INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 03 STOCKHOLM, AUGUST 19-21, 2003

CONCEPTUAL MODELING TOOL AT THE EARLY DESIGN PHASE

Ferran Naya, Manuel Contero, Joaquim Jorge, Julián Conesa

Abstract

Although CAD systems have evolved considerably in functionality, expressiveness and modeling power over the last decades, their user interfaces are still tied to legacy principles and are not suited to the initial stages of product development. They exhibit steep learning curves, cumbersome and overly structured dialogues, including hundreds of commands. While much of this functionality may be required by the sheer complexity of the tasks these systems are designed to help, we believe the user interface could benefit from simpler paradigms based on sketching and drawing to reduce unneeded complexity, especially in the conceptual design phase. In what follows, we present the CIGRO system that provides a reduced instruction set calligraphic interface to create polyhedral objects using an incremental drawing paradigm evocative of paper and pencil drawings. Users sketch lines on an axonometric projection, which are automatically beautified and connected to existing parts of the drawing. Such line drawings are then converted to a three-dimensional model through a reconstruction process guided by an axonometric inflation method. While much work remains to be done, comparative usability evaluation tests conducted with expert and new users yielded very promising results.

Keywords: computer-aided design, engineering drawing, geometric modeling, solid modeling

1. Introduction

In spite of recent advances in Computer Aided Design, it is still difficult to capture the drawing language used in the initial phases of product development, where designers prefer pencil and paper over computer-based tools. Available Graphical User Interfaces (GUI) for CAD applications are based on windows, menus and icons combined with mouse and keyboard interactions. For that reason, CAD systems present overly rigid dialogues to users early in the design process, where designers tend to use sketches to express creative ideas in a fast way [1].

Many techniques and idioms characteristic of hand-made drawings cannot be used directly in CAD systems. In this case, users are forced to learn a very different methodology involving the use of menus combined with certain keyboard commands or mouse actions. Therefore, we aim at exploring new interaction paradigms, geared at exploiting sketching skills of designers and engineers. In this context, we present a sketch based modeling application that is capable to manage polyhedral models using an approximate axonometric projection sketch as input. It uses a real time reconstruction engine, capable to propose a real three-dimensional model from the sketched two-dimensional representation, providing automatic beautification and a true integrated planar and spatial sketch environment, which provides for a natural user experience. Spareness and simplicity are the main characteristics of our system. Everything is made as easy as possible, trying to provide an experience close to real paper and pencil usage.

2. Related work

The new generation of calligraphic applications uses gestures and pen-input as commands [2], [3]. In contrast to conventional drawing applications, the stylus can also be used to enter continuous mode sketches and freehand strokes. Thus, there is a growing research interest on using freehand drawings and sketches as a way to create and edit 3D geometric models. Within this research area we can distinguish two main approaches. One method relies on gestures as commands for generating objects from sketches (gestural modeling). The second one, derived from computer vision, uses algorithms to reconstruct geometric objects from sketches that depict their two dimensional projection (reconstruction based modeling).

An example of gestural system is Sketch [4]. The geometric model is entered by a sequence of gestures according to a set of conventions, regarding the order in which points and lines are entered as well as their spatial relations. SKETCH-N-MAKE [5] is aimed at facilitating the machining process of simple pieces by numerical control using a gestural interface for the modeling of the pieces. Quick-Sketch [6] is a computer tool oriented to mechanical design, consisting of a 2D drawing environment which is based on constraints. From these it is possible to generate 3D models through modeling gestures. Teddy [7] allows free form surface modeling using a very simple interface of sketched curves, pockets and extrusions. Users draw silhouettes using a series of pen strokes and the system automatically proposes a surface using a polygonal mesh whose projection matches the object contour. GIDeS [8] allows data input from a single-view projection. In addition the dynamic recognition of modeling gestures provides users with contextual menus and icons to allow modeling using a reduced command set.

