EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2004)
John F. Hughes and Joaquim A. Jorge (Editors)

A Sketch-Based Interface for Collaborative Design

Zhe Fan[1†] Manuel M. Oliveira[2‡] Chi Ma[1] Arie Kaufman[1]

[1]SUNY at Stony Brook - Department of Computer Science, Stony Brook, NY, 11794-4400, USA
[2] UFRGS - Instituto de Informática, Caixa Postal 15064, 91501-970, Porto Alegre, RS, Brazil

Abstract

We present an interface for collaborative conceptual design that combines sketch elements, direct manipulation of 3D objects and non-photorealistic rendering. Such a combination results in a simple and intuitive 2D-sketch-to-3D modeling system suitable for novice users. It allows users potentially located in geographically distant areas to cooperate by sketching, exploring and modifying their ideas interactively, with immediate visual feedback. Our system prototype supports several modeling primitives and can be extended to handle user-defined objects. Potential applications of our system include early stages of urban and landscape design, rapid prototype of virtual environments, animation, education and recreational use.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodologies and Techniques - Interaction Techniques I.3.3 [Computer Graphics]: Picture/Image Generation - Display Algorithms H.5.2 [Information Interfaces and Presentation]: User Interfaces - Interaction Styles

1. Introduction

Sketching is frequently used during the early stages of conceptual design when ideas are still unfinished. The lack of precision implied by the strokes seems to increase the tolerance of the initial estimate of shape [ZHH96, LM95]. Thus, sketching has been recognized as an important tool for communicating ideas and concepts. While it is traditionally performed using pencil and paper, the ability to interactively explore and refine the original thoughts in a collaborative environment can provide a sense of shared design space, allowing the easy creation of different versions of the project and serving as a valuable teaching tool. Unfortunately, these benefits cannot be fully exploited with the use of pencil and paper. Traditional direct-manipulation 3D design systems (*e.g.*, CAD systems), on the other hand, provide considerable editing support, but usually require accurate descriptions of the models, making them less attractive for brainstorming. Although computers can be used to assist in the sketching process [ZHH96, Pug92, BCF94, LM95], creating 3D models directly from a series of 2D strokes is an ill-posed problem, often lending to undesirable results. As such, just a few applications have tried to mimic the process of sketching 3D objects and only do so under some restrictions [Pug92, ZHH96]. A more flexible approach for sketching 3D shapes of objects presenting symmetry and repeated substructures has been recently proposed [IH01].

We are interested in the use of sketching for conceptual design by distributed teams. In these situations, it would be desirable to allow project members and clients to remotely engage into collaborative design sessions. In practice, however, researchers have reported that distributed design often degenerates into co-located design, requiring participants to physically meet in order to overcome the expressive limitations of traditional communication media, such as phone, fax, e-mail and videoconferencing [LHF*98]. In order to face this challenge, we have designed and implemented an interface for conceptual design that supports collaborative work among groups of users over a network.

In order to be successful such an interface has to be easy to use. In particular, no artistic skills are required to effectively use our system. In order to avoid the difficulties of creating 3D models directly from 2D strokes, we constrain the use of free-hand strokes to the creation of 2.5D objects and to the drawing of 2D profiles that can be extruded to create 3D objects. In general, truly 3D objects are built from an available set of simple 3D primitives, or imported as polygonal

