Calligraphic Interfaces: Mixed Metaphors for Design

João P. Pereira[1], Joaquim A. Jorge[2], Vasco A. Branco[3], and F. Nunes Ferreira[4]

[1] Dep. of Informatics Engineering, ISEP/INESC-Porto, R. S. Tomé, Porto, Portugal
jpp@dei.isep.ipp.pt
[2] Computer Science Department, IST/UTL, Av. Rovisco Pais, Lisboa, Portugal
jorgej@acm.org
[3] Communication & Arts Department, Univ. of Aveiro, Aveiro, Portugal
vab@ca.ua.pt
[4] Dep. Of Electrical and Computer Engineering, FEUP, R. dos Bragas, Porto, Portugal
fnf@fe.up.pt

Abstract. CAD systems have yet to become usable at the early stages of product ideation, where precise shape definitions and sometimes even design intentions are not fully developed. To overcome these limitations, new approaches, which we call *Calligraphic Interfaces,* use sketching as the main organizing paradigm. Such applications rely on continuous input modalities rather than discrete interactions characteristic of WIMP interfaces. However, replacing direct manipulation by sketching alone poses very interesting challenges. While the temptation to follow the paper-and-pencil metaphor is great, free-hand sketch recognition remains an elusive goal. Further, using gestures to enter commands and sketches to draw shapes requires users to learn a command set – sketches do not enjoy the self-disclosing characteristics of menus. Moreover, the imprecise nature of interactions presents additional problems that are difficult to address using present-day techniques.
In this paper we approach the three problems outlined above through a combination of different paradigms*:* First, *a calligraphic* sketching metaphor provides for a paper-like interaction. Second, dynamic menus – *expectation lists* – try to expose the state of the application without interfering with the task. Third, an incremental drawing paradigm allows precise drawings to be progressively constructed from sketches through simple *constraint satisfaction*. Finally, reducing instruction set and command usage allow for a simple and learnable approach in contrast with the complexity of present-day interactive systems. We present a system, GIDeS, which embodies these approaches. Usability testing carried so far yielded encouraging results to warrant further research.

1 Introduction

Evolution of CAD systems over the last decades is characterized mostly by the remarkable improvement on their power as design tools, achieved at the expense of increased complexity of operation. While rigid and overly structured interaction styles restrict designer freedom and obstruct the creative flow of ideas, cognitive load on users is exerting, given the large number of commands they need to master. Thus it

doesn't come as a surprise that creators prefer paper and pencil for the early creative stages of object design [1] and resort to computers only at a latter stage, when the shape of the object they have in mind is already settled, and rough sketches get converted into precise technical drawings.

Our first approach to address these problems, IDeS [3] a menu-oriented CAD prototype, relied on drawings sketched by users, from which approximate models of the desired 3D object were generated. A second iteration, GIDeS [14, 15], replaced menus with a calligraphic interface [10, 11] to assist users in creating models from sketches. The current version explicitly addresses ambiguity and imprecision natural to human-generated sketches, using those as strengths to bring computers closer to the paper and pencil metaphor.

To deal with ambiguous input we developed expectation lists [14], non-intrusive context-based dynamic menus that free users from memorizing modeling gestures and constructs. Whenever users' strokes are ambiguously recognized, the application displays a menu with icons that correspond to two or more possible different interpretations of the input. Expectation lists reduce cognitive load on the user by providing a self-disclosing way of showing otherwise hidden functionality. Also, they provide a means to readily show system state without overly encumbering user's drawing task. Moreover, they provide an ergonomic and technically interesting way of addressing gesture recognition errors and ambiguous input. Finally, expectation lists are dynamic menus which allow us to reduce the set of independent instructions – both command and primitive instantiating gestures – clearing the way to a new generation of user interfaces we call RISC – Reduced Instruction Set Calligraphic Interfaces [16].

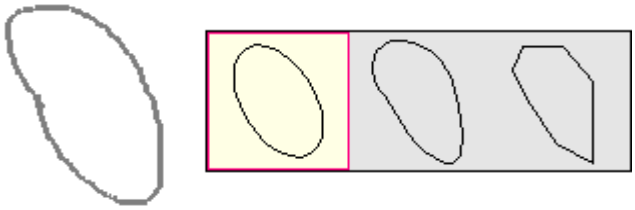


Fig. 1. Expectation Lists

Fig. 1 illustrates how expectations lists deal with ambiguity and explore it to user's benefit. In this case the designer sketched a stroke that resembles an ellipse. The resulting expectation list thus suggests an accurate ellipse as its default option. Nevertheless it is possible that the user wanted a generic smooth curve that happens to resemble an ellipse, instead of the ellipse itself. Or maybe the drawing primitive recognizer described in the previous subsection made a mistake and interpreted the stroke as smooth instead of rectilinear. The remaining two options in the expectation list reflect these possibilities.

However, ambiguity and imprecision are not limited to command semantics. In converting sketches to engineering drawings, draftspeople often need to specify precise dimensions on objects, accurate spatial relations between constituents,

manufacturing tolerances, etc. The underlying paradigm to the development of GIDeS new version relies on what we call incremental drawing, a constructive procedure of generating accurate models from sketches that tries to bridge the gap between pure sketch drawing applications and commercially available CAD systems.
GIDeS continues to rely upon sketches and drawings, and therefore it stays suitable for the early stages of design, because its ease of interaction keeps it close to the paper and pencil feel. Nevertheless this new version includes some improvements that allow users to proceed to the last phases of precise model design without loosing the simplicity and intuitiveness that characterized the previous version.
The accuracy needed for the final stages of design is achieved by means of a continuous but non-intrusive assistance to user actions, which takes place at all phases – 2D and 3D – of the creating and editing processes. The system helps users by means of context-based anchoring constraints that allow rough sketches to be converted into precise drawings and objects. This is not to say that numerical information becomes irrelevant in CAD systems. Rather, that *it is possible to specify rigorous alignment and positioning without resorting to coordinates,* through the judicious use of constraints and visual commands.
The remainder of this paper describes our approach, comparing it to other related work and presenting our method to handle ambiguous interactions and imprecise information in order to create reasonably complex scenes. We provide examples to show how the system works and present the results of early experimental evaluation. Finally we describe ongoing research directions and future work.

