GIDes++

Joaquim A Jorge
Dep. Engª. Informática, IST
Av. Rovisco Pais, 1000 Lisboa
jorgej@acm.org

Nelson F Silva
Instituto Superior Técnico
Lisboa
nelson.faria@netcabo.pt

Tiago D Cardoso
Instituto Superior Técnico
Lisboa
tiagocardoso@netcabo.pt

Abstract

GIDes ++ is a gesture based calligraphic 3D modeller using expectation lists, and extends the notion of natural and efficient interface. A brand new look and a new modeller kernel empower the modelling capabilities. A measures layer enables the user to visualize and change measures, giving a new dimension in precise and accurate modelling. Adding new modalities like handwriting and speech, GIDes ++ present a natural and intelligent multimodal interface enabling quick models prototyping creation. Visually, much effort is being made to create a more appealing and elegant application, but keeping always in mind the usefulness of these visual effects.

Keywords

Interaction Techniques, 3D Modeling, Sketching, Gesture Interfaces, Calligraphic Interfaces, Handwriting Recognition, Voice Recognition.

1. INTRODUCTION

Intelligent Multimodal Interfaces have come a long way and are currently being applied with success in many areas. Providing an interface that enables a user to interact with a system in a more natural way is one of the main goals of this upcoming research subject.

Looking at the usual workflow of any manufactured piece one can see that most of the creative work is done by designers, the following stage consists of having a specialized CAD technician to take the product of this conception phase, which is usually drawn in freehand, and port it to an exact computer three-dee model.

By providing a more natural and pleasant, interface instead of the traditional WIMP ones found in commercial packages, we hope to close this gap and "build" a bridge between these two stages, motivating designers to draw free-hand directly on a CAD system, allowing them to quickly prototype models that can easily be changed to follow the so usually strict requirement measures and constraints.

2. BACKGROUND WORK

Using sketching as the main organizing paradigm was the aim of a CAD system called GIDes [Pereira00], which called this approach *Calligraphic Interfaces*.

Replacing direct manipulating by sketching alone posed several very interesting challenges as well as requiring user to learn a given command set in order to draw shapes.

On the most noticeable solutions that resulted from the development of this system were *Expectation Lists*.

These, as shown in the right figure, allowed the system to deal with the ambiguity of the input provided by user's strokes by displaying a list of the possible most likely interpretations given a certain context. These non-intrusive context-based dynamic lists free users from memorizing modeling gestures and constructs.

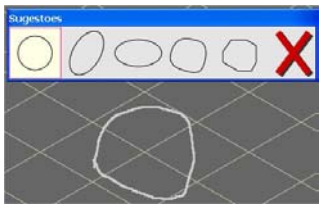

Figure 1: Expectation Lists

2.1 Manipulating Tools

In order to accomplish Three-Dee edition a Cutting tool was presented. With this tool one could carry out free-hand cuts on faces. There were two kinds of cuts: inside cuts, where the user could draw a close line in a surface, removing the enclosing area; the other type was edge to edge cuts, which were made with a line from an edge to another, eliminating the outside portion of the object. Both operations cut objects through the inverse-normal vector given at the point of the surface where the stroke began.

Other important tools only modified the object position in space, not changing the object at all. With *Positioning t*he user picked a point in an object and one in an aim object and the first of these will be translated so that both chosen points coincide. On the other hand in the *Glue tool* the initial object would be glued to the other one. To accomplish this restriction, the initial object would be rotated as well as translated so that the picked face would have the inverse normal vector of the destiny face. If two vertexes were chosen, another restriction would be triggered and an additional rotation would be performed so that both edges leaving those vertexes coincided.

Finally, with the *Adjust tool,* not only one could use it as the Positioning tool, but if the initial object was glued to a face and the destination point was also on that face, the object will translate, but respecting that face's plane, allowing both faces to continue to overlap and respecting the glue constrain.