† {zfan|mchi|ari}@cs.sunysb.edu
‡ oliveira@inf.ufrgs.br

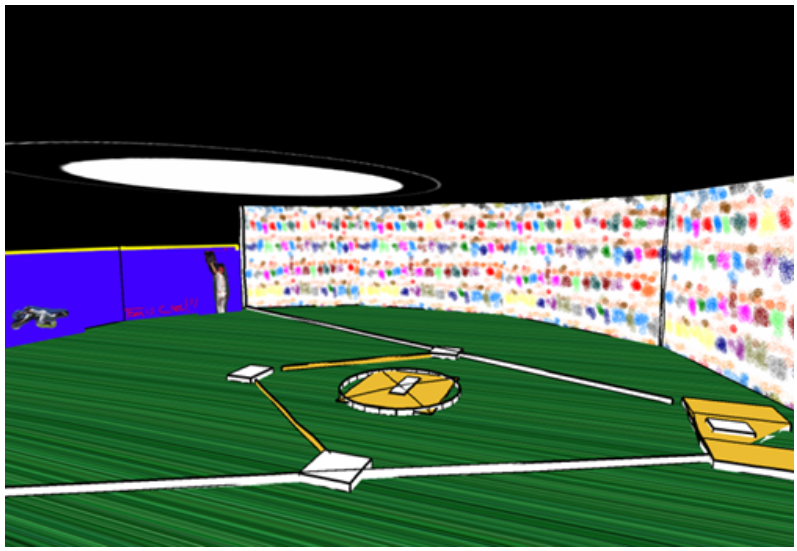

Figure 1: *A baseball field created by a new user of our system after a five-minute tutorial session on how to use it.*

meshes, and rendered using non-photorealistic (NPR) techniques. The rendering process may be complemented with animated billboards and textures, impostors and background images. Traditional interaction techniques are used for creating and editing 3D geometry. Applications of this kind of interface include early stages of urban and landscape design, rapid prototyping of virtual environments, animation, and educational and recreational use.

We demonstrate the effectiveness of the proposed system by having users with no specific artistic skills create pleasing 3D environments in just a few minutes. Figure 1 shows a baseball field created by a new user who was given a five-minute tutorial on how to use the system. This result was produced right after the tutorial in his first solo interactive session with the system. Figure 2 illustrates the use of our interface for the case of a collaborative urban design. The two images correspond to the views of two designers, who independently explore the space but whose modeling actions become immediately visible to all participants. During the exploration, the users are given correct perspective views of the scene.

The remaining of the paper is structured as follows: Section 2 discusses some related work. Section 3 provides an overview of our system's architecture and presents the details of our sketch-based interface. Section 4 comments some of the results obtained with our current prototype. Section 5 concludes the paper with a summary and directions for future exploration.

2. Related Work

Several researchers have explored the use of sketch-based interfaces for creating and editing approximate representations of 3D objects and scenes. This paper describes our experience with the design and implementation of an interface for creating sketch-like 3D scenes primarily intended for collaborative conceptual design. Unlike some related works [Pug92, AHKS94, BCF94, IMT99, IH01, KHR02, TBSR04], we do not focus on object or curve [CMZH99] design, but emphasize the creation of entire scenes. We should point out that most of the elements used in our system have been seen before, in one or another system. However, our work is unique in integrating such elements into a single working prototype that is intuitive and easy to use.

SKETCH is an interface for creating and editing 3D sketches of scenes [ZHH96]. It uses simplified (2D) drawing commands that are interpreted as operations to be applied to objects within a 3D world. Like SKETCH [ZHH96], our system is oriented towards rapid creation of approximate representation of environments. Unlike SKETCH, however, our system is a distributed application and can import 3D models, allowing designers to take advantage of the large collections of 3D polygonal models available in computer graphics. NetSketch [LHF*98] is an application based on SKETCH that supports distributed conceptual design. In spirit, this is the closest to our work, but it is constrained to the shapes that can be created and rendered using SKETCH. NetSketch uses a peer-to-peer network topology and cannot always guarantee model consistency among all users. Our system is based on a client-server model, enforcing a consistent view among all participants.

