Eurographics Workshop on Sketch-Based Interfaces and Modeling (2005)
Takeo Igarashi, Joaquim A. Jorge (Editors)

3D6B Editor: Projective 3D Sketching with Line-Based Rendering

Kiia Kallio

Media Lab, University of Art and Design Helsinki UIAH

Abstract

In this work, a system for 3D sketching is presented. Instead of trying to process 3D data as geometric models constructed of surfaces or solids, the data is stored as 3D lines, preserving their form exactly as entered by the user. The 3D input of the system works by projecting 2D input from a single viewpoint to a grid surface that can be manipulated in real time. This enables creation of sketches with complex non-planar 3D strokes while still retaining the essence of pen and paper based sketching.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Interaction techniques; J.5 [Computer Applications]: Architecture; J.5 [Computer Applications]: Fine Arts

1. Introduction

Freehand sketching has an important role in the early phases of the design process. Even today, most of the designers prefer doing this by hand with pen and paper. There are many reasons for this: pen and paper are extremely intuitive to use and they are always accessible, the preliminary character of the design can be better communicated with sketchy presentation [SSLR96] and moreover, sketching serves the cognitive processes of the designer [Lim03].

Many design tasks involve thinking in three dimensions. Although skilled designers can visualize their thoughts with perspective drawings, plans, elevations and sections, a number of these 2D drawing is required in order to provide good overall view of a 3D design. This task could be improved by transferring the sketching process from 2D to 3D.

When freehand sketching is performed in 3D space instead of 2D paper, the desired result has to be something that can be visualized with 3D rendering techniques. Typically this means representing the geometry as surfaces. This requires however that strokes entered by the designer are somehow interpreted as geometric shapes suitable for rendering. This also limits the 3D sketching applications to specific domains, as the interpretations made by the application depend on the task in question. The 3D rendering techniques are also developed mainly for photorealistic rendering. This is not well suited for visualizing the incomplete nature of sketches [SPR*94], and thus various non-photorealistic rendering techniques have been developed for solving the problem.

The 3D sketching process may first transfer the strokes drawn by the user into a 3D model — solid or surface representation — and then render the model with a non-photorealistic process imitating hand-drawn graphics. Since the input data already consists of hand-drawn strokes and output data imitates them, it is reasonable to ask the question whether the stage of using a full 3D model is necessary at all in this case. Provided that the user can define the strokes accurately enough in three dimensions, the original strokes should be enough for the whole process of producing a 3D sketch.

The roots of sketch-based modeling are in the attempts of automatic transfer of paper-based engineering plans into 3D geometry dating back to the 1960s [CPC04]. Since then, the computer systems have evolved a lot and can provide real-time performance and interactive operation with very complex sets of data, thus being able to process noisy data rather than just clean and accurate geometric representations. The scope of usage for sketch-based techniques has also become much wider. The desired outcome of the process is typically not accurate engineering data, but incomplete and ambiguous in nature. This makes accurate surface representations irrelevant for the sketching purposes — they are only needed at the later stages of the design process.

Current computer systems can render massively large polygonal models. However, the increase in performance has also made them faster with line rendering. Whereas the first computer-generated 3D renderings were simple wireframe models, an interactive line-based rendering architecture is no longer limited to this level of simplicity. Lines can now be used creatively in the same manner as with 2D sketches: for emphasizing the importance of some feature with multiple overlapping strokes or for describing a surface with hatch patterns. When sketching with pen and paper, the designer doesn't usually draw a perfect shape with a single stroke, but the final outline is defined by successive overlapping strokes — see Figure 1. This is an intuitive way of working for the designer, yet a rarely supported feature in sketch-based interfaces, which rely on the input based on single strokes. Also, the imperfect result of this drawing style can actually make the sketching process more rapid as the designer concentrates on the whole rather than on the details [MB98].