Reconstruction based systems, uses techniques in the field of computer vision to build three-dimensional geometric shapes extracted from two-dimensional images, representing their axonometric projections. The systems we surveyed use two main techniques. The first is based on Huffman-Clowes labeling scheme [9], [10]. An example of these systems is Digital Clay [11] that supports basic polyhedral objects and implements a calligraphic interfaces for data input. The sketch is then adjusted and transferred to a reconstruction engine that uses Huffman-Clowes algorithms to derive three-dimensional geometry. The second approach treats reconstruction as an optimization problem using a psychological analysis of humans' capability to identify 3D models from 2D images, deriving a simple set of perceptual heuristics. Marill [12], Leclerc [13] and Lipson [14] provide interesting ideas in this direction. Thus geometric reconstruction, when cast as a problem of perception, can be described in terms of mathematical optimization. An example of this kind of system is Stilton [15], where a calligraphic interface is directly implemented in a VRML environment, ant the reconstruction process uses the optimization approach based on genetic algorithms.

3. The CIGRO system

The CIGRO application allows interactive reconstruction of polyhedral objects from hand-input sketches, to yield three-dimensional models. This restricted geometric domain provides enough flexibility to model many shapes, because in the initial stages of conceptual design, it is not necessary to reflect detailed geometry, and also in other fields like architecture, block geometry is dominant. Our system integrates a gesture analyzer, a line drawing beautifier and a axonometric inflation engine. The integration of these elements provide a much richer vocabulary than that of simpler approaches based on extrusion and constructive geometry, which resort to a comparatively simple set of basic shapes.

In that sense it is arguable that our interface allows greater freedom of modeling without the need to learn special codes or interaction idioms to build given shapes. This design paradigm allows composition of complex shapes directly, from a collection of lines, representing edges drawn as orthogonal projections rather than a composition of simpler shapes using extrusions, cuts, pockets or holes (handling of curve forms will be incorporated in future work). Further, the adoption of a line-based paradigm for interactive reconstruction allows users to directly edit edges and create new vertices into models with rapid feedback. The advantage in reducing the models than can be created to polyhedral objects, is that we can apply straightforward reverse projection techniques (axonometric inflation) to directly generate a three-dimensional model, without having to resort to expensive optimization techniques such as hill-climbing, genetic algorithms and simulated annealing.

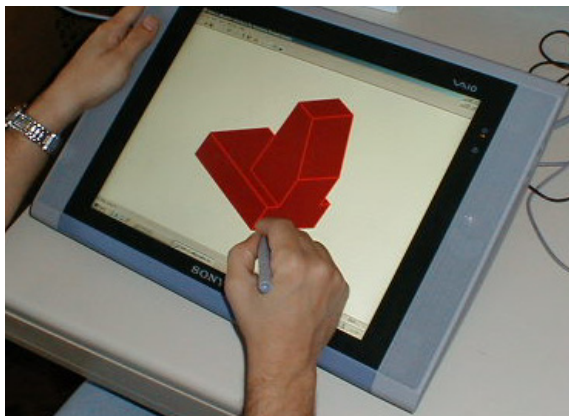

Figure 1. Example showing CIGRO system.

3.1 Gesture Alphabet

The interface of the application is responsible for user-application interaction and its functionality is designed to minimize (we call it minimalist interface) the interaction with the system in an attempt to approximate it to the traditional use of pen and paper. This application automatically generates the corresponding three-dimensional geometric model, and it does not need any additional information from the user. The system updates the 3D model as the user refines the geometry, so the user can see the evolution of the model, whenever any change in the input sketch takes place (Figure 2).

The gesture set used by the interface is reduced to the following commands: new edge, new auxiliary edge and remove edge (auxiliary or not). These gestures are very similar to those used when drawing on paper. The application is, therefore, in charge of recognizing the type of stroke drawn by the user, and to try and capture the designer's intention (parallelism, proximity, close points). The aim for the development of this interface was allow the user to centre his attention on creativity. This frees the user of having to navigate through menus or searching for icons in toolbars.