3. GIDES ++

When recreating GIDes, it was decided to change the strictly imperative language paradigm of C and encode this new version in an Object Oriented Language like C++. This would not only have a good impact on the development of GIDes++ but also in the following versions as it would be all much more organized and gaining from all Object Oriented paradigm advantages.

Visually, a lot was changed too. One tried to create a more pleasant visual ambient which could be more catchy and enjoyable for the user. The first differences noticeable reside on colour changes, like the background and *userstroke* and remaining lines. Some other changes were introduced in the visual indication resulting from user interaction. As for the *Expectation List*, one concluded that imposing the upper-left position on the screen of this had many disadvantages and could turn the experience of selecting a suggestion into an annoying process. Consequently, it was changed so that the user could move it around like a floating frame and every time this list would appear, it would be in the last place left. Objects were also part of the "face-lifting". It was wished to be created a use experience that resembled somehow computer games, which try to be attractive and visually appealing, involving the user in a deeper level. As part of this plan, some graphic effects were added enriching not only the application's aspect, but also the visual information received by the user.

The first effect applied was Motion Blur to camera view changes. When a determined view angle is chosen an animation is performed, changing the camera's orientation and zooming on the desired entity. During the animation, the well known motion blur effect provides realism since humans see real life in this manner and gives better notion of how the camera is moving and in what way is the scene being watched. Another effect was applied to object selection. When an object is selected, a glow effect appears, highlighting the object. This is not only stunting and elegant, but this glow as the particularity of glowing even when the object is behind another. A sort of ghost object appears when the object hides at the back of another, enlightening the user where the object is, what orientation does it have and how is he shaped. This information can be important when using real time object manipulators, like real time rotation or translation, since we can follow the object in every time, allowing us to adjust the object even when it is under another.

When doing boolean operations, this effect could be of a great help too, as one can watch an object part inside the other, seeing, if the case is a subtraction or an intersection, what will result from that operation.

4. MEASURES LAYER

Being a sketch-based modeller, no precision in the object creation input can be assured. Unlike other CAD systems, where dimensions of objects are defined as they are being constructed, in GIDes ++ 3D primitives are assembled from strokes which can't guarantee any precision when it comes to measures. Even with the help of some kind of ruler or a grid with defined unit size, the user can not be accurate enough.

One efficient way to define proportions, but not real measures, is to use support lines with geometric constructions. Even though this is of great help and has an enormous potential, it doesn't unravel the entire problem.

Firstly, how would the user change to this layer? Would there be any explicit information that would point us out how to do so? And then, the other logic questions, how could an interface that would both maintain the naturalness be created, following the paper sketch analogy but always having in mind that the less activity the user needs to do, the better!

As for the layer switcher, an explicit command was chosen. These operations had to be kept in an individual and independent layer since many of the commands performed in this layer were just like normal drawing commands or selection, and if the user was allowed to change measures in the drawing layer, commands could be mistaken. Besides, as has been stated, one intends for users to first develop a prototype and only then concentrate on measures, this way his focus is solely on creativity.

One other main reason for this decision was that much of the operations and simple mouse over actions required lots of computation, therefore in some situations the application would stop responding to user's queries, which isn't pleasant to happen when the user is drawing and wants to see everything he is doing in real time. As a result one can change the layer – drawing or measures – by pressing a button on the fixed toolbar, or by issuing a voice command.

Staying in the measures layer is obvious that the user wants to visualize as easily as possible the object's dimensions. The first approach to this problem was to show every edge measure from a selected object. But soon enough one realized that this was too much information and could even be impossible to show it all at once. It was also noticed that if one knows an edge's size, by proportion analogy, the size of the others can be easily speculated and consequently there were no reasons to report more that one measure at a time. Well, this probably isn't entirely true, but knowing which edges were important to inform at a given moment is a much too complex and non-trivial and we would have efforts would have been wasted and other, more important, areas might have been neglected.