CALVIN [LJVD96] is a networked collaborative environ-

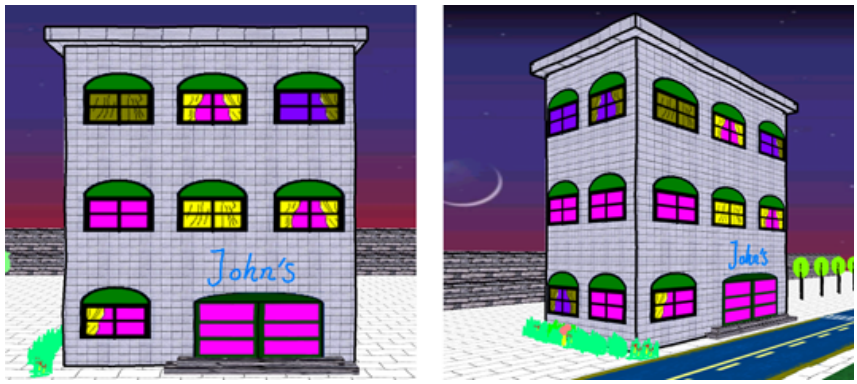

Figure 2: *Independent views by two users during a collaborative design session. The scene is rendered with correct perspective.*

ment for designing indoor architectural spaces that was designed to run in a CAVE and requires the use of VR equipment. Our system, on the other hand, is based on commodity PCs, uses a much simpler interface (CALVIN also supports verbal and video communication channels) and exploits non-photorealistic rendering.

Alice [PBC*95] is a prototyping tool for virtual reality applications that allows programmers to focus on the content of the application. Alice is a flexible and extensible environment, but it requires its users to have some programming skills. Tolba, Dorsey and McMillan [TDM01] showed how a 2D projective drawing system could be used to mimic 3D-like capabilities.

Computer Supported Cooperative Work (CSCW) is the use of computers to support and enhance the work activities of groups [BL99, Gru91]. The technical aspects involving the design of collaborative systems and their infrastructures are particularly challenging, especially because most CSCW applications can potentially support a large number of users. Our system, however, focus on conceptual design, a task involving a relatively small number of concurrent users, which significantly simplifies its design. Compared to collaborative electronic blackboard, our system extends the 2D sketching to an immersive 3D shared environment. Compared to collaborative CAD system [SLS98], our sketch-based interface makes the communication and interaction easier, faster and more creative.

3. System Architecture

This section presents an overview of our system's architecture and describes the details of our sketch-based interface.

The design of our collaborative infrastructure was based on the observations that the number of concurrent users involved in a design session is usually limited to just a few designers and that the rate at which the environment changes is relatively slow. Changes in the design need to be perceived by the users at interactive rates, but not necessarily in real time (*i.e.*, delays of up to several seconds can be tolerated).

Based on these premises, we conceived our system as a client-server application over the Internet. Clients (user sessions) connect to a server using BSD sockets and TCP/IP protocol [Ste94]. TCP's reliable communication and ordering semantics greatly simplifies the system implementation. Such benefits more than compensate for the small delay imposed by the ordering semantics itself. The server is responsible for verifying and maintaining a consistent state of the design and for forwarding updates to all clients in the session. The small number of clients justifies the use of a single server. In the event of a server failure, clients can explore their cached copies, save changes to local databases and later send the updates to the server.

As a user creates a new object, this information is immediately sent to the server who updates its database and forwards the new information to all other clients sharing the session. An equivalent sequence of messages is exchanged when an object is deleted or modified.

From a user's viewpoint, the most important part of the system is the user interface, which is used for sketching and exploration of the resulting 3D environments.

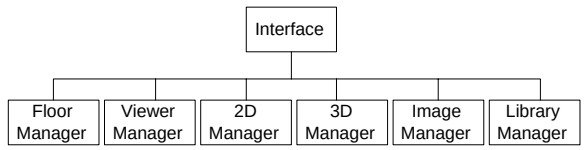

Figure 3: *The user interface is logically organized into six modules.*

3.1. User Interface

The system's interface combines sketching and non-photorealistic rendering and is logically organized in six

modules, called managers, as shown in Figure 3. Each of these modules is explained next.