2 Related Work

Although the idea of using pen-based interfaces in tasks related to the automatic recognition of hand written or drawn information is over a generation old, it did not make significant progress for three major reasons: early gesture recognition systems had had low recognition rates; dealing with the ambiguity inherent to human-provided information was not easily solved; and the advent of easy-to-use direct manipulation WIMP interfaces led to a widespread lack of interest for gesture-based interfaces.
However a significant amount of research has been done recently in pen-based computer applications (CAD systems are one amongst several examples).
Sketch [19] is an example of a system that allows users to create 3D scenes based on CSG-like primitives instantiated by 2D strokes. All interaction relies on a three-button mouse, occasionally combined with modifier keys on the keyboard. Sketch uses two types of gestural elements – five classes of strokes (draw while pressing the first mouse button) and two classes of interactors (made with the second button). Camera manipulation is performed with the third mouse button. Another feature of Sketch are direction dependent gesture strokes to infer CSG operations. Zeleznik's et al. work has proceeded with Jot [6], an extension to Sketch's interface that relies not only on pen input but also on immersive virtual reality technology such as six-degrees-of-freedom physical props and stereo glasses.
Our approach differs from Sketch's in three main ways. First, GIDeS uses a stylus for all operations. Second, except for camera manipulation, which uses the tap-and-hold,

all commands, drawing and 3D primitives are inferred only from the available (drawn) information. Lastly, all gestures are independent of direction, relying on perspective to retain the pencil-and-paper flavor.

Encarnação et al. [4, 2] developed a system that combines traditional desktop metaphors with a virtual reality interface. This allows the user to directly create simple objects in true 3D, through the use of iconic gestures that resemble the contours of the top-down projections of objects. Their system uses very sophisticated equipment such as transparent pen and pad, shutter glasses, magnetic trackers and a virtual table display device. GIDeS relies on a minimalist approach that sticks as much as possible to the paper and pencil metaphor, avoiding the need for sophisticated and expensive hardware and forgoing immersive environments.

Igarashi et al. developed Teddy [9], a system that allows modeling of freeform 3D objects from sketched 2D outlines. However, resulting models are constrained to a sphere-equivalent topology and Boolean operations were not taken into account.

Igarashi et al. describe a technique for rapid geometric design called interactive beautification [8]. Freehand strokes drawn by users are converted by their system – Pegasus – in line segments that must satisfy certain geometric constraints such as perpendicularity, congruence and symmetry amongst others. Pegasus also uses context information to deal with ambiguity. It generates multiple candidates by combining inferred constraints appropriately and evaluates them in order to find the most plausible ones and reject the others. The user is then allowed to select the candidate that meets his or her wishes by tapping on it directly. The procedure is completed as soon as the user taps outside the candidates or draws the next stroke. However, the problem with this way of handling ambiguity is that it is difficult for users to perform the selection they want amongst a large number of overlapping candidates.

GIDeS extends the use of 2D constraints to 3D modeling applications. We can generate primitives with accurate dimensions and relationships to other parts of the model, without the need to specify numerical data, because freehand gestures that instantiate primitives can be converted into precise drawings that satisfy certain 2D and 3D geometric constraints. Moreover, complex models resulting from combining several primitives are also accurate because object placement procedures and geometric transformations can be subject to 3D geometric constraints such as face parallelism, edge alignment and vertex coincidence. Finally, designers may also use 2D constrained auxiliary lines to help them in the task of precisely editing the solids they created. In short, our approach based on expectation lists to handle ambiguity avoids the above-mentioned problem of selecting intended choice amongst a large amount of overlapping candidates in a self-disclosing manner.

More recently, Fonseca et al. [5] developed $C_{AL}I$, a library of components for calligraphic interface design that relies on fuzzy logic and geometric feature analysis to recognize basic geometric shapes and gesture instantiated commands, which are being incorporated into our system.

Mankoff et al. [13] present a survey on interfaces that make use of various forms of recognition such as gesture, handwriting and speech interpretation. Their work also focuses on the problem of handling recognition errors and ambiguity by means of dialogues between the system and the user – a process they call mediation. Based on that survey, the authors created a user interface toolkit called OOPS – organized

option pruning system. OOPS consists of a library of reusable mediation techniques combined with architectural solutions to model and provide support for ambiguity at the level of user input events.
Bloomental et al. [12] developed a system for sketching parts for manufacturing. While they can provide dimensioning information, we believe that our dynamic menus are a more efficient interface technique than their expanded gesture sets.
Gross et al. [7], Back of an Envelope (BoE) project applies pen-based interfaces to a wide range of domains such as databases, simulation programs and 3D modeling. Their approach tries to combine the virtues of highly structured interfaces for creating accurate drawings and models with the benefits inherent to freehand drawing interfaces. One of the systems built in the context of the BoE project is Digital clay, an application that generates 3D models of objects from 2D line drawings. Digital Clay allows users to sketch a 3D rectilinear geometric form and the program uses constraint propagation to determine the topology of the object and to assign 3D coordinates to its representation. Our system relies on a constructive, incremental approach, where a set of relatively simple primitives can be assembled to create more complex objects, in a hierarchical fashion. Also both rectilinear and non-rectilinear geometry are equally supported by our system.
Turner et al. [18] designed Stilton, a sketch modeling system that resembles a desktop VRML browser, allowing users to interact with a 3D model in perspective projection, or panoramic photographs mapped onto the scene as a “floor” and “walls”. The system can be used to reconstruct geometry from panoramic images or to add new objects to an existing scene. Object creation relies on geometric information sketched by users on an imaginary 2D drawing plane.
In the next section we describe our system’s main organizing principles and relate these to usability features and findings during usability evaluation.