The idea was to have a way of easily and quickly displaying an edge's measure permitting the user to verify a vast number of edges in a short period of time. A solution was found, one that is plain, simple and efficient: as the

mouse moves over and edge, automatically that edge – from one vertex to the other - is highlighted and numeric indication appears, stating the edge's length.

Finally, the only thing left was a method to allow measure changes. Initially, the objective was to copy the way designers and architects set measures. This is usually done by stroking two lines, both starting near vertexes and stating the measure in the middle of them. As a result, changing a measure in GIDesS++ would be performed in the same way. The user draws a line starting from a desired vertex and then from another and the length between these two vertexes would come into sight.

Posterior to this, the user would specify the new length and the change was performed. As another possibility, the users can simply ignore this measure and continue to perform other action, enabling the visualization of distances between vertexes not linked by an object edge.

Still the user had always to stroke two lines and just then would he be able to make any changes. So and keeping in mind that simple and easy is better, a complementary way of setting measures was implemented. Simply by clicking over an object's edge, the user is allowed to change it. Of course this solution only has some vertexes combinations in mind, but we could always recur to the previous technique for others.

A particular and important moment was when the user would specify the new length. The use of the keyboard was out of the question as this application was, since the beginning, thought to be interacted with non-conventional input like mouse or keyboard.

Showing a little numeric keyboard in a frame with special characters like coma or point wasn't very convincing, stylish or even efficient. The logic way was to turn to handwriting recognition. This was the most natural, pleasant and efficient way to input measures. Not only is it quick, but designers are used to write measures down on paper. Therefore, a grey frame appears in the screen and the user can simply write down the desired number. To recognize handwriting, new Handwriting Recognition is used which is part of Windows XP Tablet PC Edition. The recognizer gives the power to recognize, in an efficient way, human writing. Some tests were performed to test the recognizer liability and success rate.

5. SPEECH RECOGNITION

As stated before using voice in order to invoke commands and switch between layers seemed to be a very feasible way of eliminating the need for a keyboard and even the right mouse button.

At the time it was decided to use an English version, hand-writing recognition that took advantage of the Microsoft Windows Tablet PC recognizers, was already at use, so keeping things simple the speech recognizer from the same software house was used. Very good things about the system had already been heard and applications that use it successfully have, in fact, been seen. One other advantage was its use of the XML format to describe the grammar providing greater flexibility.

5.1 Grammar

Voice commands could have been applied to all the functionality of the system but the main goal was to remove the need for a right mouse button on the pen. As such most of the commands enable the user to switch tools or layers, and also allow for the opening and saving of files as well as exiting from the program.

One of the main problems that came up was the fact that having no keyboard didn't enable a simple way to engage or disengage speech recognition. As a result the system keeps trying to recognize commands from what the user is mumbling while working. Fortunately one was able to reduce this to a minimum and have had only a few changes of view while working which are of minor effect to the work in progress.

The main solution for this problem was the use of commands with at least two words. Thus for every command regarding and object the word "this" was added and for camera operations the word "view" was used. A special case was for the paste command where the keyword "this" wouldn't make much sense, so both "object" and "objects" were used. The same is applied to the copy command.

For switching layers voice commands were also introduced. These are composed simply of the layer intended, drawing or measures, followed by the keyword "layer".

The command for exiting the program is, as implied by this description, "exit program".

One has to call attention for a special character used in the description of the grammars used by the MSSpeech API which is the "+" sign. By putting this before a given word in a grammar rule the recognizer ir requested to have greater confidence before stating that it recognized that given word. This way one can avoid much misrecognition. On the other hand this will require the user to pronounce the words in a clearer way for the system to be able to recognize them with confidence. What was done was trying to reach a balance. For this the "+" has been applied to the main keyword in a given command.