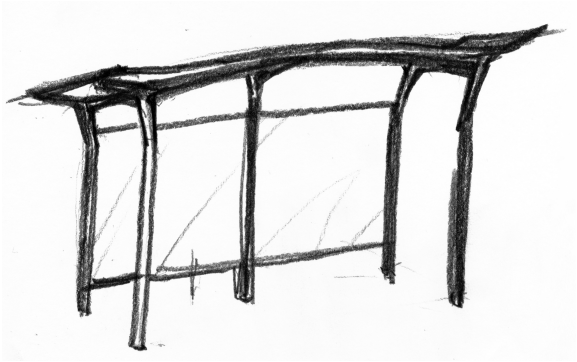

Figure 1: *A sketch drawn by hand with a 6B pencil.*

2. Related work

The research on sketch-based 3D modeling revolves around the idea of how to transform the most natural way of producing images, drawing by hand, to production of 3D data.

One approach to this is to move the process completely to 3D by using 3D input devices, such as tracking devices, data gloves or armatures. Various such 3D input devices are in commercial production, but typically they are used for other areas of 3D input. Systems using 3D input devices for sketching are for instance 3-Draw [SRS91], 3DM [BDHO92], HoloSketch [Dee95] and the system presented by Diehl et al. [DML04]. Since the approach used with 3D6B is not using 3D input devices, these methods won't be discussed in detail here.

One system that uses special 3D input hardware but still resembles the input mechanism of the 3D6B editor is Tractus [SS05], a device where a drawing tablet is attached to a stand that allows vertical movement. This makes the device able to maintain direct spatial mapping between real and virtual spaces. Notable is the method how Tractus allows drawing of non-planar curvature. While drawing with a pen on the tablet surface and moving the tablet vertically, the user can construct complex 3D paths that would otherwise be difficult to input with 2D methods.

The other approach is to use 2D input and map that to 3D. The 3D6B editor belongs to this category.

There are two main research paths in this area: gesture and reconstruction-based models. Gestural modeling systems use predefined gestures for geometric modeling operations. Such systems are for instance SKETCH [ZHH96], Teddy [IMT99] and one presented by Cherlin et al. [CSSJ05]. Reconstructional modeling systems use geometric reconstruction for building a 3D model from a 2D view of the object. Very thorough discussion of these techniques is provided by Company et al. [CPC04].

Typical for gestural and reconstructional models is the goal of producing 3D model data, i.e. 3D surfaces or solids from 2D path input. This is useful if the technique is applied to CAD or some other process that requires accurate results. However, in the sketching context, accurate 3D rendering is not desirable. Therefore some systems, such as aforementioned SKETCH [ZHH96] and Teddy [IMT99], use non-photorealistic rendering methods.

A third path in the research has been emerging more recently. These systems use the strokes or paths directly without attempting to convert them to 3D models. Cohen et al. [CMZ*99] present a system in which a 3D curve is defined by first drawing its screen plane projection and then its shadow on the floor plane. Because of the two-pass nature of the process, it is mostly suitable for entering camera paths. Tolba et al. [TDM99] present a system where sketches of 3D scenes with fixed camera position can be made. However, because of the fixed viewpoint, the results are panoramic sketches, not full 3D with six degrees of freedom.

Most closely related previous works use projection of strokes to a 3D plane in order to determine the 3D coordinates. This is similar to the method used by the 3D6B editor.

Sketchpad+ [Pic98] uses a large video display along with pen input. One operation mode of the software is line sketching, where sketched strokes are projected to a user-definable grid in the 3D space. The strokes are rendered as lines, but it is also possible to use them for defining edges of NURBS surfaces.

Bourguignon et al. [BCD01] use a method where the projection is done to a plane aligned to the screen plane, with an additional possibility to draw strokes that join two separate objects. The strokes are rendered as thick strokes with a method that gives more emphasis to strokes that are drawn from a viewing direction close to the view of the camera.