3 GIDeS System Architecture

The architecture of GIDeS calligraphic interface consists basically of a set of three recognition subsystems, one for commands, one for two-dimensional sketches and another for three-dimensional objects, an expectation list generator and a set of constraint-based interaction modes that allow users to perform geometric transformations, constructive geometry and cuts on objects.

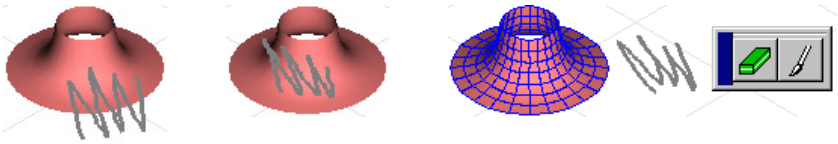
3.1 Gesture Recognition

Command gesture interpretation relies on two recognition subsystems. The first one is an improved version of Rubine’s trainable recognizer [17], changed to add some new features. Our recognizer provides support for multiple-stroke gesture recognition and the sequence by which strokes are drawn is irrelevant. It can also recognize strokes regardless of direction in which they were drawn. A major departure is to force recognition to depend on context aspects not necessarily related to gesture geometry. Last but not the least, we provide support for handling ambiguity.

Three-dimensional primitive instantiation relies on a second recognition subsystem that takes into account both the topology and geometry of each gesture. A detailed description of this recognizer including the changes we made to Rubine's algorithm can also be found in [15].
The third recognition subsystem is responsible for interpreting linear strokes and to decide whether these strokes shall be interpreted as polylines or curved (smooth) drawing elements. This subsystem can recognize line segments, polylines, circles, ellipses and generic curves represented as cubic splines. The recognition system is capable of learning and adapting to a user's drawing style, since the distinction between smooth and non-smooth strokes relies on a parameter that changes dynamically according to the success or failure of previous interpretation attempts.

3.2 Expectation Lists

For gesture recognition systems to work adequately we need to strike a compromise between two extremes. On one hand the interpretation of strokes must be flexible enough to deal with uncertainty, otherwise the rate of incorrectly rejected gestures (we call them false negatives) will be high. On the other hand recognition must be rigid enough to provide selectivity and to reduce the rate of erroneously interpreted gestures (false positives).
Our approach based on the combination of gesture recognition with expectation lists changes this paradigm. Instead of trying to find some heuristic (heuristics, no matter how good they are, are always subject to failure due to human unpredictability) that selects the most probable candidate and rejects all others in order to avoid ambiguity, we made our recognition systems more tolerant than usual to gesture uncertainty and use expectation lists as dynamic menus to allow users to exercise control and choose amongst the most likely candidates. In other words we have significantly reduced the false negative rate and, since the options presented to users by expectation lists are mere suggestions, the corresponding increase of false positives is not a problem, because users can simply ignore these suggestions in the same way they ignore other unwanted options. That is, instead of trying to avoid ambiguity, we encourage its occurrence and explore it to user's benefit.
We have tried to extend expectation lists to all levels of user interaction. With the exception of command expectation lists that use icons to make suggestions, all lists prompt the user with small-scale models of the objects that can be created in relation to the existing context.
For example, Fig. 2 shows how expectation lists can significantly reduce the number of needed command gestures, thus minimizing cognitive load on users. In this case two commands – *delete* and *apply texture* – share the same "scratch" gesture. The difference is that the *delete* stroke must cross the object boundary (Fig. 2a), while the *texture* stroke must be entirely drawn over the object's surface, i.e. inside its two-dimensional projection (Fig. 2b). The user may also opt to delete or conversely, to apply a texture to a previously selected object. In that case GIDeS does not have enough contextual information to identify what command to apply. Therefore, the application generates a command expectation list and prompts the user to select a command (Fig. 2c).



a) delete object **b)** apply texture to object **c)** ambiguous command

Fig. 2. Command Expectation Lists

Fig. 3 shows an example of a 3D primitive expectation list. Again notice how RISC interfaces explore ambiguity in order to reduce the required repertoire of recognized gestures. In this case the same idiom can instantiate four distinct objects, namely a truncated cone, a surface of revolution – the most obvious choices – and two less evident kinds of cylindrical sections with different orientations. The designer may also ignore the suggestions and proceed with the drawing.

