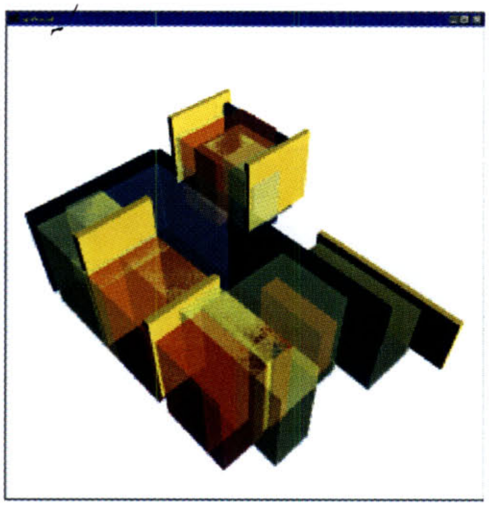


Figure 3.11 the lightFilter

Structure Filter

The structure filter looks for elements that are not touching the ground and verifies that they are supported by other elements. If the element is supported, it is rendered the color of concrete. If the element is only partially supported, it is rendered in translucent red. If the element is not at all supported, it is rendered in bright red. All the pieces that could be used to support are rendered green. The user can then quickly get a sense of where structural issues are, and can see pieces that might be used to remedy the situation. Figure 3.8 shows a 'structureFilter' view. See section 3.3.7 for more information about the implementation of the structure filter.

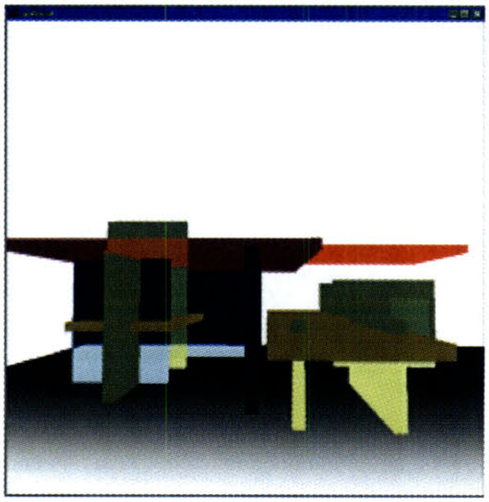


Figure 3.8 the structureFilter

Chair Filter

The chair filter looks for places to sit in the world. It searches the world for elements which are at an appropriate height for sitting, and superimposes an approximation of human size, to see if a human could sit in this location. Possible obstructions are rendered in green. Part of the chair filters criteria consults the structure filter to see if this seat is supported. When a possible seat is found, it is rendered as concrete, and all supporting pieces are rendered in yellow. When in 'chairFilter' locale, if a chair is clicked on, the supporting pieces are related to the seat, so that the chair can be manipulated, moved, resized etc. as a chair. Meaning has been constructed. In the 'worldLocale' if the same object is clicked on, the supporting pieces are not related to the seat. Figure 3.9 shows this idea. Figure 3.10 shows a 'chairFilter' view. See section 3.3.9 for more information about the implementation of the chair filter.

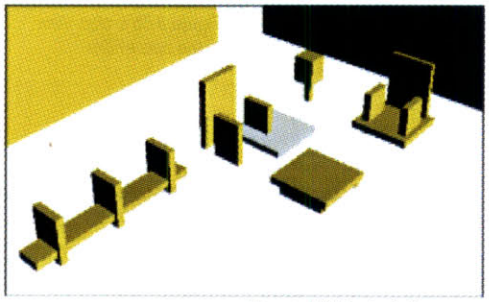


Figure 3.9 Manipulating a chair in the worldLocale

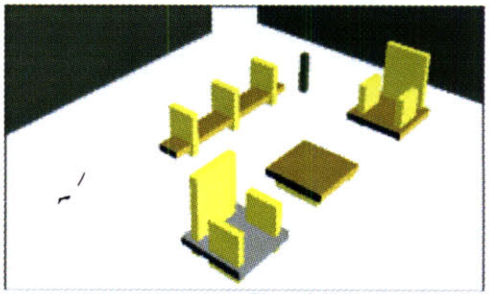
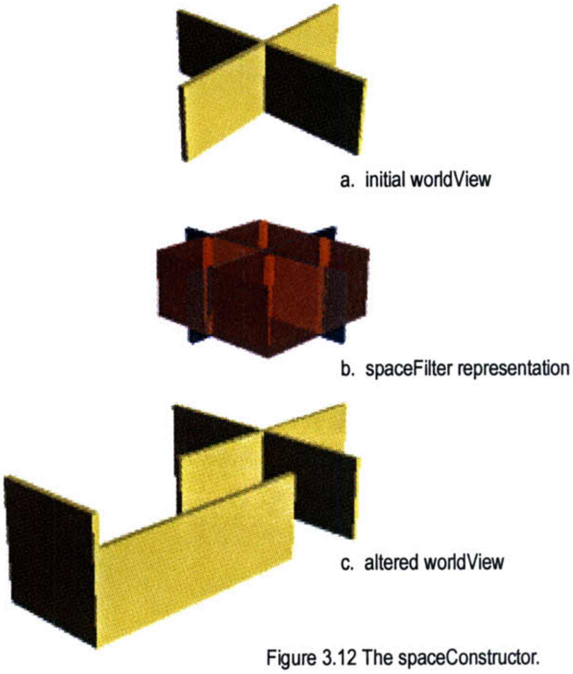


Figure 3.10 Manipulating a chair in the chairLocale

3.1.2 CONSTRUCTORS

Constructors analyze a filtered view, and alter the 'worldLocale' to improve the view, based on the constructors criteria. From a filtered view, the user can push one of the Constructor buttons, and the program takes the user back to the 'worldLocale' and improves the world based on it's criteria.. The process of constructing involves automatically deleting, transforming, or generating a block in the 'worldLocale'. The constructors below are very simple, but they do point to some useful roles for more elaborate constructors. Constructors can generate possibilities, based on the current state of the design, allowing users to freely generate. Constructors can automate tasks, taking care of the details and leaving designers free to do the more explorative work. Constructors can find the most efficient solutions, advising the users of some possibilities not considered. Finally, many constructors, when equipped with filters that allow them to constantly reconsider their generative strategies based on the current state of the design, can work together to produce many possible designs.


a. initial worldView

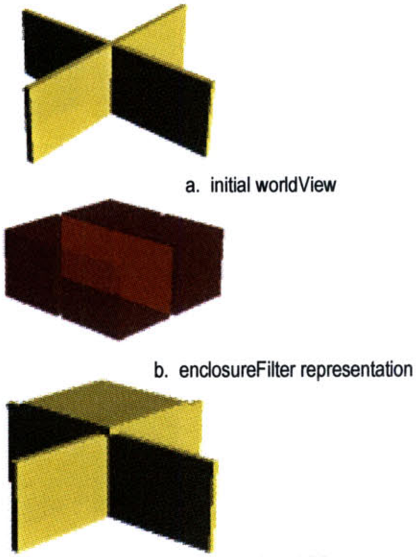
b. spaceFilter representation

c. altered worldView

Figure 3.12 The spaceConstructor.

Space Constructor

The space Constructor has as criteria, user input stating the number and size of the spaces required. A score is generated which estimates the constructors level of satisfaction. If a space is too small, the space constructor may try to enlarge it, or it may construct new space by placing new walls in the 'worldLocale'. Figure 3.12 shows a space constructor's action. See section 3.3.2 for more information about the implementation of the space constructor.


a. initial worldView

b. enclosureFilter representation

c. altered worldView

Figure 3.12 The enclosureConstructor.

Enclosure Constructor

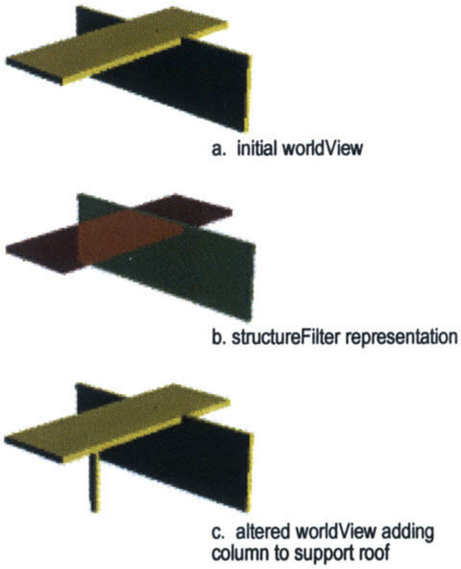
The enclosure Constructor takes the enclosureFilter's opinion about what spaces need to be covered, and generates a score which estimates the constructors level of satisfaction. The constructor covers one of the offending spaces, either by adjusting an existing roof, or generating a new roof. Figure 3.13 shows an enclosure constructor's action. See section 3.3.4 for more information about the implementation of the enclosure constructor.

Light Constructor

The light constructor takes the lightFilter's opinion about what spaces in the world need more light, generates a score which estimates the constructors level of satisfaction, and fixes one of the spaces by changing one of the spaces walls from concrete to glass. Figure 3.15 shows a light constructor's action. See section 3.3.6 for more information about the implementation of the light constructor.

Structure Constructor

The structure constructor takes the structureFilter's opinion about what structural problems there are in the world, generates a score which estimates the constructors level of satisfaction, and fixes one of the problems by dropping a column for support. Figure 3.14 shows a structure constructor's action. See section 3.3.8 for more information about the implementation of the structure constructor.


a. initial worldView

b. structureFilter representation

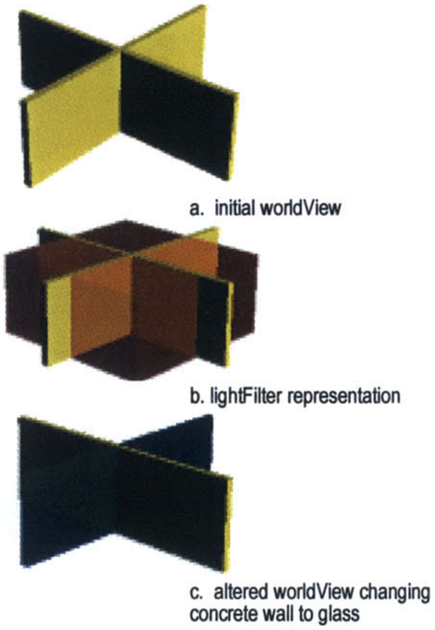
c. altered worldView adding column to support roof

Figure 3.14 The structureConstructor.

3.1.3 MEDIATORS

Figure 3.16 shows how a design can unfold through the interactions of filters and constructors with the design world. This thesis is in part about exploring the process of generating coherence from multiple perspectives. This example shows that coherence can begin to emerge when individual criteria are aloud to generate and judge in a shared world. However, the thesis also proposes a role for mediators, processes whose role is to coerce cohesion from distributed intelligence.

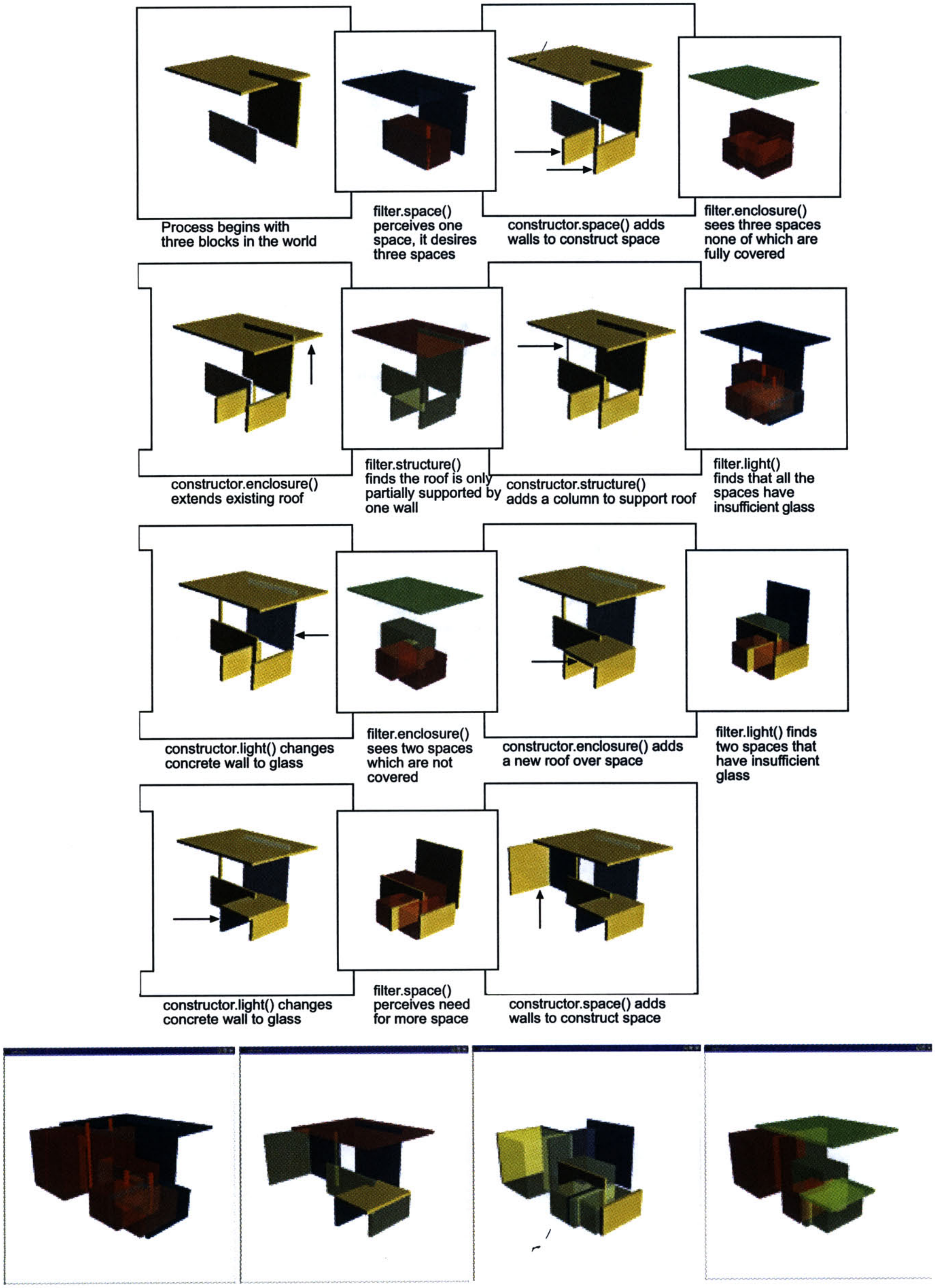
The program implements one simple mediator, but shows how a number of different mediating strategies could be explored in place of the implemented one. The current mediator allows a user to select which filter/constructor pairs to use in the session. The user than enters specific requirements for each of the filters (i.e. for space, the number and size of spaces would be entered) and the number of iterations the user wants to run. The mediator asks all participating filters to judge the world, and collects the scores estimating the constructors satisfaction with the current state of the design. The constructor with the lowest score is allowed to modify the world, and another iteration is begun. Figures 3.17 and 3.18 shows the results of two runs of the mediator, and the scores of each constructor after 20 iterations.


a. initial worldView

b. lightFilter representation

c. altered worldView changing concrete wall to glass

Figure 3.15 The lightConstructor.



Process begins with three blocks in the world

filter.space() perceives one space, it desires three spaces

constructor.space() adds walls to construct space

filter.enclosure() sees three spaces none of which are fully covered

constructor.enclosure() extends existing roof

filter.structure() finds the roof is only partially supported by one wall

constructor.structure() adds a column to support roof

filter.light() finds that all the spaces have insufficient glass

constructor.light() changes concrete wall to glass

filter.enclosure() sees two spaces which are not covered

constructor.enclosure() adds a new roof over space

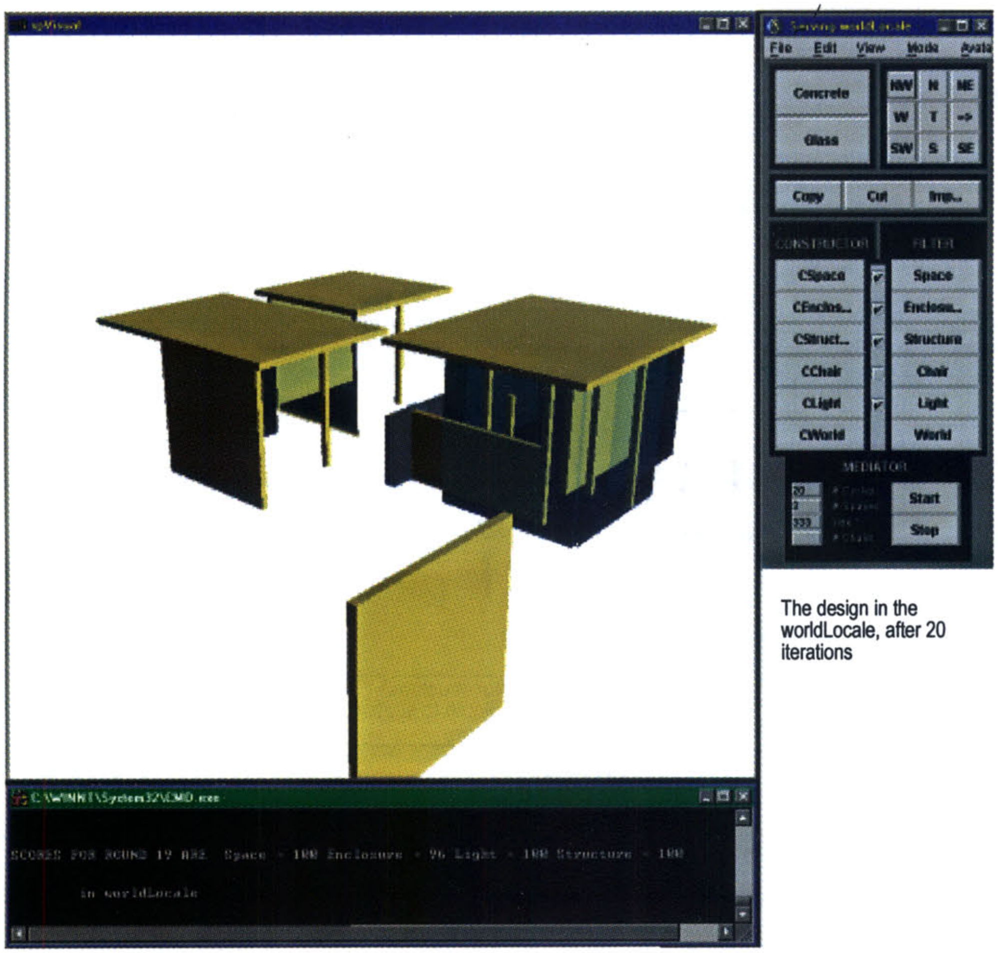
filter.light() finds two spaces that have insufficient glass

constructor.light() changes concrete wall to glass

filter.space() perceives need for more space

constructor.space() adds walls to construct space

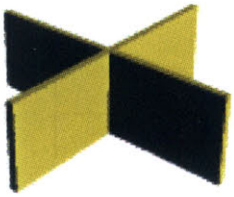
Figure 3.16 Task specific filters and constructors generate an object. Images along bottom show space, structure, light, and enclosure evaluations of the final design



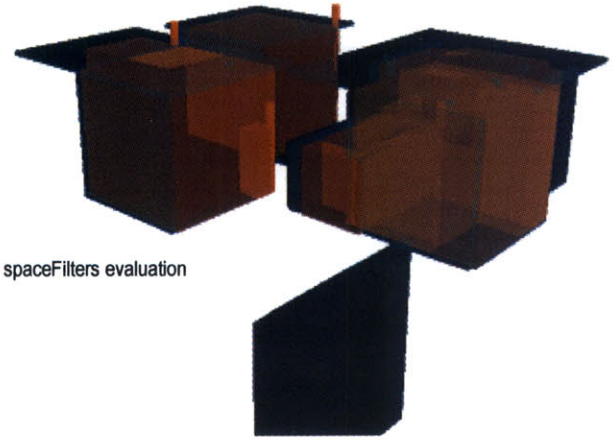
```
SCORES FOR ROUND 19 ARE  Space = 100 Enclosure = 96 Light = 100 Structure = 100

        in worldLocale
```

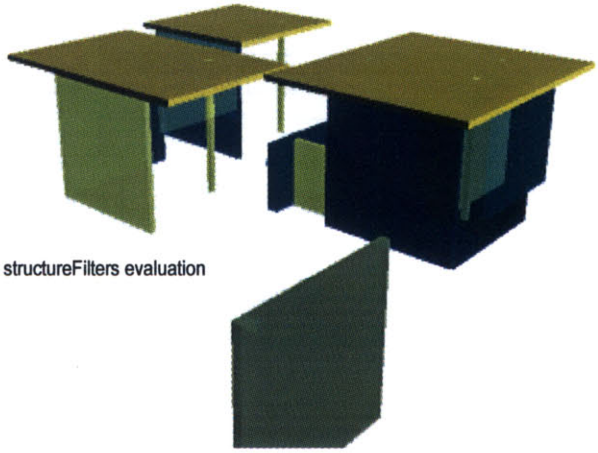
The design in the worldLocale, after 20 iterations



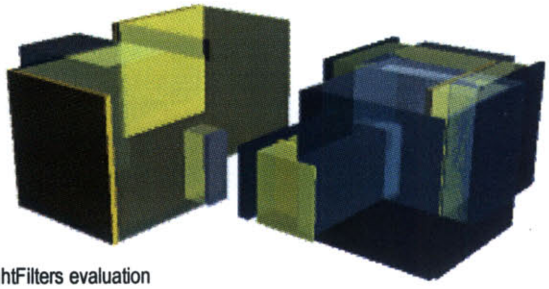
The design at Round 0



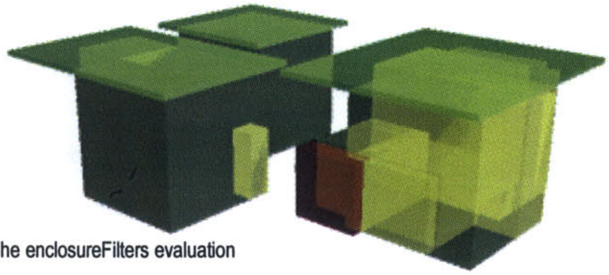
The spaceFilters evaluation



The structureFilters evaluation

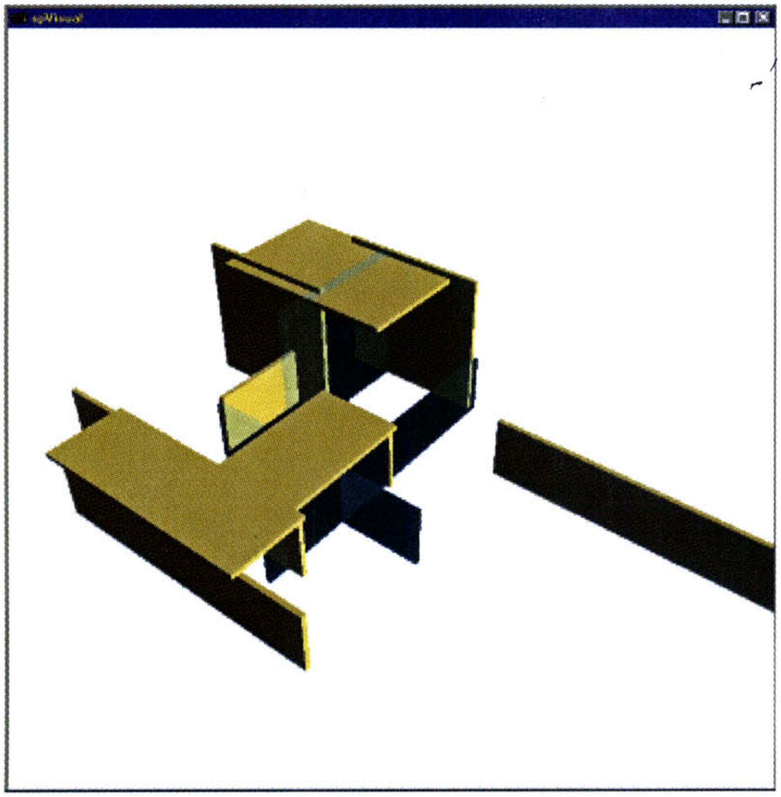


The lightFilters evaluation

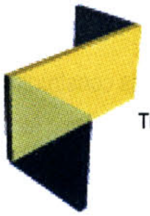


The enclosureFilters evaluation

Figure 3.17 A mediator runs for 20 iterations, allowing the constructor with the lowest level of satisfaction to generate.



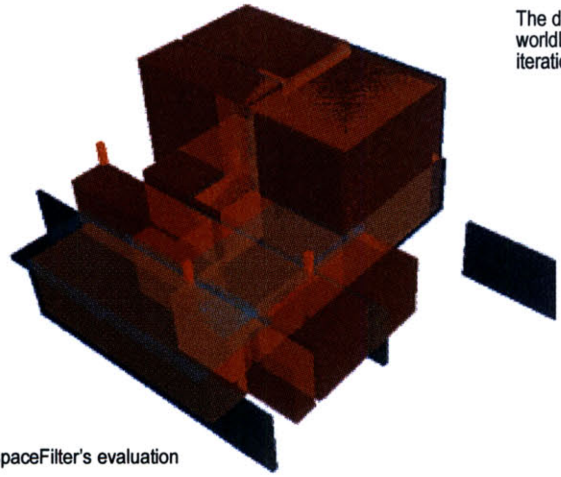
The design in the worldLocale, after 20 iterations



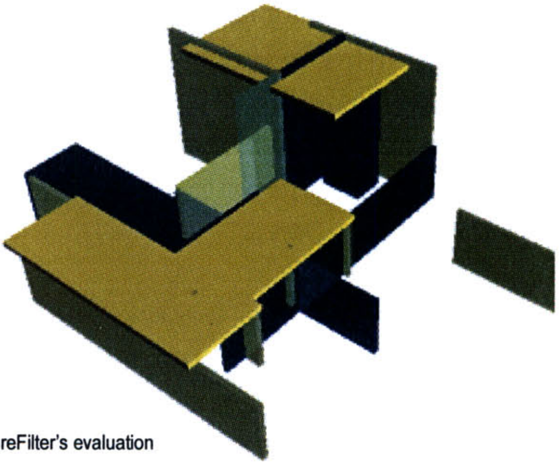
The design at Round 0

```
ROUND 0: Space = 0 Enclosure = 0 Light = 100 Structure = 100
ROUND 1: Space = 0 Enclosure = 0 Light = 100 Structure = 100
ROUND 2: Space = 0 Enclosure = 0 Light = 100 Structure = 100
ROUND 3: Space = 33 Enclosure = 0 Light = 100 Structure = 100
ROUND 4: Space = 33 Enclosure = 7 Light = 100 Structure = 50
ROUND 5: Space = 33 Enclosure = 23 Light = 100 Structure = 50
ROUND 6: Space = 33 Enclosure = 26 Light = 100 Structure = 0
ROUND 7: Space = 33 Enclosure = 26 Light = 100 Structure = 50
ROUND 8: Space = 33 Enclosure = 19 Light = 100 Structure = 0
ROUND 9: Space = 33 Enclosure = 19 Light = 100 Structure = 50
ROUND 10: Space = 33 Enclosure = 38 Light = 100 Structure = 50
ROUND 11: Space = 33 Enclosure = 26 Light = 97 Structure = 50
ROUND 12: Space = 33 Enclosure = 34 Light = 97 Structure = 50
ROUND 13: Space = 66 Enclosure = 26 Light = 96 Structure = 50
ROUND 14: Space = 66 Enclosure = 10 Light = 96 Structure = 50
ROUND 15: Space = 66 Enclosure = 14 Light = 96 Structure = 0
ROUND 16: Space = 66 Enclosure = 14 Light = 96 Structure = 50
ROUND 17: Space = 66 Enclosure = 17 Light = 96 Structure = 50
ROUND 18: Space = 66 Enclosure = 21 Light = 96 Structure = 100
ROUND 19: Space = 66 Enclosure = 21 Light = 96 Structure = 50
```

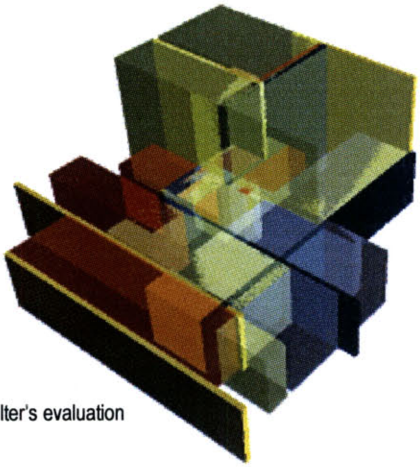
The scores at each iteration



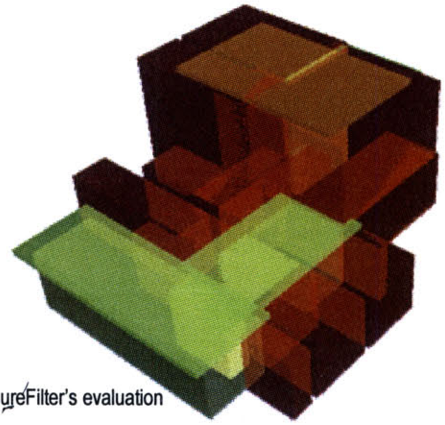
The spaceFilter's evaluation



The structureFilter's evaluation



The lightFilter's evaluation



The enclosureFilter's evaluation

Figure 3.18 A mediator runs for 20 iterations, allowing the constructor with the lowest level of satisfaction to generate.

3.2 Open Community

The implementation is constructed on top of Open Community, a prototype platform for sharing large amounts of data over the web. The implementation is constructed on a local area network (LAN). Multiple users and agents can simultaneously add, modify and delete data from a shared world model from multiple workstations. The implementation is written in Java, and communicates to Open Community, which is written in C, through a layer which modifies the Java Native Interface(JNI).

"Open Community is developed by Mitsubishi Electric Information Technology Center America. It is based on SPLINE technology developed at MERL (A Mitsubishi Electric Research Lab, and has been implemented through a joint development effort of MERL, Horizon Systems Lab (HSL), and the Information Technology Center in Ofuna, Japan. HSL is now supporting the platform.

Open Community is a software library, callable from ANSI C or Java, designed to provide many of the essential services necessary to make real-time multi-user cooperative environments possible. Open Community takes care of issues like network communications, real-time audio transport, application-neutral transport of large or complicated objects such as VRML models, and region-of-interest filtering.

Applications linked to the Open Community library see these services as a specialized form of shared memory in the form of an object database, called the World Model, that is replicated in each application. All objects in the world model are instances of classes provided by the Open Community library or user-defined classes derived from the Open Community base classes. These replicated objects are called "shared". **"[16]**

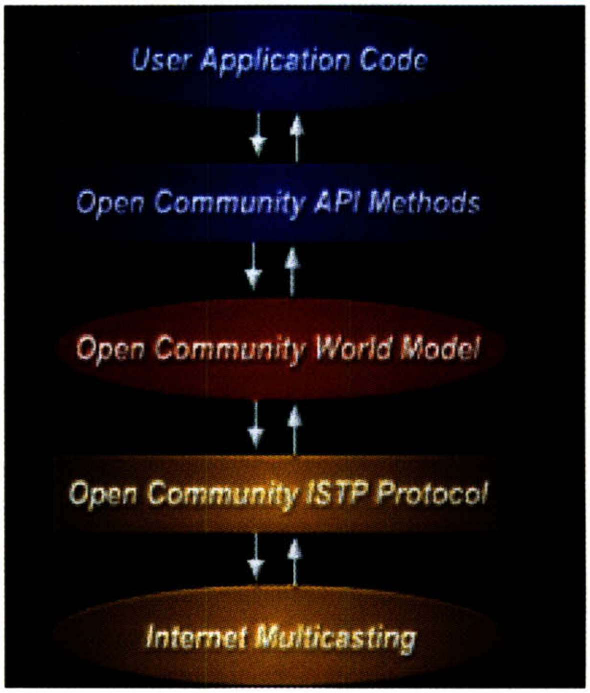

User Application Code
Open Community API Methods
Open Community World Model
Open Community ISTP Protocol
Internet Multicasting

Figure 3.19: The Open Community software Library

The Open Community library takes care of everything in the green space of Figure 3.19 - the user application code sees the in-memory database and access methods, and the Open Community method library makes calls onto the standard UDP Multicast Internet protocol.

3.3 The Program - System Architecture

The program is constructed using a notion of "Locales", a feature of Open Community. Locales uses a spatial metaphor to partition data. If a process is in one locale, it has access to all the information in that locale. Figure 3.20 shows the general structure of the program. The 'worldLocale' contains the shared design information. This is the ambiguous database described in section 2. Filters read this data, and construct domain specific representations in their specific locales. Constructors then read this filtered view, and act based on what they read. Constructors' actions, however, are performed in the 'worldLocale'. This paradigm is consistent with the role of perception as a mediator between our experiences in the world and our actions in the world.

Figure 3.21 shows a diagram of the distributed nature of the prototype, allowing multiple users to participate. This diagram suggests that each user might have their own specific domain, with filters, constructors, and mediators specific to their domain. Other users have access to information produced by these users, but the individual domain is responsible for maintenance of the filters, constructors, and data produced. Open Community then copies the information to users who are interested in the data. In this way, the design world is infinitely scalable allowing any number of designers with varying criteria to join the design process.

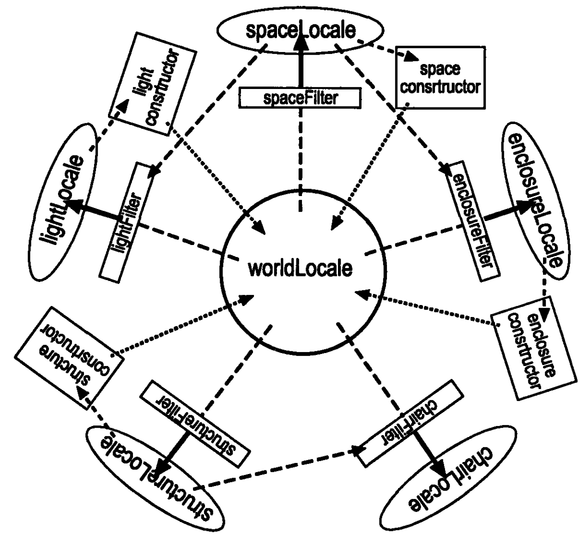


Figure 3.20: Program structure of Filter Mediated Design

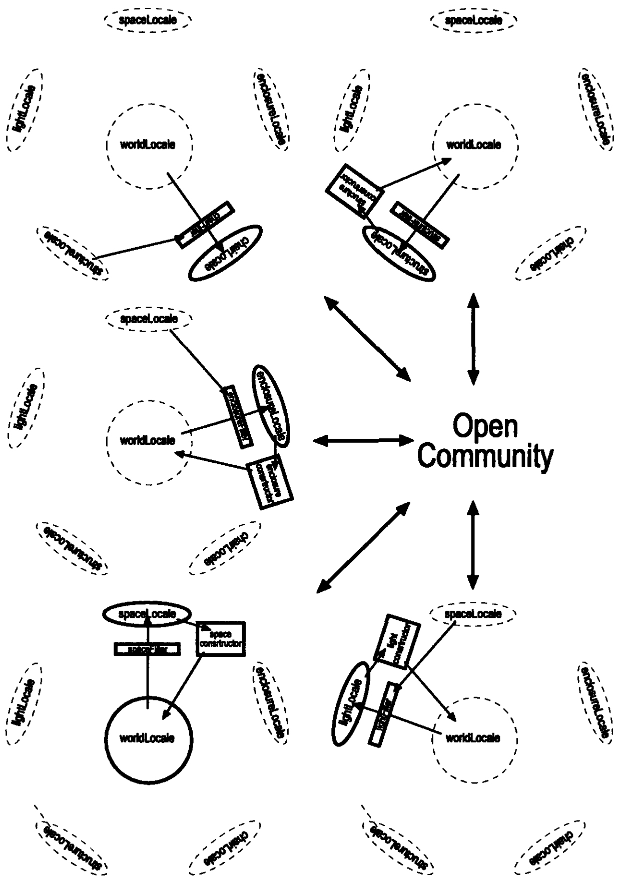


Figure 3.21 The distributed Intelligence Model, utilizing Open Community

3.3.1 filter.space()

1. filter.selectSpaceData()
World Locale Consists of collection of spBlock's, all parented to the locale. The space filter begins in filter.selectSpaceData() which selects every spBlock and places it in one vector.

2. filter.goToFilter("spaceFilter")
Moves the avatar and P.O.V. to the space Locale.

3. filter.constructSpaceData(VectorselectedBlocks)
Constructs two vectors, Roofs, and Spaces. The algorithm goes through selected Blocks, and copies the referenced spBlock into the spaceLocale. A reference is kept in the new block to the old block. If a block is horizontal, it is placed in Roofs, otherwise, a spatialWall is constructed, and this spatialWall is added to a temporary Vector. When all of the spatialWalls have been constructed, the algorithm compares every spatialWall to every other spatialWall in calcSpaces:

3a. SpatialWalls (spBlock wall)
Contain the wall, and four rectangles, which represent the shadow of thewall in the four orthagonal directions. The taller the wall the deeper the rectangle. The notion is a first order aproximation of the fact that taller might have more spatial impact.

3b. filter.calcSpaces(spatialWall1, spatialWall2)
This algorithm compares the rectangles of eachspatialWall with the rectangles of all the other spatialWalls. Where a rectangle crosses spaceis created. The vertical dimension is also considered. Once a crossing is found, a spaceObject is created which bundles references to the two spatialWalls and creates a new spBlock, which represents the space found. calcSpaces then calls :

3c. filter.addSpaceObject(spaceObject obj)
this algorithm manages the Vector spaces. Spaces contains a series of groupObjects, all of which contain spaceObjects. As each spaceObject is processed, it is compared to all the other spaceObjects, to see if they intersect (in the same manner as calcSpaces did for the spatialWalls). when an intersection is found, this spaceObject is added to the groupObject, and the space is childed to the groupObjects "key" (the bright red small blocks). The agorithm then goes through the remaining spaceObjects in the other groupObjects, and if there are any more crosses, the algoritm takes the current groupObject, copies all it's spaceObjects into the firstGroupObject in which a cross occured, and deletes the new groupObject.

4.groupObject.addVolume(spaceObject so)
when a spaceObject is added to a groupObject, this is called to add it to the groupObjects Vector of spaceObjects, and to calculate the groupObjects area, which does not count overlapped space twice.

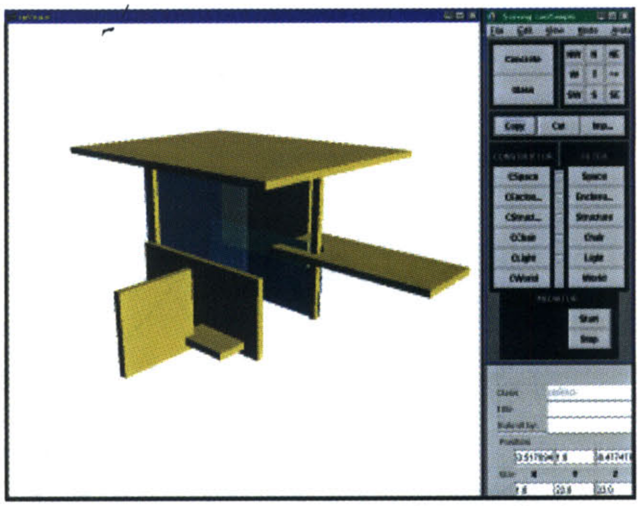


Figure 3.22 The World Locale

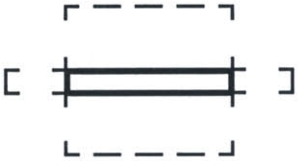


Figure 3.23 A Spatial Wall

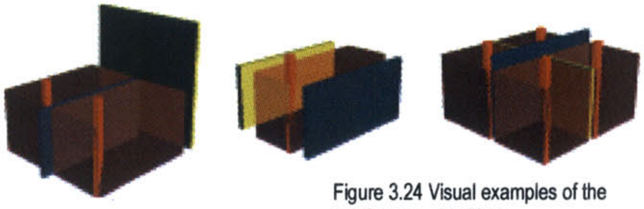


Figure 3.24 Visual examples of the calcSpaces algorithm

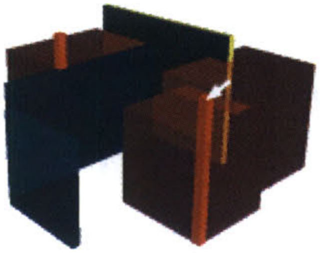


Figure 3.25 Image Illustrates how multiple spaces are children of the key block, so that overlapping spaces become one space in the group object

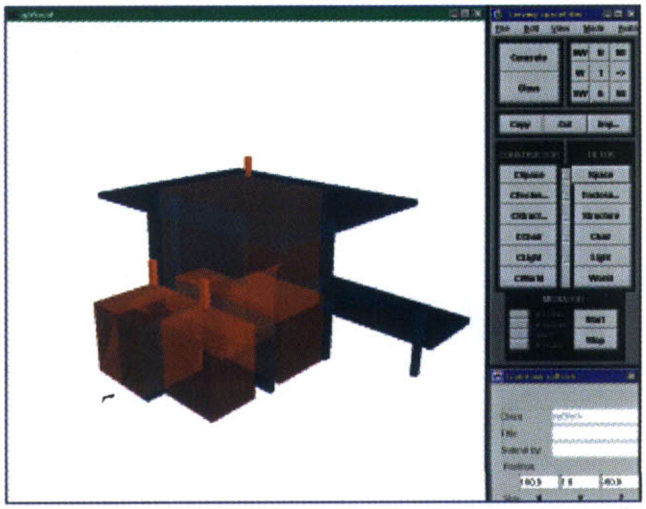


Figure 3.26 The Space Locale

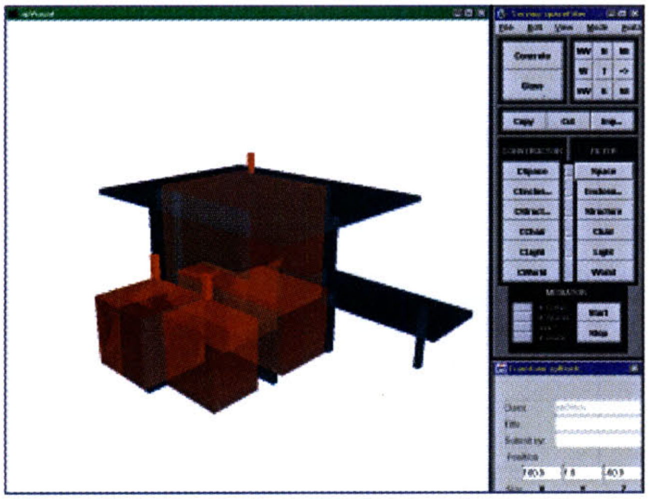


Figure 3.27 The Space Filter

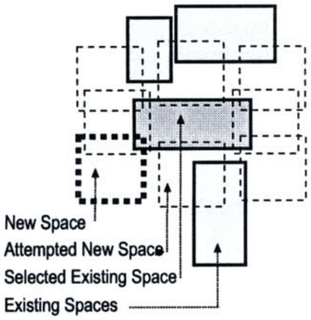


Figure 3.28 Modifying Existing Space

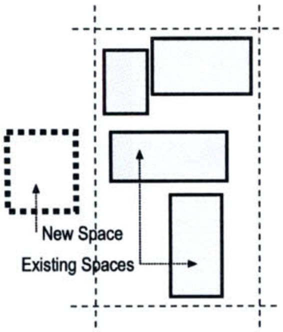


Figure 3.29 Creating New Space

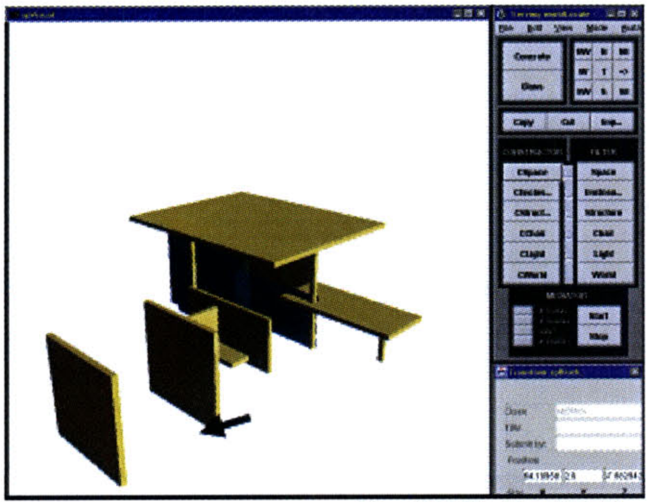


Figure 3.30 The Altered World

3.3.2 constructor.space()

1. constructor.judgeSpaceView()
filter.space() provides a vector of groupObjects called 'spaces', which represent spaces. The algorithm has hardcoded in that it wants 5 spaces at 300 sqft. each, but this can be overridden by user input. The algorithm simply goes through spaces, and counts how many spaces are the appropriate dimension. As it looks at each space, if the space is big enough it increments it's count of big spaces, otherwise, it puts the undersized space in a new vector called 'needsWork'. The algorithm then scores the scene by answer = ((bigSpaces/numSpaces * 100));. Another version subtracts 7 for each smaller space to try to clean up scattered small spaces.

2. constructor.changeSpace()
When this algorothm is called, the avatar goes to the worldLocale, with knowledge gained in judgeSpaceView.
This algorithm has a number of options for how to construct space based on the current state of the design:

if(spaceWork.isEmpty() && (bigSpaces == numSpacesReq)) :
first it checks if it's criteria are met, if so, the program doesn't construct anything, and returns.

if(bigSpaces > numSpacesReq):
if there are too many big spaces, the program looks for a bigspace and deletes any walls of this big space which do not help to make up another space.

else if(bigSpaces == numSpacesReq)
Here there is the right number of bigspaces, but there seems to be some small spaces left in needsWork, so i try to delete one of these small spaces by deleting any of this spaces walls which are not part of another space.

else if(!spaceWork.isEmpty())
if it gets here, there are not enough big spaces, and there are some smaller spaces in needsWork, so the algorithm picks a smaller space, and add some walls to it to make it bigger. It makes the walls just big enough so that the created space meets the criteria. See Figure 3.12 for modifying existing space.

else
there are no small spaces to work with, and the program still needs some larger spaces. It looks in the design space for some territory close to the the existing spaces, but not so close as to risk overlapping, and constructs a new space there. See the Figure 3.30 for creating new space.

3.3.3 filter.enclosure()

1. filter.selectEnclosureData()
If the algorithm is not currently in space, a space view must first be constructed. Once that is done, the algorithm goes through all of the groupObjects in 'spaces' extracting the spaceObjects, and puts them in a new vector called 'selectedSpaces'. Potential roofs for this algorithm were determined during the space filter algorithm.

2. filter. constructEnclosureView(Vector selectedBlocks)
The algorithm goes to the enclosureFilter, and copies all the spaces, and the roofs into the new locale. And puts the new spaces in a vector called encSpaces, and the new roofs in a vector called encRoofs.

The algorithm then takes each space, and generates an array of 9 points which are located as shown in the diagram. Then, the algorithm goes through each roof in encRoofs, and sees which ones are covering any of these points. A groupObject is created, (see below). Back in constructEnclosureView, all of the groupObjects are stored in a vector called enclosureObjs

2a. groupObject(spBlock spaceBlock, Point[] spacePoints, boolean[] spacePInt, Vector intRoofs)
This group object contains a space, a vector of the space's points, a boolean array of which points were covered, and all the partially covering roofs. If the boolean array is all true, the space is entirely covered, so the space is set to yellow color, otherwise, the space is set to red color.

3.3.4 constructor. enclosure()

1.constructor. judgeEnclosureView()
This algorithm simply counts the number of covered spaces in enclosureObjs and divides this number by the total number of spaces in enclosureObjs, and multiplies by 100. When it finds a space that is not covered, it places that groupObject in a new vector called enclosureWork.

2. constructor.changeEnclosure()
This algorithm selects a groupObject at random from enclosureObjs. If the space is not covered at all, it makes a new roof, which just covers this space. If the space does have some roofs which partially covers, the algorithm goes through to find which one covers the most, and extends that roof so that it covers this space.