The application provides both real geometry and auxiliary drawing entities. This emulates an extended practice for making sketches. First, the user draws a set of auxiliary lines (thin lines) to define the main geometric features of the object, and then, using this skeleton as a drawing template the designer refines the sketch, making pressure with the pencil, and drawing over the previous template. Our application takes in account the pressure made on the pencil. Of this way it distinguishes the different geometrical entities of a sketch representing auxiliary constructions or object outlines from those that define the model, by applying a pressure level threshold.

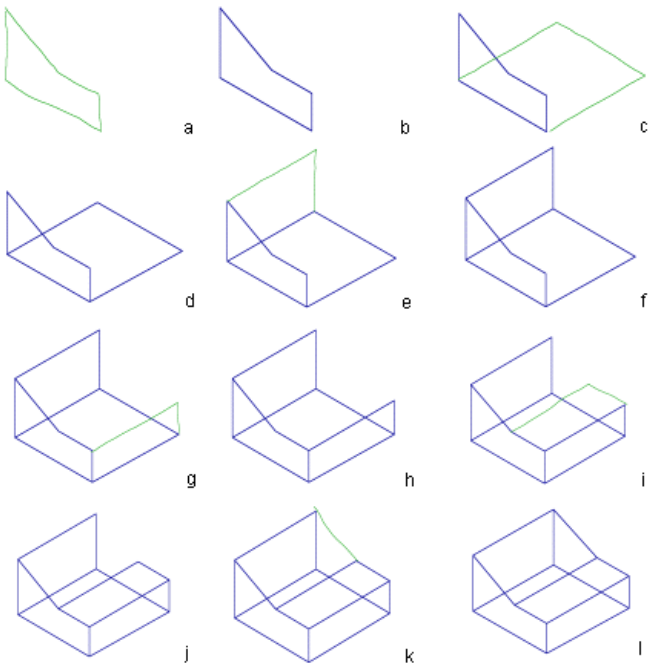

Figure 2. Example showing sequence of actions.

Auxiliary constructions are usually lighter and thinner (less pressure is applied) whereas the models strokes are darker (higher pressure on the pen). Also, the auxiliary strokes tend to serve as references (constraints) for the intended model strokes (Figure 3). This is implemented in the application using Wintab API (http://www.pointing.com), an open industry interface that directly collects pointing input and passes it in a standardized fashion to applications. This API allows the possibility to retrieve the pressure the user applies at each point of the stroke over the tablet. This information is used by the application to distinguish between those auxiliary entities (less pressure) from those of the model (higher pressure).

To obtain this, a library of components has been used to develop calligraphic interfaces: CALI [3]. This library is based on a recognizer of elemental geometric forms and of gestural commands in real time which, using fuzzy logic, discriminates the nature of the geometric shape.

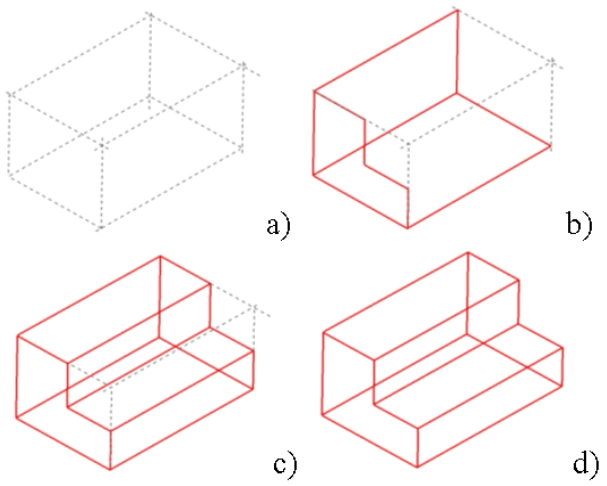

Figure 3. Using auxiliary lines and oversketching.

The recognizer receives the sketch from the application and computes the geometric features for it and then identifies the correct gesture or gestures based on the values computed before. The recognized gestures are inserted in a list order by degree of certainty, and returned to the application. CALI recognizes the elemental geometric shapes, like triangles, rectangles, circles, ellipses, lines, arrows, etc, and some gestural commands, such as delete, move, copy, etc. Among the different geometric shapes and gestural commands, the application only selects the sketched segments which can be recognized as a geometric form of the type "line" (Figure 4) or as a gestural command of the type "delete" (Figure 4.c). In this way, the application analyzes the typology of the sketched entity, and if it corresponds to a line or to the command "delete", the entity is processed.