Floor and Viewer Managers: The Floor Manager provides a top view of the entire environment, allowing users to immediate locate, select, group, move, delete, add, replicate or re-scale any object in the scene. It consists of a scrollable grid showing 2D representations of the camera (position and orientation) and of all objects in the scene (Figure 4 right). The Viewer Manager provides a perspective view of the environment from the camera's viewpoint, allowing interactive exploration of the scene (Figure 4 left).

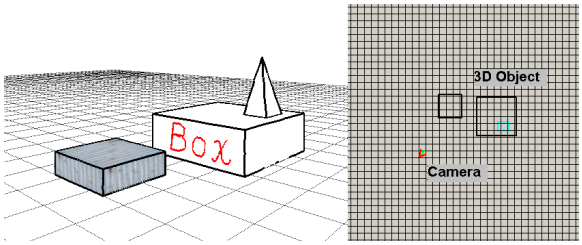

Figure 4: *User interface. 3D view used for creating, editing and exploring the scene (left). The floor manager (right). The v-shaped icon indicates the position and orientation of the camera.*

2D Manager: Sketching is performed directly on the perspective window. As the user sketches on the screen, the drawing is buffered as a set of 2D line segments, which are later projected into the 3D environment. First, the bottommost vertex is projected onto the ground plane, defining a root point (Figure 5) (a similar technique is used in [ZHH96] to decide where to position an object in 3D). The coordinates of the new 3D vertices are obtained by projecting the buffered 2D vertices onto the plane passing through the root point and parallel to the camera's plane (Figure 5). Finally, the sketch is stored and rendered as a series of line segments in 3D. As the user walks through the scene, the sketch is rotated so that it always faces the viewer. We refer to this kind of objects as sketchy billboards, which were used to create the trees and grass shown in various figures in this paper.

Support is also provided for sketching directly on 3D objects. This feature is illustrated in Figures 2 and 4, where the words *John's* and *Box* were handwritten on the building façade and on one side of the box, respectively. The process is similar to the one just described, but, in this case, the planes onto which the sketches are projected on are found by casting rays through the vertices of the strokes into the scene. The new sketches then become attributes of the closest intersected faces. The process is performed in real time on the 3D view.

3D Manager: The construction of 3D environments can be simplified if some 3D primitives are at hand and if the user can take advantage of the large number of 3D models

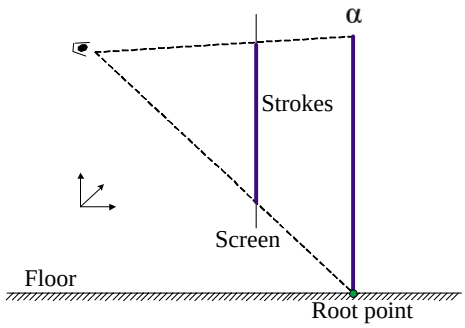

Figure 5: *Mapping 2D sketches to 3D.*

available as polygonal meshes. Our prototype supports geometric primitives and polygonal meshes, which are rendered using NPR techniques. This way, complex 3D scenes can be created in just a few minutes. This flexibility also allows the system to both import and export 3D models from and to other modelers.

The supported geometric primitives include boxes, pyramids, wedges, cylinders and polygons. In order to give these primitives a sketch-like appearance, we assign artificial edges to each primitive by subdividing its original edges into a certain number of segments (this number depends on the length of the edge itself) and adding random offsets to the internal vertices of the segments. The offsets are computed using Perlin's noise function [Per85] during rendering time. Since the noise function returns the same values given the same parameters, the artificial edges are not stored, but computed on the fly during rendering time. For rendering, each primitive face is broken into a triangle fan [WNDS99] with a central vertex at the average position of all the vertices of the original face. A triangle in the fan is defined by the central vertex and the end points of the perturbed segments. All triangles of the model are rendered without outlines followed by the rendering of the artificial edges. This algorithm is a variation of the procedure used to render polygonal objects in SKETCH [ZHH96]. Figure 6 shows a side-by-side comparison of the renderings of some 3D primitives using conventional OpenGL rendering (left) and the sketchy rendering procedure described (right). The buildings shown in Figures 2 and 10, the farm house shown in Figure 7, and the indoor scene shown in Figure 12 were created by texture mapping 3D geometric primitives. Imported polygonal meshes are rendered using silhouette and importance edges [RC99]. The cow shown in Figure 7 and the statue visible in Figure 10 are examples of imported 3D models.