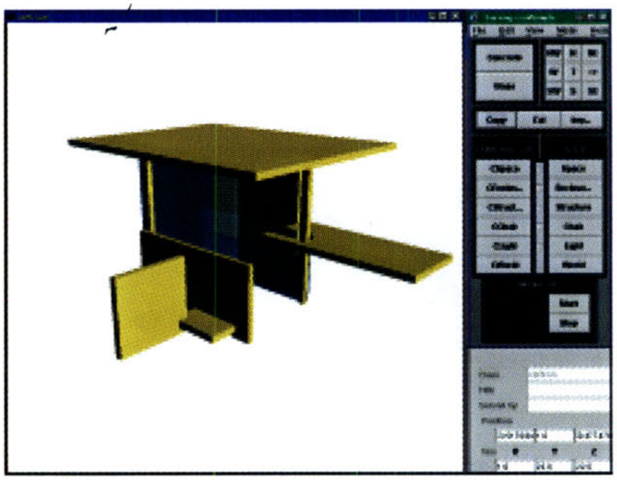


Figure 3.31 The World Locale

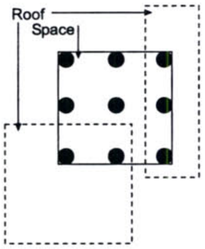


Figure 3.32 9 points are calculated on the space, and the roofs are then passed over to see which of the points are coverred

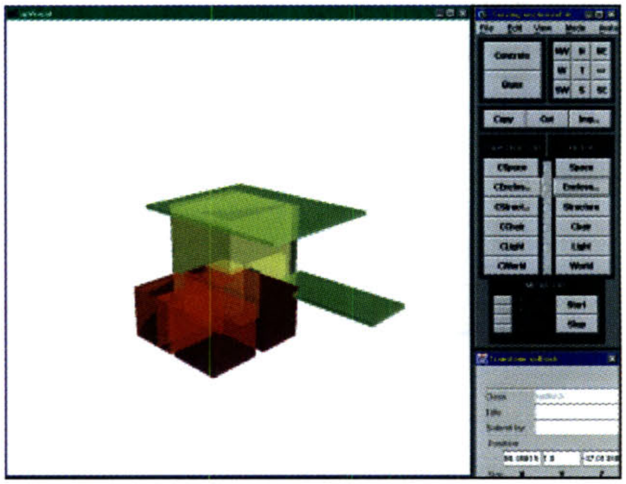


Figure 3.33 The Enclosure Filter

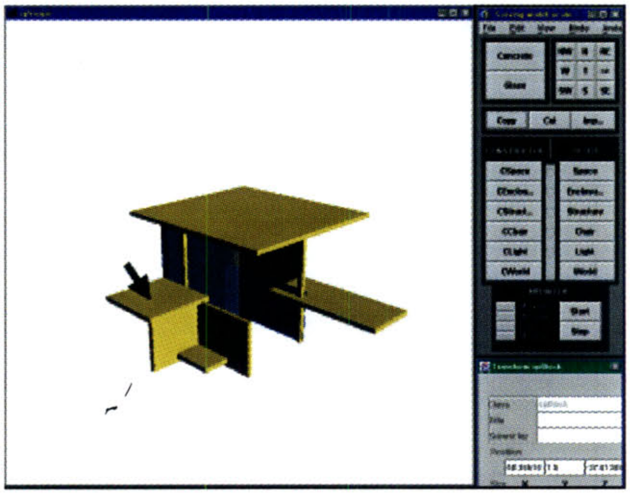


Figure 3.34 The Altered World Locale with an added roof

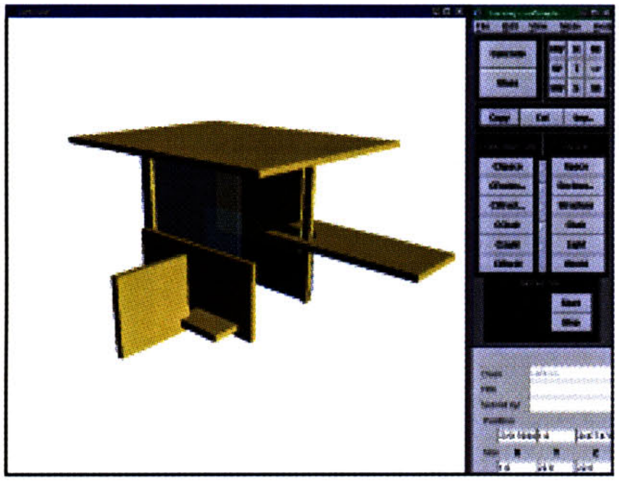


Figure 3.35 The World Locale

3.3.5 filter.light()

1. filter.selectLightData()
If the algorithm is not currently in space, a space view must first be constructed. Once that is done, the algorithm simply copies 'spaces' into a new vector called lightObjs.

2. filter. constructLightView(Vector selectedBlocks)
This algorithm goes through lightObjs, extracts the spaceObjects from the groupObject, and checks each space in relation to the walls that define this space. As it goes it creates a new space, and if these walls have not already been made in relation to a previous spaceObject, two new walls, to be placed in the lightFilter. If the space is defined by two concrete walls, the space is made red. If it is defined by one glass, and one concrete wall, the space is made yellow, if it is defined by two glass walls, the space is made the color of glass. New spaceObjects are made (each containing the space and the two walls that define it, and the spaceObject is placed in a new vector called lightObjects.

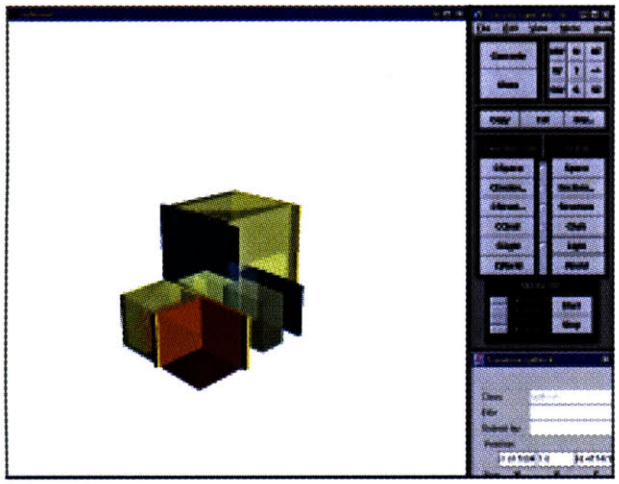


Figure 3.36 The Light Filter

3.3.6 constructor. light()

1.constructor. judgeEnclosureView()
This algorithm counts the number of well lit spaces and divides that number by the total number of spaces, and multiplies by 100. When it finds a space that is not well lit, it places that spaceObject in a new vector called lightWork.

2. constructor.changeLight()
The simplest of constructor, this algorithm picks a spaceObject at random from lightWork, and changes a wall at random from concrete to glass.

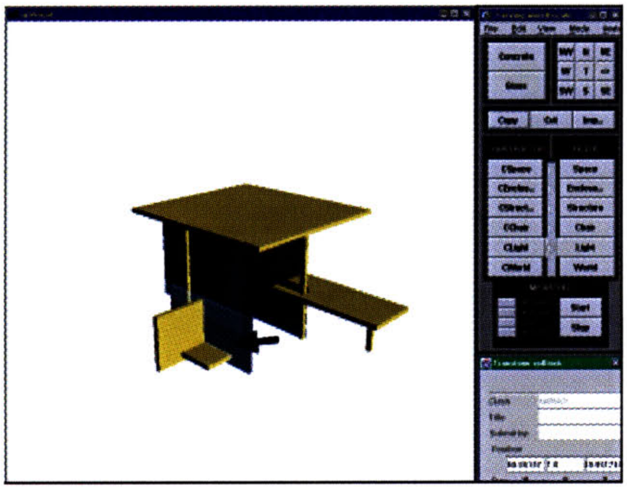


Figure 3.37 The Altered World Locale
with a concrete wall changed to glass

3.3.7 filter.structure()

1. filter.selectStructureData()
This algorithm goes through the world data, and selects all blocks that are above 1' in the air, as well as any blocks that are concrete, and touching the ground. Generally, glass touching the ground is not selected, because it is not useful to support things.

2. filter.constructStructureView(VectorselectedBlocks)
First, this algorithm goes through selectedBlocks, and creates new blocks for the structureFilter, and seperates these blocks into two vectors, 'couldSupport', and 'needsSupport'. The algorithm then takes each block in needsSupport, creates four booleans initialized to false which represents four sectors of the roof (see diagram) and calls blocksIntersect4(beam, col) on every member of couldSupport. The algorithm keeps track of which sectors of the roof have been intersected. Once the beam has been inspected with every potential support, a groupObject is created which references the supported roof with the supporting columns, and the boolean array which states where it is supported. If the beam is wholly supported, it is colored concrete, if it is partially supported, it is colored half red, and if it is not supported at all, it is colored bright red. The groupObjects which need support are stored in a new vector called 'structureWork'.

2a. blocksIntersect4(beam, col)
This algorithm creates four rectangles representing the four sectors of the roof, and checks if these rectangles are intersected by this element of could Support. It returns a boolean array representing which of the four sectors are intersected.

3.3.8 constructor.structure()

1. constructor.judgeStructureView()
The algorithm is aggressive, If there is one element not supported, it scores a 50, if two elements are not supported it scores a 0.

2. constructor.changeStructure()
This method takes a groupObject at random from structureWork, and sees where it needs support, and adds a column. Future work on this algorithm would have it look for a close wall that it might be able to use.

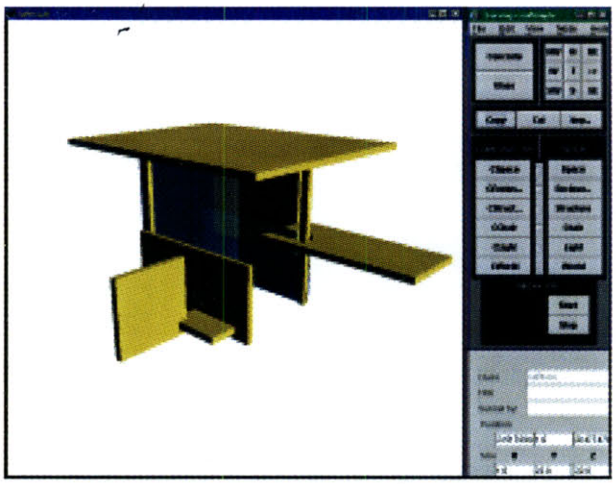


Figure 3.38 The World Locale

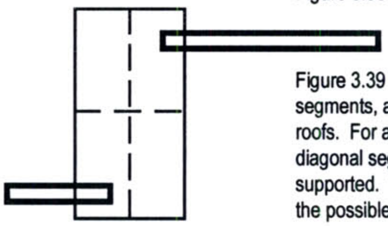


Figure 3.39 The roof is divided into four segments, and is checked against all the roofs. For a roof to be suported, the diagonal segments need to be supported. The algorithm assures that the possible support is tall enough

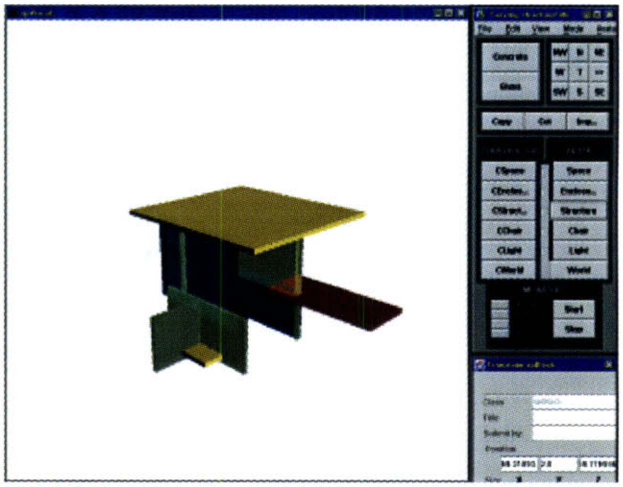


Figure 3.40 The StructureFilter

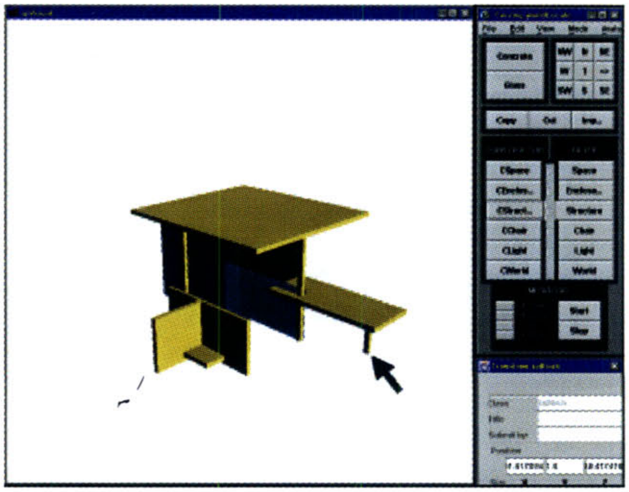


Figure 3.41 The Altered World Locale with an added column to support roof

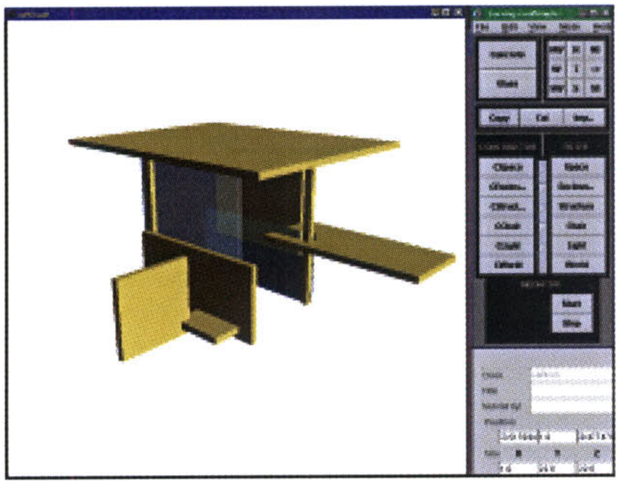


Figure 3.42 The World Locale

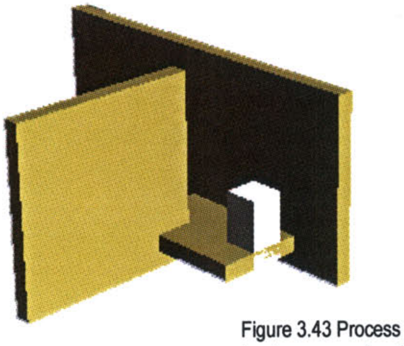


Figure 3.43 Process of finding a seat involves occupying the seat with a simulation of the human form

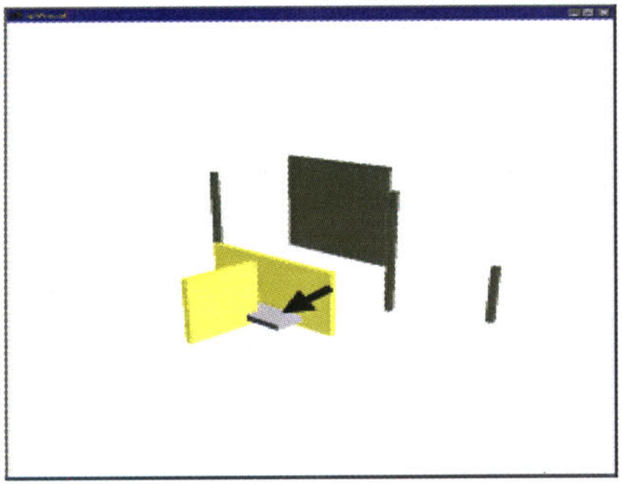


Figure 3.44 A chair is manipulatable as a chair in the chairFilter

3.3.9 filter.chair()

1. filter.selectChairData()

If structure has not been determined, the algorithm first calls structure. It then goes through structures 'structureWork', and selects any blocks which are of reasonable sitting height (ie, top of block is below 4') The algorithm puts all the possible seats in a new vector called 'possibleSeats' and all other blocks in a new vector called 'other'.

2. filter.constructChairView(VectorselectedBlocks)

The algorithm takes each possibleSeat element, and constructs a new vector of new blocks which simulates all the possible places to sit on the possible seat. Figure 3.43 shows one possible position, the vector contains all possible positions, by arraying these positions around the outside of the possibleSeat. The algorithm then cycles through each possible position, and compares this to each member of the vector 'other' to see if they intersect. The algorithm continues until it finds a position on the seat that does not intersect with anything in the world, this position is therefore open for sitting, and the search is called off. A groupObject is created of the chair with all the blocks in the world that intersect this chair.

3. tdBlockEditor.eventAction(tdVisualEvent event)

This algorithm checks if the user is inside the 'chairFilter', and if so, and the user has clicked on a seat, the algorithm dynamically sets all the seats intersecting blocks as children to the seat. Therefore, when in chairFilter, it is possible to manipulate the blocks as a chair. Meaning has been created.

3.3.10 constructor.chair()

not yet implemented

4.0 Future Work and Implications

"In order to think about thinking,
it is important to think about thinking about something"

Seymour Papert [13]

The prototype is intended as a tool to think with, as a design representation to be reflected upon in determining the next course of action. Processes observed in design can be situated in this framework, and suggest future developments of the framework. Areas of future work can be found in the ontological representation of the world, in the 'filter', 'constructor', and 'mediator' mechanisms, and in the communication between these mechanisms, as the prototype is taken from a coarse approximation to a more refined model of the processes of design. The process of constructing and operating the mechanism to date suggests future work in the areas of Ontology, Perception, and Communication.

4.0.1 FUTURE WORK ON ONTOLOGY

Filter Mediated Design implements a shared database consisting of geometric and topological data. Of course, this too is a representation of the world, but one that the prototype proves to be fairly useful in geometrically centered design. The main idea for multi-criteria information processing proposed in this thesis is to find a representation of the data to be negotiated that allows perceptual mechanisms to construct different semantic readings.

Design is not always geometry centered, sometimes it can start with a word, a financial situation or a social philosophy. The hypothesis of 'filter mediated design' is that the negotiation and construction of coherence occurs in the same way regardless of the domain. A word is just a collection of sound waves with intonation when it is inserted into the shared database. The word 'window' means something different to the architect than to the mechanical engineer, based on their many experiences. With geometry, one can choose a representation of 'blocks', or choose a representation that is more ambiguous, such as a shape grammar [12] representation. Similarly, with a negotiation over words, the central representation could be reduced to words, letters, or sound waves. It is the task of the 'filters' to determine the meaning, and to propose possibilities consistent with these meanings. The greater the ambiguity of the database, the more meanings that can be constructed. Understanding useful representations of ontology is a major subject for future research. Expanding the mechanisms to other, non geometrically centered domains will shed light on how to represent data to allow temporal, conceptual, financial and other meanings to be constructed.

4.0.2 FUTURE WORK ON PERCEPTION

Perception constructs meaning from ambiguous data. Developing a better understanding of how perception works, and how to make this process useful as a computational tool for users is an area of future work. The 'filters' constructed in the prototype were very simple, but a number of interesting observations emerged. The first was the exponential escalation of complexity with the growth of the database. Human perception necessarily struggles with a similar 'information overload' but through heuristics such as focus, the complexity seems to be greatly reduced. Ask a person to look about the room for things that are red. After they have done so, ask them what they saw that was blue, and they generally will not be able to answer. Perception seems to be very selective about what it chooses to process. Examining strategies to partition the database to limit the required processing will also be useful. Our perception benefits from only needing to account for what is relatively close. Future work involves learning from work done in both cognitive science and computer science in the realm of management of complexity.

Another interesting lesson involved the ability of 'filters' to make use of representations of other 'filters'. The 'chair filter' used the 'structure filter's' concept of support, and both 'enclosure' and 'light' made use of the 'space filter's' representation of spaces. A network of relationships begins to emerge, suggesting a hierarchy, or at least a heterarchy[13] of knowledge. More complex concepts, such as chair, are constructed from more fundamental concepts, such as structure. See Figure 4.1. Understanding and applying theories of concept formation from the domains of developmental psychology will be a major focus for future work.

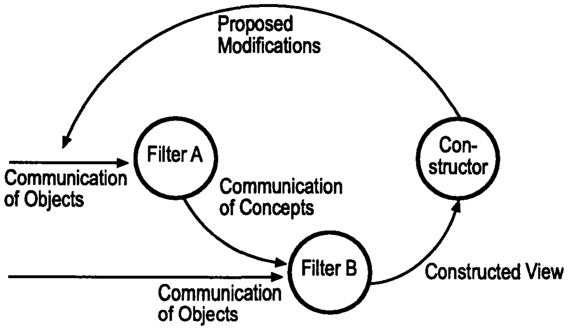


Figure 4.1: Filters can "share meaning" learning to use the concept structure of other 'filters' to develop their own meaning.

Beyond the construction of 'filters', exploiting the possibilities for designers using 'filters' is another major focus of the future work. Understanding how a user could manipulate a 'filter' to preserve intent is one area of work. A user could assign meaning to a given configuration, perhaps with some tolerances, then when another designer or agent tries to manipulate this object beyond the tolerances, the 'filter' could notify the designers of the conflict.

Filters promise to be useful design tools for understanding the design in different ways. The 'space filter' helps the user gain a different understanding of the design from the 'structure filter'. Providing users the ability to easily modify existing filters, and construct new filters, is an important avenue for future work.

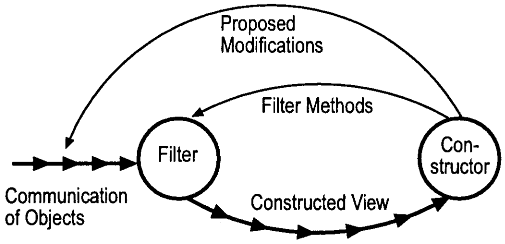


Figure 4.2a: Currently filters interpret the world "offline" giving the user a view which is seperated from the 'worldview'. Other functionality could explore a more active relationship between the filter and the world, with dynamic filtering.

Currently, a filtered view is constructed separately from the shared worldview. This is useful as an explorative device, as the user can then manipulate the filtered view to try different configurations, without affecting other 'filters'. It will be interesting to have other functionality, where objects manipulated in the filtered view are registered immediately in the worldview so that other 'filters' can give feedback. See Figure 4.2a. Other functionality could explore overlapping 'filter' views, so that, for example, 'chairs' can be understood in terms of the 'space' that they occupy. See Figure 4.2b. The mediation of perception provides many opportunities for exploring and communicating design possibilities.

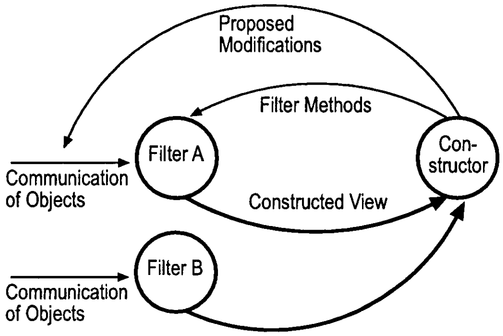


Figure 4.2b: Overlapping Filter Views can be used to give users or computational 'constructors' an understanding of the world that no single filter can give

Exploiting the creative and powerful processes of perception in design is a major and potentially fruitful focus of future work.

4.0.3 FUTURE WORK ON COMMUNICATION

Design usually starts from a vague concept, and some artifact representing this concept is produced and shared. Different perspectives then perceive this artifact, construct meaning based on it's own criteria, and change the artifact, or produce a new one, and share it. Design is a peculiar game of catch, where the object is refined and clarified with each iteration. This artifact can be geometrical, verbal, temporal or anything else that can be perceived. This artifact can not be semantic. Meaning is constructed, not shared. Communication must pass through the ambiguous database, and meaning is reconstructed on the other side. This game of catch, it could be argued, even exists within a single designer. Sketching, model building and talking to oneself could all be seen as ways of communicating design possibilities from one set of mental processes to another.

The 'Future Work on Perception' section stated that certain 'filters' are able to make use of other 'filter's' representations. The 'chair filter' consults structure. Intelligent systems that do not rely on perception for communication may not have to go so far in reducing artifacts to ambiguous representations to share concepts. Perhaps a higher level of meaning sharing can occur. Certain processes may learn to tap into the meaning structures of other processes. Mediation of the design process with 'filters' may allow us to begin to share meaning in interesting and useful ways. An architect could pass an electronically recorded understanding of space to the lighting engineer in the form of a 'filter'. Understanding the heterarchy of knowledge and the patterns of communication that could enable the construction of larger knowledge structures is an area of future work.

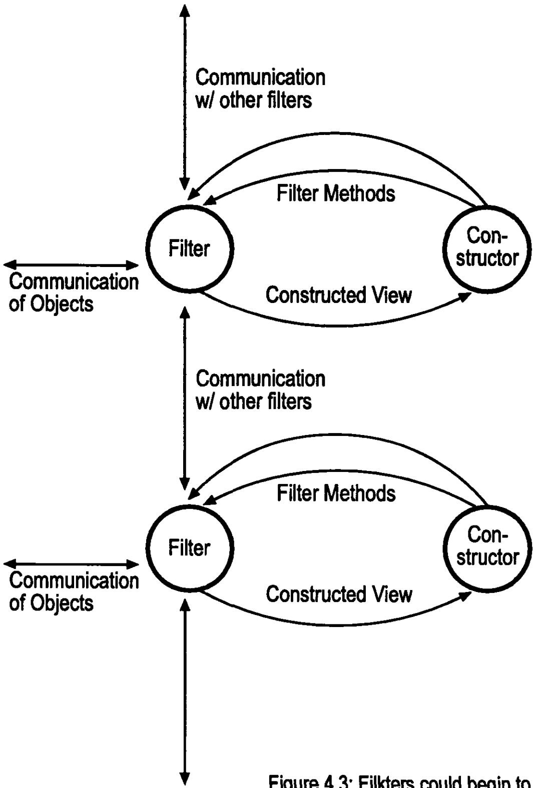


Figure 4.3: Filkters could begin to negotiate solutions directly, submitting consensus changes to the world

The negotiation of design possibilities is a related area of exploration for future work. Perhaps ‘filters’ could communicate directly with other ‘filters’ when negotiating possible changes to the database. See Figure 4.3. The ‘chair filter’ could learn to always consult the ‘structure filter’ when constructing a chair, and their ‘constructors’ could then submit a joint proposal to the world. A network of communication between ‘filters’ could emerge, and ‘mediators’ could learn to manage the flow of communication. Filters could also learn to negotiate and communicate issues like agreement, as shown in Figure 1.5, or notify users and agents of areas that need their attention. Research in issues such as conflict management from the management domain, and concept formation from developmental psychology will inform this line of inquiry.

‘Mediators’ could also learn to structure the design flow by constructing branching tree searches, and preserving a recent history of design to allow for backtracking. See Figure 4.4. Understanding how designers traverse these trees, and developing computationally similar processes will be an aspect of future work, applying advances made in design inquiry and computer science to these concerns.

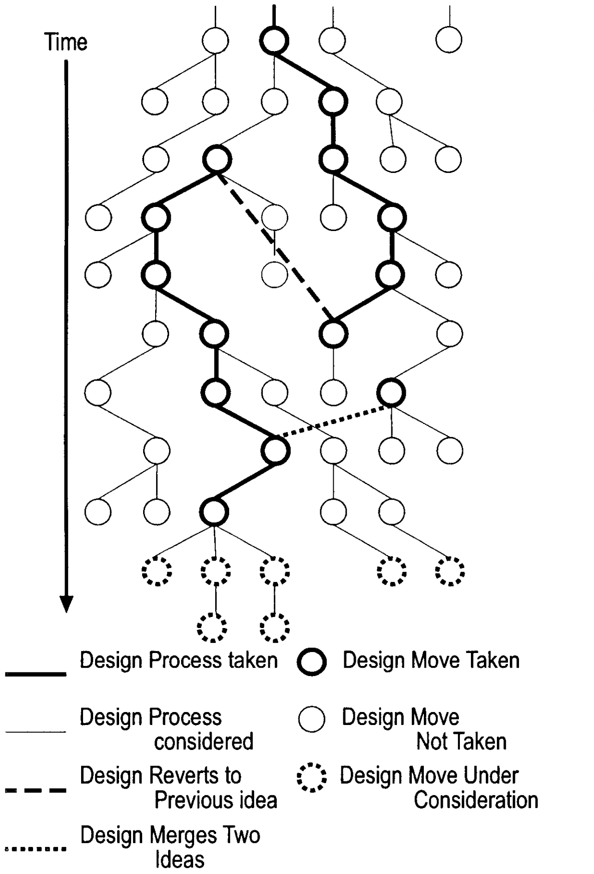

Time
Design Process taken
Design Move Taken
Design Process considered
Design Move Not Taken
Design Reverts to Previous idea
Design Move Under Consideration
Design Merges Two Ideas

Figure 4.4: A sample design process shows how a mediator might track design progress, consider multiple branches of designs, backtrack to a previous design possibility, or merge two possible solutions

4.1 Conclusion

Architectural design involves the integration of diverse, sometimes conflicting, concepts and requirements into a single composition. Multiple semantic readings are constructed from the current state of the design, and the design is altered to improve the design based on domain specific criteria. Coherence emerges when multiple points of view find satisfaction in the design artifact.

Collaborative design evolved because of the forces of specialization, delegation, cooperation, and distribution, described in Section 1 of this thesis. These forces projected the construction of coherence outward, from occurring within one designer, to occurring between many. The thesis proposes that, the process of generating coherence among multiple points of view within one designer is similar to generating coherence between multiple designers. However, there is one important difference. Within an individual designer, the pathways for sharing meaning are more evolved. Certain processes have learned to use the meaning structures of other processes. Meaning sharing among designers on the other hand is necessarily mediated by perception.

Filter Mediated Design proposes a framework of fundamental processes involved in design, and a representation of design data required for these processes. The work identifies perception as a powerful device in constructing meaning from ambiguity, and proposes a similar mechanism, called 'filters', for computational design systems. The thesis explains how current attempts to predetermine and explicitly represent semantics in the database are difficult and severely limiting, and how 'filters' overcome these limitations. Based on 'filters' representations, 'constructors' modify the design world, improving the design based on specific criteria. 'Mediators' manipulate the flow of information, employing different strategies for coercing the different perspectives to work cohesively. 'Filter mediated design' does not claim to replicate any particular strategy of design, but rather claims that these processes of evaluation, generation and mediation are fundamental across strategies.

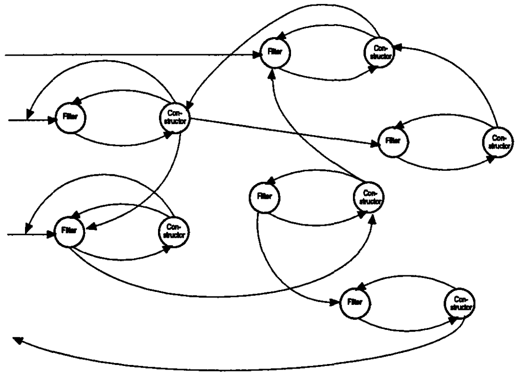


Figure 5: Schematically shows a network of possible relationhips multiple filters, constructors, in the environment. With Filter Mediated Design, these relationships can occur within, or between, individuals.

"behavior becomes intelligent as pathways between the subject and the environment cease to be simple, rigid and unidirectional, but become more reversible and mobile, as the scope of the interaction goes beyond immediate and momentary contacts to achieve far reaching and stable relations." [15]

Jean Piaget

The theoretical framework is implemented in a prototype that makes explicit the strategy for representing design data and the framework of processes for accessing and manipulating the data. 'Filters' analyzing 'enclosure', 'structure', 'space', 'light', and 'chair' access a geometrical and topological database to construct task specific representations. 'Constructors' use these representations to determine appropriate modifications to the world to address domain specific criteria. 'Mediators' moderate the actions of the 'filters' and 'constructors'. The prototype has illustrated that this mechanism of 'filters' is successful in dynamically producing multiple, task specific representations from an evolving, central, semantically sparse database. Generative and evaluative strategies are reconsidered in light of each other through the evolving design. Users interact with the system, able to assume the role of privileged 'constructors', or 'mediators'.

The work shows some promising opportunities towards understanding and manipulating the flow of information in the design process. The goal is to improve the 'collective intelligence' in design. By ambiguously representing the shared design data, multiple meaning structures can be constructed. These meaning structures, once constructed, can form the basis for design operations. The mediation of this act of perception provides an opportunity for meaning to be shared in ways not previously possible, sometimes avoiding the need for communication to pass through the ambiguous database.

The thesis makes two main contributions:

1. A strategy for representing and accessing design data.
2. A set of mechanisms which account for the fundamental processes of design.

Endowing computers with the power of perception promises to:

1. Allow multiple computational processes and humans with diverse criteria to access and manipulate a database.
2. Mediate the construction of meaning, allowing meaning to be shared across domains.
3. Provide the capability to computationally search for the emergence of coherence in an evolving database.

Modeling the processes of design provides the ability to:

1. Interact with the design system, allowing the computer to share some of the generative or evaluative load.
2. Specify strategies of design search by privileging certain filter's, constructor's, or mediators.
3. construct a tool to think with, to understand issues such as concept formation, system dynamics, and design inquiry.

References:

1 Simon, H. A. The Sciences of the Artificial, MIT Press, Cambridge, MA (1981)

2 Cross, N 'Natural intelligence in design' Design Studies, Vol 20 No 1 January 1999) pp 25-39

3 Eastman, C.A. and Nirva Fereshetian 'Information Models for Use in Product Design' Computer-Aided Design Vol. 26, No. 7 (July 1994) pp 551 – 572

4 Rosenman, M. A. and Gero, J. S. 'Modelling multiple views of design objects in a collaborative CAD environment', CAD, Special Issue on AI in Design 28(3) (1996) pp 207-21

5 MacKellar, B. & Peckam, J., 'Multiple Perspectives of Design Objects', Artificial Intelligence in Design '98, ed. John Gero and Fay Sudweeks, Klewer Academic Publishers (1998) pp. 87 – 106

6 Pahng F., Bae S., Wallace D., A Web-based Collaborative Design Modeling Environment, (1998) http://cadlab..mit.edu/publications/98-wetice-wallace

7 Eastman C., Jeng, T.S. Chowdbury, R 'Integration of design applications with building models' CAAD Futures'97 ed. R. Junge, Kluwer Academic Publishers(1997) pp 45 – 59

8 Campbell, M. Cagan, J., Kotovsky, K. 'A-Design: Theory and implementation of an adaptive agent-based method of conceptual design' Artificial Intelligence in Design '98, ed. John Gero and Fay Sudweeks, Klewer Academic Publishers (1998) pp. 579 – 598

9 Kolarevic, Schmidt, Hirschberg, Kurman, & Johnson 'An Experiment in Design Collaboration', Acadia '98, Association for Computer-Aided Design in Architecture (1998) pp 91 – 98

10 Lettvin, J. Y. 'What the Frogs Eye Tells The Frogs Brain.' The Mind: Biological Approaches to it's Functions, Ed. William C. Corning and Martin Balaban, Interscience Publishers, New York, (1968) pp 233-258

11 Porter, W. L. 'Notes on the inner logic of designing: Two thought experiments' Design Studies Vol 9 No 3 (July 1988) pp169

12 Stiny G, 'Introduction to shape and shape grammars", Environment and Planning B 7 (1980) pp 343-351

13 Minsky, M. Society of Mind Simon and Shuster, New York (1985) p. 72

14 Kahneman D. and Tversky A. 'Judgment Under Uncertainty: Heuristics and Biases' Science 185 (1974) pp 1124-1131

15 Ackermann, E. 'Circular reactions and sensori-motor intelligence: When Piaget's theory meets cognitive models' Archives de Psychologie No 38 pp. 65 – 78 (1990)

16 Mitsubishi Electric Research Laboratory, http://www.meitca.com/opencom/ (1999)

17 Persig, R. Zen and the Art of Motorcycle Maintenance

18 Christopher Alexander, Notes on the Synthesis of Form (Cambridge, Massachusetts: Harvard University Press, 1964).

APPENDIX A - BIBLIOGRAPHY

While many of the following readings are not directly referenced in the thesis, they were influential in developing the ideas contained.

READING LIST in ARTIFICIAL INTELLIGENCE

AUTHOR	TITLE
Herbert Simon	The Sciences of the Artificial
John H. Holland	Hidden Order
Brian Charles Williams	Invention From First Principles via Topologies of Interaction
Gerald Lafael Roylance	Causality, Constraint, & Mechanism in a Computer Program for Designing Circuits
H. Dreyfus	What Computers Still Can't Do
L. Franz Sinlang Ruecker	mSIBYL - A Design Documentation and management System
A. M. Turing	Computing Machinery and Intelligence
Shimon Ullman	Sequence Seeking and Counter Streams
Marvin Minsky	Steps toward Artificial Intelligence
David Marr	Artificial Intelligence—a personal view
Feldman and Ballard	Connectionist models and their properties
Marvin Minsky	K-lines: A Theory of Memory
Shimon Ullman	Visual Cognition and Visual Routines
Sajit Rao	Visual Routines and Attention
Laird,Newell,Rosenbloom	SOAR: An Architecture for General Intelligence
Guha and Lenat	Enabling Agents to Work Together
Yip and Sussman	A computational Model for the Acquisition and Use of Phonological Knowledge
Simon Kirby	Language Evolution without Natural Selection
Yuret	Discovering regularities in language
Atkeson	Learning motor routines by practice
Patrick Winston	Reasoning and learning by precedent
Letvin et al	What the frog's eye tells the frog's brain
Wilson and McNaughton	Reactivation of Hippocampal Ensemble Memories DuringSleep
Roe, Pallas, Kwon, and Sur	Visual Projections Routed to the Auditory Pathway
Spelke et al	The Development of Object Perception
Marvin Minsky	A Framework for Representing Knowledge
Borchart	Causal reconstruction
Chomsky	Language and Nature
Wilson and Tonegawa	Synaptic plasticity, place cells and spatial memory
Newell and Simon	The general problem solver, GPS
Marvin Minsky	The Society of Mind
Pearl	Bayes nets
Patrick Winston	Artificial Intelligence
K. Forbus and Gentner	Towards a computational Model of evaluating and using analogical references
K.Forbus and Gentner	A Model of Similarity Based retrieval
Dedre Gentner	Structure Mapping in Analogy and Similarity
Jintae Lee	A Decision Rationale Management System
Chaim D. Shen-or	Automated Design of Complex Gear trains
Karl Thatcher Ulrich	Computational and Pre-parametric Design

READING LIST in DESIGN COMPUTATION

AUTHOR	TITLE
Stiny, G	“New Ways to Look at Things” Environment and Planning B: Planning andDesign,
Stiny, G	“Commentary”,Massachusetts Institute ofTechnology, Cambridge
Holland, John	“Hidden Order”, Addison-Wesley
March, L.	“A boolean description of a class of built forms” in The Architecture of Form, pp.41-
Chomsky, N.	“Syntactic Structures”, Mouton, The Hague, Paris, pp.11-33, and pp. 49-60, and ..
Simon, H.	“The Sciences of the Artificial”
Stiny, George	“Design Machines”,Envirnoment and Planning B: Planning and Design, pp. 245-
Lewis Thomas	“The Lives of a Cell”
M Field and M Golubitsky	“Symetry in Chaos”
David Darling	“Equations of Eternity”
Richard Dawkins	“The Selfish Gene”
C. F. Earl	Rectangular Shapes
L. March, C. F. Earl	On continuing architectural plans
William J. Mitchell	Articulate Design of Free-Form Structure
Kristina Shea J. Cagan	Generating Structural Essays from Languages of DiscreteStructures
Matthew Campbell,J Cagan	A-Design: Theory and Implementation of AdaptiveAgent-Based Method of Conceptual Design
Karl Sims	Evolving 3D Morphology and Behaviour by Competition
Karl Sims	Evolving Virtual Creatures
John Frazer	An Evolutionary Architecture
John Frazer	The Co-operative Evolution of Buildings and Cities
Pegor Papazian	Incommensurability of Critiria and Focus in Design Generation
George Stiny	Shape
Lionel march	The Architecture of Form
March and Steadman	The geometry of environment
Jose Duarte	Using Grammars to customize Mass Housing
George Stiny	The Algebras of design
D’Arcy Thompson	On Growth and Form
Terry Knight	Designing a Shape Grammar: Problems of Predictability
Tim Smithers	Towards a Knowledge Level Theory of Design Process
MacKellar and Peckham	Multiple Perspectives of Design Objects
L. March, C Earl	On Counting Architectural Plans
C. Eastman, N. Fereshetian	Information Models for Use in Product Design: A Comparison
C. Eastman, A. Siabris	A Generic Building Product Model Incorporating Building Type Information
Eastman, Parker, Jeng	Managing the Integrity of Design Data Generated by Multiple Applications: Principle of Patching

READING LIST in PHILOSOPHICAL and MISCELLANEOUS ISSUES

Winograd & Flores	Understanding and Being - chpt. 3
Robert Persig	Zen and The Art of Motorcycle Maintenance
F. Nietzche	Beyond Good an Evil
J. Baidrillard	Simulcra and Simulation - excerpts
Robert grudin	The grace of great things - creativity and innovation
Neal Stephenson	Snow Crash
R.W. Emerson	The Over-Soul
Deleuze/Guitari	A Thousand Plateaus
James Bailey	After Thought
Deepak Chopra	The Higher Self
John Polkinghorne	Quarks, Chaos and Christianity
Bers and Bergman	A constructionist perspective on Values:
Booth, Colomb, Williams	The Craft of Research
Hermann Hesse	Siddhartha
Dan Milman	The Way of the Peaceful Warrior

READING LIST in DESIGN INQUIRY

AUTHOR	TITLE
William Porter	Notes on the inner logic of designing: Two thought-experiments
Anthony Vidler	The Idea of Type
Quatremere de Quincy	Type
Roy Ascott	The Architecture of Cyberception
Marcos Novak	transArchitecture
Christopher Alexander	Notes on the Synthesis of Form
Donald Norman	The Design of everyday Things
Christopher Alexander	The Timeless Way of Building 1979
Christopher Alexander	Pattern Language
Gaston Bachelard	The Poetics of Space
Janet Murray	The Future Narrative of Cyberspace
John Habraken	concept design Games
John Habraken	The Structure of the Ordinary
William Mitchell	City of Bits
Richard Boland	The Tyranny of space in Organizational Analysis
Karl Sims	The Art-Work: Interpretation Beyond decoding
Rodney Brooks	Planning is just a way of avoiding figuring out what to do next
Douglas Ingber	The Architecture of Life
Paul Starr	The seductions of Sim
Stan Allen	from Object to Field
Tony Brackenberg	Let the Problem of Your Mind Disolve in Your Mind
Walter Benjamin	The Work of Art in the Age of Mechanical Reproduction
Rafael Moneo	On Typology

READING LIST in DEVELOPMENTAL PSYCHOLOGY (and Collective Intelligence)

AUTHOR	TITLE
Michael J. Reddy	The Conduit Metaphor
Nelson Goodman	Ways of Worldmaking
Ludwig Wittgenstein	The Blue and Brown Books - excerpts
Nelson Goodman	Languages of Art
Donald Norman	Knowledge in the Head and in the World
Donald Norman	The Psychopathology of Everyday Things
Edith Ackermann	Tools for Constructive Learning: Rethinking Interactivity
Mitchell Resnick	Turtles, Termites, and Traffic Jams
Ginsburg and Opper	Piagets Theory of Intellectual Development
Edith Ackermann	Circular Reactions and Sensori-Motor Intelligence: When Piagets Theory meets. cognitive models
Edith Ackermann	Enactive Representations in Learning: Pretense, Models and Machines
Jean Mandler	How to Build a Baby: II Conceptual Primitives
Kenneth Gergen	The Saturated Self
David Kirsh	Complementary Strategies, Why We use our Hands when we Think
Kenneth Gergen	The Saturated Self
Strohecker and Friedlander	The Greek Chorus as a model for agents in interactive Stories
J. Wertsch	Voices of the Mind
Eugene Ferguson	Notes on Engineering and the Minds Eye
Daniel Dennett	Darwins Dangerous Idea
David Perkins	The Minds Best Work

Appendix A: The Code

The following java code was generated in support of this thesis, to construct the prototype described in Section 3. The appendix does not include the Open Community code, which is required to run the program. This code interfaces with the OpenCommunity C code, through a modified JNI layer.