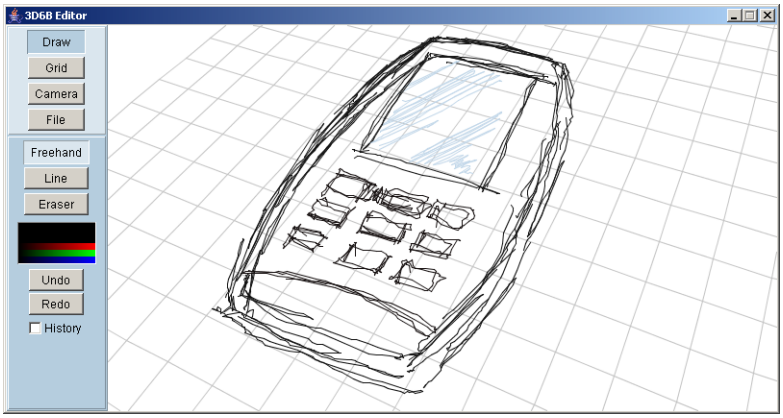

Figure 2: *A screenshot of the 3D6B editor interface.*

3. Overview

The 3D6B editor is a Java-based program for producing 3D sketches. It uses 2D input for generating projective strokes on a user-definable 3D grid. The rendering is done with a line-based renderer, and the data is never converted to 3D surface models but rather stored as strokes. Figure 2 displays a screen capture of the 3D6B editor.

The 3D6B editor was developed as a part of the 3D6B project, visualization and 3D interface study project at Media Lab of University of Art and Design Helsinki UIAH. The 3D6B project was researching and developing collaborative methods for design projects. The outcome — only parts of which were developed into a software implementation — was a web-based service where participants could create and edit, archive and share new project ideas in their initial stages as 3D sketches. The web site of the project is at `http://mlab4.uiah.fi/projects/3D6B/` and the 3D6B editor package, along with full source code, is available at `http://mlab.uiah.fi/~kkallio/3d6b/`.

The goal of the project was to allow designers the same freedom as with pen and paper, but with two additional aspects: 3D and communication. The “6B” in the name comes from 6B pencil, a common tool of designers used for sketching.

Since the freedom of pen and paper was desired, no special 3D input devices were considered. Instead a drawing tablet — or even just a mouse — was chosen as the input method. Also, the program had to be a web-based, cross-platform program in order to be accessible from any computer with Internet access, not just at the design office.

The communication framework, a kind of 3D message board, was mostly developed separately of the editor. The editor, however, has some features, such as editing history, that support the communicative aspect.

The web-based revisioning system of the 3D sketches works with the same principle as a message board: each version of the sketch is a separate “message”, and these are arranged into threads of successive versions. Sketches that are sent to the web service can’t be edited anymore, but when the author — or somebody else in the design team — wishes to edit or annotate the sketch, a new version of the sketch is created and the earlier version remains accessible.

3.1. Drawing

The 3D sketching operations in the 3D6B editor are implemented with projective strokes. The 3D space where the sketch is constructed contains a user controllable grid surface. The user draws the shapes with a 2D pointing device, typically a mouse or a drawing tablet. These 2D coordinates are mapped to the viewplane in the 3D coordinate system. The final 3D position of the point is calculated from a ray originating at the viewpoint, going through the viewplane in the defined position and finally intersecting the grid surface, as illustrated by Figure 3. From the point of view of the user this means that when the stroke is rendered, the 3D points are projected to exactly the same 2D coordinates from which they originated, and thus the sketching process is very similar to 2D sketching.

The grid can be manipulated with full six degrees of freedom, thus making it possible to transform the grid surface freely in the 3D space. Even if the transformation of the grid surface can be freely defined, projecting strokes to a fixed plane can produce only planar geometry. For most usage cases this is enough. Some shapes however, for instance 3D spirals, are impossible to draw with only planar projection. 3D6B provides two ways for drawing non-planar shapes. First, the grid is not limited to a plane, but can be bent to a parabolic shape. This makes it possible to draw non-planar curvature directly or in smaller segments. Second, the grid can be manipulated interactively using keyboard commands. This makes it possible to use the 3D6B editor in a similar

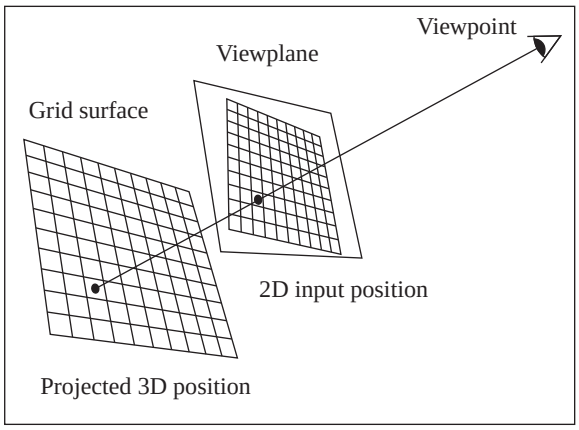

Figure 3: *Projection to the grid.*

manner as Tractus [SS05] for drawing non-planar curves. For instance a 3D spiral can be drawn easily by moving the grid with the keyboard while drawing 2D circles, as illustrated by Figure 4. This technique is also used for drawing the human scale figure at the bus stop in Figure 5.