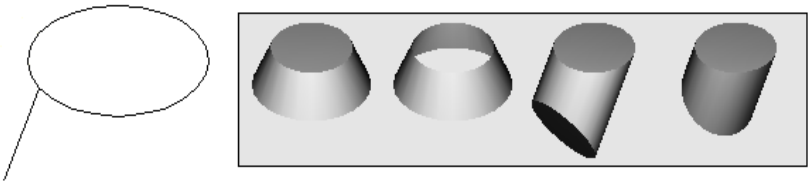


Fig. 3. 3D primitive expectation list

We have found the expectation mechanism a nice extension to gestural interaction, because it expands the functionality of the system without the need to increase the set of gestures recognized by the system. Users were able to accommodate the concept and noted one significant enhancement that has been incorporated into the system since: *expanding the expectation list with an* undo *option.* Even with this addition, the longest expectation list generated by the system does not contain more than six elements, which is cognitively acceptable. To cope with possibly longer lists of possible objects and ambiguous commands, we have evaluated *temporal menus*, in which a predefined gesture (tap), allows users to cycle between the different interpretations of each command, but users found it too confusing and interfering with the drawing task.

We have also evaluated fixed and adaptive expectation lists. In the latter case, the system *remembers* which choices the user made and adjusts the order in which items are presented (e.g. different renderings of a solid, choice between spline and polyline renderings of a sketch, etc.). In general we have found that there are *less* interactions with adaptive lists, suggesting an internal consistency of user's interpretations of their own actions during a given drawing task.

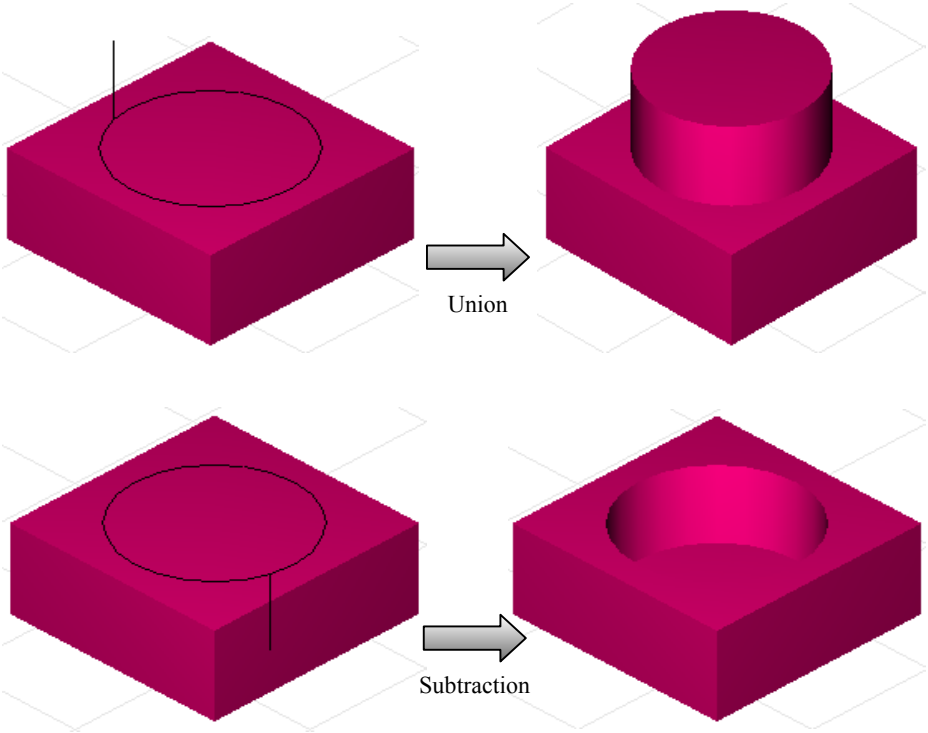


Fig. 4. Recognizing Boolean operations

Whenever designers sketch a primitive over an existent solid in the scene, the newly created object is properly placed and attached to that solid. Under these circumstances GIDeS attempts to find the appropriate Boolean operation – union or subtraction – based on gesture orientation. Fig. 4 illustrates this procedure. The gesture orientation (in the example the orientation provided by the line segment together with the ellipse attached to it) is compared with the object's surface normal. The system chooses to perform union or subtraction depending on whether the sign of the dot product is positive or negative in a manner similar to that of Sketch [19]. What is different in our system is the interpretation of ambiguous drawings: Some primitives such as spheres provide no orientation information. It may also happen that the above-mentioned dot product equals zero (i.e. the gesture orientation and the surface's normal are perpendicular to each other). These circumstances make it impossible to identify the desired Boolean operation. Again, an expectation list is generated, allowing the user to choose between union and subtraction as shown in Fig. 5.

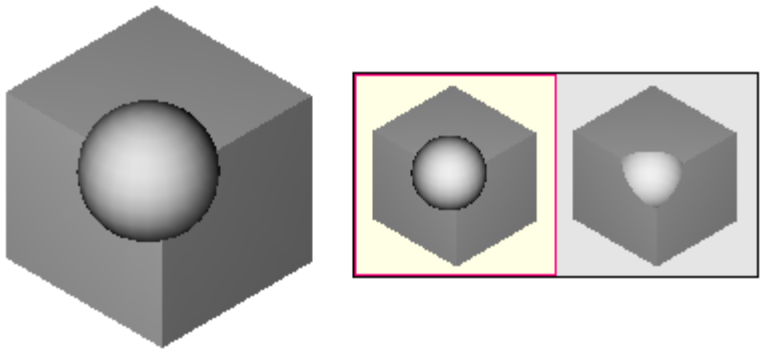


Fig. 5. Boolean operation expectation list

3.3 2D Editing: Sketch Correction

GIDeS helps users in correcting strokes. Designers often want to correct previous sketches by drawing over curves. GIDeS partially supports this interaction style by allowing the user to draw directly over the section of the stroke they want to change, as they usually do with paper and pencil. GIDeS detects this situation and automatically removes the unwanted portion of the drawing, as shown in Fig. 6. This operation, called oversketching, allows users to save several low-level interactions. A similar technique was proposed by Baudel [20], which we have adopted for sketching on three-dimensional objects using cutting as described in the next section.