// public class spBlock extends spThing

```
iimport java.util.*; import java.awt.*;
/**  This class contains info about Blocks. */ public class spBlock extends spPart {
   private spBlock referenced;//these are used to place tags in the filtered views
   private String label;
   public void setMyTransform(float [] t)
   { setX(t[spTransformSX]);
      setZ(t[spTransformSZ]);
      setY(t[spTransformSY]);
      //super.setTransform(t);
   }
   /** Called from C.*/
   public spBlock(int ptr) {
      super(ptr);
      spBlockInit() ; }
   /** Called from Javas newPart.*/
   public spBlock(String parameters, String idfPath) {
      super(0);
      tdx.spThingSetAppearance(this, idfPath, this);//HAYMAKER to try to get prop
dialog to work
      spBlockInit();
      label = null;
      setParent(tdAvatarPlatform.getAvatarsLocale());
      setInited(true);
   }
   //called by filters
   public spBlock(spBlock wBlock)
   { super(0);      //tdx.spThingSetAppearance(
      spBlockInit();
      //setParent(tdAvatarPlatform.getAvatarsLocale());  //COULD NEED THIS
      setTransform(wBlock.getTransform());
      setX(wBlock.getX());
      setZ(wBlock.getZ());
      setY(wBlock.getY());
      setMaterial(wBlock.getMaterial());
      setMass(wBlock.getMass());
      referenced = wBlock;
      setInited(true);
   }
   /** Hash table of all Blocks.*/
   transient static private Hashtable blocks = new Hashtable();
   /** Returns the hash table of all Blocks.*/
   public static Hashtable getHash() {return blocks;}
   /** Initialize the block.*/
   void spBlockInit() {
      blocks.put(new Integer(this.sp), this );
   }
   /** Return the Block by its cptr */
   public static spBlock getBlockBySp(int cptr){
      return (spBlock) blocks.get(new Integer(cptr));
   }
   /** Removes this spBlock. Should only called via JNI. */
   public void remove() {
      blocks.remove( new Integer(this.sp) );
      super.remove();
   }
   public void setReferenced(spBlock r)
   {  referenced = r;  }
   public spBlock getReferenced()
   {  return referenced;  }
   public void setLabel(String m)
   {  label = m;    }
   public String getLabel()
   {  return label; }
   public float  getMyWeight()
   {return (getZ() * getX() * getY() * getMass());}
   //get the bottom coordinates of the block
   public tdPoint getLocaleTransform()
   { return tdx.spThingGetLocaleTransform(this).copy();
   }
   public tdPoint getSWbot()
   {return tdx.spThingGetLocaleTransform(this).copy();
   }
   public tdPoint getSEbot()
   {  tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      temp.pt[spThing.spTransformX] = temp.pt[spThing.spTransformX] + getX();
      return temp;
   }
   public tdPoint getNWbot()
   { tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      temp.pt[spThing.spTransformZ] = temp.pt[spThing.spTransformZ] - getZ();
      return temp; }
   public tdPoint getNEbot()
   {  tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      temp.pt[spThing.spTransformX] = temp.pt[spThing.spTransformX] + getX();
      temp.pt[spThing.spTransformZ] = temp.pt[spThing.spTransformZ] - getZ();
      return temp; }
   //Get the top coordinates of the block
   public tdPoint getSWtop()
   { tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      temp.pt[spThing.spTransformY] = temp.pt[spThing.spTransformY] + getY();
      return temp;
   }
   public tdPoint getSEtop()
   { tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      temp.pt[spThing.spTransformX] = temp.pt[spThing.spTransformX] + getX();
      temp.pt[spThing.spTransformY] = temp.pt[spThing.spTransformY] + getY();
      return temp;
   }
   public tdPoint getNWtop()
   {tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      temp.pt[spThing.spTransformZ] = temp.pt[spThing.spTransformZ] - getZ();
      temp.pt[spThing.spTransformY] = temp.pt[spThing.spTransformY] + getY();
      return temp;
   }
   public tdPoint getNEtop()
   { tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      temp.pt[spThing.spTransformX] = temp.pt[spThing.spTransformX] + getX();
      temp.pt[spThing.spTransformZ] = temp.pt[spThing.spTransformZ] - getZ();
      temp.pt[spThing.spTransformY] = temp.pt[spThing.spTransformY] + getY();
      return temp;
   }
   public float getXpos()
   {   tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      return temp.pt[spThing.spTransformX];}
   public void setXpos(float x)
   {   tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      temp.pt[spThing.spTransformX] = x;
      this.setTransform(temp);}
   public float getYpos()
   {   tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      return temp.pt[spThing.spTransformY];}
   public void setYpos(float y)
   {   tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      temp.pt[spThing.spTransformY] = y;
      this.setTransform(temp);}
   public float getZpos()

   {   tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      return temp.pt[spThing.spTransformZ];}
   public void setZpos(float z)
   {   tdPoint temp = tdx.spThingGetLocaleTransform(this).copy();
      temp.pt[spThing.spTransformZ] = z;
      this.setTransform(temp);}
    /** Sets this blocks Width.*/
   public void   setMass(float f)        {tdWM.setBlockMass(this.sp,f);}
   /** Returns this blocks Width.*/
   public float  getMass()              {return tdWM.getBlockMass(this.sp);}
   /** Sets this blocks length, and changes transform*/
   public void   setX(float f)          { float[] t = this.getTransform();
                                         t[spTransformSX] = f;
                                         this.setTransform(t);
                                         tdWM.setBlockX(this.sp,f);}
   /** Returns this blocks length.*/
   public float  getX()                {return tdWM.getBlockX(this.sp);}
   /** Sets this blocks Width., and changes transform*/
   public void   setZ(float f)          {float[] t = this.getTransform();
                                         t[spTransformSZ] = f;
                                         this.setTransform(t);
                                         tdWM.setBlockZ(this.sp,f);}
   /** Returns this blocks Width.*/
   public float  getZ()                {return tdWM.getBlockZ(this.sp);}
   /** Sets this blocks Height.*/
   public void   setY(float f)          {float[] t = this.getTransform();
                                         t[spTransformSY] = f;
                                         this.setTransform(t);
                                         tdWM.setBlockY(this.sp,f);}
   /** Returns this blocks Height.*/
   public float  getY()                {return tdWM.getBlockY(this.sp);}
   /** Sets this blocks Type (0 = conc, 1 = glass.)*/
   public void   setMaterial(int t)        {
                                         if(t == 0)
                                         {filter.assignNormalConc(this);}
                                         if(t == 1)
                                         {filter.assignNormalGlas(this);}
                                         tdWM.setBlockMaterial(this.sp,t);}
   /** Returns this blocks Type (0 = conc, 1 = glass.).*/
   public int  getMaterial()               {return tdWM.getBlockMaterial(this.sp);}
}
```

// public class tdBlockEditor extends tdEditor

```
import java.awt.*; import java.util.*; import java.awt.event.*;
 import com.sun.java.swing.*; import com.sun.java.swing.JLabel;
import com.sun.java.swing.JTextField;
/** This class implements a Block editor. */
 public class tdBlockEditor extends tdEditor {
   static tdBlockEditor me = null; // Set once.
   static int thisEditorsIndex;
   static spMarker myMarker;
   tdVisualListener deleteLis, cloneLis, selectLis, dragLis, upLis, rightLis;
   public static tdBlockEditor get() {return me;}
   public spMarker getMarker() {return myMarker;}
   public int blockSelectMode = 0;
   public tdTransformDialog currentBlockDialog;
   public tdPropertiesDialog currentPropDialog;

   //Buttons in Schmoozer to make blocks
   private static JButton concButton = new JButton(“Concrete”);
   private static JButton glasButton = new JButton(“Glass”);
               private static JPanel makeBlockPanel;
               private static ButtonGroup makeBlockGroup = new ButtonGroup();

               private static JButton copyButton = new JButton(“Copy”);
   private static JButton cutButton = new JButton(“Cut”);
   private static JButton importButton = new JButton(“Import”);
               private static JPanel copyCutImportPanel;
               private static ButtonGroup copyCutImportGroup = new ButtonGroup();

               private static JLabel filterLabel = new JLabel();
   private static JButton spaceButton = new JButton(“Space”);
   private static JButton enclosureButton = new JButton(“Enclosure”);
   private static JButton lightButton = new JButton(“Light”);
   private static JButton structureButton = new JButton(“Structure”);
   private static JButton chairButton = new JButton(“Chair”);
   private static JButton worldButton = new JButton(“World”);

   private static JLabel constructorLabel = new JLabel();
   private static JButton CspaceButton = new JButton(“CSpace”);
   private static JButton CenclosureButton = new JButton(“CEnclosure”);
   private static JButton ClightButton = new JButton(“CLight”);
   private static JButton CstructureButton = new JButton(“CStructure”);
   private static JButton CchairButton = new JButton(“CChair”);
   private static JButton CworldButton = new JButton(“CWorld”);
   private static JPanel filterPanel;
               private static ButtonGroup filterGroup = new ButtonGroup();
               private static JPanel constructorPanel;
               private static ButtonGroup constructorGroup = new ButtonGroup();
               private static JPanel medInsertPanel;
               private static JPanel medLinkPanel;
               private static JPanel mediatorInPanel;
               private static JPanel mediatorPanel;
               private static JPanel fmdPanel;
               private static JLabel mediatorLabel = new JLabel();

               private static JCheckBox [] medConstructors = new JCheckBox[5];
               private static ButtonGroup medGroup = new ButtonGroup();
               private static JButton medStartButton = new JButton(“Start”);
   public static JButton medStopButton = new JButton(“Stop”);

   private static JTextField JTFnumCycles = new JTextField();
   private static JTextField JTFnumSpaces = new JTextField();
   private static JTextField JTFsizeSpaces = new JTextField();
   private static JTextField JTFnumChairs = new JTextField();

   private static JLabel numCyclesLabel = new JLabel();
   private static JLabel numSpacesLabel = new JLabel();
   private static JLabel sizeSpacesLabel = new JLabel();
   private static JLabel numChairsLabel = new JLabel();

               private SymAction lSymAction = new SymAction();

   /** Snap distance.*/
   public static final float HOOK_SNAP_DISTANCE = 16;
   /** The editors constructor. Called once. */
   public tdBlockEditor() {
     super();
     if (me == null) blockEditorInit();
                                        //{{INIT_CONTROLS
                       //}}
}
               /** Initalize the editor. */
   public void blockEditorInit() {
     if (me != null) return;
```

```
   me = this;
   myMarker = (spMarker) new spMarker(“spBlock”,schmoozer.getRootURL()
                         +”Editors/Pointer_Dog.idf”,2000);

   // mouse listners
   selectLis = new tdVisualListener(lLisAction,tdVisualListener
                         .EVENT_MOUSE_DOWN,false);      //onBlockAnyPick
   upLiis= new tdVisualListener(lLisAction,tdVisualListener
                         .EVENT_MOUSE_UP,false);          //onUp
   deleteLis= new tdVisualListener(lLisAction,tdVisualListener
                         .EVENT_MOUSE_RIGHT_DOUBLE,false); //onBlockDel
   dragLis  = new tdVisualListener(lLisAction,tdVisualListene
                         r.EVENT_MOUSE_DRAG,false); // onBlockDrag
   rightLis = new tdVisualListener(lLisAction,tdVisualListener
                         .EVENT_MOUSE_RIGHT,false);
   // ADD TO EDIT MENU
           com.sun.java.swing.KeyStroke jkey = com.sun.java.swing.KeyStroke
.getKeyStroke(java.awt.event.KeyEvent.VK_R,java.awt.Event.CTRL_MASK,false);
    thisEditorsIndex = schmoozer.registerEditor(me,”Edit Block”,'b'
           ,tdVisualListener.KEY_CTRL_R,jkey,”images/edBlock.gif”,null);
   //HAYMAKER add Make Buttons to schmoozer
   makeBlockPanel = new JPanel();
                         makeBlockPanel.setLocation(5,5);
                         makeBlockPanel.setSize(100, 100);
                         makeBlockPanel.setVisible(true);
                         makeBlockPanel.setBackground(Color.black);
                         makeBlockPanel.setLayout(null);
                         makeBlockPanel.show();
          schmoozer.getSchmoozer().getContentPane().add(makeBlockPanel);

                         copyCutImportPanel = new JPanel();
                         copyCutImportPanel.setLocation(5,110);
                         copyCutImportPanel.setSize(205, 40);
                         copyCutImportPanel.setVisible(true);
                         copyCutImportPanel.setBackground(Color.black);
                         copyCutImportPanel.setLayout(null);
                         copyCutImportPanel.show();
      schmoozer.getSchmoozer().getContentPane().add(copyCutImportPanel);
                         copyCutImportPanel.add(copyButton);
                         copyCutImportPanel.add(cutButton);
                         copyCutImportPanel.add(importButton);
                         copyCutImportGroup.add(copyButton);
                         copyCutImportGroup.add(cutButton);
                         copyCutImportGroup.add(importButton);

                         copyButton.setLocation(5,5);
                         copyButton.setSize(65,30);
                         cutButton.setLocation(70,5);
                         cutButton.setSize(65,30);
                         importButton.setLocation(135,5);
                         importButton.setSize(65,30);
                         copyButton.addActionListener(lSymAction);
                         cutButton.addActionListener(lSymAction);
                         importButton.addActionListener(lSymAction);

                         concButton.setLocation(5,5);
                         concButton.setSize(90,45);

                         glasButton.setLocation(5,50);
                         glasButton.setSize(90,45);

                         makeBlockPanel.add(concButton);
                         makeBlockPanel.add(glasButton);
                         makeBlockGroup.add(concButton);
                         makeBlockGroup.add(glasButton);

                         concButton.addActionListener(lSymAction);
                         glasButton.addActionListener(lSymAction);



                         filterPanel = new JPanel();
                         filterPanel.setLocation(105,0);
                         filterPanel.setSize(100, 220);
                         filterPanel.setVisible(true);
                         filterPanel.setBackground(Color.black);
                         filterPanel.setLayout(null);
                         filterPanel.show();
   filterPanel.add(filterLabel);
                         filterPanel.add(spaceButton);
                         filterPanel.add(enclosureButton);
                         filterPanel.add(lightButton);
                         filterPanel.add(structureButton);
                         filterPanel.add(chairButton);
                         filterPanel.add(worldButton);
```

```
            filterGroup.add(spaceButton);
            filterGroup.add(enclosureButton);
            filterGroup.add(lightButton);
            filterGroup.add(structureButton);
            filterGroup.add(chairButton);
            filterGroup.add(worldButton);

            filterLabel.setBackground(Color.black);
            filterLabel.setForeground(Color.lightGray);
            filterLabel.setLocation(5,5);
            filterLabel.setSize(90,30);
            filterLabel.setText("FILTER");
filterLabel.setHorizontalAlignment(SwingConstants.CENTER);
            filterLabel.setOpaque(true);
            spaceButton.setLocation(5,35);
            spaceButton.setSize(90,30);
            enclosureButton.setLocation(5,65);
            enclosureButton.setSize(90,30);
            structureButton.setLocation(5,95);
            structureButton.setSize(90,30);
            chairButton.setLocation(5,125);
            chairButton.setSize(90,30);
            lightButton.setLocation(5,155);
            lightButton.setSize(90,30);
            worldButton.setLocation(5,185);
            worldButton.setSize(90,30);

            spaceButton.addActionListener(lSymAction);
            enclosureButton.addActionListener(lSymAction);
            structureButton.addActionListener(lSymAction);
            chairButton.addActionListener(lSymAction);
            lightButton.addActionListener(lSymAction);
            worldButton.addActionListener(lSymAction);

            ///constructor group and panel/////
            constructorPanel = new JPanel();
            constructorPanel.setLocation(0,0);
            constructorPanel.setSize(100, 220);
            constructorPanel.setVisible(true);
            constructorPanel.setBackground(Color.black);
            constructorPanel.setLayout(null);
            constructorPanel.show();
            constructorPanel.add(CspaceButton);
            constructorPanel.add(CenclosureButton);
            constructorPanel.add(ClightButton);
            constructorPanel.add(CstructureButton);
            constructorPanel.add(CchairButton);
            constructorPanel.add(CworldButton);
            constructorPanel.add(constructorLabel);

            constructorGroup.add(CspaceButton);
            constructorGroup.add(CenclosureButton);
            constructorGroup.add(ClightButton);
            constructorGroup.add(CstructureButton);
            constructorGroup.add(CchairButton);
            constructorGroup.add(CworldButton);


            constructorLabel.setBackground(Color.black);
            constructorLabel.setForeground(Color.lightGray);
            constructorLabel.setLocation(3,5);
            constructorLabel.setSize(95,30);
            constructorLabel.setText("CONSTRUCTOR");
constructorLabel.setHorizontalAlignment(SwingConstants.CENTER);
            constructorLabel.setOpaque(true);
            CspaceButton.setLocation(5,35);
            CspaceButton.setSize(90,30);
            CenclosureButton.setLocation(5,65);
            CenclosureButton.setSize(90,30);
            CstructureButton.setLocation(5,95);
            CstructureButton.setSize(90,30);
            CchairButton.setLocation(5,125);
            CchairButton.setSize(90,30);
            ClightButton.setLocation(5,155);
            ClightButton.setSize(90,30);
            CworldButton.setLocation(5,185);
            CworldButton.setSize(90,30);

            CspaceButton.addActionListener(lSymAction);
            CenclosureButton.addActionListener(lSymAction);
            CstructureButton.addActionListener(lSymAction);
            CchairButton.addActionListener(lSymAction);
            ClightButton.addActionListener(lSymAction);
            CworldButton.addActionListener(lSymAction);
```

```
                /////
                medLinkPanel = new JPanel();
                medLinkPanel.setLocation(90,35);
                medLinkPanel.setSize(25, 180);
                medLinkPanel.setVisible(true);
                medLinkPanel.setBackground(Color.black);
                medLinkPanel.setLayout(null);
                medLinkPanel.show();

                medInsertPanel = new JPanel();
                medInsertPanel.setLocation(5,5);
                medInsertPanel.setSize(15, 180);
                medInsertPanel.setVisible(true);
                medInsertPanel.setBackground(Color.lightGray);
                medInsertPanel.setLayout(null);
                medInsertPanel.show();

                medLinkPanel.add(medInsertPanel);

                mediatorPanel = new JPanel();
                mediatorPanel.setLocation(15,215);
                mediatorPanel.setSize(175, 150);
                mediatorPanel.setVisible(true);
                mediatorPanel.setBackground(Color.black);
                mediatorPanel.setLayout(null);
                mediatorPanel.show();
                mediatorLabel.setBackground(Color.black);
                mediatorLabel.setForeground(Color.lightGray);
                mediatorLabel.setLocation(40,0);
                mediatorLabel.setSize(90,30);
                mediatorLabel.setText("MEDIATOR");
        mediatorLabel.setHorizontalAlignment(SwingConstants.CENTER);

                medStartButton.setLocation(100,30);
                medStartButton.setSize(65,30);
                medStartButton.addActionListener(lSymAction);
                medStopButton.setLocation(100,60);
                medStopButton.setSize(65,30);
    medStopButton.addActionListener(lSymAction);
    mediatorPanel.add(medStartButton);
    mediatorPanel.add(medStopButton);
                int yPos = 5;
                for(int i = 0; i < 5; i++)
                {
                  medConstructors[i] = new JCheckBox();
                  medInsertPanel.add(medConstructors[i]);
                  medConstructors[i].addActionListener(lSymAction);
                  medConstructors[i].setBackground(Color.black);
                  medConstructors[i].setForeground(Color.green);
                  medConstructors[i].setLocation(1, yPos);
                  medConstructors[i].show();
                  medConstructors[i].setVisible(true);
                  medConstructors[i].setSize(15, 15);
                  medConstructors[i].isOpaque();
                  medConstructors[i].enable();

                  yPos = yPos + 30;
                }

                mediatorPanel.add(mediatorLabel);

                fmdPanel = new JPanel();
//JLayeredPane fmdLPane = fmdPanel.getRootPane().getLayeredPane();
                fmdPanel.setLocation(5,155);
                fmdPanel.setBackground(Color.gray);
                fmdPanel.setLayout(null);
                fmdPanel.show();
          fmdPanel.add(mediatorPanel);
          fmdPanel.add(medLinkPanel);
                fmdPanel.add(filterPanel);
                fmdPanel.add(constructorPanel);

                //fmdLPane.add(mediatorPanel, new Integer(3));
                //fmdLPane.add(filterPanel, new Integer(2));
                //fmdLPane.add(constructorPanel, new Integer(1));
                fmdPanel.setSize(210, 320);
                fmdPanel.setVisible(true);

                JTFnumCycles.setToolTipText("How Many Iterateions ?");
                JTFnumCycles.setEnabled(true);
                mediatorPanel.add(JTFnumCycles);
                JTFnumCycles.setBackground(java.awt.Color.lightGray);
        JTFnumCycles.setFont(new Font("SansSerif", Font.PLAIN, 10));
                JTFnumCycles.setBounds(5,30,30,15);
```

```
            numCyclesLabel.setBackground(Color.black);
            numCyclesLabel.setForeground(Color.lightGray);
            numCyclesLabel.setLocation(45,30);
            numCyclesLabel.setSize(50,15);
            numCyclesLabel.setText(“# Cycles”);
            numCyclesLabel.setFont(new Font(“SansSerif”, Font.PLAIN, 10));
            numCycesLabel.setHorizontalAlignment(SwingConstants.LEFT);
            mediatorPanel.add(numCyclesLabel);

            JTFnumSpaces.setToolTipText(“How Many Spaces ?”);
            JTFnumSpaces.setEnabled(true);
            mediatorPanel.add(JTFnumSpaces);
            JTFnumSpaces.setBackground(java.awt.Color.lightGray);
            JTFnumpaces.setFont(new Font(“SansSerif”, Font.PLAIN, 10));
            JTFnumSpaces.setBounds(5,45,30,15);

            numSpacesLabel.setBackground(Color.black);
            numSpacesLabel.setForeground(Color.lightGray);
            numSpacesLabel.setLocation(45,45);
            numSpacesLabel.setSize(50,15);
            numSpacesLabel.setText(“# Spaces”);
            numSpacesLabel.setFont(new Font(“SansSerif”, Font.PLAIN, 10));
            numSpacesLabel.setHorizontalAlignment(SwingConstants.LEFT);
            mediatorPanel.add(numSpacesLabel);

            JTFsizeSpaces.setToolTipText(“How big should the spaces be?”);
            JTFsizeSpaces.setEnabled(true);
            mediatorPanel.add(JTFsizeSpaces);
            JTFsizeSpaces.setBackground(java.awt.Color.lightGray);
            JTFsizeSpaces.setFont(new Font(“SansSerif”, Font.PLAIN, 10));
            JTFsizeSpaces.setBounds(5,60,30,15);

            sizeSpacesLabel.setBackground(Color.black);
            sizeSpacesLabel.setForeground(Color.lightGray);
            sizeSpacesLabel.setLocation(45,60);
            sizeSpacesLabel.setSize(50,15);
            sizeSpacesLabel.setText(“size \””);
            sizeSpacesLabel.setFont(new Font(“SansSerif”, Font.PLAIN, 10));
            sizeSpacesLabel.setHorizontalAlignment(SwingConstants.LEFT);
            mediatorPanel.add(sizeSpacesLabel);

            JTFnumChairs.setToolTipText(“How Many Chairs?”);
            JTFnumChairs.setEnabled(true);
            mediatorPanel.add(JTFnumChairs);
            JTFnumChairs.setBackground(java.awt.Color.lightGray);
            JTFnumChairs.setFont(new Font(“SansSerif”, Font.PLAIN, 10));
            JTFnumChairs.setBounds(5,75,30,15);

            numChairsLabel.setBackground(Color.black);
            numChairsLabel.setForeground(Color.lightGray);
            numChairsLabel.setLocation(45,75);
            numChairsLabel.setSize(50,15);
            numChairsLabel.setText(“# Chairs”);
            numChairsLabel.setFont(new Font(“SansSerif”, Font.PLAIN, 10));
            numChairsLabel.setHorizontalAlignment(SwingConstants.LEFT);
            mediatorPanel.add(numChairsLabel);


            schmoozer.getSchmoozer().getContentPane().add(fmdPanel);

  }
public LisAction lLisAction = new LisAction();;
class LisAction implements tdVisualListener.visualListener

{
            public void eventAction(tdVisualEvent event){
    zprint(3,”tdBlockEditor eventAction\n”);
                        tdVisualListener lis = event.getSource();
                        //HAYMAKER Changed onBlockAnyPick to MakeBlock
                        //added the blockdragmode code
    if (lis == selectLis)      {
                  if(event.getThing() instanceof spBlock)
                  {
                    localeParticulars(event);
                    spBlock beam = (spBlock) event.getThing();

                    if (event.getThing().getLocallyOwned())
                      {
                System.out.println(“HAYMAKER item picked is locally owned”);
                      }
     else System.out.println(“HAYMAKER item picked NOT locally owned”);

                      if (blockSelectMode == 0)
                       {
                         onBlockAnyPick(event);
                        if (currentBlockDialog == null)
                        {
                        currentBlockDialog = tdKeyboardEditor.me.Transform();
                        currentBlockDialog.setLocation(710,500);
                        }
                        else
                        {
                 currentBlockDialog.target = event.getThing();
                currentBlockDialog.initFields(currentBlockDialog.target);
                 currentBlockDialog.displayTransFields(currentBlockDialog.trans);
                                   currentBlockDialog.displayTypeFields();
                                            currentBlockDialog.zeroSlider();
                        }
                        currentBlockDialog.show();
                       }
                  }
                  else
                { System.out.println(“Event.getThing() NOT instanceof spBlock “);
                    schmoozer.activateEditor(0);
                  }
         }
         else if (lis == deleteLis)  {onBlockDelete(event);}
         else if (lis == dragLis)    {
                 if(event.getThing() instanceof spBlock)
                 {        if (blockSelectMode == 0)
                            {onBlockDrag(event);}
                          if (blockSelectMode == 1)
                            {onBlockScaleXZ(event);}
                          if (blockSelectMode == 2)
                            {onBlockScaleY(event);}
                 }
                          }
         else if (lis == upLis)      {onBlockUp(event);}
         else if (lis == rightLis)   {
                          tdPoint tempPoint;
                          spBlock temp = ((spBlock)event.getThing());
                          tempPoint = temp.getSWbot();
       System.out.println(“getSWbot X,Z,Y = “  + tempPoint.pt[spTransformX]+ “ , “
                                                + tempPoint.pt[spTransformZ]+ “ ,
“

                                                + tempPoint.pt[spTransformY] );
          tempPoint = temp.getNEtop();
      System.out.println(“getNEtop X,Z,Y = “  + tempPoint.pt[spTransformX]+ “ , “
                                                + tempPoint.pt[spTransformZ]+ “ ,
“

                                                + tempPoint.pt[spTransformY] );



                          }

                          }
            }

/* MOUSE ————————————————————————————— */
   /** Continues or starts a drag. */
   public void onBlockDrag(tdVisualEvent event) {
      if (!inDrag && dragTarget != null) {
 // Start Note, cliking on blocks may make them drag, (unhook)
         spOMI.makeStaticCall(“onUnhookEvent”, “tdBlockEditor”, dragTarget);

      }
      onDrag(event);
   }
    /** Continues or starts a scale in the XZ. */
    public void onBlockScaleXZ(tdVisualEvent event){

      if (!inDrag && dragTarget != null) { // Start Note, cliking on blocks
         spOMI.makeStaticCall(“onUnhookEvent”, “tdBlockEditor”, dragTarget);

      }
      onScaleXZ(event);
   }

   public void onBlockScaleY(tdVisualEvent event){

      if (!inDrag && dragTarget != null) { // Start Note, cliking on
         spOMI.makeStaticCall(“onUnhookEvent”, “tdBlockEditor”, dragTarget);

      }
      onScaleY(event);
   }

   /** Ends a drag. */
   public void onBlockUp(tdVisualEvent event) {
      zprint(3,”onBlockMouseUp\n”);
      if (inDrag && dragTarget != null) {
         spOMI.makeStaticCall(“onRehookEvent”, “tdBlockEditor”, dragTarget);

      }
   }

   /** Mouse function to pick a Block and move the marker. */
   public void onBlockAnyPick(tdVisualEvent event) {
      spBlock p;
      boolean inLocale = event.getThing() != null &&
  tdWM.getLocale( event.getThing().sp ) ==
tdAvatarPlatform.getAvatarsLocale().sp;
      if (!inLocale) {zprint(1,”Pick outside locale “+spLab(event.getThing())+” \n”);
                   p = null;}
      else if (event.getThing() instanceof spBlock) p = (spBlock) event.getThing();
      else if (event.getThing() == myMarker.getVisualThing())
 p = myMarker.getTargetedBlock(); // drag marker, dont change target
      else p = null;
      dragPick(p);
      moveMyMarker(p);


   }



   public void moveMyMarker(spBlock p) {
    if ((p != null) && (p instanceof spBlock) && (p != myMarker.getTargetedBlock()) )
      {
         spBlock lastBlock = myMarker.getTargetedBlock();
         if (lastBlock != null)//could be first block
         {   int lastBlockType = lastBlock.getMaterial();
            if (lastBlockType == 0)//if the previous block selected is concrete
            {   if (lastBlock.getLocallyOwned())//if it’s locally owned set it to
               lastBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale()
                                          .getSchmoozerLocaleVDefNorConc());
               else          //else ask it’s owner to set it to unselected concrete
            spOMI.makeStaticCall(“setVDtoNormalConc”, “tdBlockEditor”, lastBlock);
            }
     else if (lastBlockType == 1)//if previous block was glass
            {   if (lastBlock.getLocallyOwned())//if it’s locally owned set it to
               lastBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale()
                                          .getSchmoozerLocaleVDefNorGlas());
               else                           //else ask it’s owner to set it to unselected glass
             spOMI.makeStaticCall(“setVDtoNormalGlas”, “tdBlockEditor”, lastBlock);
            }
         }
         myMarker.setTarget(p,true);//move the marker on to the selectecd block
         if (p.getMaterial() == 0)//if the newly selected block is concrete set it t
            {
               if (p.getLocallyOwned())//if it’s locally owned
               p.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale()
                                          .getSchmoozerLocaleVDefSelConc());
               else                           //else ask it’s owner
               spOMI.makeStaticCall(“setVDtoSelectedConc”, “tdBlockEditor”, p);
            }
          else if (p.getMaterial() == 1)//if the block is glas set it to selected
            {
               if (p.getLocallyOwned())//if it’s locally owned
               p.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale()
                                          .getSchmoozerLocaleVDefSelGlas());
               else                   //else ask it’s owner
               spOMI.makeStaticCall(“setVDtoSelectedGlas”, “tdBlockEditor”, p);
            }
       }
   }

   /** Mouse function to delete a Block by calling “onDeleteTree”(spOMI). */
   public void onBlockDelete(tdVisualEvent event) {
      zprint(3,”onBlockDelete\n”);
      spBlock c = (event.getThing() instanceof spBlock)
                                          ? (spBlock)event.getThing() : null;
      if (c != null) {
         moveMarkerOff(c);
         myMarker.setTarget(c,false);
         spOMI.makeStaticCall(“onDeleteTree”, “tdEditor”, c);
      } else {
         zerror(“No marked block to delete!”);
      }
   }


   public void onBlockStretchY(tdVisualEvent event)
         {}
   /** Enable/disable this editor.
     * @param show  True to enable.
     */
```

```
            public void enable(boolean show){
               if (on != show) {
               on = show;

zprint(2,”enable spBlockKit  “+on+”\n”);
         if (myMarker.putIntoLocale()) moveMarkerOff(null);
         myMarker.setVisable(on);
         selectLis.enable(on);
         deleteLis.enable(on);
         dragLis.enable(on);
         upLis.enable(on);
         rightLis.enable(on);
      }
            }
            public spBlock makeNewBlock()
            {
   spBlock concBlock = (spBlock) (tryToMakeBlock(“spBlock”,””, “file:
                        ///s:/demo/trains/definitions/blocks/concBlock.idf”, 0));
   if (concBlock.getLocallyOwned())
               {
               concBlock.setX(1);
               concBlock.setZ(1);
               concBlock.setY(1);
               concBlock.setMass(250);
               concBlock.setMaterial(0);
               }
               else
               {
               spOMI.makeStaticCall(“initConcBlock”, “tdBlockEditor”, concBlock);
               }
               return concBlock;
            }
            //HAYMAKER added  NOTE: these are hacks
            public void makeConcBlock()
            {
              if (!on) schmoozer.activateEditor(thisEditorsIndex);
               spBlock concBlock = (spBlock) (tryToMakeBlock(“spBlock”,””, “file:
                        ///s:/demo/trains/definitions/blocks/concBlock.idf”, 0));
               if (concBlock.getLocallyOwned())
               {
               concBlock.setX(1);
               concBlock.setZ(1);
               concBlock.setY(1);
               concBlock.setMass(250);
               concBlock.setMaterial(0);
               }
               else
               {
               spOMI.makeStaticCall(“initConcBlock”, “tdBlockEditor”, concBlock);
               }

               moveMyMarker(concBlock);
            }
 public void makeGlasBlock()
            {
               if (!on) schmoozer.activateEditor(thisEditorsIndex);
               spBlock glasBlock = (spBlock) (tryToMakeBlock(“spBlock”,””, “file:
                        ///s:/demo/trains/definitions/blocks/glasBlock.idf”, 0));
              if (glasBlock.getLocallyOwned())
               {
               glasBlock.setX(1);
               glasBlock.setZ(1);
               glasBlock.setY(1);
               glasBlock.setMass(100);
               glasBlock.setMaterial(1);
               }
               else
               {
               spOMI.makeStaticCall(“initGlasBlock”, “tdBlockEditor”, glasBlock);
               }

               moveMyMarker(glasBlock);
            }
/** Places a spBlock in the locale then calls “onPlaceBlock” or “onLSta
   * @param p spBlock to be moved.
   */
 public spBlock setBlock(spBlock p) {
   if (p == null) return null;
   tdPoint next,oset;
   float offset = 30;
   if (!on) schmoozer.activateEditor(thisEditorsIndex);
   spBlock t = myMarker.getTargetedBlock(); // Makes sure its in WM.
   next = (tdx.spThingGetLocalePosition(tdAvatarPlatform.getCamera())).copy();
   oset = new tdPoint(0,0,-offset); // -offset for locotive mover’s backward
   next.multiply(oset);
```

```
      //Making sure rotation is 0
      //next.pt[spTransformRZ] = 0;
      //next.pt[spTransformRX] = 0;
      next.pt[spTransformRA] = 0;
      p.setTransform(next);
      //moveMyMarker(p);//maybe htis needs to go here, not in make block
      tdx.spDonateToLocale(p); // owner must change before start attempt.
      schmoozer.WMupdate(); // Make sure ownership change gets out.
      //spOMI.makeStaticCall(“onPlaceBlock”, “tdBlockEditor”, p);

      return p;
   }


/* ————————————————————————— OMI ————————— */
   /** spOMI function that animates teh placing of a block.*/
   public static void onPlaceBlock(spOMI omi){
      sp target = omi.getParent();
      zprint(1,”tdBlock onPlaceBlock\n”);
      //if ( target instanceof spBlock )
         //tdxs.animateGrowWiggleAndStop((spBlock)target,null,.002f,3000);
         // ShrinkDown wiggle
   }

   /** spOMI function that starts the engine.*/
   public static void onLStartEvent(spOMI omi){
      zprint(1,”tdBlock onLStartEvent\n”);
      sp parent = omi.getParent();

   }


   /** spOMI function that un hooks the parent block.*/
   public static void onUnhookEvent(spOMI omi){
      sp target = omi.getParent();
      zprint(1,”tdBlock onUnhookEvent\n”);
      if ( target instanceof spBlock ){
         spBlock c = (spBlock) target;

      } else
zprint(1,”tdBlock onUnhookEvent had no Block! “+spLab(target)+”\n”);
   }
   /** spOMI fuction to rehook blocks check the parent block.*/
   public static void onRehookEvent(spOMI omi){
      sp target = omi.getParent();
      zprint(1,”tdBlock onCheckHookUps\n”);
      boolean stoplooking = false; // THIS IS A TEMP HACK
      if ( target instanceof spBlock) {
         spBlock c1 = (spBlock) target;
zprint(1,”tdBlock onCheckHookUps “+spLab(c1)+”\n”);

      } else
zprint(1,”tdBlock onCheckHookUps had no Block! “+spLab(target)+”\n”);
   }

    public static void setVDtoSelectedConc(spOMI omi){
      spBlock target = (spBlock)omi.getParent();
      spSchmoozerLocale myLocale = (spSchmoozerLocale)target.getLocale();
      target.setVisualDefinition(myLocale.getSchmoozerLocaleVDefSelConc());
   }
    public static void setVDtoNormalConc(spOMI omi){
      spBlock target = (spBlock)omi.getParent();
      spSchmoozerLocale myLocale = (spSchmoozerLocale)target.getLocale();
      target.setVisualDefinition(myLocale.getSchmoozerLocaleVDefNorConc());
   }

    public static void setVDtoSelectedGlas(spOMI omi){
      spBlock target = (spBlock)omi.getParent();
      spSchmoozerLocale myLocale = (spSchmoozerLocale)target.getLocale();
      target.setVisualDefinition(myLocale.getSchmoozerLocaleVDefSelGlas());
   }
    public static void setVDtoNormalGlas(spOMI omi){
      spBlock target = (spBlock)omi.getParent();
      spSchmoozerLocale myLocale = (spSchmoozerLocale)target.getLocale();
      target.setVisualDefinition(myLocale.getSchmoozerLocaleVDefNorGlas());
   }

   public static void setVDtogreen50(spOMI omi){
      spBlock target = (spBlock)omi.getParent();
      spSchmoozerLocale myLocale = (spSchmoozerLocale)target.getLocale();
      target.setVisualDefinition(myLocale.getSchmoozerLocaleVDefgreen50());
   }
   public static void setVDtogreen100(spOMI omi){
      spBlock target = (spBlock)omi.getParent();
      spSchmoozerLocale myLocale = (spSchmoozerLocale)target.getLocale();
```

```
      tdPoint tran = new tdPoint(cop.getTransform());
      tran.pt[spTransformX] = (tran.pt[spTransformX] + 2);
      tran.pt[spTransformZ] = (tran.pt[spTransformZ] - 2);
      cop.setTransform(tran);
      if (cop.getMaterial() == 0)
        {cop.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale()
                               .getSchmoozerLocaleVDefNorConc());
        }
        else if (cop.getMaterial() == 1)
        {cop.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale()
                               .getSchmoozerLocaleVDefNorGlas());
        }
        else {cop.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale()
                               .getSchmoozerLocaleVDefred50());}
      tdx.spDonateToLocale(cop);
    }
  }
  else if (object == cutButton)
  {
    spBlock c = myMarker.getTargetedBlock();
    if (c != null)
    {
     moveMarkerOff(c);
    myMarker.setTarget(c,false);
    spOMI.makeStaticCall(“onDeleteTree”, “tdEditor”, c);
    }
    else System.out.println(“No Block Selected to Delete”);
  }

  else if (object == spaceButton)
  {
   filter.perceive(“spaceFilter”);
  }
  else if (object == enclosureButton)
  {
   filter.perceive(“enclosureFilter”);
  }
  else if (object == structureButton)
  {
    filter.perceive(“structureFilter”);
  }
  else if (object == chairButton)
  {
    filter.perceive(“chairFilter”);
  }
  else if (object == lightButton)
  {
    filter.perceive(“lightFilter”);
  }
  else if (object == worldButton)
  {
    if((tdAvatarPlatform.getAvatarsLocale().getTitle()).
                               compareTo(“worldLocale”) != 0)
    {
      myMarker.setTarget(null,false);
      filter.goToFilter(“worldLocale”);
    }
  }
  else if (object == importButton)
  {
                          boolean replace = schmoozer.getSchmoozer(
                       ).replaceScene_Action(); //find out from user
    filter.importToWorld(replace);
    System.out.println(“Import done”);
  }
  else if (object == CspaceButton)
  { constructor.construct(“space”); }
  else if (object == CenclosureButton)
  { constructor.construct(“enclosure”);}
  else if (object == CstructureButton)
  { constructor.construct(“structure”);}
  else if (object == CchairButton)
  { constructor.construct(“chair”);}
  else if (object == ClightButton)
  { constructor.construct(“light”);}
  else if (object == CworldButton)
  {}
  else if (object == medStartButton)
  {
    int numCycles = tdxs.stringToInt(JTFnumCycles.getText());
    int numSpaces = tdxs.stringToInt(JTFnumSpaces.getText());
    int sizeSpaces = tdxs.stringToInt(JTFsizeSpaces.getText());
    int numChairs = tdxs.stringToInt(JTFnumChairs.getText());
spMediator.start(medConstructors, numCycles, numSpaces,
                               sizeSpaces, numChairs);
```

```
            medLinkPanel.repaint();
          }
        else if (object == medStopButton)
        {}
      }
  }
  public void localeParticulars(tdVisualEvent event)
  {
if((tdAvatarPlatform.getAvatarsLocale().getTitle()).compareTo("chairFilter") == 0)
    {
      Enumeration c = filter.chairObjs.elements();
      while(c.hasMoreElements())
      {
        groupObject chair = (groupObject)c.nextElement();
        if(chair.getKey().equals(event.getThing()))
        {
          spBlock seat = chair.getKey();
          tdPoint seatPoint = new tdPoint(seat.getTransform());
          tdPoint seatTemp = seatPoint.copy();
          Enumeration l = chair.getConnected().elements();
          while(l.hasMoreElements())
          {
            tdPoint seatINV = seatTemp.copy();
            seatINV.inverse();
            spBlock leg = (spBlock)l.nextElement();
            if(leg.getParent() != seat)
            {
              tdPoint legPoint = new tdPoint(leg.getTransform());
              seatINV.multiply(legPoint);
              leg.setParent(seat);
              leg.setTransform(seatINV);
            }
            else{}
          }
        }
      }
    }
if((tdAvatarPlatform.getAvatarsLocale().getTitle()).compareTo("spaceFilter") == 0)
    {
      Enumeration s = filter.spaces.elements();
      while(s.hasMoreElements())
      {
        groupObject groupSpace = (groupObject)s.nextElement();
        if(groupSpace.getKey().equals(event.getThing()))
        {
          spBlock spaceKey = groupSpace.getKey();
          tdPoint keyPoint = new tdPoint(spaceKey.getTransform());
                    // this code sets the new spaceBlock as a child to the key
          tdPoint keyTemp = keyPoint.copy();
          Enumeration l = groupSpace.getConnected().elements();
          while(l.hasMoreElements())
          {
            tdPoint keyINV = keyTemp.copy();
            keyINV.inverse();
            spaceObject nextSpaceO = (spaceObject)l.nextElement();
            spBlock nextSpace = nextSpaceO.getSpaceBlock();
            if(nextSpace.getParent() != spaceKey)
            {
              tdPoint spacePoint = new tdPoint(nextSpace.getTransform());
              keyINV.multiply(spacePoint);
              nextSpace.setParent(spaceKey);
              nextSpace.setTransform(keyINV);
            }
          }
    System.out.println("This space objects area = " + groupSpace.getTotalArea());
        }
      }
    }
  }
}
```