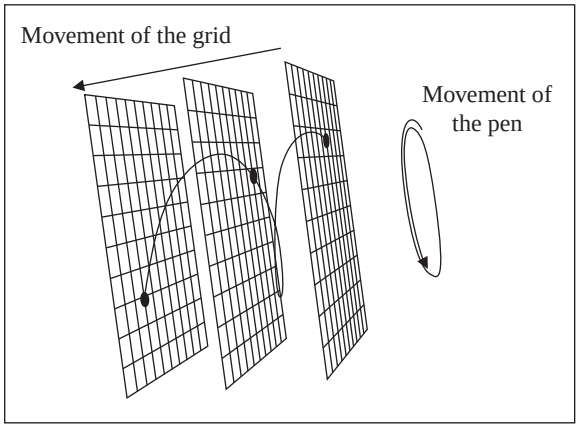

Figure 4: *Drawing a spiral.*

The bending equation of the grid surface is $y = ax^2 + bz^2$, where a and b define the bend values. In addition to this, the grid has a transformation matrix that defines its position and orientation in the 3D space. The projection to the grid surface is done in the local coordinate system of the grid, so that when a ray from the camera through the viewplane is calculated for finding the intersection, it is transformed to the coordinate system of the grid. The equation for the intersection of the ray and the surface is an equation of second degree, and is thus trivial to solve.

The equation can produce zero, one or two solutions. The result is obvious when one solution is found. In the case of two solutions, the one closer to the camera is used. Zero solutions means that the ray doesn't hit the grid surface at all and therefore there is no stroke generated. When a point is found, it is transformed back to the world coordinate system.

Since the strokes entered by the user are stored and rendered without making any interpretations about their meaning, it is completely up to the user to decide how to use the strokes in the sketch. Figure 5 illustrates this: the skeleton of the bus stop is represented by heavy overstroking that defines a 3D volume, the wall and the roof are mainly represented as character lines of the edges, and the ground is defined with a hatch pattern that depicts the shadow of the construction.

Figure 6 illustrates various levels of sketchy visualization. As the amount of detail in the sketch increases, the sketch is perceived as a more accurate representation of the object in question.

3.2. Rendering

The rendering in the 3D6B editor is done with a line-based 3D engine. The architecture of the engine is similar to a polygon-based engine, except that the only supported 3D primitive is a set of lines. In freehand drawing, each stroke consists of a list of smaller line segments. The engine uses a 2D line rendering API. Since the rendering is done in 2D, line segments are sorted and rendered in back-to-front order. Depending on the version of the Java runtime, 2D rendering can use antialiasing for higher display quality.

In 3D sketching, there are several benefits of using a rendering architecture based on lines.

First, lines directly represent the data the user feeds to the system. By making no assumptions about the meaning of the entered strokes and leaving the interpretation to the user, the software doesn't interfere with the cognitive processes involved in the sketching. The cycling process going on between the brain, hands, sketch and eyes [Las80] is not interrupted by the interpretations made by the computer.

Second, since the system renders the strokes inputted by the user directly, the results will look sketchy automatically. Sketchy visualization is important for communicating the preliminary nature of the design [SPR*94]. If the internal data representation of the system is based on 3D surfaces, non-photorealistic rendering techniques are required for reaching this goal. When the internal data is stored as strokes, there is no need for this process.

3.3. Communication

One of the main motivations in the 3D6B project was to enhance communication and collaboration in the design process. Although this paper concentrates on the 3D6B editor, the requirements of the project as a whole played a vital part in the development of the editor.