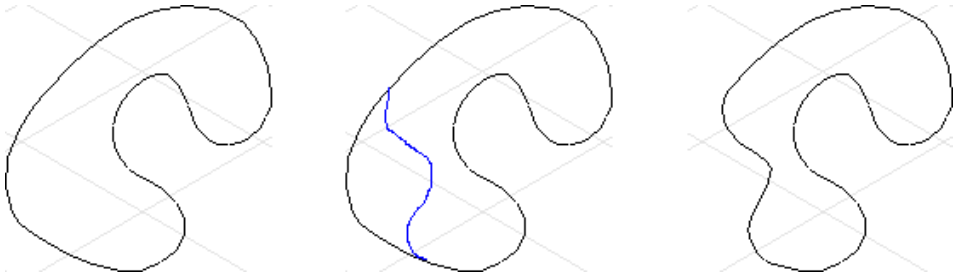


Fig. 6. Oversketching

3.4 3D Editing

Geometric Transformations: Our approach to the task of performing rigid-body geometric transformations – translations and rotations – is different from what can be found in traditional CAD systems. Instead of thinking of which geometric transformations the user must apply in order to achieve the desired result, there is a set of three interaction modes in which the user draws simple strokes and the system automatically infers the necessary constraint-based transformations that shall be carried out.

The first mode is called *gluing*. The user draws a stroke connecting two objects and the system performs the necessary set of transformations in order to attach (glue) the first object to the second through the specified faces (Fig. 7). Any objects already glued to the first one undergo the same transformations in order to stay glued.

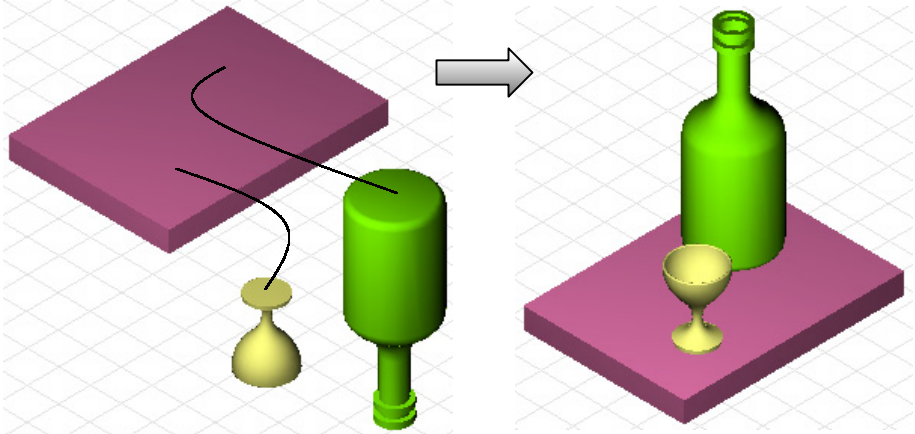


Fig. 7. Gluing primitives together

Whenever an object is already glued to another one, the second mode – called *adjusting* – allows the user to adjust the position of the first object in relation to the second. The system identifies the restrictions that must be applied to the translation process in order that the object is only allowed to slide along the face of the other object to which it is attached. Eventually glued objects undergo the same transformation.

The third interaction mode allows users to freely place an object in the scene. Translation is automatically applied to other objects eventually glued to it. A detection mechanism allows the user to place objects over other existing solids in the scene. Applying simple constraints (face co-planarity, co-axiality, edge alignment) allows the system to place the objects saving unnecessary interactions.

Cutting: GIDeS includes an additional interaction technique to allow users to perform cuts on 3D objects in a rather effective way. Designers only have to outline the cut they intend to do over the surface of the object and the system automatically evaluates, constructs and subtracts the adequate extrusion solid primitive in order to achieve the desired result. Cutting profiles can either be open or closed. In the first case the stroke must meet the boundary of the face that is going to be cut. In the second case cutting strokes can either be entirely or partially drawn over the object's above-mentioned face.

As we have seen in this section, our system presents an innovative approach as compared to traditional CAD systems by using a mixed metaphor – while we try to stay close to interaction modalities evocative of the paper-and-pencil organizing principles, we use an incremental drawing approach to allow draftspeople to use sketching gestures to construct rigorous models through a judicious combination of simple, yet powerful commands and constraints to allow users to perform more

complex operations with fewer commands. Fig. 8 exemplifies a reasonably complex object drawn using GIDeS, while Fig. 9 shows the available solid primitives.