public class tdViewEditor extends tdEditor

```
/*
** $Id: tdViewEditor.java,v 1.18 1998/10/07 15:16:38 derek Exp $
** Copyright © 1998 MITSUBISHI ELECTRIC ITA.  ALL RIGHTS RESERVED
//  haymaker altered
**
** 05/18/98 DLS Comments
**
*/
import java.awt.*;
import java.util.*;
import com.sun.java.swing.*;
/**  This editor implements a mouse view editor.
Drag Functions:
mouse   right:    Y=Translate up/down        X=Translate left/right
control right:    Y=Tilt up/down around X     X=Translate left/right??
mouse   left:     Y=Translate forward/Backward  X=Rotate left/right around Y
Shift key always aligns to grid.
*/
public class tdViewEditor extends tdEditor {
   static tdViewEditor me = null; // Set once.
   public static tdViewEditor get() {return me;}
   public spMarker getMarker() { zprint(4,"tdViewEditorget: has no marker\n");
 return null;}
   static int thisEditorsIndex;

   public tdViewEditor(){
      ViewInit();
                        //{{INIT_CONTROLS
                        //}}
              }
   private static final int   VIEWS = 20;
   private static final int   CAMERAPOSITIONS = 9;
   private static final int   DEFAULTPOSITION = CAMERAPOSITIONS-1;
   private static final float PERCENT_GOTO = .85f;
   private static final float MAXSPEED = 128;
   private static final float MINSPEED = .001f;
   private static final float DEFAULT_TRAN_SCALE = 20;
   private static final float DEFAULT_ROT_SCALE = 1.8f;
   private static final float DEF_ANGLE = -0.25f;
   private static tdPoint fixedcampos []= {
      new tdPoint( -35,50,35,   0,1,0,    -0.785f), //From SouthWest
      new tdPoint( 50,50,70,    0,1,0,    0), //From South
      new tdPoint( 135,50,35,   0,1,0,    0.785f), //From SouthEast
      new tdPoint( 170,50,-50,  0,1,0,    3.14f), //From East  CHANGED to look
away
      new tdPoint( 135,50,-135, 0,1,0,    2.355f), //From NorthEast
      new tdPoint( 50,50,-170,  0,1,0,    3.14f), //From North
      new tdPoint( -35,50,-135, 0,1,0,    -2.355f), //From NorthWest
      new tdPoint( -70,50,-50,  0,1,0,   -1.57f), //From West
      new tdPoint( 50,150,-50,  0,1,0,    0), //From Top
   };

   public static tdPoint getFixedViewPoint(int n)
   {
      return fixedcampos[n];
   }

   private tdVisualListener secLis, selectLis, keyLis, dragLis,
                                    upLis, gotoLis, frameLis, modeLis;
   private tdPoint deltaXaxis = new tdPoint(tdPoint.axisX);
   private tdPoint deltaYaxis = new tdPoint(tdPoint.axisY);
   private tdPoint tmp = new tdPoint();
   private tdPoint tar = new tdPoint();
   private tdPoint ntar = new tdPoint();
   private int dragStartX;
   private int dragStartY;
//   private float dx,dy,dz,dr,tr;
   private float max_x=500, max_y=500;
   private float tranScale = DEFAULT_TRAN_SCALE;
   private float rotScale = DEFAULT_ROT_SCALE;
   private float KeySpeed = 1;
   private boolean inDrag;
   private static int VPs = 0;
            private static spViewPoint VP[] = new spViewPoint[VIEWS];
            private static JMenuItem miStats;
            private static JMenuItem miView[] = new JMenuItem[VIEWS];
            private static JMenu rideMenu;
            private static JMenu viewMenu;
            private static JMenuItem [] miSpeed = new JMenuItem[4];
            private static JMenu SpeedMenu;
            private static JRadioButtonMenuItem [] miCamera
                        = new JRadioButtonMenuItem[CAMERAPOSITIONS];
```

```
  private static ButtonGroup CameraGroup = new ButtonGroup();
        private static JMenu CameraMenu;
        //HAYMAKER
        private static JButton [] miViews = new JButton[CAMERAPOSITIONS];
        private static JPanel ViewPanel;
        private static ButtonGroup ViewGroup = new ButtonGroup();

  private static tdPredicateCallback ViewCB;
        private LisAction lLisAction = new LisAction();
        private SymAction lSymAction = new SymAction();
  private java.awt.Font regFont, boldFont;
  private boolean shift;
  private int cameraPosition = DEFAULTPOSITION;
  //private float[] tiltTransDown = {0 ,0, 0, 1, 0, 0, -1.57f, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0};
  //private float[] tiltTransAngle = {0 ,0, 0, 1, 0, 0, -.0785f, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0};
  /** Initialize this editor.*/
  public void ViewInit() {
    if (me != null) return;
    me = this;
                on = true; //Assumes its active

    selectLis = new tdVisualListener(lLisAction,tdVisualListener.
                EVENT_MOUSE_DOWN,on); // onMouseDown
    upLis    = new tdVisualListener(lLisAction,tdVisualListener.
                EVENT_MOUSE_UP,on); // onMouseUp
    dragLis  = new tdVisualListener(lLisAction,tdVisualListener.
                EVENT_MOUSE_DRAG,on); // onNavDrag
    frameLis  = new tdVisualListener(lLisAction,tdVisualListener.
                EVENT_EACH_FRAME,false); // onNavFrame
    gotoLis  = new tdVisualListener(lLisAction,tdVisualListener.
                EVENT_MOUSE_LEFT_DOUBLE,on); // onNavGoto
    // THESE ARE GLOBAL Never disabled...
    modeLis  = new tdVisualListener(lLisAction,tdVisualListener.
                EVENT_MOUSE_BOTH,true); // onNavModeChange
    keyLis   = new tdVisualListener(lLisAction,tdVisualListener.
                EVENT_KEY_PRESS|tdVisualListener.
                EVENT_KEY_RELEASE,true); // onNavKey

    secLis   = new tdVisualListener(lLisAction,tdVisualListener.
                EVENT_EACH_SECOND,true); // onNavSec

    regFont = schmoozer.getRegFont();
    boldFont = schmoozer.getBoldFont();

                zprint(3,"NavInit regiserting self\n");
                // Register Myself, and add MenueItem
        KeyStroke jkey = KeyStroke.getKeyStroke(java.awt.event.KeyEvent
                         .VK_B,java.awt.Event.CTRL_MASK,false);
        thisEditorsIndex = schmoozer.registerEditor(me,"Browse mode",'B',
        tdVisualListener.KEY_CTRL_B,jkey,"images/browseMode.gif",null);
                zprint(3,"NavInit making menuItems \n");
    // Speed Menue Items
                SpeedMenu = new JMenu("Speed");
                SpeedMenu.setMnemonic('p');
                SpeedMenu.setFont(regFont);
        SpeedMenu.setToolTipText("Change the avatars navigation speed.");

                miSpeed[0] = new JMenuItem("Faster",'F');
                miSpeed[1] = new JMenuItem("Slower",'S');
                miSpeed[2] = new JMenuItem("Default",'D');

                schmoozer.addIcon(SpeedMenu,"images/speed.gif");
                schmoozer.addIcon(miSpeed[0],"images/
speedFaster.gif");
                schmoozer.addIcon(miSpeed[1],"images/
speedSlower.gif");
                schmoozer.addIcon(miSpeed[2],"images/
speedModerate.gif");
                for (int i=0; i<3; i++) {
                  if (i==3-1) SpeedMenu.addSeparator();
                  SpeedMenu.add(miSpeed[i]);
                  miSpeed[i].addActionListener(lSymAction);

                  miSpeed[i].setFont( regFont );
                }
                miSpeed[2].setFont( boldFont ); // Default
                schmoozer.addMenu("Avatar",SpeedMenu);

                // Camera Menue Items
                CameraMenu = new JMenu("Camera Position");
                CameraMenu.setFont( regFont );
                CameraMenu.setMnemonic('C');
        CameraMenu.setToolTipText("Move the camera relative to the
avatar.");
```

```
                // HAYMAKER Panel for view buttond
                ViewPanel = new JPanel();
                ViewPanel.setLocation(110,5);
                ViewPanel.setSize(100, 100);
                ViewPanel.setVisible(true);
                ViewPanel.setBackground(Color.black);
                ViewPanel.setLayout(null);
                ViewPanel.show();

        schmoozer.getSchmoozer().getContentPane().add(ViewPanel);

                miCamera[0] = new
JRadioButtonMenuItem("SouthWest");
                miCamera[1] = new JRadioButtonMenuItem("South");
                miCamera[2] = new JRadioButtonMenuItem("Southeast");
                miCamera[3] = new JRadioButtonMenuItem("East");
                miCamera[4] = new JRadioButtonMenuItem("NorthEast");
                miCamera[5] = new JRadioButtonMenuItem("North");
                miCamera[6] = new JRadioButtonMenuItem("NorthWest");

                miCamera[7] = new JRadioButtonMenuItem("West");
                miCamera[8] = new JRadioButtonMenuItem("Top");


                miViews[0] = new JButton("SW");
                miViews[0].setSize(30,30);
                miViews[0].setLocation(5,65);
miViews[0].setMargin(new Insets(0,0,0,0));

                miViews[1] = new JButton("S");
                miViews[1].setSize(30,30);
                miViews[1].setLocation(35,65);
                miViews[1].setMargin(new Insets(0,0,0,0));

                miViews[2] = new JButton("SE");
                miViews[2].setSize(30,30);
                miViews[2].setLocation(65,65);
                miViews[2].setMargin(new Insets(0,0,0,0));

                miViews[3] = new JButton("=>");
                miViews[3].setSize(30,30);
                miViews[3].setLocation(65,35);
                miViews[3].setMargin(new Insets(0,0,0,0));

                miViews[4] = new JButton("NE");
                miViews[4].setSize(30,30);
                miViews[4].setLocation(65,5);
                miViews[4].setMargin(new Insets(0,0,0,0));

                miViews[5] = new JButton("N");
                miViews[5].setSize(30,30);
                miViews[5].setLocation(35,5);
                miViews[5].setMargin(new Insets(0,0,0,0));

                miViews[6] = new JButton("NW");
                miViews[6].setSize(30,30);
                miViews[6].setLocation(5,5);
                miViews[6].setMargin(new Insets(0,0,0,0));

                miViews[7] = new JButton("W");
                miViews[7].setSize(30,30);
                miViews[7].setLocation(5,35);
                miViews[7].setMargin(new Insets(0,0,0,0));

                miViews[8] = new JButton("T");
                miViews[8].setSize(30,30);
                miViews[8].setLocation(35,35);
                miViews[8].setMargin(new Insets(0,0,0,0));

                for (int i=0; i < 9; i++)
                {
                miViews[i].addActionListener(lSymAction);
                }

                miCamera[0].setMnemonic('Z');
                miCamera[1].setMnemonic('X');
                miCamera[2].setMnemonic('C');
                miCamera[3].setMnemonic('D');
                miCamera[4].setMnemonic('E');
                miCamera[5].setMnemonic('W');
                miCamera[6].setMnemonic('Q');
                miCamera[7].setMnemonic('A');
                miCamera[8].setMnemonic('S');
```

```
                    /*HAYMAKER
            schmoozer.addIcon(CameraMenu, "images/camView.gif");
            schmoozer.addIcon(miCamera[0],"images/camSouthEast.gif");
            schmoozer.addIcon(miCamera[1],"images/camSouth.gif");
            schmoozer.addIcon(miCamera[2],"images/camSouthWest.gif");
            schmoozer.addIcon(miCamera[3],"images/camWest.gif");
            schmoozer.addIcon(miCamera[4],"images/camNorthWest.gif");
            schmoozer.addIcon(miCamera[5],"images/camNorth.gif");
            schmoozer.addIcon(miCamera[6],"images/camNorthEast.gif");
            schmoozer.addIcon(miCamera[7],"images/camEast.gif");
            schmoozer.addIcon(miCamera[8],"images/camTop.gif");
                    */
                    for (int i=0; i<CAMERAPOSITIONS; i++) {
              if (i == (CAMERAPOSITIONS-1)) CameraMenu.addSeparator();
                       CameraMenu.add(miCamera[i]);
                       ViewPanel.add(miViews[i]);
                       CameraGroup.add(miCamera[i]);
                       ViewGroup.add(miViews[i]);
                       miCamera[i].setFont( regFont );
                       miCamera[i].addActionListener(lSymAction);
                    }
                    miCamera[DEFAULTPOSITION].setFont( boldFont );

                    miCamera[DEFAULTPOSITION].setSelected( true );


                    schmoozer.addMenu("View",CameraMenu);
                    zprint(3,"ViewInit setting up Dynamic View Points\n");

                    // Dynamic View Points
      ViewCB = new tdPredicateCallback(this,null,"AnyChange","spViewPoint",
                    "onViewPointChanged"); // should be newOrChanged

            rideMenu = new JMenu("Ride");
            viewMenu = new JMenu("Goto Position");
                    rideMenu.setMnemonic('R');
                    viewMenu.setMnemonic('G');

            schmoozer.addMenu("View",rideMenu);
            schmoozer.addMenu("View",viewMenu);
            onViewPointChanged(null,null); // display "reset" options.
            rideMenu.setFont( regFont );
            viewMenu.setFont( regFont );
                    rideMenu.setToolTipText("Make the Avatar Ride on this.");
                viewMenu.setToolTipText("Move the avatar to this view point.");

                    schmoozer.addIcon(rideMenu,"images/ride.gif");
                    schmoozer.addIcon(viewMenu,"images/gotoPosition.gif");

      //setCameraFlyDistances(); // read the user defined properties
            // Stats
                    miStats = new JMenuItem("Statistics");
                    miStats.setMnemonic('S');
                    miStats.setFont( regFont );
                    miStats.setToolTipText("Display statistics");
                    schmoozer.addMenuItem("View",miStats,-1);
                    miStats.addActionListener(lSymAction);
                    schmoozer.addIcon(miStats,"images/stats.gif");

      on = true;
    }
    /** Sets the camera fly distances based on user preferences. */
    public static void setCameraFlyDistances(){
       /*
     float south  = schmoozer.getFloatPreference("camSouthDistance");
    float southeast   = schmoozer.getFloatPreference("camSouthEastDistance");

    float east  = schmoozer.getFloatPreference("camEastDistance");
    float northeast = schmoozer.getFloatPreference("camNorthEastDistance");

    float north = schmoozer.getFloatPreference("camNorthDistance");
    float northwest   = schmoozer.getFloatPreference("camNorthWestDistance");

    float west    = schmoozer.getFloatPreference("camWestDistance");
    float southwest   = schmoozer.getFloatPreference("camSouthWestDistance");
    float top   = schmoozer.getFloatPreference("camTopDistance");

//HAYMAKER NOTE, THESE NEED TO BE CALCULATED
      if (southwest != 0)   {fixedcampos[0].pt[1] = 0.37f*south;
fixedcampos[0].pt[2] = 0.92f*south;}
      if (south != 0)    {fixedcampos[1].pt[1] = southeast;}
      if (southeast != 0)   {fixedcampos[2].pt[1] = east;}
      if (east != 0)   {fixedcampos[3].pt[1] = northeast;}
      if (northeast != 0)   {fixedcampos[4].pt[1] = -0.1f*north;
```

```
fixedcampos[4].pt[2] = -0.99f*north;}
      if (north != 0)    {fixedcampos[5].pt[0] =  0.99f*northwest;
 fixedcampos[5].pt[1] = -0.006f*northwest;}
      if (northwest != 0)     {fixedcampos[6].pt[0] = -0.99f*west;
  fixedcampos[6].pt[1] = -0.006f*west;}
      if (west != 0)     {fixedcampos[6].pt[0] = southwest;}
      if (top != 0)     {fixedcampos[6].pt[0] = top;}
      */
    }
    /** Activate a View Points.
      * @param i   View point to display.
      */
    public void viewPoint_Action(int i){
      td.zprint(3,"viewPoint_Action "+i+"\n");
      spViewPoint vp = VP[i];
      tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();

      if (vp != null) {
         String title = vp.getTitle();
         String state = vp.getState();
         boolean camOnly = ( title.startsWith("cam") || title.startsWith("Cam")
                                                              || vp.getCamera() );
         boolean parent = state.equalsIgnoreCase("ride") || vp.getRide() ;
         tdPoint pnt = parent ? null : tdx.spThingGetLocaleTransform(vp);
         if (camOnly)    platform.cameraGoto(pnt,parent ? vp :
                                                        null ,3000,false);
         else            platform.avatarGoto(pnt,parent ? vp :
                                                        (sp)vp.getLocale(),6000,false);

         td.zprint(1,"\n\nVVVVVVVV viewPoint_Action  camOnly="
                                        +camOnly+"  parent="+parent+"\n\n");
                 highlightView(i); // Highlight the selection
      } else { // RESET (has no VP) End flight,ride restore Transforms
         boolean keepPositon = (i == VPs - 1); // ride reset
         flyCamera(false);
         highlightView(-1);
         platform.resetAvatarsTransforms(keepPositon,3000);
      }
    }
    /** Make the active view bold.
      * @param n  view to highlight.
      */
    private void highlightView(int n){
      zprint(1,"highlightView "+n+"  VPs="+VPs+".\n");
               for (int i=0; i<VPs-2; i++)
                  if (miView[i]!=null)
                     miView[i].setFont((i == n)? boldFont : regFont);
    }
    // if the thing moving around is the viewpoint itself,
     //this code could execute every frame, to be fixed...
    // if the viewpoint moves every frame, this code could be called very often,
    // and we know setting/unsetting menue items is a lot of work. Plus there could
    // be many view points in the WM and they all could be moving. So we limit
    // this work to once per frame.

    /** True if any view point has changed.*/
    boolean viewPointChanged;
    /** Callback that detects changed viewpoints.*/
    public void onViewPointChanged(Integer cbid, Integer spid) {
      viewPointChanged = true;
    }

    /** The current locale.*/
    spSchmoozerLocale locale;

    /** Check avatars and viewpoints for changes and adjust UI.*/
    public void onNavSec() {

      // check the avatar's locale (in case we lost it).
  // It's also checked after each localize i.e. every move
      tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();

    platform.checkAvatarsLocale(); // THIS LINE SOMETIMES CAUSES LOCK UP...
  // PERHAPS AT SCHMOOZER SET BROWSER TITLE????

      // This would be better if it was in schmoozer.java:
     tdxs.removeExpiredMovers(); // remove delete animations and expired movers.
      // See if we've changed locales  spSchmoozerLocale.getHash();
       // (sp) spHash.get(new Integer(sp));
      if ( platform.getAvatarsLocale() != locale  ) {
         zprint(1,"onNavSec locale changed.\n");
         locale = platform.getAvatarsLocale();
         viewPointChanged = true;
      }
    if (viewPointChanged) {
      viewPointChanged = false;
      zprint(3,"onNavSec viewPointChanged\n");

      JMenuItem mi;
      spViewPoint vp;

      // Remove all
      viewMenu.removeAll();
      rideMenu.removeAll();
      for (int i = 0; i<VPs; i++)
        if (miView[i] != null) {
          miView[i].removeActionListener(ISymAction);
          miView[i] = null;
        }
      VPs = 0;
      // I want to recompute the names on the fly incase someone changed
      // Add each viewpoint back
      Hashtable hash = spViewPoint.getHash();
      Enumeration e = hash.elements();
      String title;
      while (e.hasMoreElements()) {
        if ((vp = (spViewPoint) e.nextElement()) != null ) {
  boolean camera = vp.getCamera();// || (vp.getState().equalsIgnoreCase("ride"));
  boolean ride = vp.getRide();// || (vp.getState().equalsIgnoreCase("ride"));
spPart parent = vp.getParent() instanceof spPart ? (spPart)vp.getParent() : null;
          if (parent==null)   title = vp.getTitle() + " " + vp.getType();
          else
           title = parent.getTitle() + " " + parent.getType() + " " + vp.getTitle() + " "
                                                      + vp.getType();
          //if (title.length() > 1) {
          // If it's camera only, it should be inside this locale.
          if (camera && vp.getLocale() != locale ) continue;

          td.zprint(3,"onEachView "+VPs+" "+title+"\n");
          mi = new JMenuItem(title);
          if (ride) { // add to ride list
            rideMenu.add(mi);
          } else { // add to view list
            viewMenu.add(mi);
          }
          miView[VPs] = mi;
          VP[VPs++] = vp;
          mi.addActionListener(ISymAction);
          if (++VPs > VIEWS - 3) break; // Save room for resets
          //}
        }
      }// while

      // Add resets
      viewMenu.addSeparator();
      rideMenu.addSeparator();
      viewMenu.add(miView[VPs++] = (mi = new JMenuItem("Reset")) );
      mi.addActionListener(ISymAction);
      rideMenu.add(miView[VPs++] = (mi = new JMenuItem("Reset")) );
      mi.addActionListener(ISymAction);
      td.zprint(3,"\n onViewPointChanged ENDS "+VPs+" viewPoints\n\n");
    } // viewPointChanged
  }
  /** Camera only view points.
   * @param n     Fixed camera position.
   * @param move  If true move the camera.
   */
          public void miAvatar_Action(int n,boolean move)
          {
    zprint(1,"miAvatar_Action n="+n+" move="+move+".\n");
    tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();

             cameraPosition = n;
                     if (move)
                     {
               float[] tran = platform.getTilt().getTransform();

               tran[spThing.spTransformRX] = 1;
               tran[spThing.spTransformRY] = 0;
               tran[spThing.spTransformRZ] = 0;
      tran[spThing.spTransformRA] = (n < 8) ? -0.333f : -1.57f;
      platform.getTilt().setTransform(tran);
     platform.avatarGoto(fixedcampos[n],platform.getAvatarsLocale(),3500,true);

               miCamera[n].setSelected(true);
               miViews[n].setSelected(true);
                       for (int i=0; i<CAMERAPOSITIONS; i++)
                        miCamera[i].setFont( i == n ? boldFont : regFont );
```

```
   * @param p            Part to set the target to.
   * @param moveToTarget  If true, animate the move.
   */
  public void setActiveMarkersTarget(spPart p, boolean moveToTarget) {
    zprint(4,"setActiveMarkersTarget by tdViewEditor\n");
    tdEditor editor;
    spMarker marker;
    if ((null != (editor = schmoozer.getActiveEditor())) &&
      (null != (marker = editor.getMarker())))
      marker.setTarget(p,moveToTarget);
  }

  /** Changes the navigatoin mode by clicking on a target,
                              //then the right editor is found for that part */
  public void onNavModeChange(tdVisualEvent event) {
    zprint(1,"onNavModeChange\n");
    int next;
    spThing t = event.getThing();
    if (on) { // Was navagating, and hit a target.
      next = schmoozer.lastEditorUsed; // getLastEditorUsed()???
      if (t != null) {
        if (t instanceof spTrack)   next = 1;
        if (t instanceof spRailCar) next = 2;
        if (t instanceof spScenery) next = 3;
        if (t instanceof spBlock) next = 4;
      }
    } else // Was not navigating.
      next = 0;
    schmoozer.activateEditor(next);
    if (t != null && t instanceof spPart)
      setActiveMarkersTarget((spPart)t,true);
  }
 /** Process a key from spVisuals keyboard.*/
  public void onNavKey(tdVisualEvent event) {
    float dx=0,dy=0,dz=0,dr=0,tr=0;
    zprint(1,"onNavKey "+event.getValue()+"\n");

    if ((event.getEventType() & tdVisualListener.EVENT_KEY_RELEASE) != 0) {
      onMouseUp();
    } else if (!inDrag) { // Press
      tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();
      if ((platform.getAvatarsLocale()) == null) return; // No locale

      float RotStep = 0.2f * (float) java.lang.Math.sqrt(KeySpeed);
      float MoveStep = 20 * KeySpeed;
      switch(event.getValue()) {
        case tdVisualListener.KEY_END:
        case tdVisualListener.KEY_1:       tr = -RotStep; break; // Tilt Down
        case tdVisualListener.KEY_DOWN:
        case tdVisualListener.KEY_2:       dz =  MoveStep; break; // Back
        case tdVisualListener.KEY_PAGE_DOWN:
        case tdVisualListener.KEY_3:       dy = -MoveStep/2.0f; break; // Down
        case tdVisualListener.KEY_LEFT:
        case tdVisualListener.KEY_4:       dr =  RotStep; break; // rot left
        case tdVisualListener.KEY_RIGHT:
        case tdVisualListener.KEY_6:       dr = -RotStep; break; // rot right
        case tdVisualListener.KEY_HOME:
        case tdVisualListener.KEY_7:       tr =  RotStep; break; // tilt up
        case tdVisualListener.KEY_UP:
        case tdVisualListener.KEY_8:       dz = -MoveStep; break; // Forward
        case tdVisualListener.KEY_PAGE_UP:
        case tdVisualListener.KEY_9:       dy =  MoveStep/2.0f; break; // Up
        case tdVisualListener.KEY_0:       flyCamera(false);
          // End flight and zero tilt, keep position&rot but reset transform.
                              highlightView(-1);
                           platform.resetAvatarsTransforms(true,2000); break;
        case tdVisualListener.KEY_5:       flyCamera(false);
                                   // End ride,flight, restore Transforms
                              highlightView(-1);
                           platform.resetAvatarsTransforms(false,3500); break;
        case tdVisualListener.KEY_PERIOD:   flyCamera(true); break;
        case tdVisualListener.KEY_POSITIVE: KeySpeed = KeySpeed*2.0f;
                     if (KeySpeed>MAXSPEED) KeySpeed=MAXSPEED;
                              zprint(1,"Speed="+KeySpeed+"\n"); break;
        case tdVisualListener.KEY_NEGITIVE: KeySpeed = KeySpeed*0.5f;
                    if (KeySpeed<MINSPEED) KeySpeed=MINSPEED;
                              zprint(1,"Speed="+KeySpeed+"\n"); break;
        default: ;
      }

      if ( (dx != 0) || (dy != 0) || (dz != 0) || (dr != 0) || (tr != 0) ) {
        if (tr != 0) { // Tilt
          deltaXaxis.pt[spThing.spTransformRA] = tr;
          platform.avatarGoToward(deltaXaxis,event.getShiftKey());
        } else { // Translate
```

```
               deltaYaxis.pt[spThing.spTransformX]  = dx;
               deltaYaxis.pt[spThing.spTransformY]  = dy;
               deltaYaxis.pt[spThing.spTransformZ]  = dz;
               deltaYaxis.pt[spThing.spTransformRA] = dr;
               platform.avatarGoToward(deltaYaxis,event.getShiftKey());
            }
            inDrag = true; // Continue the drag
            frameLis.enable(true);
         }

         // applyDelta(event); // Make the move.
      } // else inDrag so continue.
   }
   /** Required for all editors. Enables and diables the visual event listeners.
    * @param show  True to enable.
    */
   public void enable(boolean show){
      if (on != show) {
                  on = show;
         zprint(2,"NavEditor enable "+on+"\n");
         upLis.enable(on);
         dragLis.enable(on);
         gotoLis.enable(on);
      }
   }

   public int currentView()
   {
      for (int i = 0; i < CAMERAPOSITIONS; i++)
      {
         if(miViews[i].isSelected())
         {
            return i;
         }
      }
      return 8;
   }

                              //{{DECLARE_CONTROLS
               //}}
}
```

public class schmoozer extends JFrame

```
** $Id: schmoozer.java,v 1.13 1998/11/03 21:41:13 derek Exp $
** Copyright © 1998 MITSUBISHI ELECTRIC ITA.  ALL RIGHTS RESERVED

//HAYMAKER MODIFIED
**


*/
import java.awt.*;
import java.net.*;
import java.io.*;
import java.util.*;
import java.awt.event.*;
import java.lang.reflect.*;
import com.sun.java.swing.*;
import com.sun.java.swing.ButtonGroup;
import com.sun.java.swing.JFrame;
import com.sun.java.swing.JMenuBar;
import com.sun.java.swing.JMenu;
import com.sun.java.swing.JMenuItem;
import com.sun.java.swing.JSeparator;
import com.sun.java.swing.JCheckBoxMenuItem;
import com.sun.java.swing.JRadioButtonMenuItem;
/** Schmoozer main class creates the main menu and loads and
  * initalizes the other classes and lauches the tdUpdate thread.
  */
public class schmoozer extends com.sun.java.swing.JFrame
{
   // Most of these should be FINAL, but FINAL confuses cafe wizards
   public static String arguments[];
   public static tdWM wm = null; // Used for locks
   public static String ROOT_FILE_URL = “file:///s:/demo/trains/definitions/”;
   public static String ROOT_HTTP_URL = “file:///s:/demo/trains/definitions/”;
   public static String ROOT_URL;
   public static String getRootURL() {return ROOT_URL;};
   public static String archiveDir = “file:///s:/demo//Trains//locales//archives”;
  public static String kitDir    = “\\demo\\Trains\\locales\\kits”;
  public static String idfDir    = “\\demo\\Trains\\definitions”;
  public static String startingLocale = null; //“A”; If set to null, ask user
  public static String startingRestore = null;
   public static String propertiesFile = “schmoozer.pro”;

   static schmoozer me;
   static tdUpdate tdUp; // Thread that calls update
   static boolean restoring = false;
   static boolean starting = true;

   public static java.awt.Font regFont =
                  new java.awt.Font(“Dialog”,java.awt.Font.PLAIN,12); // PLAIN
   public static java.awt.Font boldFont =
                    new java.awt.Font(“Dialog”,java.awt.Font.BOLD, 12);
         public static java.awt.Font getRegFont() {return regFont;}
         public static java.awt.Font getBoldFont() {return boldFont;}
 public static ImageIcon schmoozerIcon = new ImageIcon(“images/
schmoozer.gif”);
         static com.sun.java.swing.JMenuBar MainMenuBar;
         static com.sun.java.swing.JMenu FileMenu;
         static com.sun.java.swing.JMenuItem miChangeLocale;
         static com.sun.java.swing.JMenuItem miDeleteLocale;
         static com.sun.java.swing.JSeparator jSeparator1;
         static com.sun.java.swing.JMenuItem miSaveLocale;
         static com.sun.java.swing.JMenuItem miRestoreLocale;
         static com.sun.java.swing.JMenuItem miLocaleOptions;
         static com.sun.java.swing.JMenuItem miPeferences;
         static com.sun.java.swing.JMenuItem miLoadTool;
         static com.sun.java.swing.JSeparator jSeparator2;
         static com.sun.java.swing.JSeparator jSeparator3;
         static com.sun.java.swing.JMenuItem miExit;
         static com.sun.java.swing.JMenu EditMenu;
         static com.sun.java.swing.JMenu ModeMenu;
         static com.sun.java.swing.JMenu ViewMenu;
         static com.sun.java.swing.JMenu HelpMenu;
         static com.sun.java.swing.JMenu AvatarMenu;
         static com.sun.java.swing.JMenuItem miAbout;
         public static xSymAction xAction;
         public static SymAction lSymAction;
         public static SymWindow aSymWindow;
         java.awt.FileDialog LoadToolDialog;
         java.awt.FileDialog RestoreLayoutDialog;
         java.awt.FileDialog SaveLayoutDialog;
         java.awt.FileDialog NewPartDialog;
   /** Hash table of all sp.*/
```

```
   transient static private Hashtable sps = new Hashtable();
   /** Hash table of all spThings.*/
   transient static private Hashtable spthings = new Hashtable();
   /** Returns the hash table of all sp's. */
   public static Hashtable getSpHash() {return sps;}
   /** Returns the hash table of all spThings's. */
   public static Hashtable getSpThingHash() {return spthings;}
            static {
MainMenuBar = new com.sun.java.swing.JMenuBar();
FileMenu = new com.sun.java.swing.JMenu(“File”);
FileMenu.setMnemonic(‘F’);
FileMenu.setActionCommand(“File”);
FileMenu.setFont(regFont);
MainMenuBar.add(FileMenu);

miChangeLocale = new com.sun.java.swing.
            JMenuItem(“Change Locale...”,’C’);
miChangeLocale.setMnemonic(‘C’);

miChangeLocale.setActionCommand(“Change Locale...”);
miChangeLocale.setFont(regFont);
miChangeLocale.setToolTipText(“Moves the avatar to
                       a new locale. It can be one you already have, or one you
know the URL for or, it will makes a new locale.”);
                   FileMenu.add(miChangeLocale);
miDeleteLocale = new com.sun.java.swing.JMenuItem(“Delete Locale...”,’D’);
miDeleteLocale.setActionCommand(“Delete Locale...”);
miDeleteLocale.setFont(regFont);
miDeleteLocale.setToolTipText(“Deletes a locale you own and are not in”);
                   FileMenu.add(miDeleteLocale);

jSeparator1 = new com.sun.java.swing.JSeparator();
jSeparator1.setFont(regFont);
                   FileMenu.add(jSeparator1);

miSaveLocale = new com.sun.java.swing.JMenuItem(“Save Locale...”,’S’);
miSaveLocale.setActionCommand(“Save Locale...”);
miSaveLocale.setFont(regFont);
miSaveLocale.setToolTipText(“Serialize a locale and its contents to a file.”);
                   FileMenu.add(miSaveLocale);

miRestoreLocale = new com.sun.java.swing.JMenuItem(“Restore Locale...”,’R’);
 miRestoreLocale.setActionCommand(“Restore Locale...”);
miRestoreLocale.setFont(regFont);
miRestoreLocale.setToolTipText(“Restores locale(s) and contents from a file.”);
                   FileMenu.add(miRestoreLocale);

miLocaleOptions = new com.sun.java.swing.JMenuItem(“Locale Options...”);
miLocaleOptions.setMnemonic(‘O’);
miLocaleOptions.setActionCommand(“Options...”);
miLocaleOptions.setFont(regFont);
miLocaleOptions.setToolTipText(“Modify locale settings.”);
                   FileMenu.add(miLocaleOptions);

jSeparator3 = new com.sun.java.swing.JSeparator();
jSeparator3.setFont(regFont);
                   FileMenu.add(jSeparator3);


miLoadTool = new com.sun.java.swing.JMenuItem(“Load Tool...”);
miLoadTool.setMnemonic(‘L’);
miLoadTool.setActionCommand(“Load Tool...”);
miLoadTool.setFont(regFont);
miLoadTool.setToolTipText(“Adds new tools (editors) to Schmoozer from a .jar
file.”);
FileMenu.add(miLoadTool);

miPeferences = new com.sun.java.swing.JMenuItem(“Preferences...”);
miPeferences.setMnemonic(‘P’);
miPeferences.setActionCommand(“Preferences...”);
miPeferences.setFont(regFont);
miPeferences.setToolTipText(“Modify environment settings.”);
FileMenu.add(miPeferences);

jSeparator2 = new com.sun.java.swing.JSeparator();
jSeparator2.setFont(regFont);
FileMenu.add(jSeparator2);

miExit = new com.sun.java.swing.JMenuItem(“Exit...”,’E’);
miExit.setActionCommand(“Exit”);
miExit.setFont(regFont);
miExit.setToolTipText(“Quit Schmoozer”);
FileMenu.add(miExit);

EditMenu = new com.sun.java.swing.JMenu(“Edit”);
```

```
EditMenu.setMnemonic(‘E’);
EditMenu.setActionCommand(“Edit”);
EditMenu.setFont(regFont);
                            MainMenuBar.add(EditMenu);

                            ViewMenu = new com.sun.java.swing.JMenu(“View”);
                            ViewMenu.setMnemonic(‘V’);
                            ViewMenu.setActionCommand(“View”);
                            ViewMenu.setFont(regFont);
                            MainMenuBar.add(ViewMenu);

                            ModeMenu = new com.sun.java.swing.JMenu(“Mode”);
                            ModeMenu.setMnemonic(‘M’);
                            ModeMenu.setActionCommand(“Mode”);
                            ModeMenu.setFont(regFont);
                            MainMenuBar.add(ModeMenu);

                            AvatarMenu = new com.sun.java.swing.JMenu(“Avatar”);
                            AvatarMenu.setMnemonic(‘A’);
                            AvatarMenu.setActionCommand(“Avatar”);
                            AvatarMenu.setFont(regFont);
                            MainMenuBar.add(AvatarMenu);

                            HelpMenu = new com.sun.java.swing.JMenu(“Help”);
                            HelpMenu.setMnemonic(‘H’);
                            HelpMenu.setActionCommand(“Help”);
                            HelpMenu.setFont(regFont);
                            MainMenuBar.add(HelpMenu);
                            miAbout = new com.sun.java.swing.JMenuItem(“About...”);
                            miAbout.setMnemonic(‘A’);
                            miAbout.setActionCommand(“About”);
                            miAbout.setFont(regFont);
                            miAbout.setToolTipText(“About Schmoozer”);
                            HelpMenu.add(miAbout);

              }
              public schmoozer() {
      me = this;
              // This code is automatically generated by Visual Cafe when you add
              // components to the visual environment. It instantiates and initializes
              // the components. To modify the code, only use code syntax
              // what Visual Cafe can generate, or Visual Cafe may be unable to
back
                            // parse your Java file into its visual environment.
                            getContentPane().setLayout(null);
                            setVisible(false);
                            setSize(215,500);
                            getContentPane().setBackground(Color.gray);
                            setBrowserTitle(“Schmoozer Starting...”);

                            //getContentPane().add(ViewMenu);

      // Experimental... can be removed...
      update( getGraphics() );
                            setIconImage(schmoozerIcon.getImage());


                            setForeground(new Color(0));
                            LoadToolDialog = new java.awt.FileDialog(this);
                            LoadToolDialog.setMode(FileDialog.LOAD);
                            LoadToolDialog.setTitle(“Load Tools”);
                            RestoreLayoutDialog = new java.awt.FileDialog(this);
                            RestoreLayoutDialog.setMode(FileDialog.LOAD);
                            RestoreLayoutDialog.setTitle(“Restore Locale”);
      RestoreLayoutDialog.setDirectory(archiveDir);
                            SaveLayoutDialog = new java.awt.FileDialog(this);
                            SaveLayoutDialog.setMode(FileDialog.SAVE);
                            SaveLayoutDialog.setTitle(“Save Locale”);
      SaveLayoutDialog.setDirectory(archiveDir);
                            NewPartDialog = new java.awt.FileDialog(this);
                            NewPartDialog.setMode(FileDialog.LOAD);
                            NewPartDialog.setTitle(“New Part”);
                            NewPartDialog.setDirectory(idfDir);


                            setJMenuBar(MainMenuBar);

                            aSymWindow = new SymWindow();
                            this.addWindowListener(aSymWindow);
                            lSymAction = new SymAction();
                            xAction = new xSymAction();
                            miLocaleOptions.addActionListener(lSymAction);
                            miPeferences.addActionListener(lSymAction);
                            miLoadTool.addActionListener(lSymAction);
                            miAbout.addActionListener(lSymAction);
```

```
                      miExit.addActionListener(lSymAction);
                      miChangeLocale.addActionListener(lSymAction);
                      miSaveLocale.addActionListener(lSymAction);
                      miDeleteLocale.addActionListener(lSymAction);
                      miRestoreLocale.addActionListener(lSymAction);

validLocale(false); // Turn off GUI
parseArguments();

                      loadPreferences();
String loadStr = getPreference(“load”);
String [] loads = tdxs.parseTokens(loadStr,”,”,true);
String avatarName = getPreference(“hoverPod”);
int desiredInterval = getIntPreference(“frameInterval”);

ROOT_URL = getPreference(“rootURL”);

tdUrlCache.init(); // tdWM init expects to find tdUrlCache class loaded.

//boolean http = 1 =
//ROOT_URL = http ? ROOT_HTTP_URL : ROOT_FILE_URL;


wm = new tdWM( schmoozer.getPreference(“localeServer”) );

// Listen for keys and window closing
lLisAction = new LisAction();
keyLis = new tdVisualListener(lLisAction,tdVisualListener.
                                     EVENT_KEY_PRESS,true);
closeLis = new tdVisualListener(lLisAction,tdVisualListener.
                                     EVENT_WINDOW_CLOSE,true);

//  Initalize the WM callbacks, new removed and OMI.
td.zprint(2,”schmoozer init WM callbacks on sp, spThing, and spOMI.\n”);
new tdPredicateCallback(this,null,”spJustNew”,    “sp”,”onSpNew”);
new tdPredicateCallback(this,null,”spJustRemoved”,”sp”,”onSpRemoved”);
new tdPredicateCallback(this,null,”OMIPredicate”,”spOMI”,”onOMI”);

                      makeMonitors();
int i = 0;
while (loads[i] != null) {

   setTitle( “Loading “+loads[i]+”...”);
   td.zprint(1,”Loading “+loads[i]+”...\n”);
   tdEditor.create(loads[i++]);
}

setTitle(“HoverPod “+avatarName+”...”);

 tdAvatarPlatform.miAvatarChange(avatarName);

moreGUIinit();

// Update WM thread
setTitle(“Starting world model...”);
tdUp = new tdUpdate(desiredInterval);
tdUp.start();
schmoozer.WMupdate(); // The first call.


if (startingRestore != null)
   restoreAction(tdxs.parseNameFromPath(startingRestore),startingRestore);

if (startingLocale != null)
   (new tdLocalesDialog()).gotoLocale(startingLocale,tdLocalesDialog.
                      GOTO_ANY_LOCALE); // goto starting layout, or make it.

                      //setBrowserTitle( “No Locale” );
starting = false; // So new title can display
                      setBrowserTitle();
td.zprint(2,”Init Completed\n”);
                      // {{ R EGISTER_LISTENERS
                      //SymMouse aSymMouse = new SymMouse();
                      //this.addMouseListener(aSymMouse);
                      // }}

                      //{{INIT_CONTROLS
                      //}}
                      //{{INIT_MENUS
                      //}}
                      //Create The Locales
                      if ((schmoozer.getPreference(“dispatcher”)).compareTo
                              (schmoozer.getPreference(“beaconServer”)) == 0)
                      {
                      String ls = schmoozer.getPreference(“localeServer”);
```

```
                         ls = “istp://” + ls;
  spSchmoozerLocale spaceFilter = new spSchmoozerLocale(ls
                                                        + “/spaceFilter”);
spSchmoozerLocale enclosureFilter = new spSchmoozerLocale(ls
                                                        + “/enclosureFilter”);
spSchmoozerLocale structureFilter = new spSchmoozerLocale(ls
                                                        + “/structureFilter”);
spSchmoozerLocale chairFilter = new spSchmoozerLocale(ls
                                                        + “/chairFilter”);
spSchmoozerLocale lightFilter = new spSchmoozerLocale(ls
                                                        + “/lightFilter”);
                         }


              }
   public static void addIcon(JMenuItem mi,String str) { mi.setIcon(
            new ImageIcon(str)); mi.setHorizontalTextPosition(JButton.RIGHT); }
   public static void moreGUIinit(){


     // Apply icons, Swing does not let us do this in static init.
     addIcon(miChangeLocale,”images/changeLocale.gif”);
     addIcon(miDeleteLocale,”images/deleteLocale.gif”);
     addIcon(miRestoreLocale,”images/restoreLocale.gif”);
     addIcon(miSaveLocale,”images/saveLocale.gif”);
     addIcon(miExit,”images/exit.gif”);
     addIcon(miLoadTool,”images/loadTool.gif”);
     addIcon(miPeferences,”images/stats.gif”);
     addIcon(miLocaleOptions,”images/stats.gif”);

   }

////////////////////////////////////////////////////////////////////////////////
///                                                                          ///
///                      CALLBACKS                                                ///
///                                                                          ///
////////////////////////////////////////////////////////////////////////////////
   /** New object callback, add to hash. */
   public void onSpNew(Integer cbid, Integer spid) {
     td.zprint(3,”schmoozer onSpNew “+spid.intValue()+”\n”);
     Object obj = sp.getSpBySp( spid.intValue() );
     if (obj instanceof sp)      sps.put(spid,obj);
     if (obj instanceof spThing) spthings.put(spid,obj);
     td.zprint(3,”schmoozer onSpNew “+spid.intValue()+” Done\n”);
   }

   /** Removed object callback, remove from hash.*/
   public void onSpRemoved(Integer cbid, Integer spid) {
     td.zprint(3,”tdx onSpRemoved “+spid.intValue()+”\n”);
     sps.remove(spid); // Notice how I dont ref the obj? just the spid.
     spthings.remove(spid);
   }

   /** New OMI initalized and parented callback, dispatch its method.*/
   public void onOMI(Integer cbid, Integer spid) {
td.zprint(2,”onOMI “+spid.intValue()+”\n”);
     sp obj = sp.getSpBySp( spid.intValue() );
     if (obj==null || !(obj instanceof spOMI)) {
        td.zerror(“onOMI Bad spOMI=”+td.spLab(obj)+”\n”);
        return;
     }
td.zprint(2,”onOMI omi=”+td.spLab(obj)+” omi.sp=”+obj.sp+”\n”);
     spOMI omi = (spOMI) obj;
     Object target = omi.getParent();
     String c = omi.getMethodClassName();
     String m = omi.getMethodName();
     if (target==null) {
        return;
     }
     if (m.equals(“”)) {
        return;
     }

     tdEditor.callMethod(target,c,m,new Class[]{omi.getClass()},new Object
[]{omi});

     omi.remove(0);
   }

////////////////////////////////////////////////////////////////////////////////
///                                                                          ///
///                      PREFERENCES                                              ///
///                                                                          ///
////////////////////////////////////////////////////////////////////////////////

   public static java.util.Properties preferences; /** Hash table of prefrences. */
```

```
/** Writes all prefrences.*/
           static public void saveAllPreferences() {
   try {
      DataOutputStream dos =
                  new DataOutputStream( new FileOutputStream(propertiesFile) );
      preferences.save(dos,”Schmoozer Properties”);
      dos.close();
   }
   catch (IOException e) { System.out.println(
                                   “Did not write “”+propertiesFile+””\n”); }
}
/** Loads all default prefrenes, then user defined preferences.*/
           static public void loadPreferences() {
              preferences = getDefaultPrefrences(); // Get defaults

   try {  // Read saved values
      DataInputStream dis = openFile(propertiesFile); // new DataInputStream(
                                        new FileInputStream(propertiesFile) );
      preferences.load(dis);
      dis.close();
   }
   catch (IOException e) { System.out.println(“ not read “”+propertiesFile+””\n”); }
 // saveAllPreferences();
 // preferences.save(System.out,PROPERTIESHEADER); // FOR DEBUG ONLY
                         td.setZprint( getIntPreference(“zprint”) );
}

/** Sets the default prefrences.*/
           static java.util.Properties getDefaultPrefrences() {
   java.util.Properties p = new java.util.Properties();
   p.put( “load”,         “tdViewEditor,tdTrackEditor,tdRailCarEditor,
                                tdSceneEditor,tdBlockEditor,tdKeyboardEditor”);
   p.put( “browser”,      “c:\\Program Files\\Netscape\\communicator\\
                                                      Program\\netscape.exe” );
   p.put( “cache”,        “c:\\temp” );
   p.put( “cacheSize”,    “10485760” );
   p.put( “offLine”,      “0” );
   p.put( “beaconServer”, “$COMPUTERNAME” );
   p.put( “localeServer”, “$COMPUTERNAME” );
   p.put( “hoverPod”,     “Bill” );
   p.put( “monitor”,      “” );
   p.put( “rootURL_alt”,  ROOT_FILE_URL);
   p.put( “rootURL”,      ROOT_HTTP_URL);
   p.put( “zprint”,       “0” );
   p.put( “maxPaths”,     “5” );
   p.put( “maxParts”,     “5” );
   p.put( “cameraBehindDistance”,  “10” );
   p.put( “cameraAboveHeight”,     “50” );
   p.put( “cameraHigherHeight”,    “700” );
   p.put( “cameraHighestHeight”,   “2500” );
   p.put( “cameraInFrontDistance”, “10” );
   p.put( “cameraRightDistance”,   “16” );
   p.put( “cameraLeftDistance”,    “16” );
   p.put( “newPartOffsetZ”,        “40” );
   p.put( “newPartOffY0”,          “0” );

              return p;
           }

/** Sets the preference by name. */
           public static void setPreference(String pref,String value) {
              preferences.put(pref,value);
              //savePreferences();
           }

/** Returns the value of a preference by name. */
           public static float getFloatPreference(String x) {
                               return tdxs.stringToFloat( getPreference(x) ); }
/** Returns the value of a preference by name. */
public static int getIntPreference(String x) {
                               return tdxs.stringToInt( getPreference(x) ); }
/** Returns the value of a preference by name.*/
           public static String getPreference(String name) {

              // This could be done once at load time if perfomance matters.
              String val = tdWM.getenv(name);
              if (val.equals(“”)) val = null;
              if (val == null) val = preferences.getProperty(name);
         if (val == null) td.zprint(1,”\nWARNING:No preference “”+name+””\n\n”);

   td.zprint(1,”  getPreference: “+name+”=”+val+”\n”);

              return val;
           }
/** Parses the command line arguments.
```

```
 * java Schmooser <localeName> <archive file (.tds)> <set file (
                                                  .set)> <prefrence file (.pro)>
 * Otherwise, the user is propted for a starting locale.
 */
         static void parseArguments() {
            for (int i =0;i<arguments.length;i++) {
               String a = arguments[i];
               String ext = “”;
               if (a.lastIndexOf(“.”)>0) ext = tdxs.parseStrAfterDot(a);
               td.zprint(1,”  parseArguments=”+a+”\n”);
               td.zprint(1,”  ext=”+ext+”\n”);

     if ( ext.equalsIgnoreCase(“”)) startingLocale = a;
     else if ( ext.equalsIgnoreCase(“tds”)) startingRestore =
                                    archiveDir+ “\\” /*(“\\”.substring(1))*/ + a;
     else if ( ext.equalsIgnoreCase(“set”)) startingRestore =
                                    archiveDir+ “\\” /*(“\\”.substring(1))*/ + a;
     else if ( ext.equalsIgnoreCase(“pro”)) propertiesFile = a;
     else System.out.println(“\nUnknown argument:”+a+”\n”);
            }
         }
static public schmoozer getSchmoozer() { return me; } /** Returns this object.*/
/** Constuctor. */
         public schmoozer(String title)
         {
                     this();
                     setTitle(title);
         }
/** Display the 2d window.*/
         public synchronized void show()
         {
                     move(710, 2);
                     super.show();
         }
/** Main.*/
         static public void main(String args[])
         {
            arguments = args;
                     (new schmoozer()).show();
         }
/** Window messaging.*/
         public void addNotify()
         {
            // Record the size of the window prior to calling parents addNotify.
            Dimension d = getSize();

                     super.addNotify();
                     if (fComponentsAdjusted)
                              return;
                     // Adjust components according to the insets
                     setSize(insets().left + insets().right +
                            d.width, insets().top + insets().bottom + d.height);
                     Component components[] = getComponents();
                     for (int i = 0; i < components.length; i++)
                     {
                              Point p = components[i].getLocation();
                              p.translate(insets().left, insets().top);
                              components[i].setLocation(p);
                     }
                     fComponentsAdjusted = true;
         }
// Used for addNotify check
         boolean fComponentsAdjusted = false;
         class SymWindow extends java.awt.event.WindowAdapter
         {
            public void windowClosing(java.awt.event.WindowEvent event)
                     {
                              Object object = event.getSource();
                              if (object == schmoozer.this)
                                       Schmoozer_WindowClosing();
                     }
         }
         void Schmoozer_WindowClosing()
         {
   /* TAKEN OUT FOR NEW RENDERER
   if (tdUp!=null) tdUp.stop(); // Mabye inside frame??? could still go???
   tdUp = null;

   tdWM.WMremove();
   */
   hide();               // hide the Frame
                     dispose();     // free the system resources
                     System.exit(0); // close the application
         }
```

```
         /** Add menue to frame. */
   public static void addMenu(String menu, com.sun.java.swing.JMenu men) {
            if (menu.equals(“File”)) {
               me.FileMenu.insert(men, me.FileMenu.getItemCount() - 2 );
            } else if (menu.equals(“Edit”)) {
               me.EditMenu.add(men);
            } else if (menu.equals(“Mode”)) {
               me.ModeMenu.add(men);
            } else if (menu.equals(“View”)) {
               me.ViewMenu.add(men);
            } else if (menu.equals(“Avatar”)) {
               me.AvatarMenu.add(men);
            } else if (menu.equals(“Main”)) {
               me.MainMenuBar.add(men);
            }
         }

         /** Adds menue Item to the frame. */
         public static com.sun.java.swing.JMenu addMenuItem(String menu,
                                    com.sun.java.swing.JMenuItem item, int index) {
            if (menu.equals(“File”)) {
               me.FileMenu.insert(item, me.FileMenu.getItemCount() - 2 );
               return me.FileMenu;
            } else if (menu.equals(“Edit”)) {
               if (index == -1) me.EditMenu.add(item);
               else me.EditMenu.insert(item, index );
               return me.EditMenu;
            } else if (menu.equals(“Mode”)) {
               if (index == -1) me.ModeMenu.add(item);
               else me.ModeMenu.insert(item, index );
               return me.ModeMenu;
            } else if (menu.equals(“View”)) {
               if (index == -1)me.ViewMenu.add(item);
               else me.ViewMenu.insert(item, index );
               return me.ViewMenu;
            } else if (menu.equals(“Avatar”)) {
               if (index == -1)me.AvatarMenu.add(item);
               else me.AvatarMenu.insert(item, index );
               return me.AvatarMenu;
            }
            return me.HelpMenu;
         }
         /** Returns avatar menu.
         public static com.sun.java.swing.JMenu getAvatarMenu() {
                                                   return me.AvatarMenu; }
*/
         class SymAction implements java.awt.event.ActionListener
         {
         public void actionPerformed(java.awt.event.ActionEvent event)
                     {
                              Object object = event.getSource();
                              if (object == miLoadTool)
                                       loadTool_Action();

                              else if (object == miLocaleOptions)
                                       miLocaleOptions_Action();
                              else if (object == miPeferences)
                                       miPreference_Action();
                              else if (object == miAbout)
                                       miAbout_Action();
                              else if (object == miExit)
                                       exit_Action();
                              else if (object == miChangeLocale)
                                       changeLocale_Action();
                              else if (object == miRestoreLocale)
                                       restoreLocale_Action();
                              else if (object == miDeleteLocale)
                                       deleteLocale_Action();
                              else if (object == miSaveLocale)
                                       saveLocale_Action();
                        else {
      for (int i=0;i<numEditors;i++)
         if (object == editorMIs[i]) activateEditor(i);
                        }

                     }
         }

/** Hack to get static class in tdAvatarPlatform to listen for events. */
         //public static xSymAction xAction = new xSymAction();
         public static xSymAction getxSymAction() {return xAction;}
         class xSymAction implements java.awt.event.ActionListener {
         public void actionPerformed(java.awt.event.ActionEvent event) {
                        tdAvatarPlatform.actionPerformed(event);
                     }
            }
/** About dialog.*/
            void miAbout_Action()
            {
               tdAboutDialog d = new tdAboutDialog();
               //d.setIconImage(schmoozerIcon);
               d.show();
            }
/** Exit.*/
            public void exit_Action()
            {
                          // Action from Exit Create and show as modal
                          //(new tdQuitDialog(this, true)).show();
                          if (com.sun.java.swing.JOptionPane.YES_OPTION ==
                          com.sun.java.swing.JOptionPane.showConfirmDialog(this,
                             “Exit Schmoozer?”,”Exit Schmoozer”,
                             com.sun.java.swing.JOptionPane.YES_NO_OPTION))

      Schmoozer_WindowClosing();
            }

            public boolean replaceScene_Action()
            {
               if (com.sun.java.swing.JOptionPane.YES_OPTION ==
                          com.sun.java.swing.JOptionPane.showConfirmDialog(this,
                             “Replace Current Scene ?”,”Integrate or Replace ?”,
                             com.sun.java.swing.JOptionPane.YES_NO_OPTION))
                             return true;
                             else return false;
            }
/** Load new .jar file.*/
            public void loadTool_Action()
            {
                          LoadToolDialog.setDirectory(kitDir);
                          LoadToolDialog.show();
   String slash = “\\”.substring(1);  // A single slash boy thats hard!
  String fullpath = LoadToolDialog.getDirectory()+slash+LoadToolDialog.getFile();
    setTitle( “Loading “+fullpath+”...”);
    td.zprint(1,”Loading “+fullpath+”...\n”);
    tdEditor.create(fullpath);
            }
            public void miLocaleOptions_Action()
            {
               spSchmoozerLocale locale = tdAvatarPlatform.getAvatarsLocale();
               if (locale != null) {
      td.zprint(1,”miLocaleOptions_Action\n”);
               tdLocaleOptionsDialog tdp = new tdLocaleOptionsDialog();
               td.zprint(1,”miLocaleOptions_Action set title\n”);
                          tdp.setTitle(locale.getTitle()+” Locale Options”);
                          tdp.setModal(true);
                          tdp.show();  // <— should wait here till dispose.
                          }
            }
            public void miPreference_Action()
            {
    td.zprint(1,”miPreference_Action\n”);
               tdPreferencesDialog tdp = new tdPreferencesDialog();
               tdp.setTitle(“Preferences”);
                          tdp.setModal(true);
                          tdp.show();  // <— should wait here till dispose.
            }
tdVisualListener keyLis;
tdVisualListener closeLis;

static tdLocalesDialog localesDialog =
                          new tdLocalesDialog(“Select a locale to join”,true);
/** Show the Connect dialog. */
            public boolean changeLocale_Action()
            {
                          td.zprint(1,”ChangeLocale_Action\n”);
    //if (localesDialog==null) localesDialog =
                          new tdLocalesDialog(“Select a locale to join”,true);

                          td.zprint(1,”ChangeLocale_Action show starts\n”);
                          localesDialog.reLoadComboBox(true);
                          localesDialog.show(); // <— should wait here till dispose.
    td.zprint(1,”Schmoozer ChangeLocale_Action returns “”
                          + (tdAvatarPlatform.getAvatarsLocale() != null) +””\n”);

                          return tdAvatarPlatform.getAvatarsLocale() != null;

            }
 static tdDeleteLocaleDialog deleteDialog = new tdDeleteLocaleDialog();
/** Show the delete dialog. */
            void deleteLocale_Action(){
```

```
                          //tdDeleteLocaleDialog tdl = new tdDeleteLocaleDialog();
                          //deleteDialog.setTitle(“Select a locale to Delete”);
                                      //deleteDialog.setModal(true);
                                      deleteDialog.reLoadDialog();
                                      deleteDialog.show();      // <— should wait here till
dispose.
                          }
   /** Listen for ^B^T^R^S in visual’s window. */
                          public LisAction lLisAction;
                          class LisAction implements tdVisualListener.visualListener
                          {
                                      public void eventAction(tdVisualEvent event) {
         if (event.getSource() == keyLis)  {
           td.zprint(4,”Schmoozer: keyLis event\n”);
           // Detect window close or esc.
           if (event.getValue()==tdVisualListener.KEY_ESC) {
             td.zprint(1,”Schmoozer: ESCAPE EXIT.\n”);
             Schmoozer_WindowClosing();
           } else {
             td.zprint(3,”Schmoozer: key listener\n”);
             activateEditorByCosmoShortcut(event.getValue()); // Pass every key
           }
         } else if (event.getSource() == keyLis)  {
           td.zprint(1,”Schmoozer: EVENT_WINDOW_CLOSE EXIT.\n”);
           Schmoozer_WindowClosing();
         }
                                      }
                          }
   public static void validLocale(boolean valid){
                                      miLocaleOptions.setEnabled(valid);
                                      miSaveLocale.setEnabled(valid);
                                      miDeleteLocale.setEnabled(valid);
   }
                          spSchmoozerLocale currentLocale;
/** Detects locale changes for closing editors */
                          int      currentEditor=-1;  /** Detects changes in the editing mode*/
   /** Sets the Browser title.*/
   public  void setBrowserTitle(String s) { setTitle(s); currentLocale = null; }
   public  void setBrowserTitle(){

       if (restoring)
          setTitle(“Restoring...”);
       else if (starting)
          setTitle(“Starting...”);
       else {
          int editor = getActiveEditorIndex();
          spSchmoozerLocale locale = tdAvatarPlatform.getAvatarsLocale();
          td.zprint(3,”setBrowserTitle()\n”);

          if ((currentLocale != locale)||(currentEditor != editor)) {
             if ( currentLocale != locale ) {
                activateEditor(editor = 0); // Recurseive yet its ok.
                currentLocale = locale;
                validLocale(locale != null);
             }
             String title = (locale == null) ? “ No Active Connection” :
           ((locale.getLocallyOwned() ? “ Serving “ : “ Browsing “) +  locale.getTitle());
             String mode = editorNames[ (currentEditor=editor) ];
             setTitle(title +” - “+ mode);
          }
          td.zprint(3,”setBrowserTitle done\n”);
       }
                          }
//////////////////////////////////////////////////////////////////////////////////////////
///                                                                               ///
///                              EDITOR SUPPORT                                   ///
///                                                                               ///
//////////////////////////////////////////////////////////////////////////////////////////

   static final int  MAX_EDITORS = 10; /** size of editor info.*/
   static String[]   editorNames = new String[MAX_EDITORS]; /** Name of editor .*/
   static int[]editorShortcuts = new int [MAX_EDITORS] ; /** Java shortcut.*/
   static int[]editorCosmocuts = new int [MAX_EDITORS] ; /** Cosmo shortcut.*/

   static tdEditor[] editors = new tdEditor[MAX_EDITORS]; /** Each editor.*/
   static com.sun.java.swing.JRadioButtonMenuItem [] editorMIs =
                 new com.sun.java.swing.JRadioButtonMenuItem[MAX_EDITORS];
   static com.sun.java.swing.ButtonGroup editorGroup
                                            = new com.sun.java.swing.ButtonGroup();
   static int        numEditors = 0;
   static int        activeEditor = 0;
   static int        lastEditorUsed = 4;

   public static tdEditor   getActiveEditor() { return editors[activeEditor]; }
   public static tdEditor   getEditorByIndex(int i) { return editors[i]; }
public static String     getEditorNameByIndex(int i) { return editorNames[i]; }
public static int        getLastEditorUsed() { return lastEditorUsed; }
public static int        getActiveEditorIndex() { return activeEditor; }
public static boolean    getNavigating() { return activeEditor == 0; }

            /** Registers a new editor
             *
  * @author derek
  * @param self          tdEditor to register.
  * @param name          Name of this editor.
  * @param cosmocut      Shortcut key for cosmo.
  * @param shortcut      Shortcut key for java window.
  * @param icon          Icon file.
  * @param mi            menue Item if already made.
             */
public static int registerEditor(
            tdEditor self,
            String name,
            char mnemonic,
            int cosmocut,
            com.sun.java.swing.KeyStroke keystroke,
            String icon,
            com.sun.java.swing.JRadioButtonMenuItem mi) {
   // isNav not used: index 0 is the default nav editor. override is TBD.
   td.zprint(3,”registerEditor “+name+”\n”);
   if (mi==null) {
      mi = new com.sun.java.swing.JRadioButtonMenuItem(name);
      mi.setAccelerator(keystroke);
                        mi.setMnemonic(mnemonic);
                     mi.setFont( regFont );
                     mi.addActionListener(lSymAction);
      addIcon(mi,icon);
                     addMenuItem(“Mode”,mi,-1);
   }
   editors[numEditors] = self;
   editorMIs[numEditors] = mi;
   editorNames[numEditors] = name;
   editorShortcuts[numEditors] =
            (keystroke.getKeyChar() > 0) ? keystroke.getKeyChar() : 9999;
   editorCosmocuts[numEditors] = (cosmocut > 0) ? cosmocut : 9999;

   mi.setFont((numEditors == 0) ? boldFont : regFont );  // First pass initialization,
   mi.setSelected((numEditors == 0));
   editorGroup.add(mi);
   return numEditors++;
}
/** Returns a editors index by name
  * @param name          Editor name.
             */
public static int getEditorIndexByName(String name) {
   for (int i=0;i<numEditors;i++) if (name.equals(editorNames[i])) return i;
   return -1;
}
/** Activate an editor by index.*/
public static void activateEditor(int n) {
   td.zprint(3,”activateEditor(“+n+”)\n”);
   if (tdAvatarPlatform.getAvatarsLocale() == null) {n=0;activeEditor=1;}
   if ( activeEditor != 0 ) lastEditorUsed = activeEditor;
   if (activeEditor != n) { // It changed
      activeEditor = n;
      td.zprint(2,”activateEditor(“+n+”) it was “+editors[n].on+”\n”);
      for (int i=0;i<numEditors;i++) editors[i].enable(n==i);
   for (int i=0;i<numEditors;i++) editorMIs[i].setFont( (n==i) ? boldFont : regFont
);
      editorMIs[n].setSelected(true);
      getSchmoozer().setBrowserTitle(); // with this editors name
   }
}
/** Activate an editor by cosmos shortcut.*/
public static void activateEditorByCosmoShortcut(int n) { // n could be any key.
   td.zprint(4,”activateEditorByCosmoShortcut(“+n+”)\n”);
   for (int i=0;i<numEditors;i++)
      if (editorCosmocuts[i] == n) { activateEditor(i); break;}
}
            /** Restore layout dialog.*/
            void restoreLocale_Action()
            {
                     if (restoring)
     setBrowserTitle(“Restoring...”);
                     else if (starting)
     setBrowserTitle(“Starting...”);
                     else {
     activateEditor(0);
                     if (npd == null) npd = new tdNewPartDialog();
                     npd.display(npd.PICK_ARCHIVES); // Not modal....
```

```
                        else {
                           td.zprint(2,”saveLocale_Action\n”);
         // TEMP
         // activateEditor(0);
         spSchmoozerLocale locale = tdAvatarPlatform.getAvatarsLocale();

         spSchmoozerVD.purgeUnusedVDs();//Anything we own thats not used
         String title = locale.getTitle();
         //title = tdWM.stripDNS(title);
   // These lines dont work:
   //      SaveLayoutDialog = new java.awt.FileDialog(this,”Save “”
                                    +locale+”” Layout”,FileDialog.SAVE);
   //                   SaveLayoutDialog.setMode(FileDialog.SAVE);
   //                   SaveLayoutDialog.setTitle(“Save “”+locale+”” Layout”);
                        SaveLayoutDialog.setFile(title+”.tds”);
                        //SaveLayoutDialog.setDirectory(archiveDir);
         td.zprint(2,”saveLocale_Action archiveDir=”+archiveDir+”\n”);
         td.zprint(2,”saveLocale_Action title=”+title+”\n”);
             new tdDialogThread(SaveLayoutDialog,SAVE_MODE); // Start thread
/*
                        SaveLayoutDialog.show();

         String slash = “\\”.substring(1);  // A single slash boy thats hard!
               String fullpath = SaveLayoutDialog.getDirectory()+slash
                                    +SaveLayoutDialog.getFile();
               String filename = SaveLayoutDialog.getFile();

               if (SaveLayoutDialog.getFile() != null)
            tdWM.serSaveLocale(tdAvatarPlatform.getAvatarsLocale().sp, fullpath);
*/
               }
             }
   /** The new part dialog is recycled. */
   private tdNewPartDialog npd=null;
   /** new part dialog.*/
             public void newPartDialog(){
                        if (restoring)
         setBrowserTitle(“Restoring...”);
                        else if (starting)
         setBrowserTitle(“Starting...”);
                        else {
                        if (npd == null) npd = new tdNewPartDialog();
                        npd.display(npd.PICK_PARTS); // Not modal....
               }
             }
             public static final int
                SAVE_MODE = 1;
                //RESTORE_MODE = 2;

   /** Dialog non-blocking threads */
   public void dialog_Action(java.awt.FileDialog dialog, int mode){
      String slash = “\\”.substring(1);  // A single slash boy thats hard!
               String fullpath = dialog.getDirectory()+slash+dialog.getFile();
               String filename = dialog.getFile();

             if (filename != null) {
         if (mode == SAVE_MODE) {
            tdWM.serSaveLocale(tdAvatarPlatform.getAvatarsLocale().sp, fullpath);
             //} else if (mode == RESTORE_MODE) {
             //   restoreAction(filename, fullpath);
             }
      }
   }
/////////////////////////////////////////////////////////////////////////////////////
///                                                                              ///
///                            MONITOR SUPPORT                                   ///
///                                                                              ///
/////////////////////////////////////////////////////////////////////////////////////
   transient static private Hashtable monitors  = new Hashtable();
   public void onLandingPad(Integer cbid, Integer spid)
                        { td.zprint(1,”onLandingPad\n”); } /** Dummy routine*/
   /** Make a monitor for layouts at this URL if it does not already exist.*/
   public static void checkLandingPadMonitor(String url) {
      String dns = tdxs.parseDNSFromURL(url);
      if (dns == null)
         dns = “istp://”+url;
      String monStr = dns+”/LandingPad*”;
      td.zprint(1,”checkLandingPadMonitor  url=””+url+”” dns=””+monStr+”” \n”);
      if (monitors.get(monStr) == null) {
         monitors.put(monStr, new tdBeaconMonitor(me,monStr,”onLandingPad”,-
1));
      }
   }
   /** make monitors as specifed in the prefrences.*/
```

```
  public static void makeMonitors() {
    td.zprint(1,"makeMonitors\n");

    // Dispatcher....     SHOULD BE /LandingPad*...
    checkLandingPadMonitor("istp://"+getPreference("Dispatcher")
              +"/LandingPad*"); //has ISTP: lower case  OLD To be deleted
     checkLandingPadMonitor("$DISPATCHER/LandingPad*");
                        // No ISTP: at all. OLD To be deleted
    checkLandingPadMonitor("istp://"+getPreference("beaconServer")
                        +"/LandingPad*"); //has ISTP: lower case
    checkLandingPadMonitor("$beaconServer/LandingPad*"); // No ISTP: at all.
    // Other Monitor=xx,xx,xx,
    String mons = getPreference("monitor");
    if (mons!=null) {
      String a[] = tdxs.parseTokens(mons,",",true);
      for (int i=0; a[i] != null; i++)
        checkLandingPadMonitor(a[i]);
    }
  }
          //{{DECLARE_CONTROLS
          //}}
          //{{DECLARE_MENUS
          //}}
  // COULD BE DELETED———v
/*
          class SymMouse extends java.awt.event.MouseAdapter
          {
          public void mouseClicked(java.awt.event.MouseEvent event)
                    {
                              Object object = event.getSource();
                              if (object == schmoozer.this)
                              Schmoozer_mouseClicked(event);
                    }
          }
          void Schmoozer_mouseClicked(java.awt.event.MouseEvent event)
          {
                    // to do: code goes here.
          }
*/
///////////////////////////////////////////////////////////////////////////////
///                         WMUPDATE                                        ///
///                                                                         ///
///////////////////////////////////////////////////////////////////////////////
  /** Events are checked every wmUpdate. */
  private static boolean running = true;
  /** Enable or disable the WMUpdate cycle.
                    (this call not syncronized with anything) */
  static public  void    blockWMUpdate(boolean stop) {running = !stop;}

  /** Events are checked every wmUpdate. */
  private static int eventArray[] = new int[tdVisualListener.
                    SIZE_OF_EVENT*tdVisualListener.MAX_EVENTS+2];
  /** Run WMupdate, and check the event array. Returns 0 if not exiting. */
  static public int WMupdate() {
    if (running) {

      int r = tdWM.WMupdate(eventArray);
      tdVisualListener.runEvents(eventArray);
      return r;
    } else
      return 0;
  }
/// STATIC FUNCTIONS COULD BE MOVED:
///////////////////////////////////////////////////////////////////////////////
///                                                                         ///
///                       FILE SUPPORT                                      ///
///                                                                         ///
///////////////////////////////////////////////////////////////////////////////
  //static boolean isPart(String s) { return s.endsWith(".idf"); }
  static boolean isURL(String s) {  return    s.startsWith("ftp:")||
                              s.startsWith("http:")||
                              s.startsWith("FTP:")||
                              s.startsWith("HTTP:"); }
          public static DataInputStream openFile(String fullpath)
                                        throws IOException {

    if (isURL(fullpath))
      return new DataInputStream( (new URL( fullpath )).openStream() );

    if (fullpath.startsWith("file:\\\\\\")) // "file:///s:/demo/xxx"
      try {fullpath = fullpath.substring(8);}
                        catch (StringIndexOutOfBoundsException ex) {}
    if (fullpath.startsWith("file:"))
      try {fullpath = fullpath.substring(5);}
catch (StringIndexOutOfBoundsException ex) {}
```

```
      return new DataInputStream( new FileInputStream(fullpath) );

  }

///////////////////////////////////////////////////////////////////////////////
///                                                                         ///
///                     NEW PART SUPPORT                                    ///
///                                                                         ///
///////////////////////////////////////////////////////////////////////////////
/** Creates a new part. Makes all children. Determines the correct editor to invoke.
          *
   * @author derek
   * @param fullpath        Path name to .idf file.
   * @param extraParameters   Extra parameters (usefull for unique children).
   * @param cPtr            C object if made.
          */
         public static spThing newSpThingFromIdf(String fullpath,
                                        String extraParameters, int cPtr){

           if (fullpath==null) { td.zprint(1,"newPartFromIdf(null)\n"); return null;}

           boolean attempted = false;
           spThing newSpThing = null;
           td.zprint(1," newPartFromIdf fullpath="+fullpath+"\n");

    // This is removed in favor of getOriginalAppearanceStr();
    // Strip the state off the fullpath, (which you could get if you clone a vd w
    // Making this alternate VD could be bad.
    // State is represented by "__xxxx".idf in the fullpath.
    // fullpath = tdxs.stripState(fullpath);
           int linenum=0;
    try {
      DataInputStream dis = openFile(fullpath); //new DataInputStream(
                                        new FileInputStream(fullpath) );
      String line;
      while (null != (line = dis.readLine())) {
        td.zprint(2,"ReadLine:"+line+"\n");
        linenum++;
        line.trim();
        if (!attempted){
          if (line.startsWith("#Type=")) {
            String str = line.substring(6);
            td.zprint(1,"Typeline:"+line+"\n");
            td.zprint(1,"Strline:"+str+"\n");
            String cls[] = tdxs.parseTokens(str,"(");
            String arg[] = tdxs.parseTokens(cls[1],")");
            String className = cls[0];
            String parameters = arg[0] + ((arg[0]==null||arg[0].length()<1)?"":";")
                                              + extraParameters;
            td.zprint(1," newPartFromIdf CLASS="+className+" PARAM="
                                              +parameters+"\n");
            newSpThing = newSpThingFromClassName(className,
                                        parameters,fullpath,cPtr);
            attempted = true;
            // break; // Done, well, keep looking for children
          }
      } else if (newSpThing != null && cPtr == 0) {
                        // made newSpThing for the first time(cPtr==0)
        if (line.startsWith("#Child=")) {
          String str = line.substring(7);
          String cls[] = tdxs.parseTokens(str,"(");
          String arg[] = tdxs.parseTokens(cls[1],")");
          String newIdf = tdxs.parsePathFromPathName(fullpath) + cls[0];
          String parameters = arg[0] + ((arg[0]==null||arg[0].length()<1)?"":";")
                                              + extraParameters;
          sp newChildSp = newSpThingFromIdf(newIdf,parameters,cPtr);
          // The newChildPart has already been donated..
          spOMI.makeStaticCall("onSetParentToSp0", "tdEditor",
                                        newChildSp, newSpThing);
        }
      }
    }
    dis.close();
  }
  catch (IOException e) { System.out.print("Error reading .idf file '"+fullpath+"':"
                                              +linenum+"\n"); }
           if (!attempted) {
    td.zprint(1,"newPartFromIdf .idf had no 'Type=' assuming its spScenery "
                                              +fullpath+"\n");
  newSpThing = newSpThingFromClassName("spScenery",extraParameters,
                                                    fullpath,cPtr);
           }
           return newSpThing;
         }
```

```
                /** Creates a new part based on the class name,
                 *
     * @author derek
     * @param className         Name of the class to be made.
     * @param parameterString   Exta parameters plus IDF parameters.
     * @param idfPath           Path name to .idf file.
     * @param cPtr              C object if made.
                 */
                static spThing newSpThingFromClassName(String className,
                                        String parameterString, String idfPath, int cPtr) {
        td.zprint(1,"newPartFromClassName("+className+",,) \n");
        if (cPtr != 0) { // Already made by C, just make Java copy
  return
tdEditor.makeInstanceFromClass(className,parameterString,idfPath,cPtr);
        } else { // Ask an Editor to place a new part
           for (int i=0;i<numEditors;i++) {
              td.zprint(2,"newPartFromClassName("+className+") test "+i+"\n");
              if (editors[i].canMakeInstance(className,parameterString,idfPath)) {
 return (editors[i].tryToMakeInstance(className,parameterString,idfPath,cPtr));
              }
           }
        }

        return null;
                }

   /** Creates a run time equivelent of a C part. This is called only from JNI.*/
  public static int newRunTimePart(int id) { // id is some type of spPart not in Java
        td.zprint(1," newRunTimePart()\n");
        // This part,class and id may not exist in Java yet. Only CPTRs may be used.
        int vdp = tdWM.getPartOriginalVisualDefinition(id);
        if (vdp==0) vdp = tdWM.getThingVisualDefinition(id);
        String link = (vdp == 0) ? "" : tdWM.getLinkURL(vdp);
        //link = tdxs.getOriginalIdfPath(link,id);


        td.zprint(1," newRunTimePart idf="+link+"\n");
        // Do I have this class? If not load it, and make the part
        newSpThingFromIdf(link,null,id);

        return 1;
   }
```

```
/*
```