5.2 Accuracy test

MS Speech	U1 - 22 / 22 = 1.000 = 100 %
	U2 - 20 / 22 = 0.909 = 91 %
	U3 - 17 / 22 = 0.773 = 77 %
	Average : **89.3 %**

Table 1 – Speech Recognition Accuracy

It is important to state that these tests were performed in a quiet environment. Most designers and modellers work in rooms with other colleagues so perhaps while one of them is issuing a command it most probably will affect others. Still, since the main purpose was to try to apply a different modality to a system as powerful and demanding as a CAD system these accuracy rates are enough to try to draw some conclusions with the experiments lead with a group of users. Results regarding comments re-

ceived from these testers will be presented further on and discussed.

6. HANDWRITING RECOGNITION

The main goal for *Measures Layer* was for it to allow the user to change the measure of a given model by simply selecting the edge to be altered and, by simply writing the new size, affect the object and make it follow the intended size. For this we a powerful handwriting recognition engine was needed.

Since the target platform when at the start of development of this application was the Tablet PC it was decided to give the inherent recognizer a try. After installing the Windows XP Tablet PC Edition in an ordinary computer one has a powerful handwriting recognizer at hand. The SDK is also freely available an using it isn't very difficult since it need not be event driven, that is to say, one can simply send it the *ink* received in a given window and ask it for the most probable result.

6.1 Accuracy Test

The accuracy tests conducted were done directly in our application. The user had to select an edge and introduce a set of ten different measures and one would determine, by a simple mouse-over, if the edge had the correct new size.

One drawback of the current implementation is that *factoids* are not being used. That is to say that the recognizer doesn't know that users are only writing numbers on the screen, thus it might return letters or symbols. If this happens the edge doesn't change.

The numbers chosen for these tests seamed adequate since they are a mix of several similar looking digits. There are five numbers with no decimal part and others with two decimal digits in order to determine if the dot is a source of problems for the recognizer.

	U1	*U2*	*U3*
Rec. Rate	**80 %**	**80 %**	**90 %**
		Avg. Rec. Rate	**83.3 %**

Table 1 – Handwriting Recognition

Looking at the results of the misrecognitions above of Table 1, one finds that these are very close to the number the user intended to insert. The fact that these tests were conducted on an ordinary PC using the mouse as an input is one of main causes of fault. Even a human could have misinterpreted the numbers which if when drawn by hand with a pen are, sometimes, already difficult to interpret; with a mouse become a real challenge to figure out.

An 83.3% recognition rate is very good taking into account the fact that factoids are still not in use. It is also important to take into account that these tests were performed in a relative early stage of development simply to determine if the recognizer would be enough, thus the small group used.

7. USABILITY TEST

One of the most important steps in software development is to test it with users. Only then can the development team get to know how does the interface work in the real world and what features should be changed, enhanced, added or even removed. The direction of development normally changes and past work is normally reviewed after these tests sessions and usually lead to product optimization in terms of its efficiency and efficacy of its interface.

In this particular case, the main goal was to test the (intelligent multimodal) interface and not the features and functionalities. As a result, it was decided that the task performed by real users should be something more abstract and not centred in a real object, like a chair or table, and should use interface features as extensively as possible.

The construction of a real object (e.g. chair) as task could direct the user attention to other matters and not exclusively to the interface itself. Drawing an abstract object with no real meaning would not distract the users, unlike a know object which could divert their opinions, expressing whether they think the application can easily create that particular object or not. This way one can focus on the feeling and naturalness of the interface for the user.