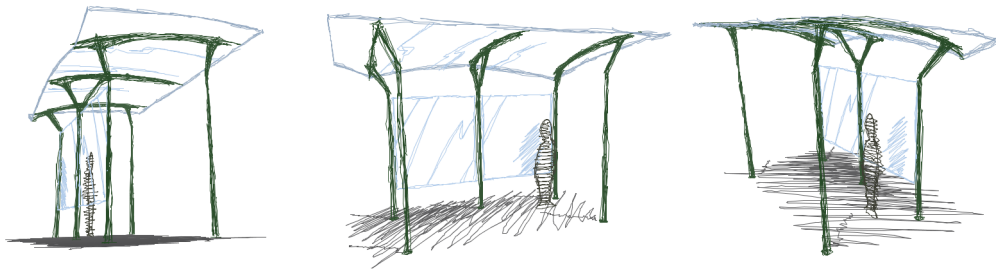

Figure 5: *Some views of a 3D sketch for a bus stop, produced with the 3D6B editor.*

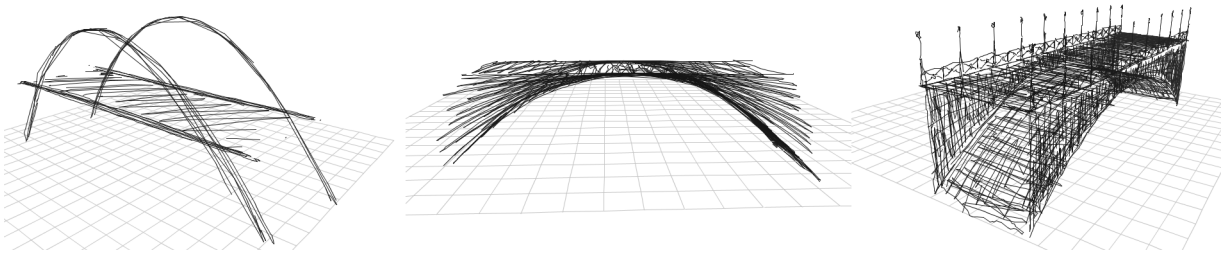

Figure 6: *3D sketches of the same theme with varying level of detail.*

An important goal in the process was to achieve the freedom of pen and paper. A crucial aspect in this is the ubiquitous nature of pen and paper: They are available everywhere. Even those who don't carry a sketchbook can always use something like beermats or cocktail napkins for sketching.

With the popularity of the Internet, computers are everywhere. Also, several applications that were originally client applications — such as e-mail — have moved to the web. With web-based interfaces the users have the ability to use the application from any location, as long as a computer with an Internet connection is available. A web service-based architecture for storing the data also means that the data is not assigned to a single storage location, but is accessible from anywhere. To fully benefit from this potential, the software must run on any Internet-enabled computer.

When this principle is applied to sketching software, no special input devices can be used, as they wouldn't be available outside the design office. Also any software that requires complex installation and maintenance processes is ruled out. Therefore, the 3D6B editor is implemented as a light-weight Java applet that can be embedded to a web page.

Sometimes a computer is available, but no Internet connection. For those situations the 3D6B editor can be used in application mode, independent of the web services. The package is the same as in the applet mode, but if launched from local storage, the application has operations for loading and saving files instead of connecting to the web-based database.

The web service of 3D6B operates with a similar concept as a message board, using both the sketches and textual descriptions as messages. Although most of the functionality is in the server side implementation, the client also has some features for this. One is the ability to display the editing history of the sketches by showing strokes entered during each editing session with different colors. The 3D6B editor can also export the data to images and VRML models when used in application mode. In the web-based mode, this export happens automatically, as the browsing of sketches in the other parts of the web service use images and VRML for displaying the sketches.

3.4. Interface

The user interface of the 3D6B editor is based on the classic WIMP paradigm. As the application is used in a context of the web page, a paradigm similar to the surrounding web browser and operation system is best suited for the task. In this sense the 3D6B editor resembles typical 2D drawing packages.

The input of the editor is handled by the keyboard and a pointer device, i.e. a mouse or a drawing tablet. The drawing tablet is supported in the same manner as a mouse, and no tablet API is used. Therefore features such as pressure sensitivity or pen angle are not available for the application.