public class tdViewEditor extends tdEditor

```
** $Id: tdViewEditor.java,v 1.18 1998/10/07 15:16:38 derek Exp $
** Copyright © 1998 MITSUBISHI ELECTRIC ITA.  ALL RIGHTS RESERVED
 haymaker altered
**
** 05/18/98 DLS Comments
**
*/
import java.awt.*;
import java.util.*;
import com.sun.java.swing.*;
/**  This editor implements a mouse view editor.
Drag Functions:
mouse   right:    Y=Translate up/down        X=Translate left/right
control right:    Y=Tilt up/down around X     X=Translate left/right??
mouse   left:     Y=Translate forward/Backward  X=Rotate left/right around Y
Shift key always aligns to grid.
*/
public class tdViewEditor extends tdEditor {
   static tdViewEditor me = null; // Set once.
   public static tdViewEditor get() {return me;}
   public spMarker getMarker() { zprint(4,"tdViewEditorget: has no marker\n")
                                                   ; return null;}
   static int thisEditorsIndex;
   public tdViewEditor(){
      ViewInit();
                        //{{INIT_CONTROLS
                        //}}
          }
   private static final int   VIEWS = 20;
   private static final int   CAMERAPOSITIONS = 9;
   private static final int   DEFAULTPOSITION = CAMERAPOSITIONS-1;
   private static final float PERCENT_GOTO = .85f;
   private static final float MAXSPEED = 128;
   private static final float MINSPEED = .001f;
   private static final float DEFAULT_TRAN_SCALE = 20;
   private static final float DEFAULT_ROT_SCALE = 1.8f;
   private static final float DEF_ANGLE = -0.25f;
   private static tdPoint fixedcampos []= {
      new tdPoint( -35,50,35,   0,1,0,    -0.785f), //From SouthWest
      new tdPoint( 50,50,70,    0,1,0,    0), //From South
      new tdPoint( 135,50,35,   0,1,0,    0.785f), //From SouthEast
      new tdPoint( 170,50,-50,  0,1,0,    3.14f),
                        //From East  CHANGED TO LOOK AWAY  was 1.57f
      new tdPoint( 135,50,-135, 0,1,0,    2.355f), //From NorthEast
      new tdPoint( 50,50,-170,  0,1,0,    3.14f), //From North
      new tdPoint( -35,50,-135, 0,1,0,    -2.355f), //From NorthWest
      new tdPoint( -70,50,-50,  0,1,0,   -1.57f), //From West
      new tdPoint( 50,150,-50,  0,1,0,    0), //From Top
   };
   public static tdPoint getFixedViewPoint(int n)
   {
      return fixedcampos[n];
   }
   private tdVisualListener secLis, selectLis, keyLis, dragLis, upLis,
                                                   gotoLis, frameLis, modeLis;
   private tdPoint deltaXaxis = new tdPoint(tdPoint.axisX);
   private tdPoint deltaYaxis = new tdPoint(tdPoint.axisY);
   private tdPoint tmp = new tdPoint();
   private tdPoint tar = new tdPoint();
   private tdPoint ntar = new tdPoint();
   private int dragStartX;
   private int dragStartY;
//   private float dx,dy,dz,dr,tr;
   private float max_x=500, max_y=500;
   private float tranScale = DEFAULT_TRAN_SCALE;
   private float rotScale = DEFAULT_ROT_SCALE;
   private float KeySpeed = 1;
   private boolean inDrag;
   private static int VPs = 0;
          private static spViewPoint VP[] = new spViewPoint[VIEWS];
          private static JMenuItem miStats;
          private static JMenuItem miView[] = new JMenuItem[VIEWS];
          private static JMenu rideMenu;
          private static JMenu viewMenu;
          private static JMenuItem [] miSpeed = new JMenuItem[4];
          private static JMenu SpeedMenu;

          private static JRadioButtonMenuItem [] miCamera =
                   new JRadioButtonMenuItem[CAMERAPOSITIONS];
   private static ButtonGroup CameraGroup = new ButtonGroup();
          private static JMenu CameraMenu;
```

```
          //HAYMAKER
          private static JButton [] miViews = new JButton[CAMERAPOSITIONS];
          private static JPanel ViewPanel;
          private static ButtonGroup ViewGroup = new ButtonGroup();

private static tdPredicateCallback ViewCB;
          private LisAction lLisAction = new LisAction();
          private SymAction lSymAction = new SymAction();
private java.awt.Font regFont, boldFont;
private boolean shift;
private int cameraPosition = DEFAULTPOSITION;
//private float[] tiltTransDown = {0 ,0, 0, 1, 0, 0, -1.57f, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0};
//private float[] tiltTransAngle = {0 ,0, 0, 1, 0, 0, -.0785f, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0};
/** Initialize this editor.*/
public void ViewInit() {
   if (me != null) return;
   me = this;
                    on = true; //Assumes its active
                    zprint(3,"NavInit setting up callbacks\n");
          selectLis = new tdVisualListener(lLisAction,tdVisualListener.
                    EVENT_MOUSE_DOWN,on); // onMouseDown
   upLis     = new tdVisualListener(lLisAction,tdVisualListener.
                    EVENT_MOUSE_UP,on); // onMouseUp
   dragLis   = new tdVisualListener(lLisAction,tdVisualListener.
                    EVENT_MOUSE_DRAG,on); // onNavDrag
   frameLis  = new tdVisualListener(lLisAction,tdVisualListener.
                    EVENT_EACH_FRAME,false); // onNavFrame
   gotoLis   = new tdVisualListener(lLisAction,tdVisualListener.
                    EVENT_MOUSE_LEFT_DOUBLE,on); // onNavGoto
   // THESE ARE GLOBAL Never disabled...
   modeLis   = new tdVisualListener(lLisAction,tdVisualListener.
                    EVENT_MOUSE_BOTH,true); // onNavModeChange
   keyLis    = new tdVisualListener(lLisAction,tdVisualListener.
                    EVENT_KEY_PRESS|tdVisualListener.
                    EVENT_KEY_RELEASE,true); // onNavKey

   secLis    = new tdVisualListener(lLisAction,tdVisualListener.
                    EVENT_EACH_SECOND,true); // onNavSec

   regFont = schmoozer.getRegFont();
   boldFont = schmoozer.getBoldFont();
                    zprint(3,"NavInit regiserting self\n");
                    // Register Myself, and add MenueItem
          KeyStroke jkey = KeyStroke.getKeyStroke(java.
                awt.event.KeyEvent.VK_B,java.awt.Event.CTRL_MASK,false);
          thisEditorsIndex = schmoozer.registerEditor(me,"Browse mode",
          'B',tdVisualListener.KEY_CTRL_B,jkey,"images/browseMode.gif",null);
                    zprint(3,"NavInit making menuItems \n");
   // Speed Menue Items
                    SpeedMenu = new JMenu("Speed");
                    SpeedMenu.setMnemonic('p');
                    SpeedMenu.setFont(regFont);
          SpeedMenu.setToolTipText("Change the avatars navigation speed.");


                    miSpeed[0] = new JMenuItem("Faster",'F');
                    miSpeed[1] = new JMenuItem("Slower",'S');
                    miSpeed[2] = new JMenuItem("Default",'D');

          schmoozer.addIcon(SpeedMenu,"images/speed.gif");
          schmoozer.addIcon(miSpeed[0],"images/speedFaster.gif");
          schmoozer.addIcon(miSpeed[1],"images/speedSlower.gif");
          schmoozer.addIcon(miSpeed[2],"images/speedModerate.gif");


                    for (int i=0; i<3; i++) {
                       if (i==3-1) SpeedMenu.addSeparator();
                       SpeedMenu.add(miSpeed[i]);
                       miSpeed[i].addActionListener(lSymAction);

                       miSpeed[i].setFont( regFont );
                    }
                    miSpeed[2].setFont( boldFont ); // Default
                    schmoozer.addMenu("Avatar",SpeedMenu);


                    // Camera Menue Items
          CameraMenu = new JMenu("Camera Position");
          CameraMenu.setFont( regFont );
          CameraMenu.setMnemonic('C');
          CameraMenu.setToolTipText("Move the camera relative to the
avatar.");
```

```
                    // HAYMAKER Panel for view buttond
                    ViewPanel = new JPanel();
                    ViewPanel.setLocation(110,5);
                    ViewPanel.setSize(100, 100);
                    ViewPanel.setVisible(true);
                    ViewPanel.setBackground(Color.black);
                    ViewPanel.setLayout(null);
                    ViewPanel.show();
            schmoozer.getSchmoozer().getContentPane().add(ViewPanel);

                    miCamera[0] = new
JRadioButtonMenuItem(“SouthWest”);
                    miCamera[1] = new JRadioButtonMenuItem(“South”);
                    miCamera[2] = new JRadioButtonMenuItem(“Southeast”);
                    miCamera[3] = new JRadioButtonMenuItem(“East”);
                    miCamera[4] = new JRadioButtonMenuItem(“NorthEast”);
                    miCamera[5] = new JRadioButtonMenuItem(“North”);
                    miCamera[6] = new JRadioButtonMenuItem(“NorthWest”);

                    miCamera[7] = new JRadioButtonMenuItem(“West”);
                    miCamera[8] = new JRadioButtonMenuItem(“Top”);
            miViews[0] = new JButton(“SW”);
                    miViews[0].setSize(30,30);
                    miViews[0].setLocation(5,65);
    miViews[0].setMargin(new Insets(0,0,0,0));
                    miViews[1] = new JButton(“S”);
                    miViews[1].setSize(30,30);
                    miViews[1].setLocation(35,65);
                    miViews[1].setMargin(new Insets(0,0,0,0));
                    miViews[2] = new JButton(“SE”);
                    miViews[2].setSize(30,30);
                    miViews[2].setLocation(65,65);
                    miViews[2].setMargin(new Insets(0,0,0,0));
                    miViews[3] = new JButton(“=>”);
                    miViews[3].setSize(30,30);
                    miViews[3].setLocation(65,35);
                    miViews[3].setMargin(new Insets(0,0,0,0));
                    miViews[4] = new JButton(“NE”);
                    miViews[4].setSize(30,30);
                    miViews[4].setLocation(65,5);
                    miViews[4].setMargin(new Insets(0,0,0,0));
                    miViews[5] = new JButton(“N”);
                    miViews[5].setSize(30,30);
                    miViews[5].setLocation(35,5);
                    miViews[5].setMargin(new Insets(0,0,0,0));
                    miViews[6] = new JButton(“NW”);
                    miViews[6].setSize(30,30);
                    miViews[6].setLocation(5,5);
                    miViews[6].setMargin(new Insets(0,0,0,0));
                    miViews[7] = new JButton(“W”);
                    miViews[7].setSize(30,30);
                    miViews[7].setLocation(5,35);
                    miViews[7].setMargin(new Insets(0,0,0,0));
                    miViews[8] = new JButton(“T”);
                    miViews[8].setSize(30,30);
                    miViews[8].setLocation(35,35);
                    miViews[8].setMargin(new Insets(0,0,0,0));
                    for (int i=0; i < 9; i++)
                    {
                    miViews[i].addActionListener(lSymAction);
                    }
                    miCamera[0].setMnemonic(‘Z’);
                    miCamera[1].setMnemonic(‘X’);
                    miCamera[2].setMnemonic(‘C’);
                    miCamera[3].setMnemonic(‘D’);
                    miCamera[4].setMnemonic(‘E’);
                    miCamera[5].setMnemonic(‘W’);
                    miCamera[6].setMnemonic(‘Q’);
                    miCamera[7].setMnemonic(‘A’);
                    miCamera[8].setMnemonic(‘S’);
                    /*HAYMAKER
            schmoozer.addIcon(CameraMenu, “images/camView.gif”);
            schmoozer.addIcon(miCamera[0],”images/camSouthEast.gif”);
            schmoozer.addIcon(miCamera[1],”images/camSouth.gif”);
            schmoozer.addIcon(miCamera[2],”images/camSouthWest.gif”);
            schmoozer.addIcon(miCamera[3],”images/camWest.gif”);
            schmoozer.addIcon(miCamera[4],”images/camNorthWest.gif”);
            schmoozer.addIcon(miCamera[5],”images/camNorth.gif”);
            schmoozer.addIcon(miCamera[6],”images/camNorthEast.gif”);
            schmoozer.addIcon(miCamera[7],”images/camEast.gif”);
            schmoozer.addIcon(miCamera[8],”images/camTop.gif”);
            */


            for (int i=0; i<CAMERAPOSITIONS; i++) {
```

```
        if (i == (CAMERAPOSITIONS-1)) CameraMenu.addSeparator();
                    CameraMenu.add(miCamera[i]);
                    ViewPanel.add(miViews[i]);
                    CameraGroup.add(miCamera[i]);
                    ViewGroup.add(miViews[i]);
                    miCamera[i].setFont( regFont );
                    miCamera[i].addActionListener(lSymAction);
                }

                miCamera[DEFAULTPOSITION].setFont( boldFont );

                miCamera[DEFAULTPOSITION].setSelected( true );


                schmoozer.addMenu(“View”,CameraMenu);
                zprint(3,”ViewInit setting up Dynamic View Points\n”);

                // Dynamic View Points
    ViewCB = new tdPredicateCallback(this,null,”AnyChange”,”spViewPoint”,
                “onViewPointChanged”); // should be newOrChanged

        rideMenu = new JMenu(“Ride”);
        viewMenu = new JMenu(“Goto Position”);
                rideMenu.setMnemonic(‘R’);
                viewMenu.setMnemonic(‘G’);

        schmoozer.addMenu(“View”,rideMenu);
        schmoozer.addMenu(“View”,viewMenu);
        onViewPointChanged(null,null); // display “reset” options.
        rideMenu.setFont( regFont );
        viewMenu.setFont( regFont );
                rideMenu.setToolTipText(“Make the Avatar Ride on this.”);
        viewMenu.setToolTipText(“Move the avatar to this view point.”);

                schmoozer.addIcon(rideMenu,”images/ride.gif”);
                schmoozer.addIcon(viewMenu,”images/gotoPosition.gif”);

    //setCameraFlyDistances(); // read the user defined properties
        // Stats
                miStats = new JMenuItem(“Statistics”);
                miStats.setMnemonic(‘S’);
                miStats.setFont( regFont );
                miStats.setToolTipText(“Display statistics”);
                schmoozer.addMenuItem(“View”,miStats,-1);
                miStats.addActionListener(lSymAction);
                schmoozer.addIcon(miStats,”images/stats.gif”);

    on = true;
  }

  /** Sets the camera fly distances based on user preferences. */
  public static void setCameraFlyDistances(){
    /*
        float south  = schmoozer.getFloatPreference(“camSouthDistance”);

 float southeast  = schmoozer.getFloatPreference(“camSouthEastDistance”);

 float east  = schmoozer.getFloatPreference(“camEastDistance”);
 float northeast = schmoozer.getFloatPreference(“camNorthEastDistance”);

 float north = schmoozer.getFloatPreference(“camNorthDistance”);
 float northwest  = schmoozer.getFloatPreference(“camNorthWestDistance”);

 float west   = schmoozer.getFloatPreference(“camWestDistance”);
 float southwest  = schmoozer.getFloatPreference(“camSouthWestDistance”);
 float top  = schmoozer.getFloatPreference(“camTopDistance”);

//HAYMAKER NOTE, THESE NEED TO BE CALCULATED
  if (southwest != 0)   {fixedcampos[0].pt[1] = 0.37f*south; fixedcampos[0].pt[2] =
                                                              0.92f*south;}
     if (south != 0)    {fixedcampos[1].pt[1] = southeast;}
     if (southeast != 0)    {fixedcampos[2].pt[1] = east;}
     if (east != 0)   {fixedcampos[3].pt[1] = northeast;}
     if (northeast != 0)   {fixedcampos[4].pt[1] = -0.1f*north; fixedcampos[4].pt[2] =
                                                              -0.99f*north;}
  if (north != 0)    {fixedcampos[5].pt[0] =  0.99f*northwest;  fixedcampos[5].pt[1] =
                                                              -0.006f*northwest;}
 if (northwest != 0)     {fixedcampos[6].pt[0] = -0.99f*west;   fixedcampos[6].pt[1] =
                                                              -0.006f*west;}
     if (west != 0)     {fixedcampos[6].pt[0] = southwest;}
     if (top != 0)     {fixedcampos[6].pt[0] = top;}
     */
  }
  /** Activate a View Points.
   * @param i   View point to display.
 */
public void viewPoint_Action(int i){
   td.zprint(3,”viewPoint_Action “+i+”\n”);
   spViewPoint vp = VP[i];
   tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();
   if (vp != null) {
      String title = vp.getTitle();
      String state = vp.getState();
      boolean camOnly = ( title.startsWith(“cam”) || title.startsWith(“Cam”) ||
                                                            vp.getCamera() );
      boolean parent = state.equalsIgnoreCase(“ride”) || vp.getRide() ;

      tdPoint pnt = parent ? null : tdx.spThingGetLocaleTransform(vp);
      if (camOnly)    platform.cameraGoto(pnt,parent ? vp
                                                   : null          ,3000,false);
      else            platform.avatarGoto(pnt,parent ? vp :
                                                  (sp)vp.getLocale(),6000,false);

      td.zprint(1,”\n\nVVVVVVVV viewPoint_Action  camOnly=”+camOnly
                                                  +”  parent=”+parent+”\n\n”);
                  highlightView(i); // Highlight the selection
   } else { // RESET (has no VP) End flight,ride restore Transforms
      boolean keepPositon = (i == VPs - 1); // ride reset
      flyCamera(false);
      highlightView(-1);
      platform.resetAvatarsTransforms(keepPositon,3000);
   }
}
/** Make the active view bold.
  * @param n  view to highlight.
  */
private void highlightView(int n){
   zprint(1,”highlightView “+n+”  VPs=”+VPs+”.\n”);
               for (int i=0; i<VPs-2; i++)
                  if (miView[i]!=null)
                     miView[i].setFont((i == n)? boldFont : regFont);


}

// if the thing moving around is the viewpoint itself,
//this code could execute every frame, to be fixed...
// if the viewpoint moves every frame, this code could be called very often,
// and we know setting/unsetting menue items is a lot of work. Plus there could
// be many view points in the WM and they all could be moving. So we limit
// this work to once per frame.


/** True if any view point has changed.*/
boolean viewPointChanged;
/** Callback that detects changed viewpoints.*/
public void onViewPointChanged(Integer cbid, Integer spid) {
   viewPointChanged = true;
}

/** The current locale.*/
spSchmoozerLocale locale;

/** Check avatars and viewpoints for changes and adjust UI.*/
public void onNavSec() {
   zprint(4,”onNavSec\n”);


   // check the avatar’s locale (in case we lost it).
   // It’s also checked after each localize i.e. every move
   tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();

   platform.checkAvatarsLocale();
           // THIS LINE SOMETIMES CAUSES LOCK UP...
           // PERHAPS AT SCHMOOZER SET BROWSER TITLE????

   // This would be better if it was in schmoozer.java:
   tdxs.removeExpiredMovers(); // remove delete animations + expired movers.

   // See if we've changed locales  spSchmoozerLocale.getHash();
   if ( platform.getAvatarsLocale() != locale  ) {
      zprint(1,”onNavSec locale changed.\n”);

      locale = platform.getAvatarsLocale();
      viewPointChanged = true;
   }

   if (viewPointChanged) {
      viewPointChanged = false;
      zprint(3,”onNavSec viewPointChanged\n”);
```

```
         JMenuItem mi;
         spViewPoint vp;
         td.zprint(3,”\n onViewPointChanged STARTS \n\n”);
         // Remove all
         viewMenu.removeAll();
         rideMenu.removeAll();
         for (int i = 0; i<VPs; i++)
            if (miView[i] != null) {
               miView[i].removeActionListener(lSymAction);
               miView[i] = null;
            }
         VPs = 0;
// I want to recompute the names on the fly incase someone changed theTitle fields
         // Add each viewpoint back
         Hashtable hash = spViewPoint.getHash();
         Enumeration e = hash.elements();
         String title;
         while (e.hasMoreElements()) {
            if ((vp = (spViewPoint) e.nextElement()) != null ) {
               boolean camera = vp.getCamera();// || (vp.getState().
                                                   equalsIgnoreCase(“ride”));
               boolean ride = vp.getRide();// || (vp.getState().
                                                   equalsIgnoreCase(“ride”));
               spPart parent = vp.getParent() instanceof spPart ?
                                                    (spPart)vp.getParent() : null;
               if (parent==null)   title = vp.getTitle() + “ “ + vp.getType();
               else             title = parent.getTitle() + “ “ + parent.getType() + “ “ +
                                                   vp.getTitle() + “ “ + vp.getType();

               //if (title.length() > 1) {
               // If it’s camera only, it should be inside this locale.
               if (camera && vp.getLocale() != locale ) continue;

               td.zprint(3,”onEachView “+VPs+” “+title+”\n”);
               mi = new JMenuItem(title);
               if (ride) { // add to ride list
                  rideMenu.add(mi);
               } else { // add to view list
                  viewMenu.add(mi);
               }
               miView[VPs] = mi;
               VP[VPs++] = vp;
               mi.addActionListener(lSymAction);
               if (++VPs > VIEWS - 3) break; // Save room for resets
               //}
            }
         }// while
         // Add resets
         viewMenu.addSeparator();
         rideMenu.addSeparator();
         viewMenu.add(miView[VPs++] = (mi = new JMenuItem(“Reset”)) );
         mi.addActionListener(lSymAction);
         rideMenu.add(miView[VPs++] = (mi = new JMenuItem(“Reset”)) );
         mi.addActionListener(lSymAction);
         td.zprint(3,”\n onViewPointChanged ENDS “+VPs+” viewPoints\n\n”);
      } // viewPointChanged
   }

   /** Camera only view points.
     * @param n     Fixed camera position.
     * @param move  If true move the camera.
     */
            public void miAvatar_Action(int n,boolean move)
            {
      zprint(1,”miAvatar_Action n=”+n+” move=”+move+”.\n”);
      tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();
                 cameraPosition = n;
                           if (move)
                           {

                float[] tran = platform.getTilt().getTransform();

                tran[spThing.spTransformRX] = 1;
                tran[spThing.spTransformRY] = 0;
                tran[spThing.spTransformRZ] = 0;
         tran[spThing.spTransformRA] = (n < 8) ? -0.333f : -1.57f;
         platform.getTilt().setTransform(tran);


      platform.avatarGoto(fixedcampos[n],platform.getAvatarsLocale(),3500,true);

                miCamera[n].setSelected(true);
                miViews[n].setSelected(true);
                          for (int i=0; i<CAMERAPOSITIONS; i++)
```

```
                        miCamera[i].setFont( i == n ? boldFont : regFont );

  }
}

/** Move the camera to a fixed view point.
 * @param next  If true increment the fixed camera position.
 */
            public void flyCamera(boolean next) {
  zprint(1,"flyCamera next="+next+".\n");

              int n = cameraPosition + 1;
              if (next) {
              if (n >= CAMERAPOSITIONS) n = 0;
            } else {
              n = DEFAULTPOSITION;
            }
              miAvatar_Action(n,next);
            }


            class SymAction implements java.awt.event.ActionListener
            {
            public void actionPerformed(java.awt.event.ActionEvent event)
                      {
                                zprint(2,"NAV actionPerformed\n");
                                Object object = event.getSource();
    if (object == miStats) (new tdStatsDialog()).show();

                        for (int i=0; i<3; i++)
                          if (object == miSpeed[i])
                            miSpeed_Action(i);
                        for (int i=0; i<CAMERAPOSITIONS; i++)
                        {
                          if (object == miCamera[i])
                            miAvatar_Action(i,true);
                          if (object == miViews[i])
                          {
                            miAvatar_Action(i,true);
                          }
                        }
                                for (int i = 0; i<VPs; i++)
                                  if (object == miView[i])
                                    viewPoint_Action(i);
                      }
            }

            class LisAction implements tdVisualListener.visualListener
            {
                      public void eventAction(tdVisualEvent event){
                                zprint(4,"NAV tdVisualListener\n");
                        tdVisualListener lis = event.getSource();
            boolean hasLis = tdVisualListener.hasListener(event.getThing());



    if (lis == frameLis)              onNavFrame(event);
    else if (lis == selectLis)
            if (event.getThing() instanceof spBlock)
              {
                schmoozer.activateEditor(1);
              }
              else                  {}
    else if (lis == secLis)           onNavSec();
    else if (lis == upLis)            onMouseUp();
    else if (lis == dragLis && !hasLis) onNavDrag(event);
    else if (lis == gotoLis && !hasLis) onNavGoto(event);
    else if (lis == modeLis)          onNavModeChange(event);
    else if (lis == keyLis)           onNavKey(event);
                      }
            }

            /** Changes the avatars speed within a range, from a menue.
 * @param n  Integer scale.
              */
            public void miSpeed_Action(int n) {
              if (n == 2) { // Default
            tranScale = DEFAULT_TRAN_SCALE;
            rotScale = DEFAULT_ROT_SCALE;
  } else {
            tranScale *= ((n == 0) ? 2.0f : .5f);
            rotScale *= ((n == 0) ? 1.2f : .8f);
  }

            for (int i=0; i<3; i++) miSpeed[i].setFont( i == n ? boldFont : regFont );
```

```
  float speed = tranScale / DEFAULT_TRAN_SCALE;
  SpeedMenu.setLabel("Speed "+speed+" ");// <== changes font, why?
  SpeedMenu.setFont(regFont);//        <== has no effect.
          }
/** Stops avatar dragging.*/
public void navDragEnd() {
  zprint(1,"navDragEnd\n");
  inDrag = false;
  frameLis.enable(false);
  tdAvatarPlatform.getPlatform().avatarStop(false);
}
/** When the mouse up is detected, dragging stops.*/
public void onMouseUp() {
  zprint(1,"onMouseUp Dragging="+inDrag+"\n");
  if (inDrag) navDragEnd();
}

/** Called every frame to drag the avatar*/
public void onNavFrame(tdVisualEvent event) {
  zprint(3,"onNavFrame\n");
  tdAvatarPlatform.getPlatform().avatarMoveEachFrame();
}

/** Detects mouse drags and builds delta movement vector for the avatar step.*/
public void onNavDrag(tdVisualEvent event) {
  int X = event.getX();
  int Y = event.getY(); // < 0 if mouse on titlebar
  shift = event.getShiftKey();

  zprint(3,"onNavDrag\n");

  if ( X > -1 && Y > -1 && X < 4096 && Y < 4096 ) {
    float dx=0,dy=0,dz=0,dr=0,tr=0;
    if (X > max_x) max_x = X;
    if (Y > max_y) max_y = Y;
    float DX = ((float)(X - dragStartX))/(max_x);
    float DY = ((float)(Y - dragStartY))/(max_y);
    tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();

    if (!inDrag) { // Start dragging
      zprint(2,"onNavDrag starting\n");
      inDrag = true;
      frameLis.enable(true);

      dragStartX = X;
      dragStartY = Y;
    } else if ((event.getMouse() & tdVisualListener.MOUSE_RIGHT)>0) {
      if ( event.getControlKey() ) {
        tr = rotScale * DY;  // Y=Tilt up/down around X
      } else {
        dy = tranScale * DY; // Y=Translate up/down
        dx = tranScale * DX; // X=Translate left/right
      }
    } else {
      dz = tranScale * -DY; // Y=Translate forward/Backward
      dr = rotScale  * -DX; // X=Rotate left/right around Y
    }

    //zprint(3,"Dx="+DX+" DR="+dr+" TR="+tr+"\n");

    if ( event.getControlKey() ) {
      deltaXaxis.pt[spThing.spTransformRA] = tr;
      platform.avatarGoToward(deltaXaxis,event.getShiftKey()); // shift
    } else {
      deltaYaxis.pt[spThing.spTransformX]  = dx * java.lang.Math.abs(dx);
      deltaYaxis.pt[spThing.spTransformY]  = dy * java.lang.Math.abs(dy);
      deltaYaxis.pt[spThing.spTransformZ]  = dz * java.lang.Math.abs(dz);
      deltaYaxis.pt[spThing.spTransformRA] = dr * java.lang.Math.abs(dr);
      platform.avatarGoToward(deltaYaxis,event.getShiftKey()); // shift
    }
  }
}

/** Travel to the part. Called when a double click is detected.*/
public void onNavGoto(tdVisualEvent event) {
  zprint(3,"onNavGoto\n");
  spThing t = event.getThing();
  if (t != null) {
    float sign = event.getShiftKey() ? -1 : 1;
    tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();
    spAvatar a = platform.getAvatar();
    tdPoint tp = tdx.spThingGetLocaleTransform(t);
    tdPoint ap = tdx.spThingGetLocaleTransform(a);
```

```
         // newAvatar = avatar + ( delta * percent )
         for (int i=0;i<3;i++) ap.pt[i] += ((tp.pt[i]-ap.pt[i])*PERCENT_GOTO*sign);

         platform.avatarGoto(ap,t.getLocale(),3000,true);
      }
   }
   /** Sets the target of the active editors marker.
     * @param p            Part to set the target to.
     * @param moveToTarget  If true, animate the move.
     */
   public void setActiveMarkersTarget(spPart p, boolean moveToTarget) {
      zprint(4,"setActiveMarkersTarget by tdViewEditor\n");
      tdEditor editor;
      spMarker marker;
      if ((null != (editor = schmoozer.getActiveEditor())) &&
         (null != (marker = editor.getMarker())))
         marker.setTarget(p,moveToTarget);
   }

   /** Changes the navigatoin mode by clicking on a target,
              then the right editor is found for that part */
   public void onNavModeChange(tdVisualEvent event) {
      zprint(1,"onNavModeChange\n");
      int next;
      spThing t = event.getThing();
      if (on) { // Was navagating, and hit a target.
         next = schmoozer.lastEditorUsed; // getLastEditorUsed()???
         if (t != null) {
            if (t instanceof spTrack)   next = 1;
            if (t instanceof spRailCar) next = 2;
            if (t instanceof spScenery) next = 3;
            if (t instanceof spBlock) next = 4;
         }
      } else // Was not navigating.
         next = 0;
      schmoozer.activateEditor(next);
      if (t != null && t instanceof spPart)
         setActiveMarkersTarget((spPart)t,true);
   }
 /** Process a key from spVisuals keyboard.*/
   public void onNavKey(tdVisualEvent event) {
      float dx=0,dy=0,dz=0,dr=0,tr=0;
      zprint(1,"onNavKey "+event.getValue()+"\n");

      if ((event.getEventType() & tdVisualListener.
                              EVENT_KEY_RELEASE) != 0) { // Release
         onMouseUp();
      } else if (!inDrag) { // Press
         tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();
         if ((platform.getAvatarsLocale()) == null) return; // No locale

         float RotStep = 0.2f * (float) java.lang.Math.sqrt(KeySpeed);
         float MoveStep = 20 * KeySpeed;
         switch(event.getValue()) {
            case tdVisualListener.KEY_END:
            case tdVisualListener.KEY_1:       tr = -RotStep; break; // Tilt Down
            case tdVisualListener.KEY_DOWN:
            case tdVisualListener.KEY_2:       dz =  MoveStep; break; // Back
            case tdVisualListener.KEY_PAGE_DOWN:
            case tdVisualListener.KEY_3:       dy = -MoveStep/2.0f; break; // Down
            case tdVisualListener.KEY_LEFT:
            case tdVisualListener.KEY_4:       dr =  RotStep; break; // rot left
            case tdVisualListener.KEY_RIGHT:
            case tdVisualListener.KEY_6:       dr = -RotStep; break; // rot right
            case tdVisualListener.KEY_HOME:
            case tdVisualListener.KEY_7:       tr =  RotStep; break; // tilt up
            case tdVisualListener.KEY_UP:
            case tdVisualListener.KEY_8:       dz = -MoveStep; break; // Forward
            case tdVisualListener.KEY_PAGE_UP:
            case tdVisualListener.KEY_9:       dy =  MoveStep/2.0f; break; // Up
            case tdVisualListener.KEY_0:       flyCamera(false);
               // End flight and zero tilt, keep position&rot but reset transform.
                                 highlightView(-1);
                              platform.resetAvatarsTransforms(true,2000); break;
            case tdVisualListener.KEY_5:       flyCamera(false); // End ride,flight,
                                 highlightView(-1);
                              platform.resetAvatarsTransforms(false,3500); break;
      case tdVisualListener.KEY_PERIOD:  flyCamera(true); break; // next fly point
            case tdVisualListener.KEY_POSITIVE: KeySpeed = KeySpeed*2.0f;
                              if (KeySpeed>MAXSPEED) KeySpeed=MAXSPEED;
                                 zprint(1,"Speed="+KeySpeed+"\n"); break;
            case tdVisualListener.KEY_NEGITIVE: KeySpeed = KeySpeed*0.5f;
                              if (KeySpeed<MINSPEED) KeySpeed=MINSPEED;
                                 zprint(1,"Speed="+KeySpeed+"\n"); break;
```

```
            default: ;
         }

         if ( (dx != 0) || (dy != 0) || (dz != 0) || (dr != 0) || (tr != 0) ) {
            if (tr != 0) { // Tilt
               deltaXaxis.pt[spThing.spTransformRA] = tr;
               platform.avatarGoToward(deltaXaxis,event.getShiftKey());
            } else { // Translate
               deltaYaxis.pt[spThing.spTransformX]  = dx;
               deltaYaxis.pt[spThing.spTransformY]  = dy;
               deltaYaxis.pt[spThing.spTransformZ]  = dz;
               deltaYaxis.pt[spThing.spTransformRA] = dr;
               platform.avatarGoToward(deltaYaxis,event.getShiftKey());
            }
            inDrag = true; // Continue the drag
            frameLis.enable(true);
         }

         // applyDelta(event); // Make the move.
      } // else inDrag so continue.
   }
   /** Required for all editors. Enables and diables the visual event listeners.
     * @param show  True to enable.
     */
   public void enable(boolean show){
      if (on != show) {
                  on = show;
         zprint(2,"NavEditor enable "+on+"\n");
         upLis.enable(on);
         dragLis.enable(on);
         gotoLis.enable(on);
      }
   }

   public int currentView()
   {
      for (int i = 0; i < CAMERAPOSITIONS; i++)
      {
         if(miViews[i].isSelected())
         {
            return i;
         }
      }
      return 8;
   }

                              //{{DECLARE_CONTROLS
               //}}
}
```