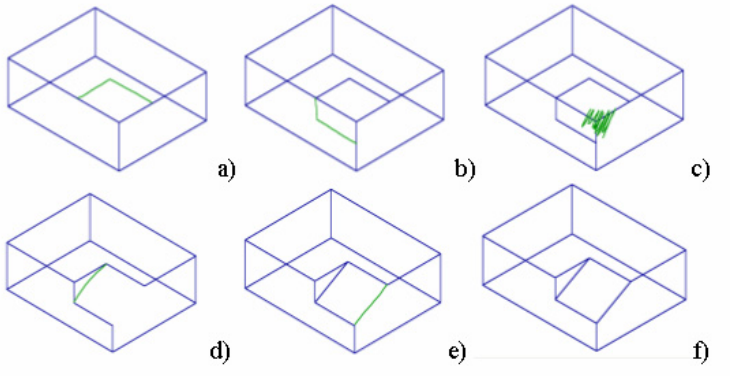

Figure 4. Editing complex from simple shapes.

3.2 Line Drawing beautification and smart snaps

Previous reconstruction based applications usually include an offline 2D reconstruction stage, where the input sketch is adjusted. In our system we propose an online 2D reconstruction [16]. Online reconstruction provides an immediate feedback to the user, because it operates as the user draws the sketch, and it offers better integration in the calligraphic interface. This concept of 2D reconstruction is similar to drawing beautification proposed by Igarashi [17].

The aim of this stage is transform the digitized data from the LCD tablet into the necessary format to be used at the stage of 3D reconstruction. To obtain functional input data, we need to cleanup input data and adjust edges to make sure they meet precisely at common endpoints. This stage includes input sketch filtering techniques which are applied to drawings generated by designers and acquired through the calligraphic interface. These consist mainly of transforming sketches into geometrically consistent figures which can then be used for generating 3D models at the next stage.

But “beautification” should not only remove defects caused by designer-made errors. Its principal task is to filter all the defects and errors of initial sketches and which are inherent to their inaccurate and incomplete nature. This problem can be illustrated with the case in which hand-drawn lines are perceived by humans as parallel although their machine representations are slightly convergent. While line endpoints at corners lie “close enough”, these endpoints do not define vertices because they are not completely coincident. A fuzzy formulation of these spatial relations can be used to make sure that e.g. “approximately parallel” maps to “exactly parallel” before applying the three dimensional reconstruction algorithms.

We now look at constructing a two-dimensional representation. At present, the stage of 2D reconstruction receives as input data either geometric shapes of type “line” or “auxiliary line”, or gesture commands of type “delete”. This will be expanded in the near future to include other primitives, such as triangles and quadrilaterals.

When processing a geometric form of type “line” or “auxiliary line”, the application has to perform several tasks to create an adequate database for the 3D geometric reconstructor. Among them, we will mention the following:

- Modifying the slope of the new lines that are nearly parallel to one of the three principal axes of the projection.
- Adjusting the start and end points of each new line in order to make them coincident with existing vertices of the model, while maintaining the appropriate orientation of the edges incident at these vertices.

In order to perform these tasks efficiently, the first step is to classify the new line, depending on its features. We then manipulate its endpoints, so as to match the existing edges of the model and proceed with the stage of 3D Reconstruction.

To perform the beautification process, the first analysis consists of checking whether the new line is parallel to any of the principal axes of the sketch, considering a certain tolerance. In the case that the straight line is nearly parallel to one axis, then we adjust one or both endpoints so that the resulting line is now precisely parallel line to one of the three main axes.