The interface of the 3D6B editor is divided to four modes. The first is the drawing mode, where the user can draw the strokes. The camera view and the grid positioning can be edited in two additional modes. The fourth mode is for file and web operation.

In the drawing mode, there are three drawing tools: freehand, line and eraser. These work in the same way as with 2D drawing software. The user can also change the current drawing color with RGB sliders. The software also has an undo buffer, and thus supports undo and redo operations.

In the grid mode, the user can modify the grid with a set of tools. This is done by selecting the tool and dragging with the mouse in the editing window. The grid manipulation operations are move, push, roll, rotate and bend. These operations are also available through keyboard shortcuts. The mapping of the shortcuts is such that it is possible to operate the grid with one hand from the keyboard while drawing the sketch with the other hand.

The camera mode operates in a similar manner as the grid mode. Tools for modifying the camera allow free movement in 3D, and rotation either around the camera position or around the grid center. The focal length of the camera can be also changed.

The file mode operates a bit differently depending on whether the editor is used through the web or as a local application. In the application mode, operations for loading and saving sketches, as well as for exporting them as images or VRML models are available. Since the web-based mode operates with a similar concept as a message board, the server takes care of that the correct file is loaded when the applet is started. Also, the saving is just an upload operation, and the server takes care of that the data is inserted to the correct location in the database. In addition to the drawing, the web-based mode allows attaching textual messages to annotate the drawings.

4. Discussion and future work

The 3D6B editor hasn't gone through rigorous user testing, and therefore the results from the usability of the prototype are only preliminary.

The prototype seems to work well in the context of sketching 3D objects. The designer has good control of the sketch being produced. These in turn are a reflection of the abilities of the designer. Also the interface manages to maintain the cognitive aspects of sketching. The visual appearance of the sketches represents the preliminary and ambiguous nature of the designs well.

However, the amount of work required for producing a 3D sketch is considerably more than for a 2D sketch of the same object. This is natural, since the amount of work should be compared to the amount of work required for producing a set of sketches of the object: plans, elevations, sections and perspective images. When compared this way, working with the 3D6B editor is actually faster. On the other hand, in the really early phases of the design process, it is not necessary to think about the full 3D appearance of the object, and a 2D sketch from a single viewpoint can be sufficient.

It is possible to use the 3D6B editor for drawing 2D sketches as well, if the grid and camera are not moved. Therefore, the system also supports 2D sketching. An interesting issue to research would be the transformation of these 2D sketches to three dimensions. This could, for instance, involve copying and pasting sets of strokes from one grid orientation to another. In this way a 2D sketch could be transformed into 3D while still retaining its sketchy nature.

Also, while the interactive grid makes it possible to draw a complex 3D shape, the grid manipulation should be extremely intuitive in order to make the sketching process fluent. With the current prototype, the user may need to spend some time in order to move the grid to a new transformation in a pleasing way. Also re-orienting the grid back to an earlier transformation can be tricky. These problems are not major, but annoying, and can probably be solved with a few features, such as a possibility to orient the grid to existing strokes or a possibility of saving the grid parameters.

Since the rendering uses lines as the only primitives, there are no surfaces that could occlude other parts of the sketch. This makes complex sketches overly crowded with lines when viewed from certain angles, this is visible for instance in Figure 6. Since the grid is rendered with lines as well, it is sometimes difficult to find out where exactly the grid is located in relation to some specific strokes.

These problems can be solved with some enhancements in the rendering engine. For instance by rendering the grid with semi-transparent polygons, the orientation of the strokes in relation to the grid would become easier to understand. Also the problems with the clutter of lines could be reduced by modifying the rendering so that the depth order of the lines is more obvious. One solution for this is to use haloed lines as originally explained by Appel et al. [ARS79].

Currently the editor runs only on personal computers, although some mobility can be gained with a laptop or Tablet PC. A modern PDA on the other hand can have screen resolution of up to 640×480 pixels, a pen-based display and a wealth of processing power. This makes PDA implementation an attractive alternative, as such system would be as mobile as a sketchpad. However, in order to achieve interactive speeds on a PDA, a native implementation would be needed instead of using Java.