public class tdWM extends td

```
/*  HAYMAKER ALTERED
** $Id: tdWM.java,v 1.22 1998/11/03 21:42:51 derek Exp $
** Copyright © 1998 MITSUBISHI ELECTRIC ITA.  ALL RIGHTS RESERVED
*/
import java.util.*;
/** This class implements all access to the spline c core.*/
public class tdWM extends td {
                public tdWM (String s) {
      int wm = WMnew(s);
                }
   static native public  synchronized int     spVisualGetWindowSizeX();
   static native public  synchronized int     spVisualGetWindowSizeY();
   static native public  synchronized int
                              spTrainStart(int locomotive,int track,int startIndex);
   static native public  synchronized int     spTrainStop(int locomotive);

   static native public  synchronized int     WMnew(String s);
   static native public  synchronized int     WMremove();
   static native public  synchronized int     WMupdate(int[] events);
                              // only called from schmoozer.java.
   static native public  synchronized String  getenv(String s);
   static native public  synchronized int     system(String s);
   static native public  synchronized void    setVisualEventMask(int value);

   static native public  synchronized void
                              serRestoreLocale(int sp,String name, String fullpath);
   static native public  synchronized void    serSaveLocale(int locale, String
fullpath);
   //sp
   static native public  synchronized int
                              newObj(String klass, String superKlass, String initStr);
   static native public  synchronized void    removeSp(int sp);
   static native public  synchronized boolean getLocallyOwned(int sp);
   static native public  synchronized int     getOwner(int sp);
   static native public  synchronized void    setOwner(int sp, int to);
   static native public  synchronized int     getLocale(int sp);
   static native public  synchronized int     getParent(int sp);
   static native public  synchronized void    setParent(int sp, int parent);
   static native public  synchronized boolean getDoNotSerialize(int sp);
   static native public  synchronized void    setDoNotSerialize(int sp, boolean init);
   //sp Experimental
   static native public  synchronized boolean getMessageNeeded(int sp);
   static native public  synchronized void    setMessageNeeded(int sp, boolean init);
   static native public  synchronized boolean getChange(int sp);
   static native public  synchronized void    setChange(int sp, boolean init);

   //thing
   static native public  synchronized float   getThingInRadius(int sp);
   static native public  synchronized void    setThingInRadius(int sp, float r);
   static native public  synchronized float   getThingOutRadius(int sp);
   static native public  synchronized void    setThingOutRadius(int sp, float r);
   static native public  synchronized void    setThingVisualDefinition(int sp, int to);
   static native public  synchronized int     getThingVisualDefinition(int sp);
   static native public  synchronized boolean
                                      localize(int obj, int loc, boolean smallest);
   static native public  synchronized void    getTransform(int sp, float[] T);
   static native public  synchronized void    setTransform(int sp, float[] T);
   static native public  synchronized void
                                      getRelativeVector(int sp1, int sp2, float[] V);
   //thing-motion
   static native public  synchronized void    thingStop(int sp);
   static native public  synchronized void    thingFlushMotionQueue(int sp);
   static native public  synchronized void    thingGoThru(int sp, float[] pt, int ms);
   static native public  synchronized void    thingStopAt(int sp,  float[] pt, int ms);
   static native public  synchronized int     getMotionTimeLeft(int sp);
   static native public  synchronized void    thingLookAt(int sp1, int sp2);
   //thing-math-via-transform
   //static native public  synchronized void
                                      getRelativeTransform(int sp1, int sp2, float[]
T);
   //static native public  synchronized void
                                      transformMult(float[] t1, float[] t2, float[]
result);
   //static native public  synchronized void    transformInvert(float[] t1, float[] result);
   //thing matrix math
   static native public  synchronized void    transformToMatrix(float[] t, float[] m);
   static native public  synchronized void    matrixToTransform(float[] m, float[] t);
   static native public  synchronized void
                                      matrixMult(float[] m1, float[] m2, float[] result);
   static native public  synchronized void    matrixInvert(float[] m, float[] result);
   static native public  synchronized void
                                      getRelativeMatrix(int sp1, int sp2, float[] m);
```

```
  // POV
  static native public  synchronized boolean getPOVIgnoreNearby(int sp);
  static native public  synchronized void    setPOVIgnoreNearby(int sp, boolean b);

  // Part
  static native public  synchronized boolean getPartInited(int sp);
  static native public  synchronized void    setPartInited(int sp, boolean init);
  static native public  synchronized boolean getPartRunTimeClass(int sp);
  static native public  synchronized void setPartRunTimeClass(int sp, boolean init);
 static native public  synchronized void
                            setPartOriginalVisualDefinition(int sp, int to);
  static native public  synchronized int     getPartOriginalVisualDefinition(int sp);
  static native public  synchronized String  getPartType(int sp);
  static native public  synchronized void    setPartType(int sp, String s);
  static native public  synchronized String  getPartTitle(int sp);
  static native public  synchronized void    setPartTitle(int sp, String s);
  static native public  synchronized String  getPartState(int sp);
  static native public  synchronized void    setPartState(int sp, String s);
  static native public  synchronized String  getPartSubState(int sp);
  static native public  synchronized void    setPartSubState(int sp, String s);
  // To be removed
  static native public  synchronized int     getPartPortBeacon(int sp);

  // Link
  static native public  synchronized String  getLinkURL(int sp);
  static native public  synchronized void    linkURLAltered(int sp);

  // RailCar
  static native public  synchronized void    setRailCarMass(int sp, float fr);
  static native public  synchronized float   getRailCarMass(int sp);
  static native public  synchronized void    setRailCarRollingFriction(int sp, float fr);
  static native public  synchronized float   getRailCarRollingFriction(int sp);
  static native public  synchronized int     getRailCarCarInFront(int sp);
 static native public  synchronized void setRailCarCarInFront(int sp, int spInFront);
  static native public  synchronized int     getRailCarCarInBack(int sp);
 static native public  synchronized void    setRailCarCarInBack(int sp, int
spInBack);

  static native public  synchronized float   getRailCarFrontTruck(int sp);
  static native public  synchronized void    setRailCarFrontTruck(int sp, float f);
  static native public  synchronized float   getRailCarBackTruck(int sp);
  static native public  synchronized void    setRailCarBackTruck(int sp, float f);
  static native public  synchronized float   getRailCarBackCoupler(int sp);
  static native public  synchronized void    setRailCarBackCoupler(int sp, float f);

   // Block
  static native public  synchronized void    setBlockMass(int sp, float fr);
  static native public  synchronized float   getBlockMass(int sp);
  static native public  synchronized void    setBlockX(int sp, float fr);
  static native public  synchronized float   getBlockX(int sp);
  static native public  synchronized void    setBlockZ(int sp, float fr);
  static native public  synchronized float   getBlockZ(int sp);
  static native public  synchronized void    setBlockY(int sp, float fr);
  static native public  synchronized float   getBlockY(int sp);
  static native public  synchronized void    setBlockMaterial(int sp, int fr);
  static native public  synchronized int     getBlockMaterial(int sp);

  // Locomotive
  static native public  synchronized int     getLocomotiveGear(int sp);
  static native public  synchronized void    setLocomotiveGear(int sp, int spInBack);
  static native public  synchronized float   getLocomotiveAccelerator(int sp);
  static native public  synchronized void    setLocomotiveAccelerator(int sp, float
fr);
  static native public  synchronized float   getLocomotiveDecelerator(int sp);
  static native public  synchronized void    setLocomotiveDecelerator(int sp, float
fr);
  static native public  synchronized float   getLocomotiveVelocity(int sp);
  static native public  synchronized int     getLocomotiveShiftForward(int sp);
  static native public  synchronized void
                             setLocomotiveShiftForward(int sp, int spInBack);
  static native public  synchronized int     getLocomotiveShiftNeutral(int sp);
  static native public  synchronized void
                            setLocomotiveShiftNeutral(int sp, int spInBack);
  static native public  synchronized int     getLocomotiveShiftReverse(int sp);
  static native public  synchronized void
                             setLocomotiveShiftReverse(int sp, int spInBack);
  static native public  synchronized int     getLocomotiveThrottleCtrl(int sp);
  static native public  synchronized void
                             setLocomotiveThrottleCtrl(int sp, int spInBack);
  static native public  synchronized int     getLocomotiveHornCtrl(int sp);
  static native public  synchronized void
                            setLocomotiveHornCtrl(int sp, int spInBack);
  static native public  synchronized int     getLocomotiveBellCtrl(int sp);
  static native public  synchronized void
                            setLocomotiveBellCtrl(int sp, int spInBack);
```

```
// Track
static native public  synchronized float   getTrackLength(int sp);
static native public  synchronized void    setTrackLength(int sp, float f);
static native public  synchronized float   getTrackDegrees(int sp);
static native public  synchronized void    setTrackDegrees(int sp, float f);
static native public  synchronized float   getTrackCrossDegrees(int sp);
static native public  synchronized void    setTrackCrossDegrees(int sp, float f);
static native public  synchronized float   getTrackGrade(int sp);
static native public  synchronized void    setTrackGrade(int sp, float f);
static native public  synchronized float   getTrackHeight(int sp);
static native public  synchronized void    setTrackHeight(int sp, float f);
static native public  synchronized void    setTrackPoints(int sp, int pts);

static native public  synchronized int     getTrackConnect0(int sp);
static native public  synchronized int     getTrackConnect1(int sp);
static native public  synchronized int     getTrackConnect2(int sp);
static native public  synchronized int     getTrackConnect3(int sp);
static native public  synchronized void    setTrackConnect0(int sp, int to);
static native public  synchronized void    setTrackConnect1(int sp, int to);
static native public  synchronized void    setTrackConnect2(int sp, int to);
static native public  synchronized void    setTrackConnect3(int sp, int to);

static native public  synchronized int     getTrackConnectIndex(int sp, int i);
static native public  synchronized void    setTrackConnectIndex(int sp, int i,int to);
static native public  synchronized void
                         setTrackPathData(int id, int pnum, int startp, int endp,
                String requires, float len, float degy, float degx, int pts, float[] fa);
//TrackSignal
static native public  synchronized int     getTrackSignalBar1(int sp);
static native public  synchronized void    setTrackSignalBar1(int sp, int to);
static native public  synchronized int     getTrackSignalBar2(int sp);
static native public  synchronized void    setTrackSignalBar2(int sp, int to);

// Audio
static native public  synchronized int
          newDoSoundPlay(int sound, int source, boolean loop, float gain);
static native public  synchronized boolean getAudioSourceLive(int sp);
static native public  synchronized void    setAudioSourceLive(int sp, boolean b);
// AudioScenery
static native public  synchronized String  getAudioSceneryAudioFile(int sp);
static native public  synchronized void
                              setAudioSceneryAudioFile(int sp, String f);
static native public  synchronized int     getAudioSceneryMode(int sp);
static native public  synchronized void    setAudioSceneryMode(int sp, int mode);
static native public  synchronized int     getAudioSceneryLength(int sp);
static native public  synchronized void    setAudioSceneryLength(int sp, int ms);
static native public  synchronized int     getAudioSceneryFixedInterval(int sp);
static native public  synchronized void
                              setAudioSceneryFixedInterval(int sp, int ms);
static native public  synchronized float   getAudioSceneryOutterRadius(int sp);
static native public  synchronized void
                              setAudioSceneryOutterRadius(int sp, float f);
static native public  synchronized float   getAudioSceneryInnerRadius(int sp);
static native public  synchronized void
                              setAudioSceneryInnerRadius(int sp, float f);
static native public  synchronized float   getAudioSceneryGain(int sp);
static native public  synchronized void    setAudioSceneryGain(int sp, float f);

// OMI
static native public  synchronized String  getOMIMethodClassName(int sp);
static native public  synchronized void
                              setOMIMethodClassName(int sp, String s);
static native public  synchronized String  getOMIMethodName(int sp);
static native public  synchronized void    setOMIMethodName(int sp, String s);
static native public  synchronized String  getOMIStr0(int sp);
static native public  synchronized void    setOMIStr0(int sp, String s);
static native public  synchronized String  getOMIStr1(int sp);
static native public  synchronized void    setOMIStr1(int sp, String s);
static native public  synchronized String  getOMIStr2(int sp);
static native public  synchronized void    setOMIStr2(int sp, String s);
static native public  synchronized String  getOMIStr3(int sp);
static native public  synchronized void    setOMIStr3(int sp, String s);
static native public  synchronized int     getOMISp0(int sp);
static native public  synchronized void    setOMISp0(int sp, int id);
static native public  synchronized int     getOMISp1(int sp);
static native public  synchronized void    setOMISp1(int sp, int id);
static native public  synchronized int     getOMISp2(int sp);
static native public  synchronized void    setOMISp2(int sp, int id);
static native public  synchronized int     getOMISp3(int sp);
static native public  synchronized void    setOMISp3(int sp, int id);
static native public  synchronized void    getOMIFloats(int sp, float[] F);
static native public  synchronized void    setOMIFloats(int sp, float[] F);
static native public  synchronized void    getOMIInts(int sp, int[] I);
static native public  synchronized void    setOMIInts(int sp, int[] I);

// Ports TO BE DELETED
static native public  synchronized String  getBeaconConnRequestToStr(int sp);
static native public  synchronized void
                         setBeaconConnRequestToStr(int sp, String s);
static native public  synchronized String
                          getBeaconConnRequestLeadsToStr(int sp);
static native public  synchronized void
                         setBeaconConnRequestLeadsToStr(int sp, String s);
static native public  synchronized String
                         getBeaconConnRequestFromStr(int sp);
static native public  synchronized void
                         setBeaconConnRequestFromStr(int sp, String s);
static native public  synchronized String
                         getBeaconConnRequestFromServerStr(int sp);
static native public  synchronized void
                         setBeaconConnRequestFromServerStr(int sp, String s);
static native public  synchronized String
                          getBeaconConnRequestPortNameStr(int sp);
static native public  synchronized void
                         setBeaconConnRequestPortNameStr(int sp, String s);
static native public  synchronized String
                         getBeaconConnRequestToPortNameStr(int sp);
static native public  synchronized void
                          setBeaconConnRequestToPortNameStr(int sp, String s);
static native public  synchronized String
                         getBeaconConnRequestReqType(int sp);
static native public  synchronized void
                          setBeaconConnRequestReqType(int sp, String s);

// TO BE DELETED spBeaconConnRequest
static native public  synchronized int
            newBeaconConnRequest(String Beacon, int parent, int ix, float[] pt);
// GenericScenery
static native public  synchronized void
                         getGenericSceneryPosition(int sp, float[] T);
static native public  synchronized void
                         setGenericSceneryPosition(int sp, float[] T);
// TouchScenery
static native public  synchronized String  getTouchLinkURL(int sp);
static native public  synchronized void    setTouchLinkURL(int sp, String s);

// Locale
static native public  synchronized String  getLocaleURLName(int sp);
static native public  synchronized int    getLocaleBoundary(int sp);
static native public  synchronized void    setLocaleBoundary(int sp, int b);

// SchmoozerLocale
static native public  synchronized boolean getSchmoozerLocaleSnapOn(int sp);
static native public  synchronized void
                         setSchmoozerLocaleSnapOn(int sp, boolean b);
static native public  synchronized float
                         getSchmoozerLocaleSnapToPosition(int sp);
static native public  synchronized void
                         setSchmoozerLocaleSnapToPosition(int sp, float f);
static native public  synchronized float
                         getSchmoozerLocaleSnapToRadians(int sp);
static native public  synchronized void
                         setSchmoozerLocaleSnapToRadians(int sp, float f);

static native public  synchronized void
                         setSchmoozerLocaleVDefSelConc(int sp, int to);
static native public  synchronized int   getSchmoozerLocaleVDefSelConc(int sp);
static native public  synchronized void
                          setSchmoozerLocaleVDefNorConc(int sp, int to);
static native public  synchronized int   getSchmoozerLocaleVDefNorConc(int sp);
static native public  synchronized void
                         setSchmoozerLocaleVDefSelGlas(int sp, int to);
static native public  synchronized int   getSchmoozerLocaleVDefSelGlas(int sp);
static native public  synchronized voi
                         setSchmoozerLocaleVDefNorGlas(int sp, int to);
static native public  synchronized int   getSchmoozerLocaleVDefNorGlas(int sp);
static native public  synchronized void
                          setSchmoozerLocaleVDefgreen50(int sp, int to);
static native public  synchronized int   getSchmoozerLocaleVDefgreen50(int sp);
static native public  synchronized void
                          setSchmoozerLocaleVDefgreen100(int sp, int to);
static native public  synchronized int   getSchmoozerLocaleVDefgreen100(int
sp);
static native public  synchronized void
                         setSchmoozerLocaleVDefred50(int sp, int to);
static native public  synchronized int   getSchmoozerLocaleVDefred50(int sp);
static native public  synchronized void
```

```
                         setSchmoozerLocaleVDefred100(int sp, int to);
  static native public  synchronized int  getSchmoozerLocaleVDefred100(int sp);
  static native public  synchronized void
                          setSchmoozerLocaleVDefyellow50(int sp, int to);
  static native public  synchronized int  getSchmoozerLocaleVDefyellow50(int sp);
  static native public  synchronized void
                          setSchmoozerLocaleVDefyellow100(int sp, int to);
  static native public synchronized int getSchmoozerLocaleVDefyellow100(int sp);
  //HaymakerAddded

  // Neighbor
  static native public  synchronized void
                         getNeighborImportTransform(int sp, float[] T);
  static native public  synchronized void
                         setNeighborImportTransform(int sp, float[] T);
  static native public  synchronized void
                         getNeighborExportTransform(int sp, float[] T);
  static native public  synchronized void
                         setNeighborExportTransform(int sp, float[] T);
  static native public  synchronized String
                         getNeighborLocaleURLName(int sp);
  // CubicBoundary
  static native public  synchronized float  getCubicBoundaryVol(int sp);
  static native public  synchronized void   setCubicBoundaryVol(int sp, float v);
  static native public  synchronized void
                         getCubicBoundaryMinVector(int sp, float[] T);
  static native public  synchronized void
                         setCubicBoundaryMinVector(int sp, float[] T);
  static native public  synchronized void
                         getCubicBoundaryMaxVector(int sp, float[] T);
  static native public  synchronized void
                         setCubicBoundaryMaxVector(int sp, float[] T);

  // spBeacon
  static native public  synchronized String  getBeaconLocaleURLName(int sp);
  static native public  synchronized String  getBeaconTag(int sp);
  // LandingPad
  static native public  synchronized String  getLandingPadDescription(int sp);
  static native public  synchronized void
                         setLandingPadDescription(int sp, String s);
  static native public  synchronized String  getLandingPadIconFile(int sp);
  static native public  synchronized void    setLandingPadIconFile(int sp, String s);
  static native public  synchronized int     getLandingPadAvatars(int sp);
  static native public  synchronized void    setLandingPadAvatars(int sp, int i);

  //ViewPoint
  static native public  synchronized boolean getViewPointRide(int sp);
  static native public  synchronized void    setViewPointRide(int sp, boolean b);
  static native public  synchronized boolean getViewPointCamera(int sp);
  static native public  synchronized void    setViewPointCamera(int sp, boolean b);

  // spMarker
  static native public  synchronized int     getMarkerTarget(int maker);
  static native public  synchronized void    setMarkerTarget(int maker, int target);

  // spSeeing
  static native public  synchronized float  getSeeingFarClip(int sp);
  static native public  synchronized void   setSeeingFarClip(int sp, float f);
  static native public  synchronized float  getSeeingNearClip(int sp);
  static native public  synchronized void   setSeeingNearClip(int sp, float f);

  // JNI support for callbacks
  static native public  synchronized int     beaconGoto(int avatar, String url, int ms);
  static native public  synchronized void
                         newBeaconMonitor(int jcbid, String Search, int ms);
  static native public  synchronized void
                         classExamine(int jcbid, String spClassName, String
Mask);
  static native public  synchronized int
                          classMonitor(int jcbid, String spClassName, String Mask);
  static native public  synchronized void
newCallback(int jcbid, String Predicate, String spClassName, int sp, int interval);
  static native public  synchronized void    removeCallback(int cbid);

  // Java Cache support
  static native public  synchronized String  cacheGetUrl();
  static native public  synchronized void    cacheSetFile(String fileName);

  // Required for JNI
  static {
    System.loadLibrary("JNIwm");
  }
}
```

public class filter extends td

```
import java.util.*;
import java.awt.*;
import java.lang.*;
public class filter extends td
{
   public static void perceive(String myFilter)
   {
      if(tdAvatarPlatform.getAvatarsLocale().getTitle().compareTo(myFilter) == 0)
      {}
      else
      {
         deleteAllBlocks(myFilter);
         Vector selectedBlocks = selectWorldData(myFilter);
         goToFilter(myFilter);
         constructView(selectedBlocks, myFilter);
      }
   }
   public static void importToWorld(boolean replace)
   {
    Vector importData = selectWorldData();//select the blocks in the current locale
    goToFilter("worldLocale");
   if(replace)
   {
     Enumeration e = tdx.spDescendants(tdAvatarPlatform.getAvatarsLocale()).
                                                                     elements();
     //myMarker.setTarget(null,false);
     while (e.hasMoreElements())
     {
        sp s = (sp) e.nextElement();
        if  (s.getLocallyOwned() && (s instanceof spBlock))
        {
           s.remove(); //this could be a problem
        }
        else if ((! s.getLocallyOwned()) && (s instanceof spBlock))
        {
           spOMI.makeStaticCall("onDeleteTree", "tdEditor", (spBlock)s);
        }
     }
    }
    constructWorldView(importData);
}
   public static Vector selectWorldData(String myFilter)
   {
      if(myFilter.compareTo("spaceFilter") == 0)
      {
         return selectSpaceData();
      }
      else if(myFilter.compareTo("enclosureFilter") == 0)
      {
         return selectEnclosureData();
      }
      else if(myFilter.compareTo("lightFilter") == 0)
      {
         return selectLightData();
      }
      else if(myFilter.compareTo("structureFilter") == 0)
      {
         return selectStructureData();
      }
      else if(myFilter.compareTo("chairFilter") == 0)
      {
         return selectChairData();
      }
      else return selectWorldData();

   }

   public static boolean goToFilter(String myFilter)
   {
      if(tdAvatarPlatform.getAvatarsLocale().getTitle().compareTo(myFilter) == 0)
      {
         System.out.println("Already in " + myFilter);
      }
      else
      {
         schmoozer.WMupdate();
         tdBlockEditor.myMarker.setTarget(null,false);
         tdAvatarPlatform platform = tdAvatarPlatform.getPlatform();
         float[] avatarsPoint = platform.getAvatar().getTransform();
         float[] avatarsTilt = platform.getTilt().getTransform();
         Enumeration e = spLandingPad.getHash().elements();//find the beacon
         String ls = schmoozer.getPreference("localeServer");
                              ls = "istp://" + ls + "/LandingPad/" + myFilter;
         while (e.hasMoreElements())
         {
            spLandingPad t = (spLandingPad) e.nextElement();
            if ((t.getTag().compareTo(ls)) == 0)
            {
               tdAvatarPlatform.avatarGoto(t);
               platform.avatarLocalize(true);
               platform = tdAvatarPlatform.getPlatform();
               platform.avatarGoto(new tdPoint(avatarsPoint),
                          platform.getAvatarsLocale(),10,true);
               platform.getTilt().setTransform(avatarsTilt);
             schmoozer.getEditorByIndex(tdBlockEditor.
                                   thisEditorsIndex).getMarker().putIntoLocale();
               schmoozer.getEditorByIndex(tdBlockEditor.
                                   thisEditorsIndex).dragTarget = null;
               return true;
            }
         }
      }
      return false;
   }
   public static boolean constructView(Vector selectedBlocks, String myFilter)
   {

      if(myFilter.compareTo("spaceFilter") == 0)
      {
         return constructSpaceView(selectedBlocks);
      }
      else if(myFilter.compareTo("enclosureFilter") == 0)
      {
         return constructEnclosureView(selectedBlocks);
      }
      else if(myFilter.compareTo("lightFilter") == 0)
      {
         return constructLightView(selectedBlocks);
      }
      else if(myFilter.compareTo("structureFilter") == 0)
      {
         return constructStructureView(selectedBlocks);
      }
      else if(myFilter.compareTo("chairFilter") == 0)
      {
         return constructChairView(selectedBlocks);
      }
      else return constructWorldView(selectedBlocks);
   }

////////////SPACE////////////////////////////////////////////////////////////
//this algorith constructs a series of spaceObjects - which contain one block
//which ocuppies the space, and the two blocks which define the space
//only vertical blocks define space  These space objects are
/ then grouped into group objects/which group all the space objects that touch
   public static Vector roofs;// keep a vector of the spBlocks that could be roofs
   public static Vector spaces;//keep a vector of the space objects
   public static boolean spacesInited = false;
   public static Vector selectSpaceData()
   {
      Vector selectedBlocks = new Vector();
      spSchmoozerLocale thisLocale = tdAvatarPlatform.getAvatarsLocale();
      Vector localeObj = tdx.spDescendants(thisLocale);
      Enumeration d =localeObj.elements();
      while (d.hasMoreElements())
      {
         Object t = d.nextElement();
         if (t instanceof spBlock)
         {
          selectedBlocks.addElement(t);// takes all  the blocks
         }
      }
      return selectedBlocks;
   }
   public static boolean constructSpaceView(Vector selectedBlocks)
   {
      if (spacesInited)
      {spaces.removeAllElements();
      roofs.removeAllElements();}
      else
      {spaces = new Vector();
      roofs = new Vector();}

      spatialWall spatialWall1;
      spatialWall spatialWall2;
      Enumeration e =selectedBlocks.elements();
      Vector spat = new Vector();
```

```
  while (e.hasMoreElements()) // go through all the blocks in the world
  {
     spBlock b = (spBlock)e.nextElement();
     spBlock newBlock = new spBlock((spBlock) b);
     newBlock.setReferenced(b);
                          //go through all the blocks and make them for world
     if(newBlock.getY() < 2)
     {roofs.addElement(newBlock);}
     else
     {
     spatialWall tmp = new spatialWall(newBlock);
     spat.addElement(tmp);//create a new "spatialBlock" from this
     }                               //counter
  }                             //  0 1 2 3 //spat1
  Enumeration spat1 = spat.elements();    // 0 x x x x
  int counter = 0;                     // 1 o x x x
  while(spat1.hasMoreElements())        // 3 o o o x
  {
     spatialWall1 = (spatialWall) spat1.nextElement();
                          //increments counter so spatialwall2 is always offset
     counter++;
     Enumeration spat2 = spat.elements();
     int i;
     for(i = 0; (i < counter); i++)
        {
        spatialWall2 = (spatialWall)spat2.nextElement();
                          //cycle through the redundant eleements
        }
     while(spat2.hasMoreElements())
     {
        spatialWall2 = (spatialWall)spat2.nextElement();
        calcSpaces(spatialWall1, spatialWall2);
     }
  }
  Enumeration s = spat.elements();
  while(s.hasMoreElements())
  {
     spatialWall temp = (spatialWall)s.nextElement();
     tdx.spDonateToLocale(temp.getMe());
     assignNormalGlas(temp.getMe());
  }
  Enumeration r = roofs.elements();
  while(r.hasMoreElements())
  {
     spBlock rf = (spBlock)r.nextElement();
     tdx.spDonateToLocale(rf);
     assignNormalGlas(rf);
  }
  schmoozer.WMupdate();
  return true;
}
public static boolean rectSize(Rectangle r)
{
 if ((r.width > 1) && (r.height > 1)) return true;
 else return false;
}
public static void calcSpaces(spatialWall sw1, spatialWall sw2)
{
     if((sw1.wallbotY >= sw2.walltopY) | (sw2.wallbotY >= sw1.walltopY))
     { return;}
     else
     {
      int botY = (sw1.wallbotY <= sw2.wallbotY) ? sw2.wallbotY : sw1.wallbotY;
      int topY = (sw1.walltopY >= sw2.walltopY) ? sw2.walltopY : sw1.walltopY;

        ///sw1.south and sw2.east
      if (sw1.getSouth().intersects(sw2.getEast()))
      {  Rectangle r = sw1.getSouth().intersection(sw2.getEast());
         if (rectSize(r)){
    spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);}}
         ///////sw1.south and sw2.North
      if (sw1.getSouth().intersects(sw2.getNorth()))
      {  Rectangle r = sw1.getSouth().intersection(sw2.getNorth());
         if (rectSize(r)){
    spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);}}
        ///////sw1.south and sw2.west
      if (sw1.getSouth().intersects(sw2.getWest()))
      {  Rectangle r = sw1.getSouth().intersection(sw2.getWest());
         if (rectSize(r)){
   spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);}}
         ///////sw1.east and sw2.south
      if (sw1.getEast().intersects(sw2.getSouth()))
```

```
      {  Rectangle r = sw1.getEast().intersection(sw2.getSouth());
         if (rectSize(r)){
     spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);}    }
         ///////sw1.east and sw2.north
      if (sw1.getEast().intersects(sw2.getNorth()))
      {  Rectangle r = sw1.getEast().intersection(sw2.getNorth());
         if (rectSize(r)){
     spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);} }
         ///////sw1.east and sw2.west
      if (sw1.getEast().intersects(sw2.getWest()))
      {  Rectangle r = sw1.getEast().intersection(sw2.getWest());
         if (rectSize(r)){
     spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);}}
         ///////sw1.north and sw2.south
      if (sw1.getNorth().intersects(sw2.getSouth()))
      {  Rectangle r = sw1.getNorth().intersection(sw2.getSouth());
         if (rectSize(r)){
     spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);}    }
         ///////sw1.north and sw2.east
      if (sw1.getNorth().intersects(sw2.getEast()))
      {  Rectangle r = sw1.getNorth().intersection(sw2.getEast());
         if (rectSize(r)){
     spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);} }
         ///////sw1.north and sw2.west
      if (sw1.getNorth().intersects(sw2.getWest()))
      {  Rectangle r = sw1.getNorth().intersection(sw2.getWest());
         if (rectSize(r)){
     spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);}  }
          ///////sw1.West and sw2.south
      if (sw1.getWest().intersects(sw2.getSouth()))
      {  Rectangle r = sw1.getWest().intersection(sw2.getSouth());
         if (rectSize(r)){
     spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);}  }
         ///////sw1.West and sw2.East
      if (sw1.getWest().intersects(sw2.getEast()))
      {  Rectangle r = sw1.getWest().intersection(sw2.getEast());
         if (rectSize(r)){
     spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);}   }
         ///////sw1.West and sw2.North
      if (sw1.getWest().intersects(sw2.getNorth()))
      {  Rectangle r = sw1.getWest().intersection(sw2.getNorth());
         if (rectSize(r)){
     spaceObject so = new spaceObject(r, sw1.getMe(), sw2.getMe(),botY,topY);
         addSpaceObject(so);}    }
     }
}
public static void addSpaceObject(spaceObject so
                                        )//groups the space objects into groups
{
   spBlock thisSpace = so.getSpaceBlock();
   boolean joinedGroup = false;
   groupObject firstIntersect = null;
   Enumeration a = spaces.elements();// all the groupObjects
   while(a.hasMoreElements())//while there are more groupObjects
   {
      groupObject grp = (groupObject)a.nextElement();
      Vector conn = grp.getConnected();
      Enumeration existSpaceO = conn.elements();
      while(existSpaceO.hasMoreElements())
                          // go through all the spaceObjects in this groupObject
      {
         spaceObject nextSO = (spaceObject)existSpaceO.nextElement();
         boolean overlap = blocksOverlap(thisSpace, nextSO.getSpaceBlock());
         if(overlap && !joinedGroup)
            // if this space Object has not yet joined a group, but overlaps
         {     // a space object in this group, add this space object to this group
            grp.addVolume(so);
            joinedGroup = true;
            firstIntersect = grp;
            break;
         }
         else if(overlap && joinedGroup)
                      // if this space Object has joined a group, and i come accross
         {    // another group with a space object that intersects,
                          //i need to  copy all this groups space
            Enumeration deadGroup = grp.getConnected().elements();
                                        //object into the group which owns this
```

```
 // spaceobject (firstIntersect), and remove the emptied groupObject from spaces
while(deadGroup.hasMoreElements())
              {
                spaceObject fromDead = (spaceObject)deadGroup.nextElement();
                firstIntersect.addVolume((spaceObject)fromDead);
              }
              spBlock keyBlock = grp.getKey();
              if(keyBlock.getLocallyOwned())
                     { keyBlock.remove(0);
                     }else
           { spOMI.makeStaticCall("onDeleteTree", "tdEditor", (spBlock)keyBlock);}
              spaces.removeElement(grp);
              schmoozer.WMupdate();
              break;
           }
        }
     }
     if(!joinedGroup)// if, after all that i still haven't joined a group, it's a new space.
     {
       spaces.addElement(new groupObject(so));
     }
   }
   public static class spatialWall
   {
     private spBlock me;
     private Rectangle south;
     private Rectangle east;
     private Rectangle west;
     private Rectangle north;
     public int wallSWX;
     public int wallSWZ;
     public int wallNEX;
     public int wallNEZ;
     public int wallbotY;
     public int walltopY;
     boolean ewWall;
     boolean nsWall;
     boolean slab;
     boolean column;
     public Vector associated;

     public spatialWall(spBlock thisBlock)
     {
       me = thisBlock;
       wallSWX = (int)me.getSWbot().pt[spTransformX];
       wallSWZ =(int)Math.abs(me.getSWbot().pt[spTransformZ]);
       wallNEX = (int)me.getNEbot().pt[spTransformX];
       wallNEZ = (int)Math.abs(me.getNEbot().pt[spTransformZ]);
       wallbotY = (int)me.getSWbot().pt[spTransformY];
       walltopY = (int)me.getSWtop().pt[spTransformY];
       if((wallNEX - wallSWX) < (wallNEZ - wallSWZ))
          { nsWall = true;  ewWall = false;}
       else if((wallNEX - wallSWX) > (wallNEZ - wallSWZ))
          { ewWall = true;  nsWall = false;}
          else { ewWall = false;  nsWall = false;}
       if(((walltopY - wallbotY) < (wallNEZ - wallSWZ))
          && ((walltopY - wallbotY) < (wallNEX - wallSWX)))
          slab = true;
       else if(((walltopY - wallbotY) > (wallNEZ - wallSWZ))
          && ((walltopY - wallbotY) > (wallNEX - wallSWX)))
          column = true;

       int southPointZ = (wallSWZ - (walltopY - wallbotY) < 0) ? 0 :
                                        (wallSWZ - (walltopY - wallbotY));
       int westPointX = ((wallSWX - (walltopY - wallbotY)) < 0) ? 0 :
                                        (wallSWX - (walltopY - wallbotY));
       south = new Rectangle(wallSWX, southPointZ, (wallNEX - wallSWX),
                                        wallSWZ - southPointZ);
       east = new Rectangle(wallNEX, wallSWZ, (walltopY - wallbotY),
                                        (wallNEZ - wallSWZ));
       north = new Rectangle(wallSWX, wallNEZ, (wallNEX - wallSWX),
                                        (walltopY - wallbotY));
       west = new Rectangle(westPointX, wallSWZ, (wallSWX - westPointX),
                                        (wallNEZ - wallSWZ));
     }

     public Rectangle getSouth() { return south;}
     public Rectangle getWest() { return west;}
     public Rectangle getEast() { return east;}
     public Rectangle getNorth() { return north;}
     public spBlock getMe() {return me;}
   }
```

```
//////////////LIGHT/////////////////////////////////////////////////////////////////
  public static Vector lightObjects;
                              // a vector of spaceObjects (a space block and two walls)
  public static boolean lightInited = false;

  public static Vector selectLightData()
  {
     if (lightInited)
     {lightObjects.removeAllElements();
     }
     else
     {lightObjects = new Vector();
     lightInited = true;}
     if(((tdAvatarPlatform.getAvatarsLocale().
                                        getTitle()).compareTo("spaceFilter") != 0))
     {
        perceive("spaceFilter");
     }
    Vector lightObjs = spaces;//copies all the space groupObjects
   // goToFilter("lightFilter");
     return lightObjs;
  }
  public static boolean constructLightView(Vector lightObjs)
  {
     Vector refWallMade = new Vector();
     Vector newWallMade = new Vector();
     Enumeration g =lightObjs.elements();//take the group objects
     while (g.hasMoreElements())
     {
       groupObject go = (groupObject)g.nextElement();
       float groupArea = go.getTotalArea();
       Enumeration s = go.getConnected().elements();
                                          //look at each spaceObject
       while(s.hasMoreElements())
                              // this code goes through each space Object, if there is
       {          //and counts how many concrete and glass in each spaceObject
         int concCount = 0;
         spaceObject so = (spaceObject)s.nextElement();
         spBlock wall1 = so.getBlock1();
         spBlock wall2 = so.getBlock2();
         float wall1Area = wall1.getX() * wall1.getY() * wall1.getZ();
         float wall2Area = wall2.getX() * wall2.getY() * wall2.getZ();
         spBlock wallBlock1 = null;
         spBlock wallBlock2 = null;
         if(wall1.getMaterial() == 1)
         {
          if(!refWallMade.contains(wall1))
                              //just making sure the wall isn't made twice
          {
            wallBlock1 = new spBlock(wall1);
            wallBlock1.setReferenced(wall1.getReferenced());
            tdx.spDonateToLocale(wallBlock1);
            assignNormalGlas(wallBlock1);
            refWallMade.addElement(wall1);
            newWallMade.addElement(wallBlock1);
          }
          else
          {
          wallBlock1 = (spBlock)newWallMade.
                              elementAt(refWallMade.lastIndexOf(wall1));
          }
          //glasArea = glasArea + wall1Area;//add
         }
         else if(wall1.getMaterial() == 0)
         {
          if(!refWallMade.contains(wall1))
          {
            wallBlock1 = new spBlock(wall1);
            wallBlock1.setReferenced(wall1.getReferenced());
            tdx.spDonateToLocale(wallBlock1);
            assignNormalConc(wallBlock1);
            refWallMade.addElement(wall1);
            newWallMade.addElement(wallBlock1);
          }
          else
          {
          wallBlock1 = (spBlock)newWallMade.
                              elementAt(refWallMade.lastIndexOf(wall1));
          }
          //concArea = concArea + wall1Area;
          concCount++;
         }
         else
         {
            wallBlock1 = new spBlock(wall1);
            wallBlock1.setReferenced(wall1.getReferenced());
            tdx.spDonateToLocale(wallBlock1);
            assignNormalConc(wallBlock1);
        }
        if(wall2.getMaterial() == 1)
        {
          if(!refWallMade.contains(wall2))
          {
            wallBlock2 = new spBlock(wall2);
            wallBlock2.setReferenced(wall2.getReferenced());
            tdx.spDonateToLocale(wallBlock2);
            assignNormalGlas(wallBlock2);
            refWallMade.addElement(wall2);
            newWallMade.addElement(wallBlock2);
          }
          else
          {
  wallBlock2 = (spBlock)newWallMade.elementAt(refWallMade.lastIndexOf(wall2));
          }
          //glasArea = glasArea + wall2Area;
        }

        else if(wall2.getMaterial() == 0)
        {
          if(!refWallMade.contains(wall2))
          {
            System.out.println("constructLightView 20");
            wallBlock2 = new spBlock(wall2);
            wallBlock2.setReferenced(wall2.getReferenced());
            tdx.spDonateToLocale(wallBlock2);
            assignNormalConc(wallBlock2);
            refWallMade.addElement(wall2);
            newWallMade.addElement(wallBlock2);
            System.out.println("constructLightView 21");
          }
          else
          {
 wallBlock2 = (spBlock)newWallMade.elementAt(refWallMade.lastIndexOf(wall2));
          }
          //concArea = concArea + wall2Area;
          concCount++;
        }
        else
        {
            wallBlock2 = new spBlock(wall2);
            wallBlock2.setReferenced(wall2.getReferenced());
            tdx.spDonateToLocale(wallBlock2);
            assignNormalConc(wallBlock2);
        }
spBlock spaceBlock = new spBlock(so.getSpaceBlock());
                                        //no value in setting reference here
        tdx.spDonateToLocale(spaceBlock);
        if(concCount == 0)
        {
          assignSelectedGlas(spaceBlock); // it's all glas
        }
        else if(concCount == 1)
        {
          assignyellow50(spaceBlock);//it's half and half
        }
        else if(concCount == 2)
        {
          assignred50(spaceBlock);
        }
        spaceObject lo = new spaceObject(spaceBlock, wallBlock1, wallBlock2);
        lightObjects.addElement(lo);//Note: I am only dealing with spaceObjects
                              //light loses the notion of groups of spaces
      }
    }
    return true;
  }
  public static Vector structureObjects;
  public static Vector couldSupport;
  public static boolean structInited = false;
```

```
/////////////STRUCTUREX/////////////////////////////////////////////////////////////////
  //Structure takes all the elemetns above 1' and constructs a groupObject with
  //all the pieces that support it.  It also collects all the pieces touching the ground
  //and constructs a vector of these called "could support
   public static Vector selectStructureData() // this phase just eliminates glass walls
  {
    if (structInited)
    {structureObjects.removeAllElements();
    couldSupport.removeAllElements();
    }
    else
    {structureObjects = new Vector();
    couldSupport = new Vector();
    structInited = true;}
    Vector needSupport = new Vector();
    spSchmoozerLocale thisLocale = tdAvatarPlatform.getAvatarsLocale();
    Vector localeObj = tdx.spDescendants(thisLocale);
                              //put all the objects in the locale into a vector
    Enumeration d =localeObj.elements();//enumeration of all these objects
    while (d.hasMoreElements())
    {
      Object t = d.nextElement();
      if (t instanceof spBlock) //if it's a block
      {
        spBlock b = (spBlock)t;
         if (b.getSWbot().pt[spTransformY] > 1) //if it's not resting on the ground
         {  needSupport.addElement(b); }
         else  //or if it's concrete it might be good for support
         {  couldSupport.addElement(b); }
      }
    }
    return needSupport;
  }
    static int beamSWX;//note: these are used in blocksIntersect4(),
                              //but are defined in constructStructureView
    static int beamSWZ;
    static int beamNEX;
    static int beamNEZ;
    static int beambotY;
    static int beamMidX;// X the point in the center of the box
    static int beamMidZ;// Z the  Point at the center
 public static boolean constructStructureView(Vector need)
                              //copies over need vector, could is held statically
 {
    lastBeam = null; //hack to initialize lastbeam, used in blocksIntersect4
    Vector needSupport = need;//
    Enumeration s = couldSupport.elements();
                              //enumeration of the selected could support blocks
    Vector holder = new Vector();
        //holds the newly created blocks to be transfered back into could support
    while (s.hasMoreElements())
    {
      spBlock cl = (spBlock) s.nextElement();
      spBlock newB = new spBlock(cl);
 newB.setReferenced(cl);//try to keep a reference to the objectin the world locale,
      tdx.spDonateToLocale(newB);
      if(newB.getMaterial() == 0)// takes the glas out of consideration
      {
        assignyellow50(newB);
        holder.addElement(newB);
      }
      else assignNormalGlas(newB);
    }
   couldSupport = holder;
    Enumeration n = needSupport.elements();
                                        //enumeration of the selected beams
    while (n.hasMoreElements())
    {
      spBlock nextBeam = (spBlock)n.nextElement();
      spBlock beam = new spBlock(nextBeam);
      beam.setReferenced(nextBeam);
      tdx.spDonateToLocale(beam);
      beamSWX = (int)beam.getSWbot().pt[spTransformX];
      beamSWZ =(int)Math.abs(beam.getSWbot().pt[spTransformZ]);
      beamNEX = (int)beam.getNEbot().pt[spTransformX];
      beamNEZ = (int)Math.abs(beam.getNEbot().pt[spTransformZ]);
      beambotY = (int)beam.getSWbot().pt[spTransformY];
      beamMidX = beamSWX + ((beamNEX - beamSWX)/2);
                                        // X the point in the center of the box
      beamMidZ = beamSWZ + ((beamNEZ - beamSWZ)/2);
                                        // Z the  Point at the center
      Enumeration c = couldSupport.elements();
      boolean plan1 = false;
      boolean plan2 = false;
      boolean plan3 = false;
```

```
        boolean plan4 = false;
        Vector columns = new Vector();
        while (c.hasMoreElements())
// goes through all the could supports, and attaches it to seat if it intersects
        {
          spBlock col = (spBlock)c.nextElement();
            //note, i could construct meaning objects here
          boolean[] spt = blocksIntersect4(beam, col);
          if (spt != null)
          {
          if(spt[0]) plan1 = true;//these track over all the possible supports
          if(spt[1]) plan2 = true;
          if(spt[2]) plan3 = true;
          if(spt[3]) plan4 = true;
          if(spt[0] | spt[1] | spt[2] | spt[3]) columns.addElement(col);
          }
        }
        if (plan3 && plan1)
          assignNormalConc(beam);
        else if (plan2 && plan4)
          assignNormalConc(beam);
        else if (plan1 | plan2 | plan3 | plan4)
          assignred50(beam);//one, or multiple lopsided supports
        else
          assignred100(beam);//no support
        boolean[] supported = new boolean[4];
        supported[0] = plan1;
        supported[1] = plan2;
        supported[2] = plan3;
        supported[3] = plan4;
        groupObject beamGroup = new groupObject(beam, supported, columns);
        structureObjects.addElement(beamGroup);
      }
      return true;
    }
    public static boolean enclosureInited = false;
    public static Vector  enclosureObjs;
    /////////////ENCLOSUREX/////////////////////////////////////////////////////////////
    //this code takes the spaceObjects, reduces them to just the space Blocks
    //then takes the roof blocks (determined during space), and constructs
    //a series of group objects where each space has a series of overlapping roofs
     public static Vector selectEnclosureData()
    {

       if(enclosureInited) {enclosureObjs.removeAllElements();}
       else {enclosureObjs = new Vector();
           enclosureInited = true;}
    if(tdAvatarPlatform.getAvatarsLocale().getTitle().compareTo("spaceFilter") != 0)
       { perceive("spaceFilter");}

     Vector selectedSpaces = new Vector();
     Enumeration go = spaces.elements();
     while(go.hasMoreElements())//go through all the group objects
     {
       groupObject g = (groupObject)go.nextElement();
       Vector con = g.getConnected();
       Enumeration so = con.elements();
       while(so.hasMoreElements())//go through each group objects space objects
       {
         spaceObject s = (spaceObject)so.nextElement();
         spBlock sp = (spBlock) s.getSpaceBlock();
         selectedSpaces.addElement(sp);
       }
     }
    return selectedSpaces;
// simply has the space Blocks, i pick up the roofs through the static vector roofs
    }
    public static boolean constructEnclosureView(Vector selectedBlocks)
    {
       Enumeration e = selectedBlocks.elements();
       Vector encSpaces = new Vector();
       while (e.hasMoreElements())
              // make all the new space blocks from the old space blocks
       {
         spBlock old = (spBlock)e.nextElement();
         spBlock spaceBlock = new spBlock(old);
         spaceBlock.setReferenced(old.getReferenced());
         tdx.spDonateToLocale(spaceBlock);
         assignred50(spaceBlock);
         encSpaces.addElement(spaceBlock);
       }
       Enumeration r = roofs.elements();//make all the roofs
       Vector encRoofs = new Vector();
       while(r.hasMoreElements())
       {
```

```
         spBlock old = (spBlock)r.nextElement();
         spBlock roofBlock = new spBlock(old);
         roofBlock.setReferenced(old.getReferenced());
         tdx.spDonateToLocale(roofBlock);
         assigngreen50(roofBlock);
         encRoofs.addElement(roofBlock);
       }

       Enumeration es = encSpaces.elements();
       while(es.hasMoreElements())// go through all the spaces
       {
          spBlock nextSpace = (spBlock)es.nextElement();//look at the next space
          Vector intRoofs = new Vector();
          int happiness = 0;
          int nextSpaceSWX = (int)nextSpace.getSWtop().pt[spTransformX];
       int nextSpaceSWZ = (int)Math.abs(nextSpace.getSWtop().pt[spTransformZ]);
          int nextSpaceNEX = (int)nextSpace.getNEtop().pt[spTransformX];
       int nextSpaceNEZ = (int)Math.abs(nextSpace.getNEtop().pt[spTransformZ]);
       int nextSpaceMidX = nextSpaceSWX + (nextSpaceNEX - nextSpaceSWX)/2;
       int nextSpaceMidZ = nextSpaceSWZ + (nextSpaceNEZ - nextSpaceSWZ)/2;
          int nextSpacetopY = (int)nextSpace.getSWtop().pt[spTransformY];

          Point pSW = new Point(nextSpaceSWX, nextSpaceSWZ);
                            //create 9 points, if these are covered
          Point pSM = new Point(nextSpaceMidX, nextSpaceSWZ);
          Point pSE = new Point(nextSpaceNEX, nextSpaceSWZ);
          Point pMW = new Point(nextSpaceSWX, nextSpaceMidZ);
          Point pMM = new Point(nextSpaceMidX, nextSpaceMidZ);
          Point pME = new Point(nextSpaceNEX, nextSpaceMidZ);
          Point pNW = new Point(nextSpaceSWX, nextSpaceNEZ);
          Point pNM = new Point(nextSpaceMidX, nextSpaceNEZ);
          Point pNE = new Point(nextSpaceNEX, nextSpaceNEZ);
          Point [] spacePoints = new Point[9];
          spacePoints[0] = pSW;
          spacePoints[1] = pSM;
          spacePoints[2] = pSE;
          spacePoints[3] = pMW;
          spacePoints[4] = pMM;
          spacePoints[5] = pME;
          spacePoints[6] = pNW;
          spacePoints[7] = pNM;
          spacePoints[8] = pNE;
          boolean[] spacePInt = new boolean[9];
                            //vector representing which spacePoints are covered
          for(int i = 0; i < 9; i++)
          {spacePInt[i] = false;}

          Enumeration er = encRoofs.elements();
          while(er.hasMoreElements())//while there are more potential roofs
          {
             spBlock nextRoof = (spBlock)er.nextElement();
             int nextRoofSWX = (int)nextRoof.getSWtop().pt[spTransformX];
        int nextRoofSWZ = (int)Math.abs(nextRoof.getSWtop().pt[spTransformZ]);
             int nextRoofNEX = (int)nextRoof.getNEtop().pt[spTransformX];
         int nextRoofNEZ = (int)Math.abs(nextRoof.getNEtop().pt[spTransformZ]);
               Rectangle roofRect = new Rectangle(nextRoofSWX, nextRoofSWZ,
                  (nextRoofNEX - nextRoofSWX), (nextRoofNEZ - nextRoofSWZ));
             for(int n = 0; n < 9; n++)
             {
                if(roofRect.contains(spacePoints[n]))
                            //if the roof rectangle intersects any of the points
                {
                   spacePInt[n] = true;    //this point is covered
                   intRoofs.addElement(nextRoof);
                }
             }
          }
   groupObject encObj = new groupObject(nextSpace,spacePoints,
                                         spacePInt, intRoofs);
          for(int p = 0; p < 9; p++)
          enclosureObjs.addElement(encObj);
       }
       return true;
    }
    public static Vector chairObjs;
    public static Vector possibleLegs;
    public static boolean chairInited = false;
   /////////////CHAIR/////////////////////////////////////////////////////////////
 //produces a set of chairObjs, which are group Objects with the seat, and a vector
//of linked legs.  It also has a boolean array which tells where it needs support
//possible Legs keeps a pointer to all the possible legs,
// so that a new chair use old legs
    public static Vector selectChairData()
   {
      Vector seats = new Vector();
      if(chairInited) {chairObjs.removeAllElements();
                  possibleLegs.removeAllElements();}
      else {chairObjs = new Vector();
            possibleLegs = new Vector();
            chairInited = true;}
 if(tdAvatarPlatform.getAvatarsLocale().getTitle().compareTo("structureFilter") == 0)
      {}
      else
      { perceive("structureFilter");}//create a structure view first
 Enumeration pos = structureObjects.elements();//enumeration of all these objects
      while (pos.hasMoreElements())
      {   groupObject grp = (groupObject)pos.nextElement();
          spBlock posSeat = grp.getKey();
 if((posSeat.getNEtop().pt[spTransformY] > 1) &&
                                        (posSeat.getNEtop().pt[spTransformY] < 4))
         {
              seats.addElement(grp);
              //returns the entire group object so i can use getConnected
         }
      }
   return seats;// this is an groupObject
   }
   public static boolean constructChairView(Vector selectedSeats)
   {
      Enumeration c = couldSupport.elements();
              //go through structures possible legs & make new ones for chair
      while(c.hasMoreElements())
      {
         spBlock lg = (spBlock)c.nextElement();
         spBlock leg = new spBlock(lg);//leg is created once here
         leg.setReferenced(lg.getReferenced());
         tdx.spDonateToLocale(leg);
         filter.assignyellow50(leg);
         possibleLegs.addElement(leg);
      }
      Enumeration s = selectedSeats.elements();// the possible chair seats
      while(s.hasMoreElements())  //while there are more possible seats
      {
         boolean chair = true;
         groupObject ch = (groupObject)s.nextElement();
         spBlock seat = ch.getKey();
         boolean[] supported = ch.getZoneSupports();


         Vector positions = possibleSeats(seat);//creates little simulated people
         Vector legs = ch.getConnected();
                              //structure has determined which blocks touch the seat
         Enumeration p = positions.elements();
         while(p.hasMoreElements())//go through all the positions on a seat
         {
           spBlock pos = (spBlock) p.nextElement();
           Enumeration o = legs.elements();
           while(o.hasMoreElements())//go through all the touching elements
           {
              spBlock leg = (spBlock)o.nextElement();
              if(blocksOverlap(pos, leg))
              {  chair = false;
                 break;// can't be sat in go to next simulated position
              }
              else
              {
                 chair = true;
              }
           }
           if(chair)
           {
           break;
//if i get to hear and chair is true, it means i've found n
// empty position, no need to look further
           }
         }
         if(chair)
         {
           spBlock chairSeat = new spBlock(seat);// seat is constructed once here
           tdx.spDonateToLocale(chairSeat);
           chairSeat.setReferenced(seat.getReferenced());
```

```
           if(supported != null)
           {
              if (supported[0] && supported[2])
              assignNormalConc(chairSeat);
              else if (supported[1] && supported[3])
              assignNormalConc(chairSeat);
              else if (supported[0] | supported[1] | supported[2] | supported[3])
              assignred50(chairSeat);//one, or multiple lopsided supports
              else
              assignred100(chairSeat);//no support
           }
           else
           assignred100(chairSeat);//no support

           Vector parts = new Vector();//my changes here 4/19
           Enumeration oo = possibleLegs.elements();
           while(oo.hasMoreElements())
           {
              spBlock part = (spBlock)oo.nextElement();
              if (blocksOverlap(chairSeat, part))
              //going through all the parts to see what i should attach to the seat
              {
                 parts.addElement(part);
                 filter.assignyellow100(part);
              }
           }
           chairObjs.addElement(new groupObject(parts, chairSeat, supported));
         }
      }
      if(chairObjs.isEmpty())
      {
         //deleteAllBlocks("structureFilter");
         return false;
      }

      else {return true;}
   }


public static Vector possibleSeats(spBlock seat)
{
   if((seat.getX() < 2.0) | (Math.abs(seat.getZ()) < 2.0))
   {
     return null;  //block is two thin to be a seat
   }
   else
   {
      float SWX = seat.getSWtop().pt[spTransformX];
      float SWZ = seat.getSWtop().pt[spTransformZ];
      float NEX = seat.getNEtop().pt[spTransformX];
      float NEZ = seat.getNEtop().pt[spTransformZ];
      float topY = seat.getSWtop().pt[spTransformY];
      float botY = seat.getSWbot().pt[spTransformY];

      spBlock template = new spBlock(seat);
      template.setX((float) 2.0);
      template.setZ((float) 2.0);
      template.setY((float) 4.0);
      tdPoint tempPoint = template.getSWbot().copy();
      tempPoint.pt[spTransformY] = topY;
                              //creates a block of the right size, and puts it at SWtop
      Vector people = new Vector();
      spBlock smallX = new spBlock(template);
                              //creates a copy of template to be used in x direction
      spBlock smallZ = new spBlock(template);
                              //creates a copy of template to be used in z direction
      spBlock largeX = new spBlock(template);
                              //creates a copy of template and moves it to NW
      spBlock largeZ = new spBlock(template);
                              //creates a copy of template and moves it to SE

      tdPoint largeXPoint = template.getSWbot().copy();
      largeXPoint.pt[spTransformZ] = NEZ + (float)2.0;
      largeX.setTransform(largeXPoint);

      tdPoint largeZPoint = template.getSWbot().copy();
      largeZPoint.pt[spTransformX] = NEX - (float)2.0;
      largeZ.setTransform(largeZPoint);

      spBlock smXmove = new spBlock(smallX);//these are redundant but ....
      spBlock lgXmove = new spBlock(largeX);
      spBlock smZmove = new spBlock(smallZ);
      spBlock lgZmove = new spBlock(largeZ);

while(smXmove.getNEtop().pt[spTransformX] < NEX)//place sblocks along x axis
```

```
         {
            people.addElement(smXmove);
            people.addElement(lgXmove);
            smXmove = new spBlock(smXmove);
            smXmove.setXpos(smXmove.getXpos() + (float)2.0);
            lgXmove = new spBlock(lgXmove);
            lgXmove.setXpos(lgXmove.getXpos() + (float)2.0);
            //tdx.spDonateToLocale(lgXmove);
            //tdx.spDonateToLocale(smXmove);
            //assignred100(lgXmove);
            //assigngreen100(smXmove);



         }
      while(smZmove.getNEtop().pt[spTransformZ] > NEZ)//places blocks along z axis
         {
            people.addElement(smZmove);
            people.addElement(lgZmove);
            smZmove = new spBlock(smZmove);
            smZmove.setZpos(smZmove.getZpos() - (float)2.0);
            lgZmove = new spBlock(lgZmove);
            lgZmove.setZpos(lgZmove.getZpos() - (float)2.0);
            //tdx.spDonateToLocale(lgZmove);
            //tdx.spDonateToLocale(smZmove);
            //assignred100(lgZmove);
            //assigngreen100(smZmove);
         }
         return people;// a vector of lots of blocks that simulate where peeople might sit
   }
 }
public static boolean blocksOverlap(spBlock blk1, spBlock blk2)
{
      int blk1SWX = (int)blk1.getSWtop().pt[spTransformX];
      int blk1SWZ = (int)Math.abs(blk1.getSWtop().pt[spTransformZ]);
      int blk1NEX = (int)blk1.getNEtop().pt[spTransformX];
      int blk1NEZ = (int)Math.abs(blk1.getNEtop().pt[spTransformZ]);
      int blk1topY = (int)blk1.getSWtop().pt[spTransformY];
      int blk1botY = (int)blk1.getSWbot().pt[spTransformY];

      int blk2SWX = (int)blk2.getSWtop().pt[spTransformX];
      int blk2SWZ = (int)Math.abs(blk2.getSWtop().pt[spTransformZ]);
      int blk2NEX = (int)blk2.getNEtop().pt[spTransformX];
      int blk2NEZ = (int)Math.abs(blk2.getNEtop().pt[spTransformZ]);
      int blk2topY = (int)blk2.getSWtop().pt[spTransformY];
      int blk2botY = (int)blk2.getSWbot().pt[spTransformY];

      Rectangle blk1Plan = new Rectangle(blk1SWX, blk1SWZ,
                              (blk1NEX - blk1SWX),(blk1NEZ - blk1SWZ));
      Rectangle blk2Plan = new Rectangle(blk2SWX, blk2SWZ,
                              (blk2NEX - blk2SWX),(blk2NEZ - blk2SWZ));

      if (blk1Plan.intersects(blk2Plan))      //sees if plans intersect
      {
         if((blk1topY > blk2botY) && (blk2topY > blk1botY))
         {
            return true;
         }
      }
      return false;
}
public static boolean blk1OverBlk2(spBlock blk1, spBlock blk2)/
                                                         / tests for partial cover
{
      int blk1SWX = (int)blk1.getSWtop().pt[spTransformX];
      int blk1SWZ = (int)Math.abs(blk1.getSWtop().pt[spTransformZ]);
      int blk1NEX = (int)blk1.getNEtop().pt[spTransformX];
      int blk1NEZ = (int)Math.abs(blk1.getNEtop().pt[spTransformZ]);
      int blk1topY = (int)blk1.getSWtop().pt[spTransformY];
      int blk1botY = (int)blk1.getSWbot().pt[spTransformY];

      int blk2SWX = (int)blk2.getSWtop().pt[spTransformX];
      int blk2SWZ = (int)Math.abs(blk2.getSWtop().pt[spTransformZ]);
      int blk2NEX = (int)blk2.getNEtop().pt[spTransformX];
      int blk2NEZ = (int)Math.abs(blk2.getNEtop().pt[spTransformZ]);
      int blk2topY = (int)blk2.getSWtop().pt[spTransformY];
      int blk2botY = (int)blk2.getSWbot().pt[spTransformY];

      Rectangle blk1Plan = new Rectangle(blk1SWX, blk1SWZ,
                              (blk1NEX - blk1SWX),(blk1NEZ - blk1SWZ));
      Rectangle blk2Plan = new Rectangle(blk2SWX, blk2SWZ,
                              (blk2NEX - blk2SWX),(blk2NEZ - blk2SWZ));

      if (blk1Plan.intersects(blk2Plan))      //sees if plans intersect
      {
         if(blk1topY > blk2topY)
```

```
         {
            return true;
         }
      }
      return false;
}
///////////////WORLD - to catch errors, and to import data/////////////////////////
   public static Vector selectWorldData()
   {
      Vector selectedBlocks = new Vector();
      spSchmoozerLocale thisLocale = tdAvatarPlatform.getAvatarsLocale();
      Vector localeObj = tdx.spDescendants(thisLocale);
      Enumeration d =localeObj.elements();
      while (d.hasMoreElements())
      {
         Object t = d.nextElement();
         if (t instanceof spBlock)//note, here is where select test should go
         {
          selectedBlocks.addElement((spBlock)t);
         }
      }
      return selectedBlocks;
   }
   public static boolean constructWorldView(Vector selectedBlocks)
   {
      Enumeration e =selectedBlocks.elements();
      while (e.hasMoreElements())
      {
         Object t = e.nextElement();
          if (t instanceof spBlock)
               {
                  spBlock newBlock = new spBlock((spBlock) t);
                  tdx.spDonateToLocale(newBlock);
                  if (newBlock.getMaterial() == 0)
                  {
                     newBlock.setVisualDefinition(tdAvatarPlatform.
                           getAvatarsLocale().getSchmoozerLocaleVDefNorConc());
                  }
                  else if (newBlock.getMaterial() == 1)
                  {
                     newBlock.setVisualDefinition(tdAvatarPlatform.
                           getAvatarsLocale().getSchmoozerLocaleVDefNorGlas());
                  }
               }
      }
      return true;
   }
   ////Material Assignment////////////
   public static void assignMaterial(spBlock newBlock)
   {
      if (newBlock.getMaterial() == 0){
         if (newBlock.getLocallyOwned())
           newBlock.setVisualDefinition(tdAvatarPlatform.
               getAvatarsLocale().getSchmoozerLocaleVDefgreen100());
         else spOMI.makeStaticCall(“setVDtogreen100”, “tdBlockEditor”,
newBlock);}
      else if (newBlock.getMaterial() == 1){
         if (newBlock.getLocallyOwned())
            newBlock.setVisualDefinition(tdAvatarPlatform.
                           getAvatarsLocale().getSchmoozerLocaleVDefred50());
         else spOMI.makeStaticCall(“setVDtored50”, “tdBlockEditor”, newBlock);}
   }
   public static void assigngreen50(spBlock newBlock)
   {  if (newBlock.getLocallyOwned())
      newBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale().
                           getSchmoozerLocaleVDefgreen50());
      else  spOMI.makeStaticCall(“setVDtogreen50”, “tdBlockEditor”, newBlock); }

   public static void assigngreen100(spBlock newBlock)
   {  if (newBlock.getLocallyOwned())
      newBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale().
                                    getSchmoozerLocaleVDefgreen100());
      else  spOMI.makeStaticCall(“setVDtogreen100”, “tdBlockEditor”, newBlock); }

   public static void assignred50(spBlock newBlock)
   {  if (newBlock.getLocallyOwned())
      newBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale().
                                    getSchmoozerLocaleVDefred50());
      else  spOMI.makeStaticCall(“setVDtored50”, “tdBlockEditor”, newBlock); }

   public static void assignred100(spBlock newBlock)
   {  if (newBlock.getLocallyOwned())
      newBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale().
                                    getSchmoozerLocaleVDefred100());
      else  spOMI.makeStaticCall(“setVDtored100”, “tdBlockEditor”, newBlock); }
   public static void assignyellow50(spBlock newBlock)
   {  if (newBlock.getLocallyOwned())
      newBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale().
                              getSchmoozerLocaleVDefyellow50());
      else  spOMI.makeStaticCall("setVDtoyellow50", "tdBlockEditor", newBlock); }

   public static void assignyellow100(spBlock newBlock)
   {  if (newBlock.getLocallyOwned())
      newBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale().
                                             getSchmoozerLocaleVDefyellow100());
      else  spOMI.makeStaticCall("setVDtoyellow100", "tdBlockEditor", newBlock); }

   public static void assignNormalConc(spBlock newBlock)
   {  if (newBlock.getLocallyOwned())
      newBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale().
                              getSchmoozerLocaleVDefNorConc());
      else  spOMI.makeStaticCall("setVDtoNormalConc", "tdBlockEditor",
newBlock); }

   public static void assignSelectedConc(spBlock newBlock)
   {  if (newBlock.getLocallyOwned())
      newBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale().
                                             getSchmoozerLocaleVDefSelConc());
else  spOMI.makeStaticCall("setVDtoSelectedConc", "tdBlockEditor", newBlock); }

   public static void assignNormalGlas(spBlock newBlock)
   {  if (newBlock.getLocallyOwned())
      newBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale().
                                             getSchmoozerLocaleVDefNorGlas());
 else  spOMI.makeStaticCall("setVDtoNormalGlas", "tdBlockEditor", newBlock); }

   public static void assignSelectedGlas(spBlock newBlock)
   {  if (newBlock.getLocallyOwned())
      newBlock.setVisualDefinition(tdAvatarPlatform.getAvatarsLocale().
                              getSchmoozerLocaleVDefSelGlas());
 else  spOMI.makeStaticCall("setVDtoSelectedGlas", "tdBlockEditor", newBlock); }


   //some useful math functions
   static spBlock lastBeam;

   public static boolean[] blocksIntersect4(spBlock beam, spBlock col)
            //returns 0 if it doesn't intersect, 1 if in quad 1, 12 if in quad 1 and 2 etc.
   {
      int colSWX = (int)col.getSWtop().pt[spTransformX];
      int colSWZ = (int)Math.abs(col.getSWtop().pt[spTransformZ]);
      int colNEX = (int)col.getNEtop().pt[spTransformX];
      int colNEZ = (int)Math.abs(col.getNEtop().pt[spTransformZ]);
      int coltopY = (int)col.getSWtop().pt[spTransformY];
      int colMidX = colSWX + (colNEX - colSWX);
      int colMidZ = colSWZ + (colNEZ - colSWZ);
      Rectangle beamPlan = new Rectangle(beamSWX, beamSWZ,
                              (beamNEX - beamSWX),(beamNEZ - beamSWZ));
      Rectangle colPlan = new Rectangle(colSWX, colSWZ,
                              (colNEX - colSWX),(colNEZ - colSWZ));
      if (beamPlan.intersects(colPlan))//sees if there is any intersection at all
      {
         //note these get made over and over again, i need to fix this
         Rectangle beam1Plan = new Rectangle(beamSWX, beamSWZ,
                              (beamMidX - beamSWX),(beamMidZ - beamSWZ));
         Rectangle beam2Plan = new Rectangle(beamMidX, beamSWZ,
                              (beamNEX - beamMidX),(beamMidZ - beamSWZ));
         Rectangle beam3Plan = new Rectangle(beamMidX, beamMidZ,
                              (beamNEX - beamMidX),(beamNEZ - beamMidZ));
         Rectangle beam4Plan = new Rectangle(beamSWX, beamMidZ,
                              (beamMidX - beamSWX),(beamNEZ - beamMidZ));

         //plan view
      //____________. beamNEX,Z
      //|    |    |
      //| 4  |  3 |
      //|_____.beamMidX,Z
      //|    |    |
      //| 1  |  2 |
      // .beamSWX,Z

         boolean tallEnough = (beambotY <= coltopY) ;
         boolean[] answer = new boolean[4];
         answer[0] = (beam1Plan.intersects(colPlan)&& tallEnough)? true: false;
         answer[1] = (beam2Plan.intersects(colPlan)&& tallEnough)? true: false;
         answer[2] = (beam3Plan.intersects(colPlan)&& tallEnough)? true: false;
         answer[3] = (beam4Plan.intersects(colPlan)&& tallEnough)? true: false;
         lastBeam = beam;
         return answer;
      }
      else
      {lastBeam = beam;
      return null;}
   }
public static void deleteAllBlocks(String myFilter)
                                          // deletes all the blocks in a locale
  {
     Enumeration e = spLandingPad.getHash().elements();//find the beacon
     String ls = schmoozer.getPreference("localeServer");
                                 ls = "istp://" + ls + "/LandingPad/" + myFilter;
     while (e.hasMoreElements())
     {
        spLandingPad t = (spLandingPad) e.nextElement();
        if ((t.getTag().compareTo(ls)) == 0)
        {
           spSchmoozerLocale loc = (spSchmoozerLocale) t.getLocale();// i have
           if(loc == null)
           {
              String curLoc = tdAvatarPlatform.getAvatarsLocale().getTitle();
              goToFilter(myFilter);
              goToFilter(curLoc);
              loc = (spSchmoozerLocale) t.getLocale();
           }

              Enumeration b = tdx.spDescendants(loc).elements();
              while (b.hasMoreElements())
              {
                 sp s = (sp) b.nextElement();
                 if (s != null)
                 {if (s instanceof spBlock)
                    {
                        if(s.getLocallyOwned())
                        { s.remove(0);
                        }else
                   { spOMI.makeStaticCall("onDeleteTree", "tdEditor", (spBlock)s);}
                    }
                   schmoozer.WMupdate();
                 }
              }
           }
        }
    }
  }
}
```