The second analysis looks for vertices close to the line endpoints, again taking into account a proximity tolerance. In the case there are several such vertices, we select the one with the closest to that line endpoint. For those endpoints of the new line which lie sufficiently near to a vertex, the system records the number of edges to this closest vertex as one of the definition points. For endpoints of the new line which do not lie close to a model vertex, the system analyzes whether the points are close to an existing edge, accounting for a given tolerance to proximity. If several edges match this criterion, we select the edge which lies closest to the given endpoint.

This smart snapping capability provides a continuous feedback to the user, because it is performed in real time. Other previous systems perform these analysis offline, when the user has finished his sketch, before to launch 3D reconstruction.

We provide a general tolerance control to soften the beautification action, because some users prefer a less automatic drawing control. We can distinguish:

- Tolerance of line parallelism. This tolerance value defines the maximum deviation of the slopes of two straight lines to be considered parallel.
- Tolerance of vertex proximity. This tolerance value defines the longest distance between the definition points of the new line and the existing vertices, to be considered adjacent.
- Tolerance of edge proximity. This tolerance value defines the longest distance between the definition points of the new line and the existing edges, to be considered adjacent.

After this stage all the 2D image data are stored in a database composed by a list of vertices and a list of edges that is processed by the axonometric inflation engine.

3.3 Axonometric inflation engine

The so-called normalons (rectangular trihedral polyhedrons, whose corners consist of three mutually perpendicular edges) benefit from a rectangularity constraint that greatly simplifies the interpretation process [18].

Thus, axonometric inflation takes profit from formulations that determine the angle (for instance Φ_Z in Figure 5.a and Figure 5.b) between every edge (OZ) and its own line-segment (O’Z’) when three orthogonal edges are connected to the same central vertex (O).

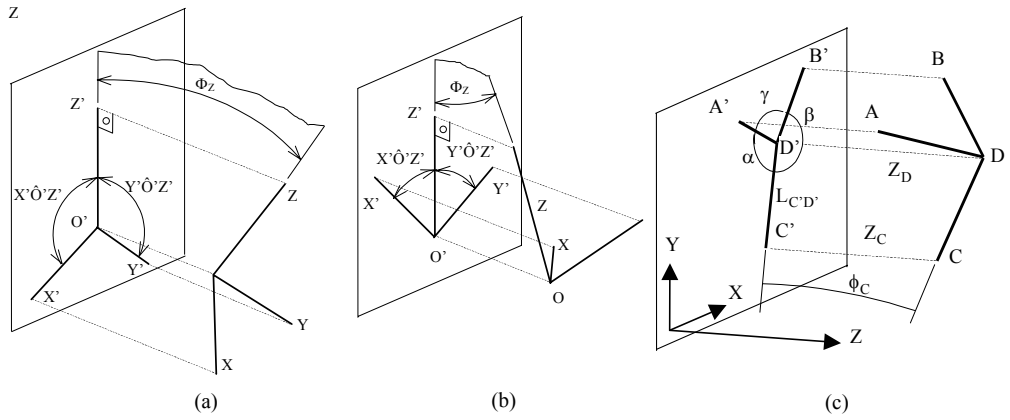

Figure 5. Dependence between angles (a) (b). Axonometric inflation of an orthogonal corner (c).

It is a function of the angles (X'Ô'Z' and Y'Ô'Z') between the orthogonal projection of OZ (O'Z') and the orthogonal projections of the other two orthogonal edges (O'X' and O'Y'):

$$\sin \phi_Z = \sqrt{\cot g\,(X'\hat{O}'Z') \cdot \cot g\,(Y'\hat{O}'Z')} \tag{1}$$