Image Manager: It provides a number of important features, including animated billboards and textures, user-defined textures, impostors and background images. A billboard is a texture-mapped polygon always facing the viewer, which replaces some amount of 3D geometry. Ani-

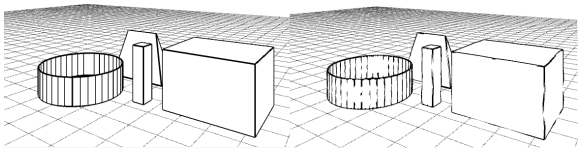

Figure 6: *Rendering 3D primitives. Conventional rendering (left) and sketchy rendering (right).*

mated billboards, introduced here, extend the traditional notion of a billboard by allowing its texture to change over time. They are especially effective for representing amorphous elements, such as clouds, fire and water, traditionally represented using particle systems and procedural modeling. In these billboards, the individual frames of the associated animation are cyclically mapped onto the corresponding polygon. Despite its simplicity, the technique is quite effective and was used to animate the water fountain shown in Figure 11 (please see the accompanying videos at http://www.cs.sunysb.edu/∼fzhe/sketch.html).

Our system provides some painting capabilities to allow the users create and edit their own textures. Such a feature was used to create the windows and doors used in the buildings of sketchy town (Figures 2) and the farm house (Figure 7). A special color is reserved to represent transparency, allowing textures to have arbitrary shapes. Distant geometry can be replaced by planar impostors [MS95]. The system also supports the use of background images. This provides an easy and efficient way to visually enrich a scene. It also provides a way to represent and transition between, for example, various times of the day and different weather conditions. Examples of background images can be seen in Figures 7 and 11.

The Image Manager also provides other features such as a simple painting system and tools for image manipulation. These can be used in combination with other tools to enhance the quality of the sketches. Figure 9 shows a scene containing a pig, a cat and five instances of the same tree. In order to create these objects, closed curves were outlined in 2D on the 3D view and filled with the proper color. The rendering of this kind of objects is performed in a similar way as other sketch objects, always facing the camera. The tree was saved in a library and used in the farm scene shown in Figure 7.

Library Manager. The capabilities of our interface can be extended in an easy way with user-defined libraries. A library is an arbitrary set of objects and provides a natural way for grouping related entities and for sharing them with other users.

4. Results

We have implemented the described interface architecture in C++ and OpenGL and used it to create several virtual environments both in solo as well as in collaborative sessions. The accompanying videos were recorded in real-time using a relatively inexpensive laptop (Pentium III with 1.0GHz, 128 MB of memory and an ATI Mobility Radeon with 16M of memory).

Figure 1 shows a baseball field designed by a new user, after receiving a five-minute demonstration of the interface usage and a five-page manual. The resulting scene was produced as a solo design during his first design session, which last a total of 50 minutes. The bases and other marks on the ground were created in 2D using the Floor Manager and extruded vertically. A green texture was used to represent grass. A texture was hand-painted to represent the spectators and was mapped onto walls created by extruding some sketched lines on the ground. The sky was created as a polygon mapped with a black texture with some white circle representing the light source.

Figure 7 shows a farm scene created in about four minutes with the support of a library. The house was created using two geometric primitives (a box and a wedge). The ground is a texture-mapped polygon. The door and windows (user-defined textures), the cow (polygonal model), the tree (sketchy billboard), the sunset sky (background image) and the fan of the windmill (animated object) were imported from a library. The flowers are sketchy objects created and replicated by the user.