public class groupObject

```
mport java.util.*;
import java.awt.*;
import java.lang.*;
public class groupObject
{
    private Vector connected;
        //heeps all the spaceObjects that are associated with this object
    private spBlock key;
    private String objectType;
    private float totalArea;
    private float overlapArea;
    private Vector alreadyDeducted;  //keeps previous intersections as rectangles
    private boolean[] zoneSupports;
    //for enclosure
    private Point[] keyPoints;
    private boolean[] covered;
    int amountCovered = 0;
    public groupObject(Vector legs, spBlock s, boolean[] b)//for Chair
    {
      totalArea = 0;
      zoneSupports = b;
      key = s;
      connected = new Vector();
      Enumeration l = legs.elements();
      while(l.hasMoreElements())
      {
        spBlock leg = (spBlock)l.nextElement();
        connected.addElement(leg);
      }
    }
    public groupObject(spaceObject so)// for space
    {
      totalArea = ((so.getSpacePlan().width) * (so.getSpacePlan().height));
                    //this is alt,ered with each added object
      connected = new Vector();
      connected.addElement(so);
      alreadyDeducted = new Vector();
      key = new spBlock(so.getSpaceBlock());
      key.setReferenced(null);
      key.setY(so.getSpaceBlock().getY()+4);
      key.setX((float)1);
      key.setZ((float)1);
      key.setMaterial(3);
      filter.assignred100(key);
      tdx.spDonateToLocale(key);
    }
    public groupObject(spBlock beam, boolean[] zSupp, Vector cols)//for structure
    {
      key = beam;
      connected = cols;
      zoneSupports = zSupp;
    }
                                      //for enclosure
    public groupObject(spBlock spaceBlock, Point[] spacePoints,
                                   boolean[] spacePInt, Vector intRoofs)
    {
      key = spaceBlock;
      connected = intRoofs;
      covered = spacePInt;
      keyPoints = spacePoints;
      for(int i = 0; i < 9; i++)
      {
        if(covered[i]) amountCovered = amountCovered + 1;
      }
      if(amountCovered > 8)
      filter.assignyellow50(key);
    }
     public int getamountCovered()
    {
      return amountCovered;
    }
    public boolean[] getCovered()
    {
      return covered;
    }
    public Point[] getkeyPoints()
    {
      return keyPoints;
    }
    public Vector getConnected()
    {
      return connected;
    }
    public spBlock getKey()
    {
      return key;
    }

    public String getObjectType()
    {
      return objectType;
    }
    public void setObjectType(String ot)
    {
      objectType = ot;
    }

    public float getTotalArea()
    {
      return totalArea;
    }
    public void setTotalArea(float n)
    {
      totalArea = n;
    }
    public void addVolume(spaceObject so)
          //this code adds a space object and recalculates groupObjects area
    {
  long addVolumeStart_time = (new java.util.Date()).getTime();
      spBlock addedSpace = so.getSpaceBlock(); // the added spaceBlock
      Rectangle addedSpacePlan = so.getSpacePlan();// the plan for this block
      Vector holdDeducted = new Vector();
      float addedSpacePlanZ = addedSpacePlan.height;
      float addedSpacePlanX = addedSpacePlan.width;
      totalArea = totalArea + (addedSpacePlanX * addedSpacePlanZ);
          //increases total area by this plan size
      Enumeration mySOs = getConnected().elements();
      while(mySOs.hasMoreElements())
          //goes trough all the existing space objects looking for intersections
      {
        spaceObject nextSpaceObject = (spaceObject)mySOs.nextElement();
        Rectangle nextSpacePlan = nextSpaceObject.getSpacePlan();
        spBlock nextSpace = nextSpaceObject.getSpaceBlock();
        if(nextSpacePlan.intersects(addedSpacePlan)
          )// everytime this space intersects with one of the existing spaces
        {
          Rectangle inters = addedSpacePlan.intersection(nextSpacePlan);
          //calculates the intersection with this space object
          float intersZ = inters.height;
          float intersX = inters.width;
          totalArea = totalArea - (intersX * intersZ);
          //decreases total area by the intersection size
          holdDeducted.addElement(inters);
         //holds the intersection for insertion to alreadydeducted, after this round
        }}
      Enumeration deducted = alreadyDeducted.elements();
      while(deducted.hasMoreElements())
          //adds back space that has been deducted previously, that this interse
      {
        Rectangle prev = (Rectangle) deducted.nextElement();
        if(prev.intersects(addedSpacePlan))
        {
          Rectangle addBack = addedSpacePlan.intersection(prev);
          float addBackZ = addBack.height;
          float addBackX = addBack.width;
          totalArea = totalArea + (addBackX * addBackZ);
         }
      }
      Enumeration hDeducted = holdDeducted.elements();
      while(hDeducted.hasMoreElements())
                    //puts the intersections into alreadyIntersected
      {
        Rectangle nextInt = (Rectangle)hDeducted.nextElement();
        alreadyDeducted.addElement(nextInt);
      }
      getConnected().addElement(so);
      //filter.addVolume_time = filter.addVolume_time
                    + (new java.util.Date()).getTime() - addVolumeStart_time;
     }
     public boolean[] getZoneSupports()
     {
     return zoneSupports;
     }
     public void setZoneSupports(boolean[] z)
     { zoneSupports = z;
     }
 }
```

public class spaceObject

```
import java.util.*;
import java.awt.*;
import java.lang.*;
public class spaceObject
  {
    private spBlock spaceBlock;
    private spBlock block1;
    private spBlock block2;
    private Rectangle spacePlan;
    public int spacebotY;
    public int spacetopY;
//used for space
 public spaceObject(Rectangle plan, spBlock b1, spBlock b2, int bottom, int top)
    {
      spacePlan = plan;
      block1 = b1;
      block2 = b2;
      spacebotY = bottom;
      spacetopY = top;

      spaceBlock = (spBlock) (tdBlockEditor.me.
                    makeInstanceFromClass(“spBlock”,“”, “file:
                    ///s:/demo/trains/definitions/blocks/concBlock.idf”, 0));

          spaceBlock.setTransform(new tdPoint((float)spacePlan.x,
                    (float)spacebotY,(0 - (float)spacePlan.y)));
              spaceBlock.setX(spacePlan.width);
              spaceBlock.setZ(spacePlan.height);//note this is the z dimension,
              spaceBlock.setY(spacetopY - spacebotY);
              spaceBlock.setMass(0);
              spaceBlock.setMaterial(2);
              filter.assignred50(spaceBlock);
              tdx.spDonateToLocale(spaceBlock);
              schmoozer.WMupdate();

              //NOTE THIS MAY NEED TO BE FIXED FOR REMOTE
            }

 public spaceObject(spBlock space, spBlock wall1, spBlock wall2)//used for light
            {
              spaceBlock = space;
              block1 = wall1;
              block2 = wall2;
            }


            public spBlock getSpaceBlock(){return spaceBlock;}
            public spBlock getBlock1(){return block1;}
            public spBlock getBlock2(){return block2;}
            public Rectangle getSpacePlan() { return spacePlan;}
           }
```

public class constructor extends td

```
import java.util.*;
import java.awt.*;
import java.lang.*;
public class constructor extends td
{
   public static void construct(String myFilter)
   {
      if(tdAvatarPlatform.getAvatarsLocale().getTitle().compareTo(myFilter+“Filter”)
!= 0)
      System.out.println(“Only valid if in “ + myFilter + “Filter”);
      else
      {
         int score = judgeFilterView(myFilter);
         changeWorld(myFilter);
      }
   }  public static int judgeFilterView(String myConstructor)
   {
      if(myConstructor.compareTo(“space”) == 0)
      {
         return judgeSpaceView(3, 250);
      }
      else if(myConstructor.compareTo(“enclosure”) == 0)
      {
         return judgeEnclosureView();
      }
      else if(myConstructor.compareTo(“light”) == 0)
      {
         return judgeLightView();
      }
      else if(myConstructor.compareTo(“structure”) == 0)
      {
         return judgeStructureView();
      }
      else if(myConstructor.compareTo(“chair”) == 0)
      {
         return judgeChairView(4);
      }
      else return judgeWorldView();
   }
      public static boolean changeWorld(String myConstructor)
   {

      if(myConstructor.compareTo(“space”) == 0)
      {
         return changeSpace();
      }
      else if(myConstructor.compareTo(“enclosure”) == 0)
      {
         return changeEnclosure();
      }
      else if(myConstructor.compareTo(“light”) == 0)
      {
         return changeLight();
      }
      else if(myConstructor.compareTo(“structure”) == 0)
      {
         return changeStructure();
      }
      else if(myConstructor.compareTo(“chair”) == 0)
      {
         return changeChair();
      }
      else return false;
   }
   public static Vector spaceWork;
   public static boolean spaceConInited = false;
   public static float bigSpaces = 0;
   public static float minAreaReq = 0;
   public static float numSpacesReq = 0;
////////////SPACE/////////////////////////////////////////////////////////
   public static int judgeSpaceView(float numSpaces, float minArea)
   {
      if((tdAvatarPlatform.getAvatarsLocale().getTitle()).compareTo(“spaceFilter”) !=
0)
      {
         System.out.println(“CAUTION - not in spaceFilter - info could be old”);
      }
      if(spaceConInited)
      {
         spaceWork.removeAllElements();
      }
      else {spaceWork = new Vector();}

      minAreaReq = minArea;
      numSpacesReq = numSpaces;
      Enumeration go = filter.spaces.elements();
      bigSpaces = 0;
      while(go.hasMoreElements())
      {
         groupObject nextGo = (groupObject)go.nextElement();
         float goArea = nextGo.getTotalArea();
         System.out.println(“GroupObject Area = “ + goArea);
         System.out.println(“Minimum Area = “ + minArea);
         if(goArea >= minArea)
         bigSpaces++;
         else
         spaceWork.addElement(nextGo);
      }
      float answer = ((bigSpaces/numSpaces * 100));
                        //this is where i could judge against smaller spaces
      if(answer > 100)
      answer = 55;
     System.out.println(“****************************************”);
      System.out.println(“Number of BigSpaces = “ + bigSpaces);
      System.out.println(“Number of smallSpaces = “ + spaceWork.size());
      System.out.println(“space Scores a “ + answer);
      return (int)answer;
   }

   public static boolean changeSpace()
            // it would be nice to try to break up a space if it is too big
   {  //which should be easy if delete space works, i just find which space
            //is biggest first(and put it in it's own vector)
      if(spaceWork.isEmpty() && (bigSpaces == numSpacesReq))
      {
         System.out.println(“I AM TOTALLY HAPPY HERE”);
         return true;
      }

      if(bigSpaces > numSpacesReq)// if i have too many big spaces i should try
                        //to delete one of them, i may delete a small space
      {
       boolean  success = false;
       int numSpaces = filter.spaces.size();
       int tries = 0;
       while(tries < numSpaces)
       {
         success = deleteSpace(filter.spaces, tries);
                        //deletes any walls not used by other spaces
         if(success) return true;
         tries++;
       }
      }
      else if(bigSpaces == numSpacesReq)//but space work isn't empty
      {
         boolean  success = false;
         int numSpaces = spaceWork.size();
         int tryNumber = 0;
         while(tryNumber < numSpaces)
         {
            success = deleteSpace(spaceWork, tryNumber);
                        //deletes any walls not used by other spaces
            if(success) return true;
            tryNumber++;
         }
      }
      else if(!spaceWork.isEmpty())
                        // here i need to find a small space and make it bigger
      {
         System.out.println(“SPACEWORK ISN'T EMPTY”);
         int numBigSpaces = filter.spaces.size();
         int numSmallSpaces = spaceWork.size();

         Vector planRectangles = new Vector();
                        // this will be a vector of all the bigspace plans
         Enumeration big = filter.spaces.elements();
         while(big.hasMoreElements())
   //this code goes through spaces and gets all the space plans of the BIG spaces
         {
            groupObject nextBig = (groupObject)big.nextElement();
            if(nextBig.getTotalArea() > minAreaReq)
            {
               Enumeration so = nextBig.getConnected().elements();
               while(so.hasMoreElements())
               {
                  spaceObject nextSo = (spaceObject)so.nextElement();
            Rectangle nextRect = nextSo.getSpacePlan();
            if(!planRectangles.contains(nextRect))
            {planRectangles.addElement(nextRect);}
          }
        }
      }
      int smallSpaceTried = 0;
      boolean success = false;

      while(smallSpaceTried < numSmallSpaces)
      {
        int sRanNum = (int)(Math.random() * spaceWork.size());
        groupObject smallGo = (groupObject)spaceWork.elementAt(sRanNum);
                    // pulls out the group Object at this position in spaceWork
        Vector smallSos = smallGo.getConnected();
        Enumeration sm = smallSos.elements();
        while(sm.hasMoreElements())
                    // while there are more space Objects in this groupObject
        {
          spaceObject nextSo = (spaceObject)sm.nextElement();
          Rectangle nextSpacePlan = nextSo.getSpacePlan();
                              // pulls out the space plan for this space
          float existArea = (nextSpacePlan.width * nextSpacePlan.height);
                                        //this isn't too accurate but ....
          float neededArea = minAreaReq - existArea;
          Point loc = new Point(nextSpacePlan.getLocation());
                    // this is where i can make the spaces not SQUARE
     Rectangle newSpacePlan = new Rectangle(loc.x, loc.y, ((int)Math.
        sqrt((double)neededArea)+ 4), ((int)Math.sqrt((double)neededArea)+ 4));

          int[] positionNumbers = new int[8];
          int num = 0;
          int zed = 0;
          while(num < 8)
          // this is all just to get an array distinct random numbers from 0 - 7
          {
            boolean contained = false;
            int r = (int)(Math.random() * 8);
            for(int i = 0; i < 8; i++)
            {
              if(positionNumbers[i] == r)
              { contained = true;}
            }
            if(!contained)
            {
              positionNumbers[num] = r;
              num++;
            }
            else if(zed == 0)
            {
              positionNumbers[num] = 0;
              num++;
              zed = 1;
            }
          }
          boolean goodPlan;
          int posCounter = 0;
          while(posCounter < 8)
          {  //make this while loop go through each possible
                    //position for the new space plan
            goodPlan = true;
            Enumeration pR = planRectangles.elements();
positionRectangle(nextSpacePlan, newSpacePlan, positionNumbers[posCounter]);
                    //positions rectangle around another rectangle
            while(pR.hasMoreElements())//go through all the rectangles
            {
              Rectangle thispR = (Rectangle)pR.nextElement();
              if(newSpacePlan.intersects(thispR))
              {
                goodPlan = false;
                break;
          //found an intersection, so i don't need to bother looking further
              }
            }
            if(goodPlan)
          //this is it!  make a space here(nextSpacePlan is in the right palce)
            {
              int ranNum = (int)(Math.random() * 4);          //     4
                                                        //    2[]3
              int ranNum2 = (int)(Math.random() * 4);         //     1
        while(ranNum == ranNum2)//make sure the numbers aren't the same
              {ranNum2 = (int)(Math.random() * 4);}
              filter.goToFilter("worldLocale");
        spBlock firstWall= new spBlock(nextSo.getBlock1().getReferenced());
       spBlock secondWall = new spBlock(nextSo.getBlock2().getReferenced());
        positionWalls(firstWall, ranNum, secondWall, ranNum2, newSpacePlan);
              tdx.spDonateToLocale(firstWall);
              tdx.spDonateToLocale(secondWall);
              return true;
            }
   posCounter++;//look at the next position of this rectangle agianst all the others
          }
        }//go to the next space Object,
        spaceWork.removeElementAt(sRanNum);
        smallSpaceTried++;
      }// go to the next groupObject
    }
    else // i don't have enough big spaces, an i have no small spaces
          // i need to create a new space in the middle of nowhere
    {
      int ranNum = (int)(Math.random() * 2);
      int dxn = (int)(Math.random() * 4);
      int ranNum1 = (int)(Math.random() * 4);
      int ranNum2 = (int)(Math.random() * 4);
      while(ranNum == ranNum2)     //make sure the numbers aren't the same
      {ranNum2 = (int)(Math.random() * 4);}
      Vector planRectangles = new Vector();
                    // this will be a vector of all the bigspace plans
      Enumeration big = filter.spaces.elements();
      int highestX = 1;
      int highestZ = 1;
      int lowestX = 1000;
      int lowestZ = 1000;
      while(big.hasMoreElements())
    //this code goes through spaces and gets all the space plans of the BIG
spaces
      {
        groupObject nextBig = (groupObject)big.nextElement();
        if(nextBig.getTotalArea() > minAreaReq)
        {
          Enumeration so = nextBig.getConnected().elements();
          while(so.hasMoreElements())
          {
            spaceObject nextSo = (spaceObject)so.nextElement();
            Rectangle nextRect = nextSo.getSpacePlan();if(highestX <
          nextRect.x + nextRect.width) highestX = nextRect.x + nextRect.width;

            if(highestZ < nextRect.y + nextRect.height) highestZ =
                                         nextRect.y + nextRect.height;
            if(lowestX > nextRect.x) lowestX = nextRect.x;
            if(lowestZ > nextRect.y) lowestZ = nextRect.y;
            if(!planRectangles.contains(nextRect))
            {planRectangles.addElement(nextRect);}
          }
        }
      }
      filter.goToFilter("worldLocale");
      spBlock wall1 = tdBlockEditor.me.makeNewBlock();
      wall1.setYpos(1);
      spBlock wall2 = tdBlockEditor.me.makeNewBlock();
      wall2.setYpos(1);
      tdx.spDonateToLocale(wall1);
      tdx.spDonateToLocale(wall2);
      int wallLength = (int)Math.sqrt((double)minAreaReq)+ 2;
      int middleX = lowestX + ((highestX - lowestX)/2);
      int middleZ = lowestZ + ((highestZ - lowestZ)/2);
      Rectangle plan = new Rectangle(0,0,wallLength, wallLength);

     if(dxn < 1){plan.setLocation((middleX * ranNum) ,
                              Math.abs((lowestZ - wallLength - 2)));}
      else if(dxn < 2){plan.setLocation(Math.
                    abs((lowestX - wallLength - 2)) , (middleZ * ranNum));}
      else if(dxn < 3){plan.setLocation((highestX + 2) , (middleZ * ranNum));}
      else {plan.setLocation((middleX * ranNum) , (highestZ + 2));}
      positionWalls(wall1, ranNum1, wall2, ranNum2,plan);
      return true;
    }
    return false;
   }
public static void positionWalls(spBlock wall1, int pos1,
                              spBlock wall2, int pos2, Rectangle plan)
{
  float planSWx = (float)plan.x;
  float planSWz = -(float)plan.y;
  float planNEx = (float)plan.x + plan.width;
  float planNEz = -(float)plan.y - plan.height;
  float planX = planNEx - planSWx;
  float planZ = Math.abs(planNEz) - Math.abs(planSWz);
  if(pos1 < 1)//set wall1 along the bottom border
  {   wall1.setXpos(planSWx); wall1.setZpos(planSWz); wall1.setX(planX);
```

```
               wall1.setZ(1); wall1.setY(planZ);}
   else if(pos1 < 2)//set it along the left border
   {   wall1.setXpos(planSWx); wall1.setZpos(planSWz); wall1.setX(1);
wall1.setZ(planZ); wall1.setY(planX);}
   else if(pos1 < 3)//set it along the left border
   {   wall1.setXpos(planNEx); wall1.setZpos(planSWz); wall1.setX(1);
wall1.setZ(planZ); wall1.setY(planX);}
   else//set it along the top border
   {   wall1.setXpos(planSWx); wall1.setZpos(planNEz); wall1.setX(planX);
wall1.setZ(1); wall1.setY(planZ);}

   //Wall2
   if(pos2 < 1)//set wall2 along the bottom border
   {   wall2.setXpos(planSWx); wall2.setZpos(planSWz); wall2.setX(planX * 2);
               wall2.setZ(1); wall2.setY(11);}//was plan z, changed to 11 also planx*2
   else if(pos2 < 2)//set it along the left border
   {   wall2.setXpos(planSWx); wall2.setZpos(planSWz); wall2.setX(1);
     wall2.setZ(planZ * 2); wall2.setY(11);} //was plan x, changed to 11 also planz*2
   else if(pos2 < 3)//set it along the left border
   {   wall2.setXpos(planNEx); wall2.setZpos(planSWz); wall2.setX(1);
     wall2.setZ(planZ * 2); wall2.setY(11);}//was plan x, changed to 11 also planz*2
   else//set it along the top border
   {   wall2.setXpos(planSWx); wall2.setZpos(planNEz); wall2.setX(planX * 2);
                wall2.setZ(1); wall2.setY(11);}//was plan z, changed to 11 also planx*2
   //this is just to catch erroers
   if((wall1.getXpos() > 100)|(wall1.getXpos() < 1))
   {    wall1.setXpos((float)Math.random() * 75);}
   if((wall1.getZpos() < -100)|(wall1.getZpos() > -1))
   {   wall1.setZpos((float)Math.random() * -75);}
   if((wall2.getXpos() > 100)|(wall2.getXpos() < 1))
   {   wall2.setXpos((float)Math.random() * 75);}
   if((wall2.getZpos() < -100)|(wall2.getZpos() > -1))
   {   wall2.setZpos((float)Math.random() * -75);}
}
public static void positionRectangle(Rectangle sun, Rectangle planet, int pos)
               //code here to position one recrtangle around another
{
   //  This code simply takes a rectangle and positions it at an
  // appropriate place around a second rectangle (with a 2' overlap)
   //     [5]   [6]   [7]
   //     [3]  [sun]  [4]
   //     [0]   [1]   [2]
   //
   int sunSWX = sun.x;
   int sunSWY = sun.y;
   int sunNEX = sun.x + sun.width;
   int sunNEY = sun.y + sun.height;

   if(pos == 0)
   {planet.setLocation((sun.x - planet.width + 2), (sun.y - planet.height + 2));}
   if(pos == 1)
   {planet.setLocation((sun.x), (sun.y - planet.height + 2));}
   if(pos == 2)
   {planet.setLocation((sun.x + planet.width - 2), (sun.y - planet.height + 2));}

   if(pos == 3)
   {planet.setLocation((sun.x - planet.width + 2), (sun.y));}
   if(pos == 4)
   {planet.setLocation((sun.x + planet.width - 2), (sun.y));}

   if(pos == 5)
   {planet.setLocation((sun.x - planet.width + 2), (sun.y + planet.height - 2));}
   if(pos == 6)
   {planet.setLocation((sun.x), (sun.y + planet.height - 2));}
   if(pos == 7)
   {planet.setLocation((sun.x + planet.width - 2), (sun.y + planet.height - 2));}
}
public static boolean deleteSpace(Vector groupOfSpaces, int tryNumber)
               //wants a vector of groupObjects (either spaces, or spaceWork
{
   int ranNum = (int)(Math.random() * groupOfSpaces.size());
   groupObject goToBeDeleted =
                              (groupObject)groupOfSpaces.elementAt(tryNumber);
   Vector sosToBeDeleted = goToBeDeleted.getConnected();
               //this is a vector of the spaceObjects I want to delete
   Vector wallsToBeDeleted = new Vector();
   Enumeration sosd = sosToBeDeleted.elements();
         while(sosd.hasMoreElements())
                                  //make a vector of the possible walls to be deleted
         {
            spaceObject nso = (spaceObject)sosd.nextElement();
            spBlock b1 = nso.getBlock1();
            spBlock b2 = nso.getBlock2();
            if(!wallsToBeDeleted.contains(b1))
            wallsToBeDeleted.addElement(b1);
```

```
            if(!wallsToBeDeleted.contains(b2))
            wallsToBeDeleted.addElement(b2);
         }
         boolean deleted = false;
         Enumeration as = filter.spaces.elements();//all the spaces
         if(!wallsToBeDeleted.isEmpty())
         {
            while(as.hasMoreElements())
            {
               if(wallsToBeDeleted.isEmpty())
               {break;}
               groupObject nextSpace = (groupObject)as.nextElement();
               if(nextSpace != goToBeDeleted)
               //make sure this group object isn't the same as the one to be deleted
               {
                  Enumeration sos = nextSpace.getConnected().elements();
                           //the spaceObjects of the nextGroupObject
                  while(sos.hasMoreElements())
                  {
                     spaceObject otherSpace = (spaceObject)sos.nextElement();
                     spBlock wall1 = otherSpace.getBlock1();
                                            // first wall of other space object
                     spBlock wall2 = otherSpace.getBlock2();//second wall
                     Enumeration dw = wallsToBeDeleted.elements();
                                            // an enumeration of the walls to be deleted
                     while(dw.hasMoreElements())
                     {
                        spBlock delWall = (spBlock)dw.nextElement();
                        if(wall1 == delWall | wall2 == delWall)
                        wallsToBeDeleted.removeElement(delWall);
                     }
                  }
               }
            }
         }
         if(!wallsToBeDeleted.isEmpty())
         {
            filter.goToFilter("worldLocale");
            Enumeration wal = wallsToBeDeleted.elements();
            while(wal.hasMoreElements())
            {
              spBlock deadWall = ((spBlock)wal.nextElement()).getReferenced();
              if(deadWall.getLocallyOwned())
                      { deadWall.remove(0);
                      }else
             { spOMI.makeStaticCall("onDeleteTree", "tdEditor", (spBlock)deadWall);}
            }
            return true;
         }
         else{
            return false;
         }
}
public static Vector lightWork;//holds space Objects that need work
public static boolean lightConInited = false;
/////////////LIGHT/////////////////////////////////////////////////////////////
   public static int judgeLightView()
   {
      if((tdAvatarPlatform.getAvatarsLocale().getTitle()).compareTo("lightFilter") != 0)
      {
         System.out.println("CAUTION - not in lightFilter - info could be old");
      }
      if(lightConInited) lightWork.removeAllElements();
      else {
         lightWork = new Vector();
         lightConInited = true;
      }
      Enumeration so = filter.lightObjects.elements();
      float numSpaces = 0;
      float numWellLit = 0;
      while(so.hasMoreElements())
      {
         spaceObject nextSO = (spaceObject)so.nextElement();
         int mat1 = nextSO.getBlock1().getMaterial();
         int mat2 = nextSO.getBlock2().getMaterial();
         if((mat1 + mat2) < 1) lightWork.addElement(nextSO);
         else numWellLit++;
         numSpaces++;
      }
      float answer = (numWellLit/numSpaces * 100);
      System.out.println("light Scores a " + answer);
      return (int)answer;
   }
```

```
  public static boolean changeLight()
  {
    if(!lightWork.isEmpty())
    {
      int ranNum = (int)(Math.random() * lightWork.size());
      if(ranNum == lightWork.size()) ranNum = ranNum - 1;
      //if i get null pointer, could be here
      spaceObject toBeFixed = (spaceObject)lightWork.elementAt(ranNum);
      int ranNum2 = (int)(Math.random() * (float) 2.0);
      if(ranNum2 < 1)
      {toBeFixed.getBlock1().getReferenced().setMaterial(1);}
      else
      {toBeFixed.getBlock2().getReferenced().setMaterial(1);}
    }
    else
    {}
    filter.goToFilter(“worldLocale”);
    return true;
  }



public static Vector structureWork;
public static boolean structureConInited = false;
  /////////////STRUCTURE////////////////////////////////////////////////////////////////
  public static int judgeStructureView()
  {
    if((tdAvatarPlatform.getAvatarsLocale().getTitle()).
                                        compareTo(“structureFilter”) != 0)
    {
      System.out.println(“CAUTION - not in structureFilter - info could be old”);
    }
    if(structureConInited) structureWork.removeAllElements();
    else structureWork = new Vector();
    float supported = 0;
    float numBeams = 0;
    Enumeration go = filter.structureObjects.elements();
    while(go.hasMoreElements())
    {
      numBeams++;
      groupObject strO = (groupObject) go.nextElement();
      boolean[] supp = strO.getZoneSupports();
      if((supp[0] && supp[2]) | (supp[1] && supp[3]))
      {
        supported++;
        spBlock beam = strO.getKey();
      }
      else
      {
        structureWork.addElement(strO);
      }
    }
    int answer;
    float temp = numBeams - supported;
    if(temp == 0)
    answer = 100;
    else if (temp == 1)
    answer = 50;
    else
    answer = 0;
    System.out.println(“structure Scores a “ + answer);
    return answer;

  }

  public static boolean changeStructure()
  {
    spBlock col = null;
    spBlock col2 = null;
    spBlock beam = null;
    boolean col1Build = false;
    boolean col2Build = false;
    float colXpos = 0;
    float colZpos = 0;
    float col2Xpos = 0;
    float col2Zpos = 0;
    float beambotY = 0;
    if(!structureWork.isEmpty())
    {
      int ranNum = (int)(Math.random() * structureWork.size());
                                              //if i get null pointer, could be here
      groupObject toBeFixed = (groupObject)structureWork.elementAt(ranNum);
      int ranNum2 = (int)(Math.random() * (float) 2.0);
      boolean[] supp = toBeFixed.getZoneSupports();
```

```
      beam = toBeFixed.getKey();
      float beamSWX = beam.getSWbot().pt[spTransformX];
      float beamSWZ =Math.abs(beam.getSWbot().pt[spTransformZ]);
      float beamNEX = beam.getNEbot().pt[spTransformX];
      float beamNEZ = Math.abs(beam.getNEbot().pt[spTransformZ]);
      beambotY = beam.getSWbot().pt[spTransformY];
      float beamMidX = beamSWX + ((beamNEX - beamSWX)/2);
                                        // X the point in the center of the box
      float beamMidZ = beamSWZ + ((beamNEZ - beamSWZ)/2);
                                        // Z the  Point at the center
      if(supp[0]| supp[1] | supp[2] | supp[3])
      {
        col1Build = true;
        if(supp[2])
        {
          colXpos = (beamMidX - (beamMidX-beamSWX)/2);
          colZpos = (0 - (beamMidZ - (beamMidZ-beamSWZ)/2));
        }
        else if(supp[3])
        {
          colXpos = (beamNEX - (beamNEX - beamMidX)/2);
          colZpos = (0 - (beamMidZ - (beamMidZ-beamSWZ)/2));
        }
        else if(supp[0])
        {
          colXpos = (beamNEX - (beamNEX-beamMidX)/2);
          colZpos = (0 - (beamNEZ - (beamNEZ-beamMidZ)/2));
        }
        else if(supp[1])
        {
          colXpos = (beamMidX - (beamMidX - beamSWX)/2);
          colZpos = (0 - (beamNEZ - (beamNEZ-beamMidZ)/2));
        }
      }
      else
      {
        col1Build = true;
        col2Build = true;
        col2 = new spBlock(beam);
        if(ranNum2 < 1)
        {
          colXpos = (beamNEX - (beamNEX-beamMidX)/2);
          colZpos = (0 - (beamNEZ - (beamNEZ-beamMidZ)/2));
          col2Xpos = (beamMidX - (beamMidX - beamSWX)/2);
          col2Zpos = (0 - (beamMidZ - (beamMidZ-beamSWZ)/2));
        }
        else
        {
          colXpos = (beamMidX - (beamMidX - beamSWX)/2);
          colZpos = (0 - (beamNEZ - (beamNEZ-beamMidZ)/2));
          col2Xpos = (beamNEX - (beamNEX-beamMidX)/2);
          col2Zpos = (0 - (beamMidZ - (beamMidZ-beamSWZ)/2));
        }
      }

    }

    filter.goToFilter(“worldLocale”);
    if(col1Build)
    {
      col = new spBlock(beam);
      tdx.spDonateToLocale(col);
      filter.assignNormalConc(col);
      col.setXpos(colXpos);
      col.setZpos(colZpos);
      col.setYpos(1);
      col.setX(1);
      col.setZ(1);
      col.setY(beambotY);
      col.setMaterial(0);
    }
    if(col2Build)
    {
      col2 = new spBlock(beam);
      tdx.spDonateToLocale(col2);
      filter.assignNormalConc(col2);
      col.setXpos(col2Xpos);
      col2.setZpos(col2Zpos);
      col2.setYpos(1);
      col2.setX(1);
      col2.setZ(1);
      col2.setY(beambotY);
      col2.setMaterial(0);
      }
    return true;;
  }


public static Vector enclosureWork;
    //contains group objects of spaces that need to be covered, and possible roofs
public static boolean enclosureConInited = false;
  //////////////ENCLOSURE/////////////////////////////////////////////////////////////////
public static int judgeEnclosureView()
  {
   if((tdAvatarPlatform.getAvatarsLocale().getTitle())
                    .compareTo(“enclosureFilter”) != 0)
    {
      System.out.println(“CAUTION - not in enclosureFilter - info could be old”);
    }
    if(enclosureConInited) enclosureWork.removeAllElements();
    else enclosureWork = new Vector();

    float numSpaces = 0;
    float covered = 0;
    Enumeration go = filter.enclosureObjs.elements();
    while(go.hasMoreElements())
    {
      numSpaces++;
      groupObject encO = (groupObject) go.nextElement();
      if(encO.getamountCovered() > 8)
      {  covered++;}// this space is totally covered
      else
      {enclosureWork.addElement(encO);}
    }
    float answer = (covered/numSpaces * 100);
    System.out.println(“Enclosure Scores a “ + answer);
    return (int)answer;
  }

  public static boolean changeEnclosure()
  {
    if(!enclosureWork.isEmpty())
    {
      int ranNum = (int)(Math.random() * enclosureWork.size());
                                       //if i get null pointer, could be here
      groupObject toBeFixed = (groupObject)enclosureWork.elementAt(ranNum);
      Vector roofs = toBeFixed.getConnected();
      int ranNum2 = (int)(Math.random() * (float) 2.0);
      spBlock space = toBeFixed.getKey();
      filter.assignNormalGlas(space);
      spBlock cover = null;
      if(roofs.isEmpty())// if there are no roofs over this space
      {
        filter.goToFilter(“worldLocale”);
        cover = new spBlock(space);
        cover.setMaterial(0);
        cover.setY(1);
        cover.setXpos(cover.getXpos() - 2);
        cover.setZpos(cover.getZpos() + 2);
        cover.setX(cover.getX() + 4);
        cover.setZ(cover.getZ() + 4);
      }
      else // there are some roofs over this space
      {
        Point[] spacePoints = toBeFixed.getkeyPoints();
        spBlock bestRoof = null;
        int biggestInt = 0;
        Enumeration r = roofs.elements();
        while(r.hasMoreElements())/
                     / find which roof intersects the space plan the most
        {
          spBlock roof = (spBlock)r.nextElement();
          filter.assignSelectedGlas(roof);
          Rectangle roofRect = new Rectangle((int)roof.getXpos(),
                     (int)(0 - roof.getZpos()), (int)roof.getX(),(int)roof.getZ());
          int intCount = 0;
          for(int n = 0; n < 9; n++)
          {  if(roofRect.contains(spacePoints[n]))
                     //if the roof rectangle intersects any of the points
            {  intCount++;}  //this point is covered
          }
          if(intCount > biggestInt)
          {
            biggestInt = intCount;
            bestRoof = roof;//make the best roof for expansion , this roof
            filter.assigngreen100(bestRoof);
          }
          else filter.assigngreen50(bestRoof);
```

```
        }
        filter.goToFilter(“worldLocale”);
        cover = bestRoof.getReferenced();
        if(cover.getXpos() > space.getXpos())
          // this code alters the transform of an existing roof to cover the space
        {  cover.setXpos(space.getXpos() - 1);}
        if(cover.getZpos() < space.getZpos())
        {  cover.setZpos(space.getZpos()+ 1);}
     if(cover.getNEbot().pt[spTransformX] < space.getNEbot().pt[spTransformX])
        { cover.setX(4 + cover.getX()+ (space.getNEbot().pt[spTransformX] -
                                   cover.getNEbot().pt[spTransformX]));}
     if(cover.getNEbot().pt[spTransformZ] > space.getNEbot().pt[spTransformZ])
        { cover.setZ(4 + cover.getZ() + (cover.getNEbot().pt[spTransformZ] -
                                   space.getNEbot().pt[spTransformZ]));}
      }
      tdx.spDonateToLocale(cover);
      if(cover.getMaterial() == 0)
      {  filter.assignNormalConc(cover);}
      if(cover.getMaterial() == 1)
      {  filter.assignNormalGlas(cover);}
      if(cover.getYpos() < (space.getYpos()+ space.getY()))
      {  cover.setYpos(space.getYpos()+ space.getY());}
    }
    filter.goToFilter(“worldLocale”);
    return true;
  }

public static Vector chairWork;
public static boolean chairConInited = false;
  //////////////CHAIR/////////////////////////////////////////////////////////////////
public static int judgeChairView(int reqChairs)
   {
  if((tdAvatarPlatform.getAvatarsLocale().getTitle()).compareTo(“chairFilter”) != 0)
    {
      System.out.println(“CAUTION - not in chairFilter - info could be old”);
    }
    if(enclosureConInited) enclosureWork.removeAllElements();
    else enclosureWork = new Vector();

    float numChairs = 0;
    float posChairs = 0;
    Enumeration go = filter.chairObjs.elements();
    while(go.hasMoreElements())
    {
      groupObject chrO = (groupObject) go.nextElement();
      boolean[] supported = chrO.getZoneSupports();
      if ((supported[0] && supported[2]) | (supported[1] && supported[3]))
      numChairs++;
      else if (supported[0] | supported[1] | supported[2] | supported[3])
      {chairWork.addElement(chrO);
      posChairs++;}
      else
      {chairWork.addElement(chrO);
    posChairs++;}// for completeness, and if there is a possible seat with no legs

    }
    float answer = (((numChairs + (posChairs /2))/reqChairs) * 100);
    System.out.println(“chair Scores a “ + answer);
    return (int)answer;
  }
  public static boolean changeChair()
  {return true;}
public static int judgeWorldView()
{
  return 0;
}
}
```