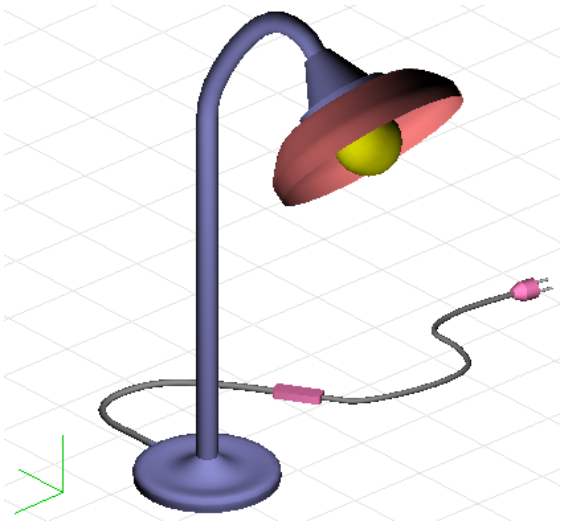


Fig. 8. Lamp created with GIDeS

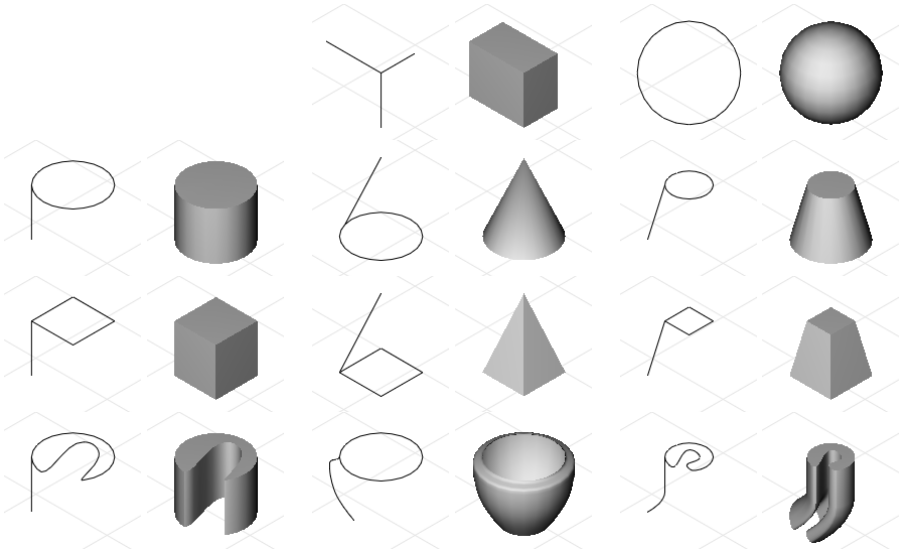
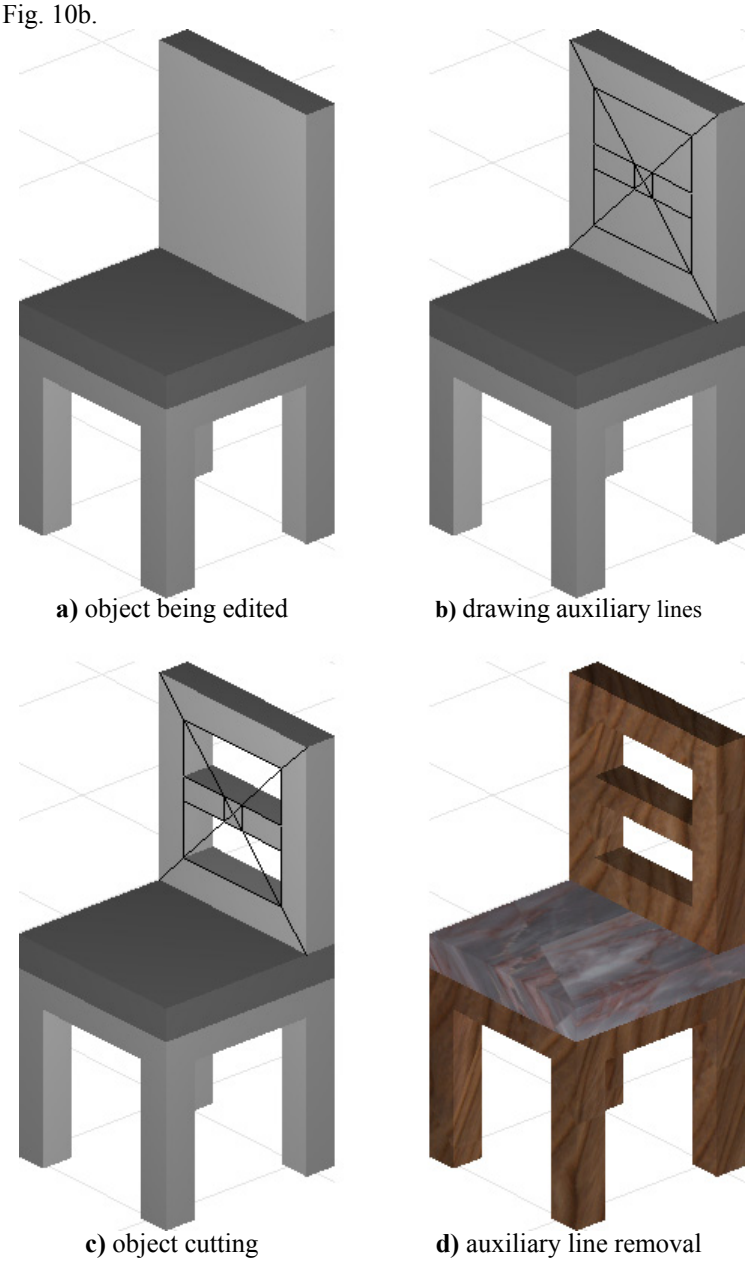


Fig. 9. 3D Objects and corresponding sketches

Fig. 10 illustrates the last steps in the process of assembling an accurate chair. Two cuts on the back of the chair are carried out in Fig. 10c. An example of the use of 2D

constrained construction lines to outline with precision the cutting profile can also be seen in Fig. 10b.



a) object being edited **b)** drawing auxiliary lines

c) object cutting **d)** auxiliary line removal

Fig. 10. Using constraints and auxiliary lines to accurately cut an object

As we have seen in this section, our system presents an innovative approach as compared to traditional CAD systems by using a mixed metaphor – while we try to

stay close to interaction modalities evocative of the paper-and-pencil organizing principles, we use an incremental drawing approach to allow draftspeople to use sketching gestures to construct rigorous models through a judicious combination of simple, yet powerful commands and constraints to allow users to perform more complex operations with fewer commands.