Let us suppose an orthogonal projection of an orthogonal corner like A'B'C'D' in Figure 5.c. To reconstruct the model (i.e. to determine x, y and z coordinates of vertices A, B, C and D) the Cartesian coordinate system associated to inflation is used to trivially obtain x and y coordinates of all four vertices ($x_A = x_{A'}$, $y_A = y_{A'}$, …). Next, relative z coordinates of lateral vertices (A, B, C) are obtained from z coordinate of central vertex (D) in the following way:

$$|z_C - z_D| = L_{C'D'} \,.\, \tan \Phi_C \tag{2}$$

Where $L_{C'D'}$ is the length of line segment C'D' (CD edge projection), and z_C and z_D are the respective z coordinates of the lateral and the central vertex that determine the reconstructed edge. Obviously, Φ_C can be obtained from (1), simply replacing X'Ô'Z' and Y'Ô'Z', by α and β:

$$z_C = z_D \pm L_{C'D'} \,.\, \tan(\operatorname{asin}(\cot g(\alpha) \bullet \cot g(\beta))^{1/2}) \tag{3}$$

To determine z coordinates of all vertices in a normalon polyhedron, a careful propagatıon tree is required to convert already determined lateral vertices into new central vertices, and in this way not to miss anything and construct a topologically consistent 3D model. Additionally, particular cases must be studied to prevent numerical inconsistencies during the process (for instance due to trigonometric operations).

4. Preliminary evaluation

One of the main claims we make with our approach is that despite its simplicity and minimalist syntax, its use compares favorably with conventional CAD systems. Indeed, the command set includes just three main concepts. Constructive strokes allow designers to create edges to build a model gradually. Erase gestures allow users to selectively alter parts of a model to possibly define more complex shapes. Finally, camera manipulation is needed to access different parts of the model being created. In contrast, conventional systems provide dozens to hundred of specialized commands for the same purpose.

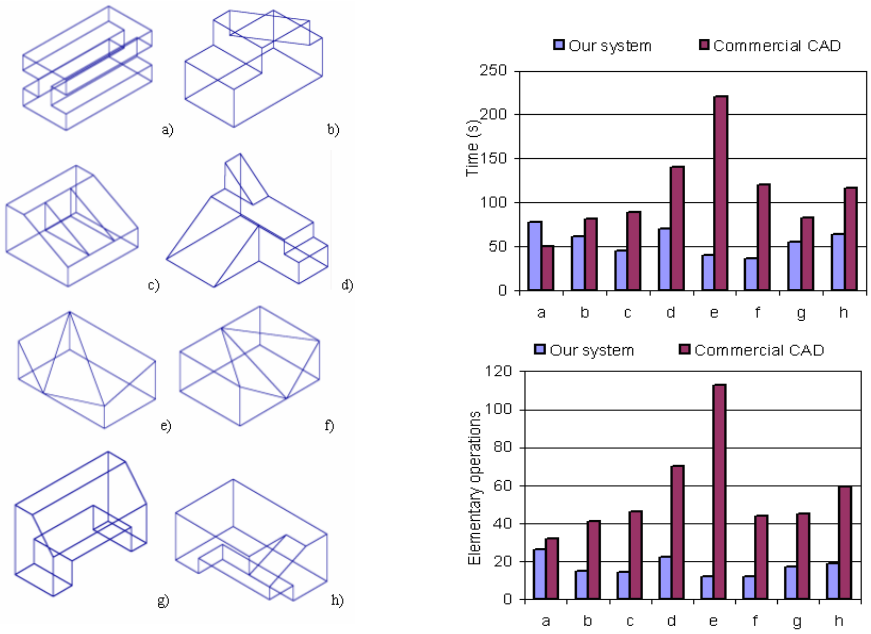

Figure 6. Time to draw shapes and elementary operations.

This is mainly because such systems are organized around maintaining a geometric representation rather than drawing which relies on ideation and user's perception of spatial form. This has the well-know effect that conventional systems emphasize on dialogue, while sketching focuses on results. Comparing the elementary operations needed to obtain the shapes drawn in Figure 6, we can see that even for an experienced user, this emphasis on protocol and articulation of commands is difficult to overcome. Figures 6 shows the number of commands and execution times needed for an experienced user using a commercial CAD system and our approach. Moreover, Figure 7 counts the number of elementary commands needed to create a model, averaged over the eight shapes in Figure 6 in the same manner. From Figure 7, we can see that interaction with conventional CAD systems requires on average 74% menu interactions, whereas in our system three-quarters of the actions are strokes.