Figure 8: *Views of the concurrent exploration of the farm scene by two users.*

Figures 10 and 11 are views of a town model sketched using our system. In this case, all objects, with the exception of the water fountain (animated billboard) and the statue (3D polygonal mesh), were created from scratch. Two users collaborated on the design, finishing the project in approximately four hours. Most of this time was spent on experimenting with different design possibilities, such as positions and sizes of the buildings, and selection and design of new textures. The brick textures used in the hotel building and on the sidewalks, and the water texture used in the pool were found on the web; the remaining textures were hand-drawn.

Figure 7: *Farm scene. It illustrates the use of several features available in our system's interface.*

Figure 2 shows a concurrent exploration of a design space by users. As one designer modifies the scene, the effect is immediately reflected in the other user's view. Alternatively, users can independently create different parts of a project locally (using the client application without connecting it to a server) and integrate them later during a joint review. Figure 8 shows the concurrent exploration of the farm scene by two users.

Figure 12 shows an indoor environment modeled using only simple primitives and textures. In this virtual environment, the computer display shows an animated texture creating the effect that a movie is playing on the screen. Figure 13 shows a house sketched during a collaborative session involving two users. In this scene, most of the elements were sketched directly on the scene window.

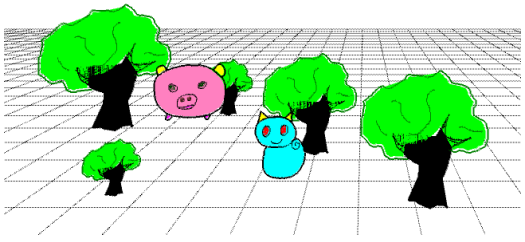

Figure 9: *The pig and the cat. A 2.5 scene sketched using our system's interface.*

5. Summary and Future Work

This paper described the design and implementation of an interface for conceptual design that combines sketch elements,

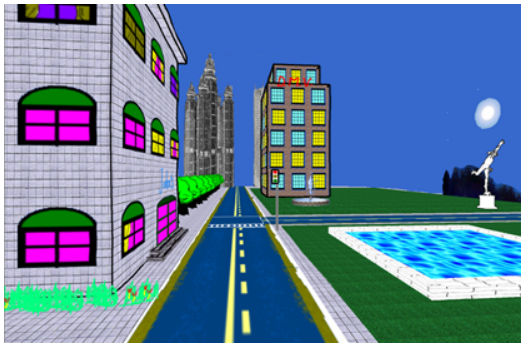

Figure 10: *View of a 3D urban design created by two users during a collaborative session. The statue on the right is a 3D polygonal model rendered using NPR techniques.*

direct manipulation of 3D objects and non-photorealistic rendering. Such a combination results in a simple and intuitive 2D-sketch-to-3D modeling system suitable for novice users. It allows users potentially located in geographically distant areas to cooperate by sketching, exploring and modifying their ideas interactively, with immediate visual feedback. The resulting system can find potentially applications in urban and landscape design, rapid prototyping of virtual environments, animation, education and recreational use.

Although our system makes use of several known techniques, it is unique in combining them to create an intuitive and easy-to-use interface for supporting collaborative design. Moreover, it provides strong evidence that scene design can be successfully performed in collaborative virtual environments.