While some of the techniques developed here have been tried in other contexts, we believe that the combination of three basic interaction paradigms bears the promise to provide highly flexible and powerful design systems. In developing our approach we have applied a consistent set of organizing principles throughout the drawing application:

Calligraphic Recognition: As a foundation to our drawing paradigm, this allows draftspeople to apply a pencil-and-paper like metaphor for creating base objects. Calligraphic recognition combined with incremental drawing allows powerful operations to be carried out through *sketching on objects* as illustrated by Fig. 11.

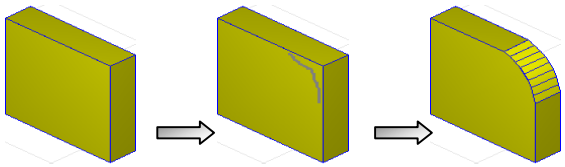


Fig. 11. Sketching on objects

Expectation lists: To make the system state and functionality self-disclosing. Expectation lists also make it possible to deal with imprecision and recognition errors in an elegant and graceful manner.

Incremental Drawing: To make it possible to gradually refine a drawing by means of implicit constraint satisfaction. To this end we have replaced explicit placement and adjustment commands by approximate manipulations afforded by gestures and using constraints to achieve final (rigorous) placement. Moreover, constraints can be and are used to determine the geometric attributes of newly created primitives if those can be obtained from context. For example, sketching an ellipse on top of a cylinder automatically makes the ellipse take on the cylinder's diameter. The user can then validate this option interacting with the dynamic expectation list.

Constraint satisfaction: To replace explicit commands. Constraint satisfaction is a powerful control mechanism. We use it implicitly on the constrained positioning (adjust) and assembly (glue) command modes described above. We believe that matching on approximate constraints and then incrementally adjust the image so that these can be met can replace many explicit commands commonly used in CAD systems for the same purpose. Often designers spend precious time adjusting details in a technical drawing to ensure that certain constraints are met. Constraint satisfaction coupled with sketched intentions makes it possible to achieve the same result in a simpler manner.

Reducing instruction set and command usage: The conjugation of the mechanisms outlined above can result in more concise and expressive user interfaces. Reducing instructions can in theory improve learnability and minimize cognitive load. We need to use expectation lists judiciously to expose functionality in a flexible manner, however.

4 Usability Evaluation

We have asked five designers and architects familiar with traditional CAD systems to participate in an evaluation session of our system. The session started with an initial questionnaire, in order to determine each participant's user profile. Then we presented GIDeS to the users, along with a brief tutorial describing its functionality, which was followed by a small session of questions and answers. The users have then answered to a first questionnaire, so that we could determine their first impression about the system. After that they performed three simple tasks, both with GIDeS and with two commercial, direct manipulation CAD systems of their choice. Task 1 consisted of creating the objects depicted in Fig. 12. In task 2 they used 3D editing tools such as cutting and gluing to modify the objects. Task 3 consisted of creating a lamp somehow simpler than the one represented in Fig. 8. The session ended up with a second questionnaire, which was basically the same than the first one. Table 1 below shows the measured times in seconds of our system as compared to conventional CAD systems.

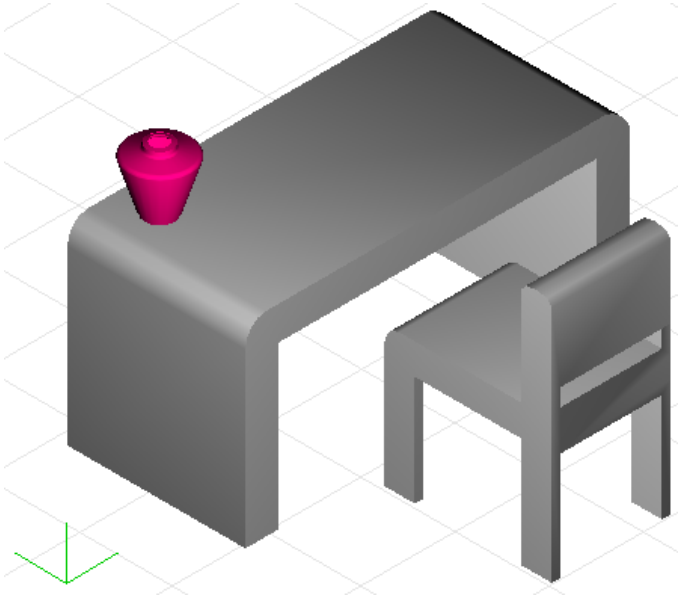


Fig. 12. Model used for Task 1

Results seem very encouraging and show a clear performance advantage to our system relative to conventional approaches. Also answers to both the first and second questionnaires were very positive, although participant's opinion about GIDeS was better in the first one (before they experimented the system) than in the second (after performing the tasks). The users pointed out several limitations such as the lack of an undo/redo command, but all participants met the calligraphic model of interaction with enthusiasm. Other suggestions were made, such as giving users an easy way of making the projection plane parallel to any face of any object (implemented since by

means of a double tap over the projection of the desired face), or adding an extra input device so that users can make use of both hands at the same time, one for drawing and the other one for controlling the camera.