There are two striking observations in this: the number of commands needed to create models is smaller in our approach. Moreover, most of the commands in our system are constructive, whereas most of the commands in the CAD system specify actions not data. The latter only account for roughly 20% of all interactions in the CAD system. This both epitomizes the different approaches and illustrates the better expressive power of calligraphic interactions. Interestingly enough camera operations average to about the same in both approaches (3 to 3.1) but represent a higher percentage in our interface since there are fewer interactions needed on average as can be seem from Figure 6. Figure 6 shows a clear advantage in time needed to complete the modeling task in favor of our approach.

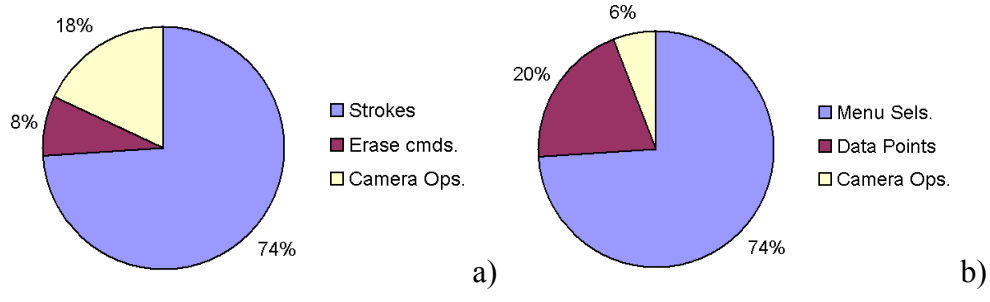

Figure 7. Elementary operations for our system (a) and for commercial CAD package (b).

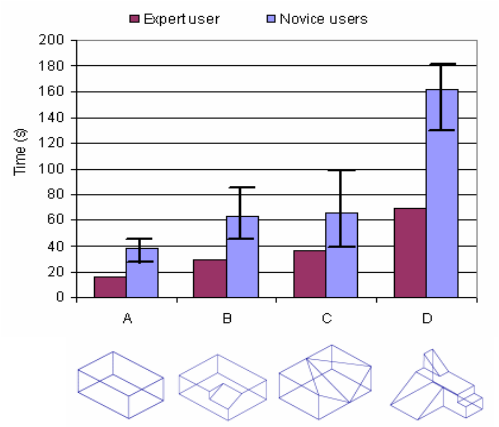

Figure 8. Novice vs. expert execution times, our system.

While the numbers above show an inarguable advantage of our system from the standpoint of an experienced user, we have conducted preliminary usability tests with six novice users. Unfortunately due to the amount of time it takes to train laypeople on full-featured commercial CAD systems it was not possible at the time of this writing to conduct a comparative experiment (we estimate that 10 to 12 hour total training time is needed to teach users to model the parts depicted in Figure 6 without getting lost too many times while navigating menus). Similarly, some of the shapes in Figure 6 also need a bit of getting used to our system in order to be created efficiently. Indeed the test was conducted using computer science (not mechanical engineering) students. Therefore, time constraints forced us to limit the study to four shapes. Figure 8 shows average, maximum and minimum times needed to create these shapes. As we can see, novice users take about 50% to 100% longer than an expert to draw some of the test models. These results are encouraging in that they suggest good usability for untrained users. The users have also commented this tool should be used in the first stages of the design process since it privileges the designer's creativity.

5. Conclusions and Future Work

The modeling approach this application uses is much more familiar than that used in current CAD systems. This is due to the analogies with the methodology used in design on paper. Preliminary studies indicate that this approach presents certain advantages to current commercial modeling systems.

The development of new graphical user interfaces opens new perspectives to the creation of tools oriented to satisfying the needs of the designer in the early stages of the design process, where sketches are used to quickly represent creative ideas.

Future work is intended to provide parametric capabilities and handwritten dimensional control. Also it is expected to support NPR rendering to maintain a hand made look.

Acknowledgments

This work was supported in part by the Portuguese Science Foundation grant 34672/99, the European Commission project IST-2000-28169 and the Spanish Generalidad Valenciana grant CTIDIB/2002/51.