The users who have tried our system prototype so far are

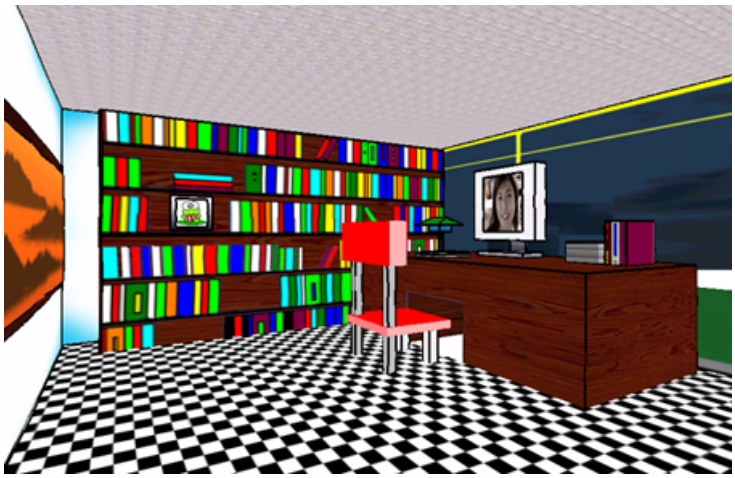

Figure 12: *Indoor scene created using simple primitives and textures. In this 3D virtual environment the computer display shows a movie represented as an animated texture.*

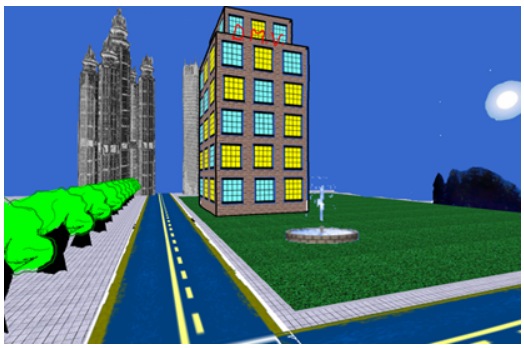

Figure 11: *View of a sketchy town. The water fountain on the grass is an animated billboard. The buildings in the back are impostors.*

not professional designers. As such, several important questions still need to be addressed. For example, can this system significantly reduce the need for co-located design? Can it change the way designers collaborate? We intend to carry out studies with real designers and evaluate their level of satisfaction. Feedback from real users should help us to identify ways of improving the system.

The current version of our prototype has some limitations. For example, one does not know about the presence of other users in the environment, unless they are performing some noticeable actions. Notifying all participants every time somebody logs in or logs out the server is a simple way of fixing this problem. We also need to improve the way sudden events, such as the removal of an object, are perceived by other participants. LaViola et al. [LHF*98] discuss some ways of handling such events.

All tests carried out so far involved computers connected to a local area network. While design teams are usually located in close proximity, the full potential of our system can be realized over the Internet by supporting true remote collaboration. Thus, we still need to measure the possible impact of network latency on the users' ability to perform collaborative work. One should notice, however, that conceptual design only requires interactive, not real-time, update rates.

Some informal experiments indicate that pre-school children are attracted by the cartoon-like renditions produced by our system. The ability to interactively create and explore virtual environments using a system like ours seems to be a powerful tool for creative storytelling.

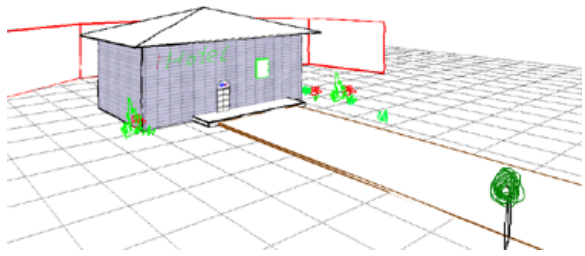

Figure 13: *A house sketched during a collaborative design session.*

Acknowledgements

The authors would like to thank Thomas Nasti, Fatma Altinbas and Lei Zhang for testing the system and providing some feedback. We would also like to thank the anonymous referees for their insightful suggestions. Figure 1 was created by Thomas Nasti.

This work was partially supported by an NSF grant CCR-0306438. The initial stages of this work have been carried out while Manuel M. Oliveira was at SUNY Stony Brook.