Table 1. Benchmarks

	GIDeS (s)	**CAD #1** (s)	**CAD #2** (s)
1	135.8	267.5	298.5
2	060.5	213.5	158.5
3	173.8	407.5	268.0

5 Conclusions and Future Work

In this paper we have presented GIDeS, a system for creating geometric models through Calligraphic Interaction. Our goal is to improve on the usability of CAD systems at the early stages of product design. To this end we have proposed an approach based on Calligraphic Interfaces, Reduced Instruction Set and Constraints, mixing metaphors to model objects by incremental drawing. We have introduced *expectation lists,* a kind of dynamic menus to make our interfaces easier to learn while showing users more of the system state. We believe this approach is highly suited for designing complex shapes and will be looking into expanding and augmenting the expectation lists to make our interface more self-disclosing. Preliminary results show a positive attitude from users and the promise to improve on traditional approaches via more flexible and expressive commands. We plan to explore more natural ways of combining constraints and error handling to improve dialogues in the near future.
While the system shows promise, we feel that further attention to task and contextual user analysis should provide more insights to make the approach more efficient. We are also working on adding dimensional and numerical constraints to the prototype to make it amenable to manufacturing tasks.

Acknowledgements

The work described in this paper has been supported in part by the European Commission Grant #IST-2000-28169 (SmartSketches project) and by the Portuguese Science Foundation under grant POSI/34672/SRI/2000.

References

1. Blinn J. F. Jim Blinn's Corner – The Ultimate Design Tool. IEEE Computer Graphics & Applications, Vl, No. 11, pp. 90–92, 1990.

2. Bimber O., Encarnação L. M., Stork A. A multi-layered architecture for sketch-based interaction within virtual environments. Computers & Graphics, Vol. 24, No. 6, pp. 851 – 867, Elsevier, Dec. 2000.
3. Branco V., Ferreira F. N., Costa A. Sketching 3D models with 2D interaction devices. EUROGRAPHICS '94 Conference Proceedings, Daehlen M., Kjelldahl L. (editors), Oslo, Blackwell Pub., pp. 489 – 502, 1994.
4. Encarnação L. M., Bimber O., Schmalstieg D., Chandler S. D. A Translucent Sketchpad for the Virtual Table Exploring Motion-based Gesture Recognition. Computer Graphics Forum, Vol. 18, No. 3, pp. C-277 – C-285, 1999.
5. Fonseca, M. J., Jorge J. A., Experimental Evaluation of an on-line Scribble Recognizer, Pattern Recognition Letters Journal, v22 n12, pp 1311-1319 2001.
6. Forsberg A. S., LaViola Jr. J. J., Markosian L., Zeleznik R. C. Seamless Interaction in Virtual Reality. Computer Graphics & Applications, IEEE, Vol. 17, No. 6, pp. 6 – 9, 1997.
7. Gross M. D., Do E. Y.-L. Drawing on the Back of an Envelope: a framework for interacting with application programs by freehand drawing. Computers & Graphics, Vol. 24, No. 6, pp. 835 – 849, Elsevier, Dec. 2000.
8. Igarashi T., Matsuoka S., Kawachiya S., Tanaka H. Interactive Beautification: A Technique for Rapid Geometric Design. Proceedings, ACM Symposium on User Interface Software Technology (UIST), 1997.
9. Igarashi T., Matsuoka S., Tanaka H. Teddy: A Sketching Interface for 3D Freeform Design. SIGGRAPH '99 Conference Proceedings, ACM, 1999.
10. Jorge J. A. Parsing Adjacency Grammars for Calligraphic Interfaces. PhD Thesis, Rensselaer Polytechnic Institute, Troy, New York, 1994.
11. Jorge J. A., Glinert E. P. Calligraphic Interfaces: towards a new generation of interactive systems. Jorge J. A., Glinert E. P. (guest editors), Computers & Graphics, Vol. 24, No. 6, pp. 817, Elsevier, Dec. 2000.
12. M Bloomenthal et al. Sketch-n-Make: Automated Machining of CAD Sketches. *Proceedings of ASME Design Engineering Technical Conferences,* September, Atlanta, Georgia, 1998.
13. Mankoff J., Abowd G. D., Hudson S. E. OOPS: a toolkit supporting mediation techniques for resolving ambiguity in recognition-based interfaces. Computers & Graphics, V24, N6, pp. 819–834, Elsevier, Dec. 2000.
14. Pereira J. P., Jorge J. A., Branco V., Ferreira F. N. Towards Calligraphic Interfaces: Sketching 3D Scenes with Gestures and Context Icons. The 8-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media 2000, Plzen, Czech Republic, Feb. 2000.
15. Pereira J. P., Jorge J. A., Branco V., Ferreira F. N. GIDeS: Uma Abordagem Caligráfica à Edição 3D. 9. º Encontro Português de Computação Gráfica, pp.101–108, Feb. 2000.
16. Pereira J. P., Jorge J. A., Branco V., Ferreira F. N. Reduced Instruction Set Calligraphic Interfaces: Sketching Complex 3D Objects with (Fewer) Gestures. d3 desire designum design, 4th European Academy of Design Conference Proceedings, pp. 194 – 196, Aveiro, Portugal, April 2001.
17. Rubine D. Specifying Gestures by Example, SIGGRAPH '91Conference Proceedings, ACM, Vol. 25, No. 4, pp. 329 – 337, 1991.
18. Turner A., Chapman D., Penn A. Sketching space. Computers & Graphics, Vol. 24, No. 6, pp. 869 – 879, Elsevier, Dec. 2000.
19. Zeleznik R. C., Herndon K. P., Hughes J. F. SKETCH: An Interface for Sketching 3D Scenes. SIGGRAPH '96 Conference Proceedings, ACM, Vol. 30, No. 4, pp. 163 – 170, 1996.
20. Baudel, Thomas, A Mark-based interaction paradigm for free-hand drawing, Proceedings, ACM Symposium on User Interface Software Technology (UIST), 1994.