public class spMediator extends sp

```
import java.util.*;
import java.awt.*;
import java.lang.*;
import com.sun.java.swing.*;
public class spMediator extends sp
{
  public static String[] availableFilters;
  public static String[] activeFilters;

  private static int[] currentScores;
  private static int[] previousScores;

 public static void start(JCheckBox[] medConstructors, int numCycles,
                          int numSpaces, int sizeSpaces, int numChairs)
 {
   boolean spaceQ = medConstructors[0].isSelected();
   boolean enclosureQ = medConstructors[1].isSelected();
   boolean structureQ = medConstructors[2].isSelected();
   boolean chairQ = medConstructors[3].isSelected();
   boolean lightQ = medConstructors[4].isSelected();
   int counter = 0;
   while((counter < numCycles))// && !tdBlockEditor.medStopButton.isSelected()
   {
     int spaceScore = 100;
     int enclosureScore = 100;
     int structureScore = 100;
     int lightScore = 100;
     int chairScore = 100;
     filter.goToFilter(“worldLocale”);
     if(spaceQ)
     {
       filter.perceive(“spaceFilter”);
       spaceScore = constructor.judgeSpaceView(numSpaces, sizeSpaces);
       filter.goToFilter(“worldLocale”);
     }
     if(enclosureQ)
     {
       if(spaceQ) filter.goToFilter(“spaceFilter”);
       filter.perceive(“enclosureFilter”);
       enclosureScore = constructor.judgeEnclosureView();
       filter.goToFilter(“worldLocale”);
     }
     if(structureQ)
     {
       filter.perceive(“structureFilter”);
       structureScore = constructor.judgeStructureView();
       filter.goToFilter(“worldLocale”);
     }
     if(lightQ)
     {
      if(spaceQ) filter.goToFilter(“spaceFilter”);
       filter.perceive(“lightFilter”);
       lightScore = constructor.judgeLightView();
       filter.goToFilter(“worldLocale”);
     }
     if(chairQ)
     {
       filter.perceive(“chairFilter”);
       chairScore = constructor.judgeChairView(numChairs);
       filter.goToFilter(“worldLocale”);
     }

     System.out.println(“SCORES FOR ROUND “ + counter + “ ARE  Space = “
                                             + spaceScore + “ Enclosure = “
       + enclosureScore + “ Light = “ + lightScore + “ Structure = “
                                  + structureScore + “ Chair = “ + chairScore);
     if((spaceScore <= enclosureScore)&& (spaceScore <= structureScore)&&
                 (spaceScore <= chairScore)&& (spaceScore <= lightScore))
     {
       constructor.changeSpace();
     }
     else if ((structureScore <= enclosureScore)&&
             (structureScore <= spaceScore)&&
              (structureScore <= chairScore)&&
              (structureScore <= lightScore))
     {
       constructor.changeStructure();
     }
     else if ((lightScore <= structureScore)&& (lightScore <= spaceScore)&&
              (lightScore <= chairScore)&& (lightScore <= enclosureScore))
     {
       constructor.changeLight();
     }
     else if ((enclosureScore <= structureScore)&&
           (enclosureScore <= spaceScore)&& (enclosureScore <= chairScore)&&
                                      (enclosureScore <= lightScore))
     {
       constructor.changeEnclosure();
     }
     else if ((chairScore <= structureScore)&& (chairScore <= spaceScore)&&
             (chairScore <= enclosureScore)&& (chairScore <= lightScore))
     {
       constructor.changeChair();
     }
     counter++;
   }
 }
}
```

Appendix C: Images

The following views were generated after a run of the mediator completed 19 iterations. The top image on each page is from the preset SouthWest viewPoint, the bottom image is taken from a perspective useful for each domain.

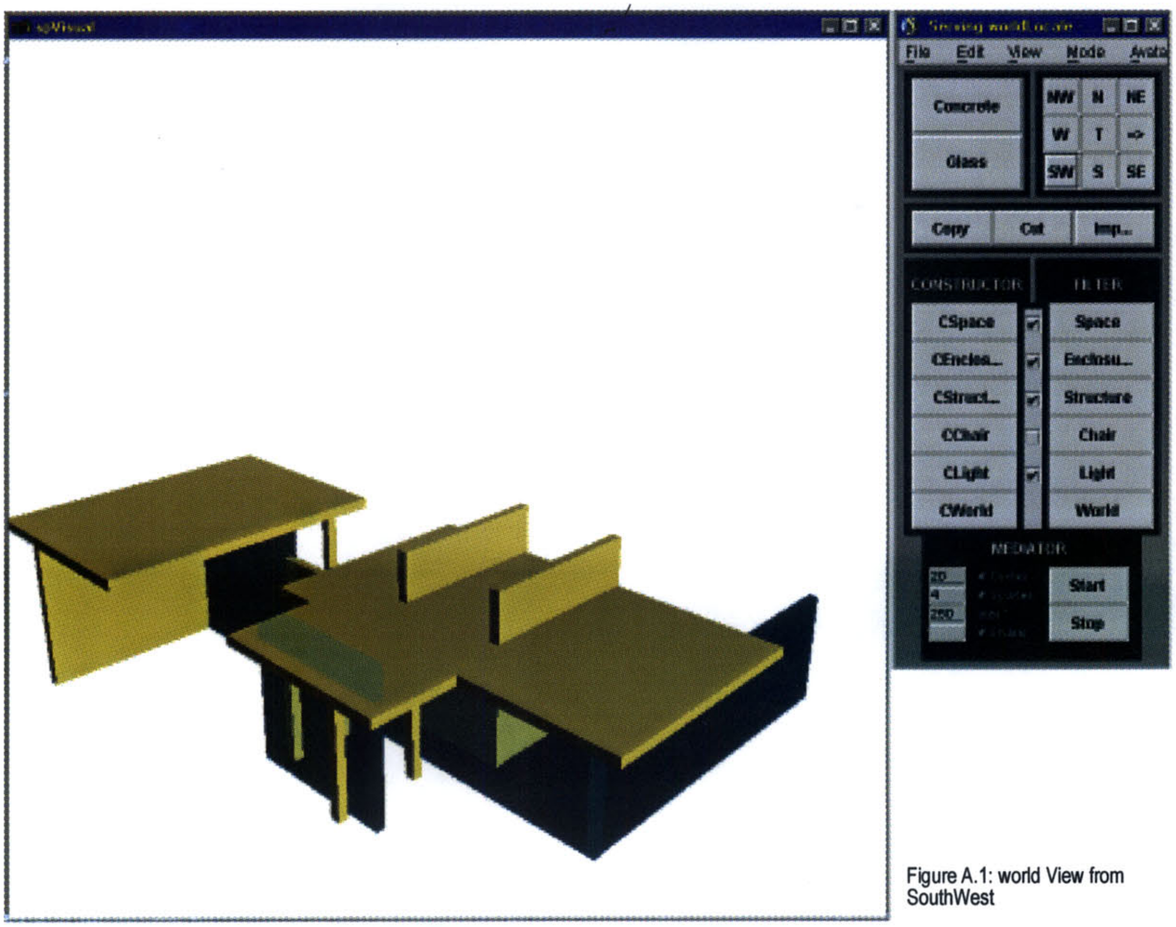


Figure A.1: world View from SouthWest

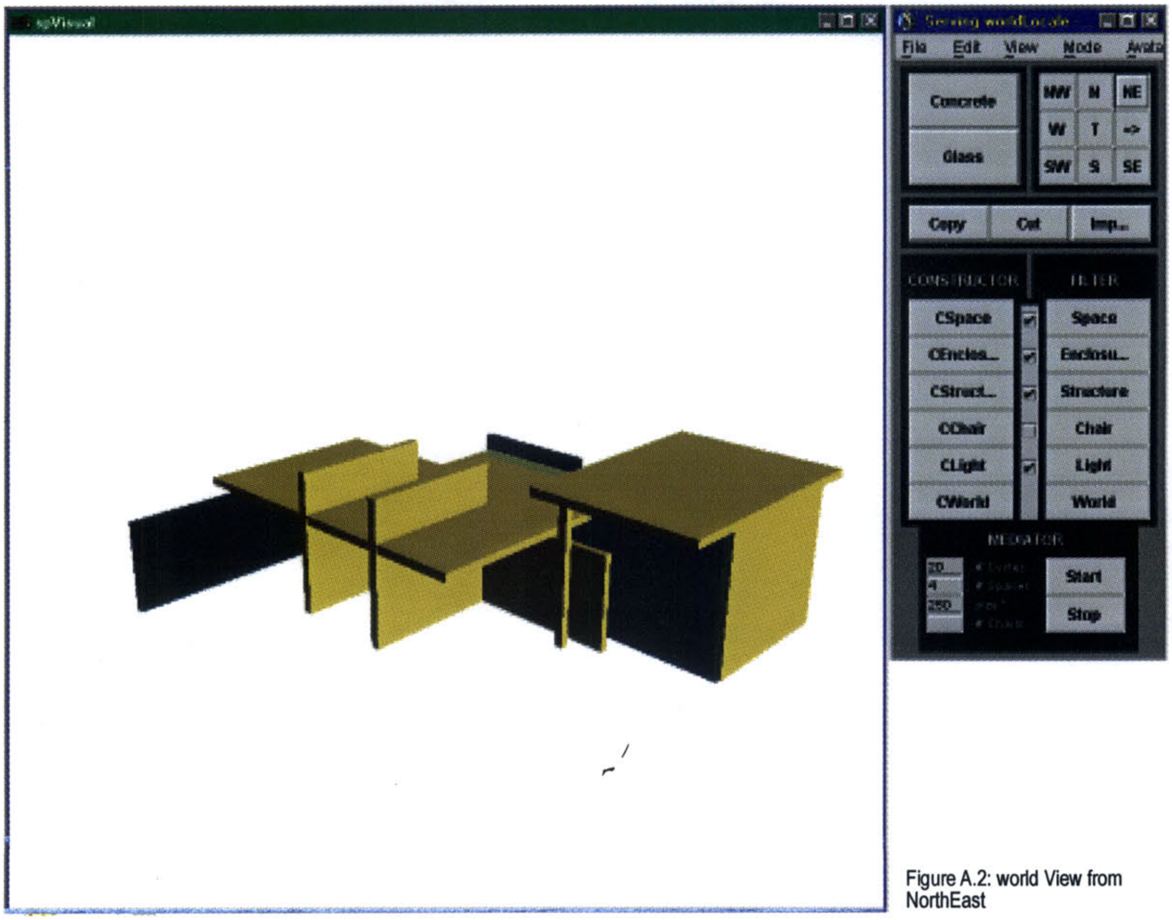


Figure A.2: world View from NorthEast

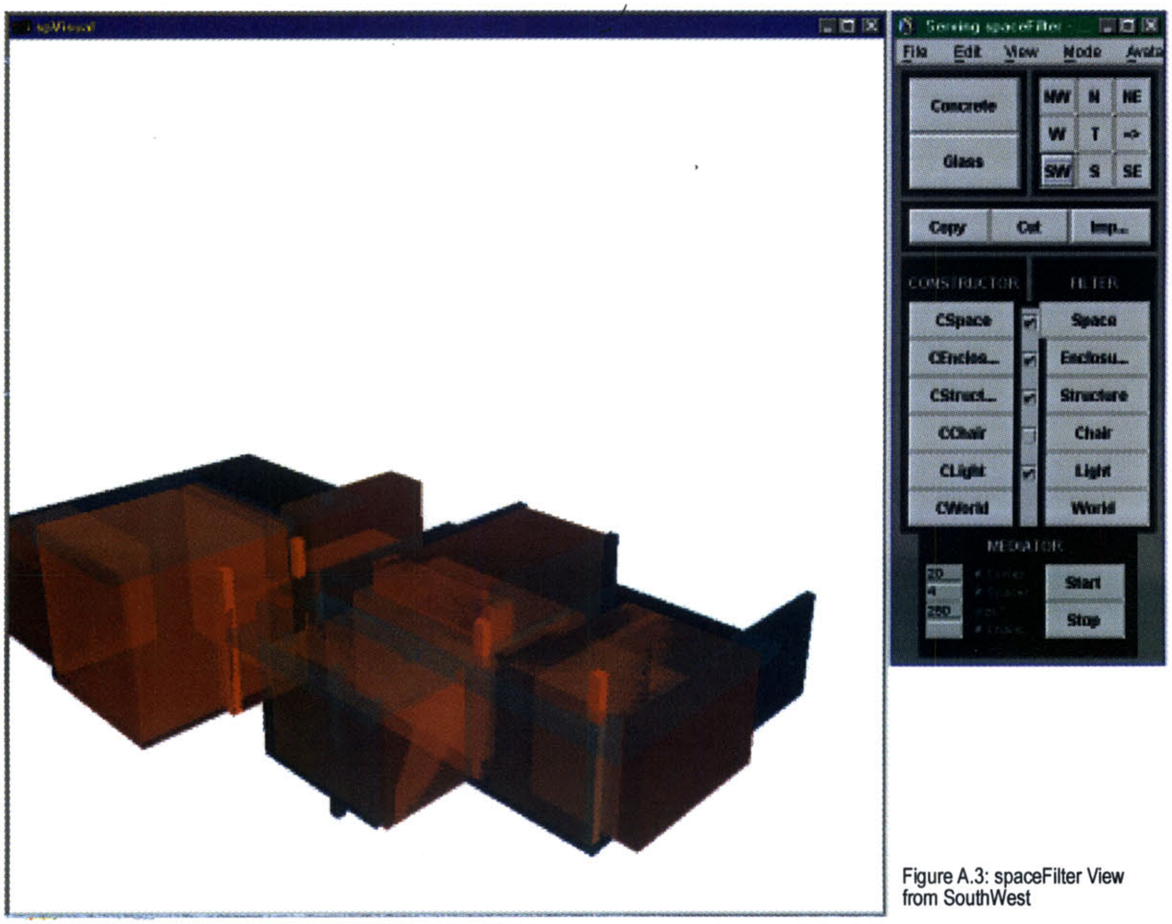


Figure A.3: spaceFilter View from SouthWest

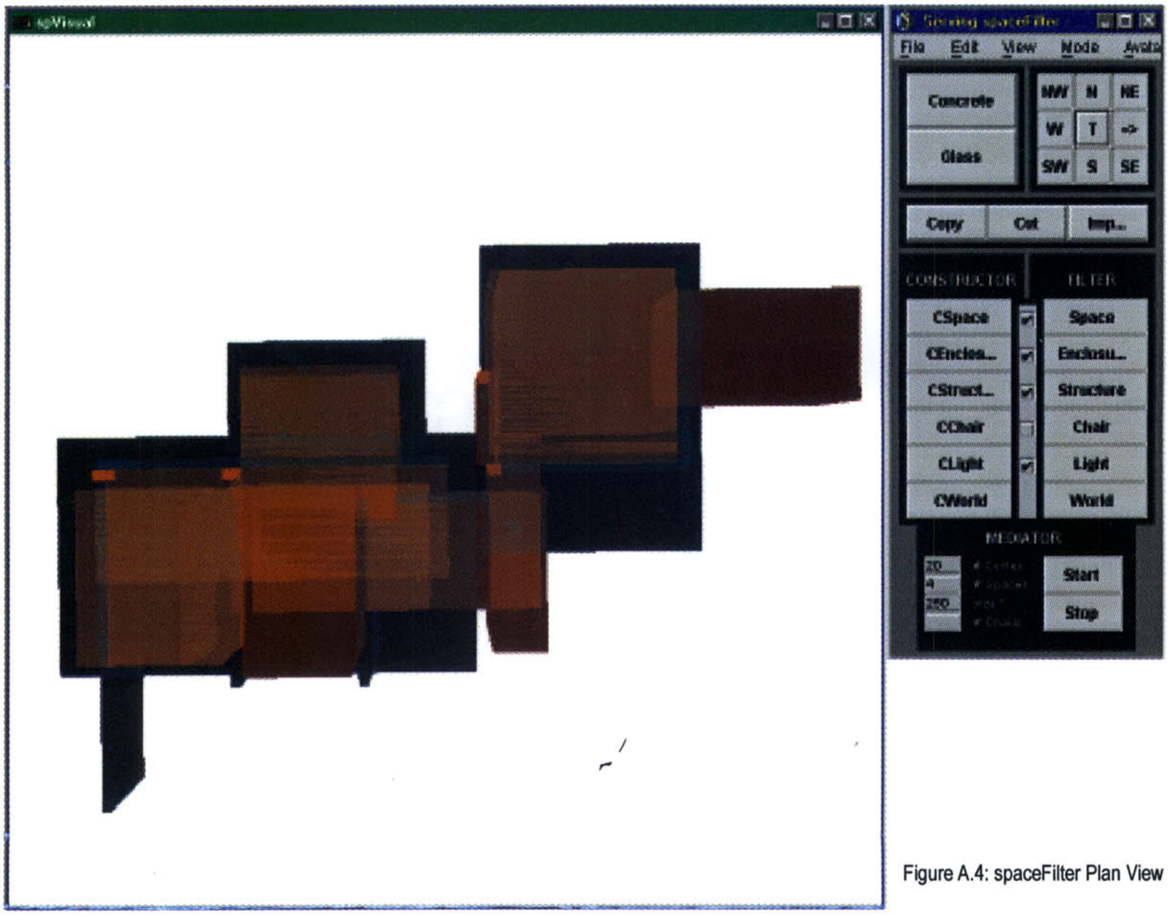


Figure A.4: spaceFilter Plan View

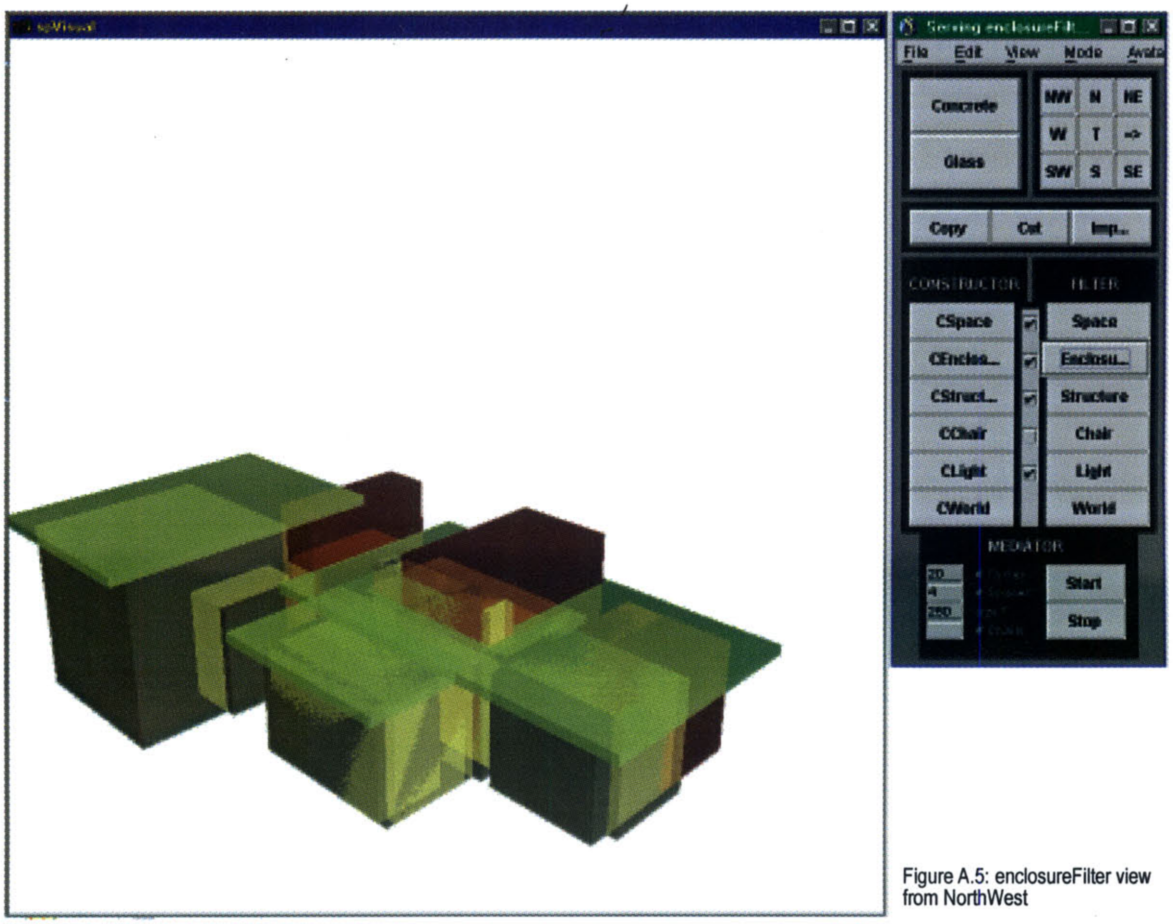


Figure A.5: enclosureFilter view from NorthWest

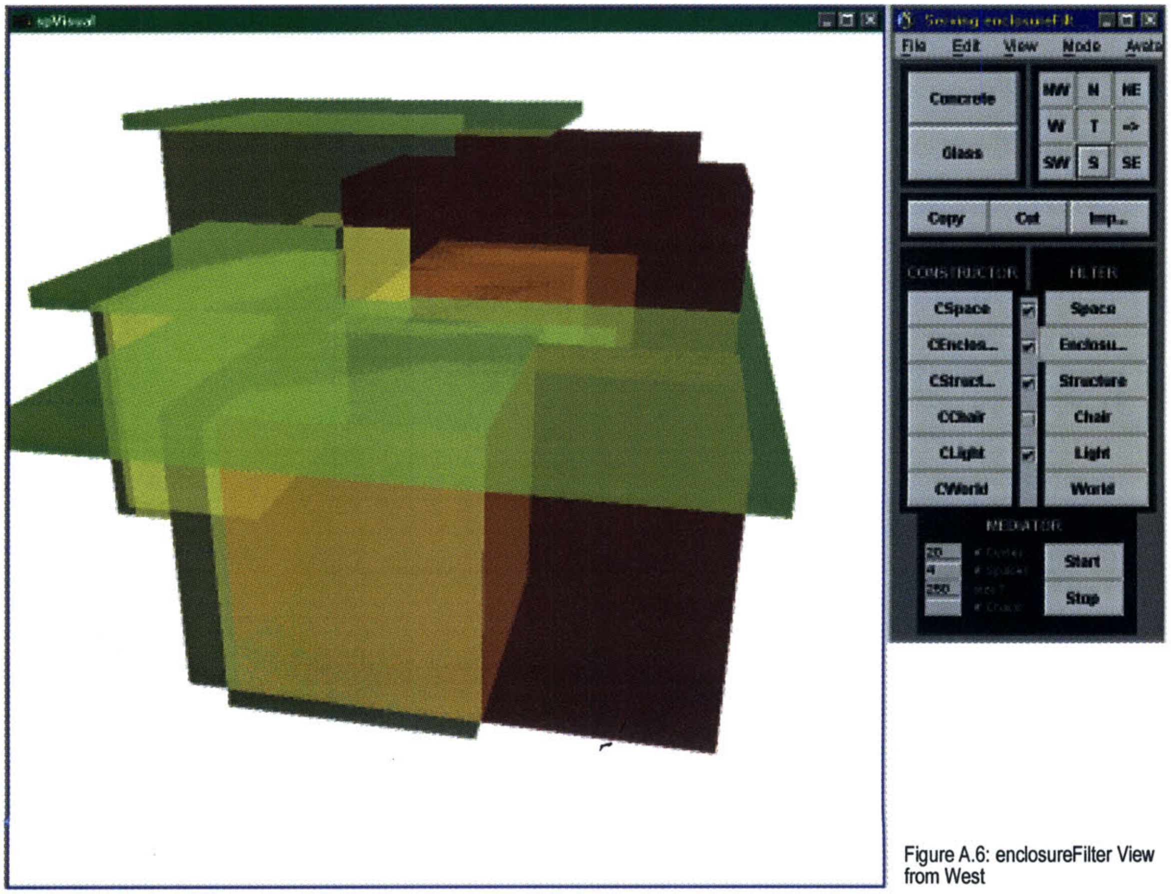


Figure A.6: enclosureFilter View from West

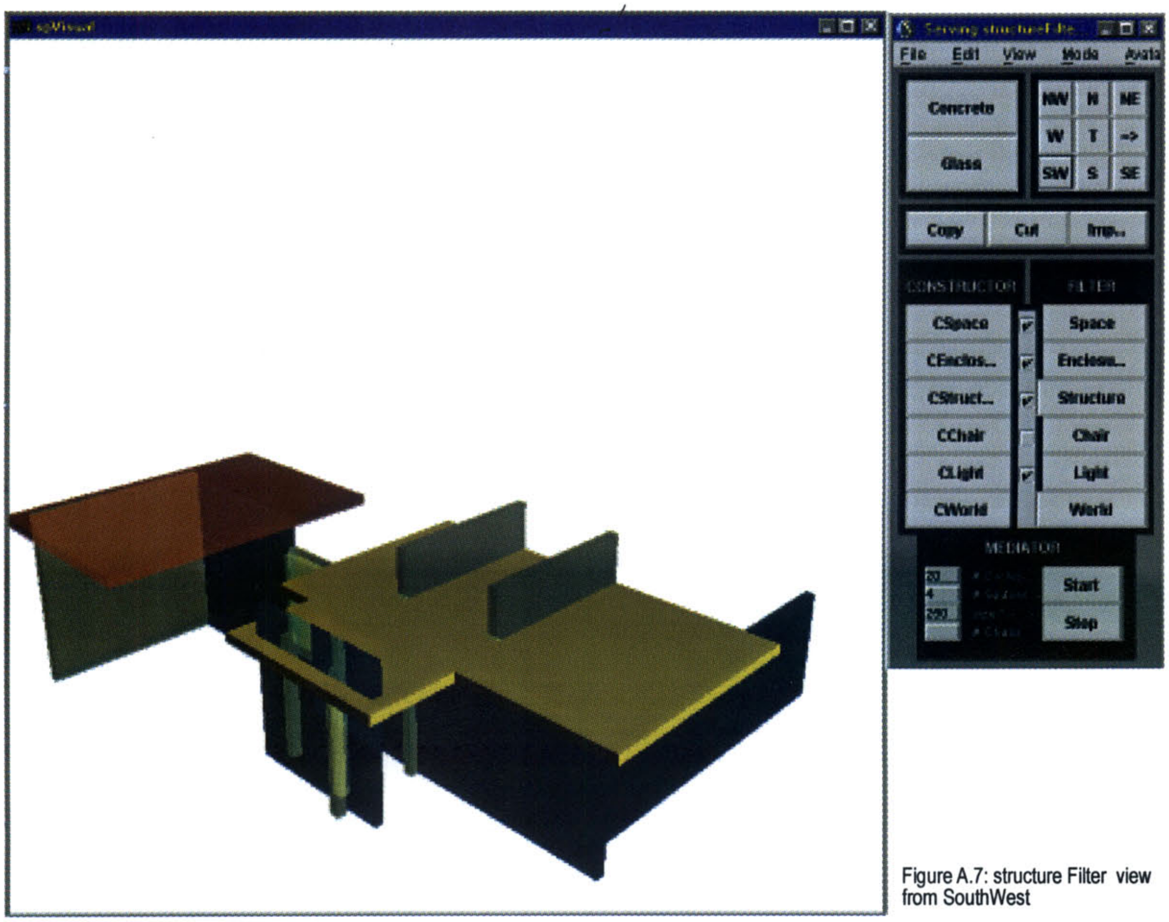


Figure A.7: structure Filter view from SouthWest

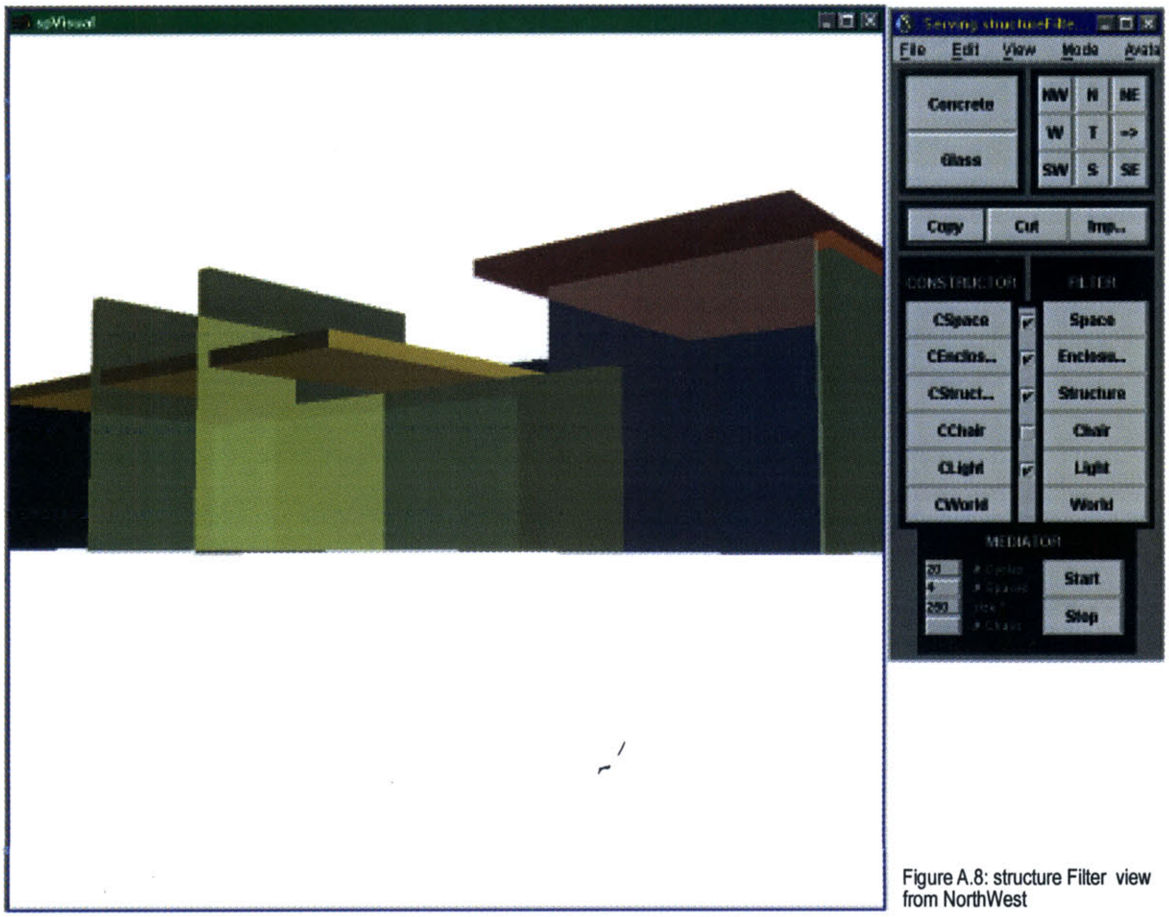


Figure A.8: structure Filter view from NorthWest

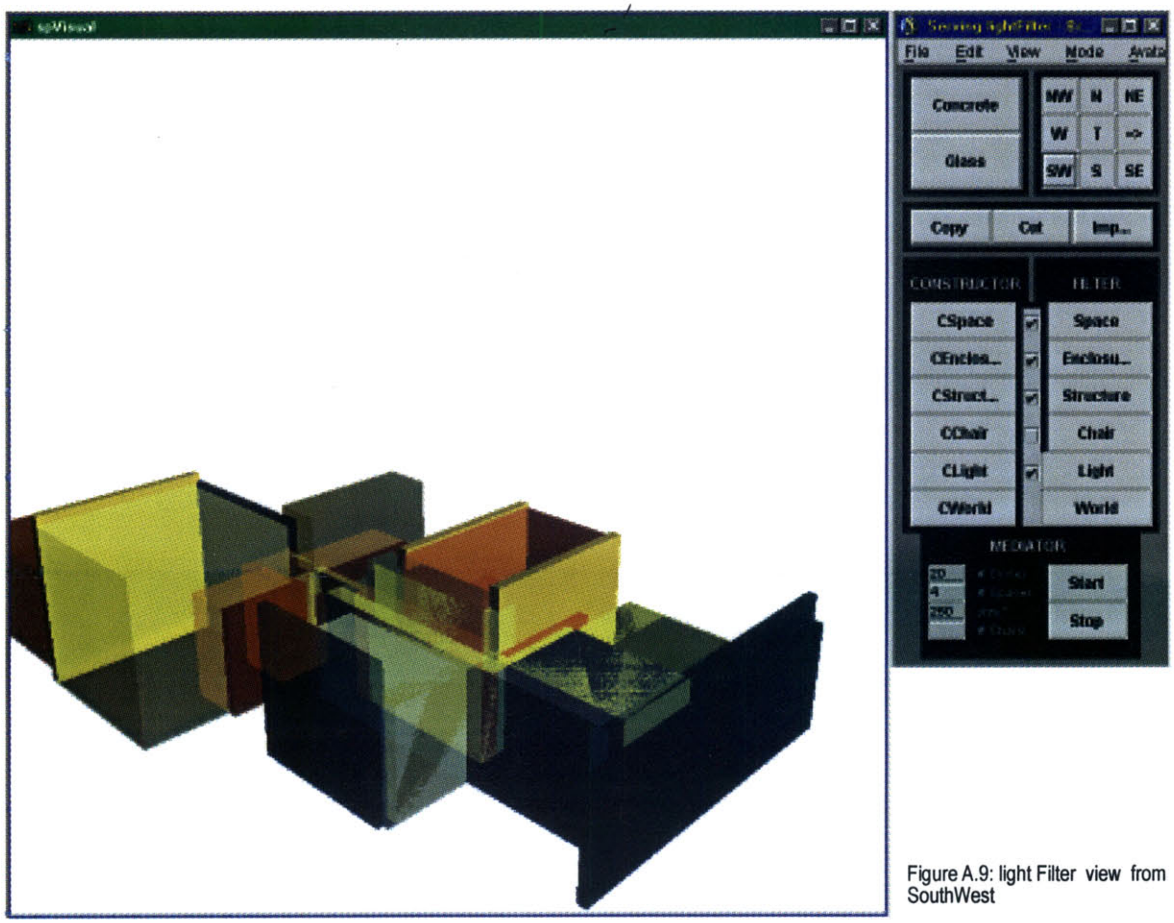


Figure A.9: light Filter view from SouthWest

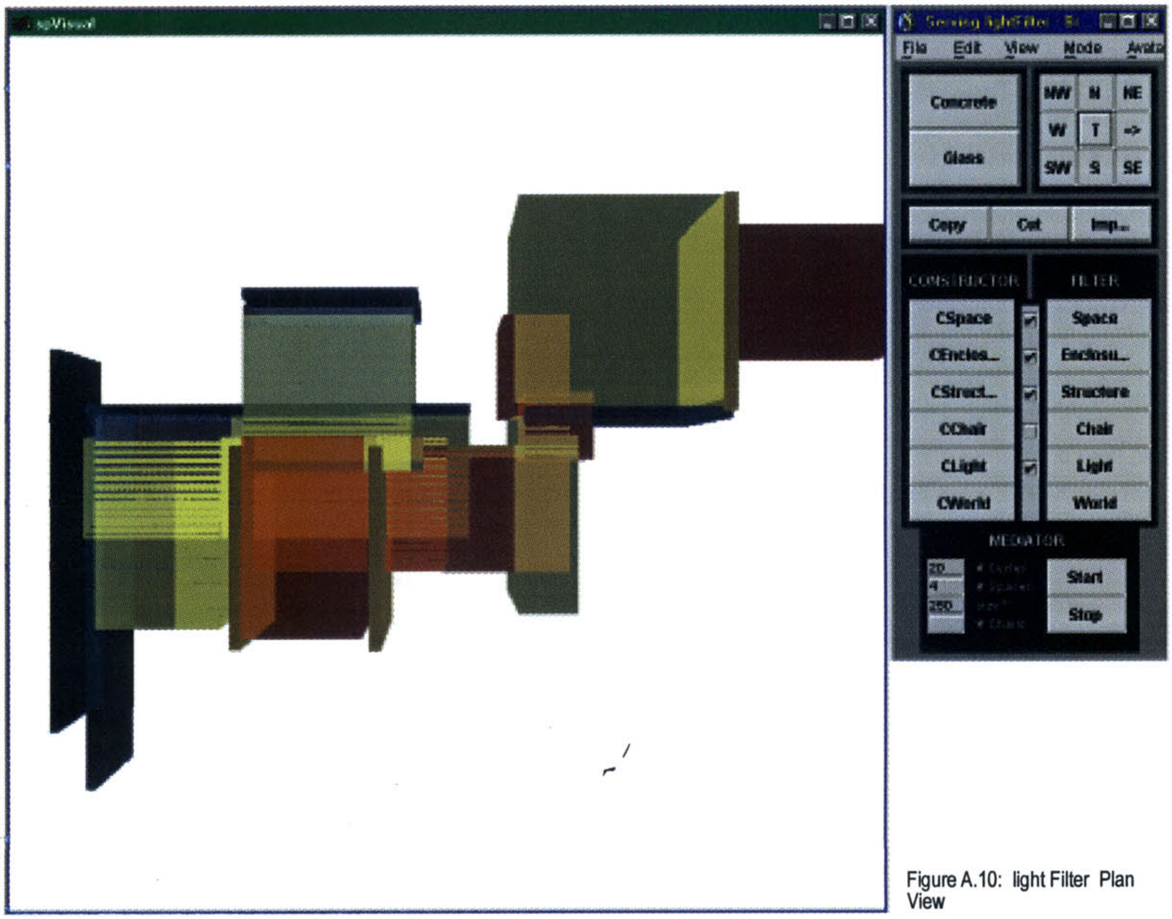


Figure A.10: light Filter Plan View