References

[AHKS94] AKEO M., HASHIMOTO H., KOBAYASHI T., SHIBUSAWA T.: Computer graphics system for reproducing three-dimensional shape from idea sketch. *Computer Graphics Forum 13*, 3 (1994), 477–488. 2

[BCF94] BRANCO V., COSTA A., FERREIRA F.: Sketching 3D models with 2D interactive devices. *Computer Graphics Forum 13*, 3 (1994), 489–502. 1, 2

[BL99] BEAUDOUIN-LAFON M. (Ed.): *Trends in Software: Computer Supported Co-operative Work*. John Wiley & Sons, 1999. 3

[CMZH99] COHEN J., MARKOSIAN L., ZELEZNIK R., HUGHES J.: An interface for sketching 3d curves. In *ACM Symposium on Interactive 3D Graphics* (1999), pp. 17–21. 2

[Gru91] GRUDIN J.: CSCW. *Communications of the ACM 34*, 12 (December 1991), 30–34. 3

[IH01] IGARASHI T., HUGHES J.: A suggestive interface for 3d drawing. In *Symposium on User Interface Software and Technology* (2001), pp. 173 – 181. 1, 2

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3d freeform design. In *SIGGRAPH* (1999), pp. 409–416. 2

[KHR02] KARPENKO O., HUGHES J., RASKAR R.: Free-form sketching with variational implicit surfaces. *Computer Graphics Forum 21*, 3 (2002), 585–594. 2

[LHF*98] LAVIOLA J., HOLDEN L., FORSBERG A., BHUPHAIBOOL D., ZELEZNIK R.: Collaborative conceptual modeling using the sketch framework. In *IASTED International Conference on Computer Graphics and Imaging* (1998), pp. 154–157. 1, 3, 8

[LJVD96] LEIGH J., JOHNSON A., VASILAKIS C., DEFANTI T.: Multi-perspective collaborative design in persistent networked virtual environments. In *IEEE VRAIS* (1996), pp. 253–260. 3

[LM95] LANDSAY J., MYERS B.: Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (1995), pp. 43–50. 1

[MS95] MACIEL P., SHIRLEY P.: Visual navigation of large environments using textured clusters. In *Proceedings of ACM Symposium on Interactive 3D Graphics* (1995), pp. 95–102. 5

[PBC*95] PAUSCH R., BURNETTE T., CONWAY M., DELINE R., GOSSWEILER R.: Alice: A rapid prototyping system from virtual reality. *IEEE Computer Graphics and Applications 15*, 3 (May 1995), 8–11. 3

[Per85] PERLIN K.: An image synthesizer. In *SIGGRAPH* (1985), pp. 287–296. 5

[Pug92] PUGH D.: Designing solid objects using interactive sketch interpretation. In *Proceedings of ACM Symposium on Interactive 3D Graphics* (1992), pp. 117–126. 1, 2

[RC99] RASKAR R., COHEN M.: Image precision silhouette edges. In *Proceedings of the Symposium on Interactive 3D Graphics* (1999), pp. 135–140. 5

[SLS98] STORK A., LUKAS U. V., SCHULTZ R.: Enhancing a commercial 3d CAD system by CSCW functionality for enabling co-operative modelling via WAN. In *Proceedings of the ASME Design Engineering Technical Conferences* (Atlanta, September 1998). 3

[Ste94] STEVENS W. R.: *TCP/IP Illustrated*, vol. 1. Addison Wesley, 1994. 4

[TBSR04] TSANG S., BALAKRISHNAN R., SINGH K., RANJAN A.: A suggestive interface for image guided 3d sketching. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (2004), pp. 591–598. 2

[TDM01] TOLBA O., DORSEY J., MCMILLAN L.: A projective drawing system. In *Proceedings of the ACM Symposium on Interactive 3D Graphics* (2001), pp. 25–34. 3

[WNDS99] WOO M., NEIDER J., DAVIS T., SHREINER D.: *OpenGL Programming Guide, 3rd Ed.* Addison Wesley, 1999. 5

[ZHH96] ZELEZNIK R., HERNDON K., HUGHES J.: Sketch: An interface for sketching 3d scenes. In *SIGGRAPH* (1996), pp. 163–170. 1, 2, 4, 5