5. Acknowledgments

I wish to thank Prof. Lily Díaz-Kommonen and members of the 3D6B project: Jaakko Latikka, Wille Mäkelä, Annina Rüst, Simona Schimanovich, Zhidi Shang, Tuomas Siitonen and Akio Wada.

References

[ARS79] APPEL A., ROHLF F. J., STEIN A. J.: The haloed line effect for hidden line elimination. In *SIGGRAPH '79: Proceedings of the 6th annual conference*

on Computer graphics and interactive techniques (New York, NY, USA, 1979), ACM Press, pp. 151–157.

[BCD01] BOURGUIGNON D., CANI M.-P., DRETTAKIS G.: Drawing for illustration and annotation in 3D. In *Computer Graphics Forum* (sep 2001), Chalmers A., Rhyne T.-M., (Eds.), vol. 20 of *EUROGRAPHICS Conference Proceedings*, EUROGRAPHICS, Blackwell Publishers, pp. C114–C122.

[BDHO92] BUTTERWORTH J., DAVIDSON A., HENCH S., OLANO M. T.: 3DM: a three dimensional modeler using a head-mounted display. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics* (New York, NY, USA, 1992), ACM Press, pp. 135–138.

[CMZ*99] COHEN J. M., MARKOSIAN L., ZELEZNIK R. C., HUGHES J. F., BARZEL R.: An interface for sketching 3D curves. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (New York, NY, USA, 1999), ACM Press, pp. 17–21.

[CPC04] COMPANY P., PIQUER A., CONTERO M.: On the evolution of geometrical reconstruction as a core technology to sketch-based modeling. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (Aug. 2004), Hughes J. F., Jorge J. A., (Eds.), Eurographics.

[CSSJ05] CHERLIN J. J., SAMAVATI F., SOUSA M. C., JORGE J. A.: Sketch-based modeling with few strokes. In *21st Spring Conference on Computer Graphics (SCCG 2005)* (May 2005).

[Dee95] DEERING M. F.: HoloSketch: a virtual reality sketching/animation tool. *ACM Trans. Comput.-Hum. Interact. 2*, 3 (1995), 220–238.

[DML04] DIEHL H., MÜLLER F., LINDEMANN U.: From raw 3D-sketches to exact CAD product models — concept for an assistant-system. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (Aug. 2004), Hughes J. F., Jorge J. A., (Eds.), Eurographics.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3D freeform design. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 409–416.

[Las80] LASEAU P.: *Graphic Thinking for Architects and Designers*. Van Nostrand Reinhold, New York, NY, USA, 1980.

[Lim03] LIM C.-K.: An insight into the freedom of using a pen: Pen-based system and pen-and-paper. In *Proc. 6th Asian Design International Conference* (Oct. 2003).

[MB98] MEYER J., BEDERSON B. B.: *Does a Sketchy Appearance Influence Drawing Behavior?* HCIL technical report no. 98-12, University of Maryland, Human Computer Interaction Lab, Sep. 1998.

[Pic98] PICCOLOTTO M. A.: *Sketchpad+ Architectural Modeling through Perspective Sketching on a Pen-based Display*. Master's thesis, Cornell University, 1998.

[SPR*94] STROTHOTTE T., PREIM B., RAAB A., SCHUMANN J., FORSEY D. R.: How to render frames and influence people. In *Computer Graphics Forum* (1994), vol. 13, pp. 455–466. Special issue on Eurographics '94.

[SRS91] SACHS E., ROBERTS A., STOOPS D.: 3-Draw: A tool for designing 3D shapes. *IEEE Comput. Graph. Appl. 11*, 6 (1991), 18–26.

[SS05] SHARLIN E., SOUSA M. C.: Drawing in space using the 3D tractus. In *2nd IEEE Workshop on New Directions in 3D User Interfaces (IEEE VR 2005)* (Mar. 2005).

[SSLR96] SCHUMANN J., STROTHOTTE T., LASER S., RAAB A.: Assessing the effect of non-photorealistic rendered images in CAD. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1996), ACM Press, pp. 35–41.

[TDM99] TOLBA O., DORSEY J., MCMILLAN L.: Sketching with projective 2D strokes. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1999), ACM Press, pp. 149–157.

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: SKETCH: an interface for sketching 3D scenes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM Press, pp. 163–170.