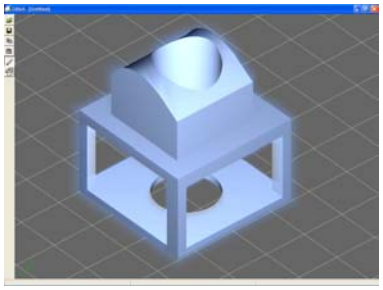

Figure 2 – Test model

This test was performed by a group of users, which had never used GIDeS++ time enough to be considered to have any expertise over this application, and a first user who was an expert in order to get some indicator over the easiness of learning of GIDeS++. In order to have some feedback regarding the use of speech, every user did the task two times: in the first, not using any speech, recurring to the "plan B", namely toolbars and view alterations in the green axis; And in the posterior attempt, only speech could be used, over the alternatives. With this, a proper evaluation of the speech modality could be reached by comparing the results for every user. Unfortunately, these experiments weren't performed in a screen equipped with a digitizer, but in a normal monitor with a mouse instead of a pen. This can deceive a bit the results as working with a mouse is far more complex than with a pen in this kind of interface. A particular part that suffers much from this is the handwriting input, since trying to write numbers with a mouse is far more time consuming than with a pen.

As for the results, shown next in table 3, the only output was the time needed to finish the task. Initially, we con-

sidered recording the number of mistakes, but as this task was quite straightforward a rather low quantity of errors were expected, thus being this information inconsequential and irrelevant. As no comparison with conventional CAD systems was made the number of clicks – or actions – weren't traced either.

	With Speech Commands			
	*U1**	U2	*U3*	*U4*
Time	1:43.02	3:07.81	3:07.97	2:06.57
Average: Inexperience Users	-	02:47,5		
Average	02:31,3			

	Without Speech Commands			
	*U1**	U2	*U3*	*U4*
Time	2:00.51	3:40.19	4:27.29	2:52.21
Average: Inexperience Users	-	03:39,9		
Average	03:15,1			

* Expert User

Table 3 – Usability Tests

As one can observe, all inexperienced users had lower performances than the expert. This was expected, but the gap between both types of users isn't that much, being the expert almost just 33% quicker in the "with speech" test. Taking into consideration the gap between the first user and the others - first time users - one can somehow state that this interface is easy to learn and to get at ease with. Another immediate observation that can be made is the difference in the Average times in both tests. When performing with speech commands, users always finished earlier. Some users even spent 1:20 less in the second trial which is a huge improvement.

One has to take into account that in the "With Speech Commands" test, users were doing the same thing for the second time, therefore, it makes sense for the times to be shorter. The tests had to be done in this way, or else, no conclusion could be taken and, having all users experienced the same test over two different methods, conclusions from users could be more useful and their comparison between with and without voice would be more founded and easy to make. Even so, looking to the expert user, that had executed many times this task before performing any of the real tests, the difference is still there.

So, it's safe to assert that speech modality is an advantage to this application, allowing users to avoid toolbar navigation and centralizing some of the commands in a way that humans are used to exercise, this is, speaking commands.

During the tests, it was noticed that users lost much time looking for a specific tool, trying to remember in what toolbar context they were in, or pointing the cursor to a determined command area of the screen. Instead, with voice, they could change tool, make view modifications, etc... And prepare them to the next action, moving the cursor (hand) to the point where they would draw next.

All users seemed most impressed with the interface potentials and the mixture of various modalities. Handwriting recognition was probably the one that had the most impact. It is not very common to see this modality in big use nowadays and mainly with so good results and good application.

Speech was noticeable, but it is already very seen on media or films for a long time. One thing most users weren't expecting was that using voice really enhanced their performance and was so pleasant to use.

All users stated that the multimodalities were put to good use and really helped the development and creation of object in GIDes++. Not only this, but they expressed their satisfaction in using such things and said they were extremely enjoyable to use, commenting that it looked like a sci-fi software, but that really worked.

8. CONCLUSION

The main goal of this project was to try several approaches, in this case modalities, for introducing several improvements to a CAD system in order to make it more natural. This added to the fact that the starting point was an already pretty natural system it makes one feel that the goal was achieved. Even if several issues are still to be covered one certainly was able to show, at least to the testers, that much can be done in order to "camouflage" the inherent complexity of a modeling system.

9. REFERENCES

[Pereira00] Pereira J P, Jorge J A, Branco V, Ferreira F N: GIDeS: Uma Abordagem Caligráfica à Edição, *9º Encontro Português de Computação Gráfica*, 2000