References

[1] Ullman D.G., Wood S. and Craig D., "The Importance of Drawing in the Mechanical Design Process", Computers and Graphics, Vol. 14 (2), 1990, pp.263-274.

[2] Rubine R., "Specifying Gestures by Example". Computer Graphics (Proceedings of ACM SIGGRAPH 91), Vol. 25, ACM, 1991, pp.329-337.
[3] Fonseca M.J. and Jorge J., "Experimental Evaluation of an On-Line Scribble Recognizer", Pattern Recognition Letters, Vol. 22 (12), 2001, pp.1311-1319.
[4] Zeleznik R.C., Herndon K.P. and Hughes J.F. "SKETCH: An interface for sketching 3D scenes", Proceedings of ACM SIGGRAPH 1996, ACM Press / ACM SIGGRAPH, New York. Rushmeier H., Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 1996, pp.163-170.
[5] Bloomenthal M., Zeleznik R., Fish R., Holden L. Forsberg A., Riesenfeld R., Cutts M., Drake S., Fuchs H. and Cohen E., "SKETCH-N-MAKE: Automated machining of CAD sketches", Proceedings of ASME DETC'98, 1998, pp.1-11.
[6] Eggli L., Hsu C.Y., Brüderlin B.D. and Elber G., "Inferring 3D Models from Freehand Sketches and Constraints", Computer-Aided Design, Vol. 29 (2), 1997, pp.101-112.
[7] Igarashi T., Matsuoka S. and Tanaka T. "Teddy: A sketching interface for 3d freeform design", Proceedings of ACM SIGGRAPH 1999, ACM Press / ACM SIGGRAPH, New York, Rockwood A. Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 1999, pp.409-416.
[8] Pereira J., Jorge J., Branco V. and Nunes F., "Towards calligraphic interfaces: sketching 3D scenes with gestures and context icons", WSCG'2000 Conference Proceedings, 2000.
[9] Huffman D.A. "Impossible Objects as Nonsense Sentences", Machine Intelligence 6, Edinburgh University Press, B. Melzer and D. Michie Eds., 1971, pp.295-323.
[10] Clowes M.B., "On Seeing Things", Artificial Intelligence, Vol. 2 (1), 1971, pp.79-116.
[11] Schweikardt M. and Gross M.D., "Digital Clay: Deriving digital models from freehand sketches", Automation in Construction, Vol. 9, 2000, pp.107-115.
[12] Marill T. "Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects", International Journal of Computer Vision, Vol. 6 (2), 1991, pp.147-161.
[13] Leclerc Y. and Fischler M., "An Optimization-Based Approach to the Interpretation of Single Line Drawings as 3D Wire Frames", International Journal of Computer Vision, Vol. 9 (2), 1992, pp.113-136.
[14] Lipson H. and Shpitalni M., "Optimization-Based Reconstruction of a 3D Object from a Single Freehand Line Drawing", Computer-Aided Design, Vol. 28 (8), 1996, pp.651-663.
[15] Turner A., Chapmann D., and Penn A. "Sketching space". Computers & Graphics, Vol. 24, 2000, pp.869-879.
[16] Oh B.S. and Kim C.H. "Progressive 3D reconstruction from a sketch drawing". Proceedings of the 9th Pacific Conference on Computer Graphics and Applications, 2001, pp.108 -117.
[17] Igarashi T., Matsuoka S., Kawachiya S. and Tanaka H., "Interactive Beautification: A Technique for Rapid Geometric Design", Proceedings of the ACM Symposium on User Interface Software and Technology UIST'97, 1997, pp.105-114.
[18] Kanatani K., "The Constraints on Images of Rectangular Polyhedra". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8 (4), 1986, pp. 456-463.

For more information please contact:

Ferran Naya Polytechnic University of Valencia, DEGI Camino de Vera, s/n E-46022 Valencia Spain
Tel: +34 963879516 Fax: +34 963879513 E-mail: fernasan@degi.upv.es