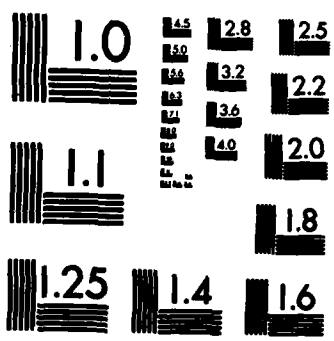
AD-A120 664 STATUS REPORT ON THE GRAPHICAL CURVE EDITOR (GRACE)(U) 1/1
CALIFORNIA UNIV BERKELEY BERKELEY COMPUTER GRAPHICS LAB
A D DEROSE 28 SEP 82 N00039-82-C-0235

UNCLASSIFIED F/G 9/2 NL

END

FILMED
1
DTIC



Status Report on the GRAphical Curve Editor (GRACE)

Technical Report

R.S. Fabry
(415) 642-2714

"The views and conlcusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government."

Contract N00039-82-C-0235
November 15, 1981 - September 30, 1983

ARPA Order Number 4031

Status Report on the GRAphical Curve Editor (GRACE)

Anthony D. DeRose

Berkeley Computer Graphics Laboratory
University of California
Berkeley, California 94720

ABSTRACT

This document describes a program called **Grace**, an interactive tool for creating two-dimensional pictures. The program consists of two communicating parts: first, a data flow network running on an Evans and Sutherland Picture System 300 (PS300), and second, a C program running on a host VAX 11/780. Scenes are created by creating and/or modifying any number of objects. Once a scene is created it may be saved for later editing, or sent to a hardcopy device for permanent imaging.

Object types currently supported are text, vector lists, rectangles and a new curve type; the **Beta-spline** (circles are scheduled for the next release). Once an object is defined it may be transformed by translating, rotating or scaling it (only translation is currently implemented). These transformations are all performed in real time, thanks to the graphics display processor contained within the PS300.

September 28, 1982

Status Report on the GRAphical Curve Editor (GRACE)

Anthony D. DeRose

Berkeley Computer Graphics Laboratory
University of California
Berkeley, California 94720

1. Introduction

Grace began as a tool used to investigate the behavior of a new curve technique. This technique, to be discussed herein, proved to be so useful that **Grace** was extended to provide a highly interactive environment for creating two dimensional diagrams and logos. One problem with programs of this type is that too many options face the uninitiated user. To minimize the learning curve an attempt was made to allow one to use **Grace** at many different levels. At the lowest level the user need only know how to create and destroy items in the scene. As the user becomes more proficient he may wish to zoom in to inspect a region of high detail or use one of the vertex placement aids. At the highest level the user may operate on an arbitrary group of items in the picture, performing transformations such as translation on all members of the group simultaneously.

Grace uses a two terminal format, much like **Caesar**[1] and **Gremlin**[2]. This format uses the PS300 screen to display the scene itself, along with other graphically oriented information. Non-graphical information is displayed on the screen of a normal ASCII terminal such as a Zenith Z19 or Concept 100. This format was chosen so that a complete command menu and secondary status information could be displayed without consuming space on the graphics screen.

2. Nomenclature

In order to more clearly present the underlying concepts of **Grace** it is convenient to introduce some terminology.

The scene is said to consist of a set of pictorial *items*, each composed of two parts. The first, the part of interest in designing the scene, is the *object* associated with the item. The second part defines, or guides, where the object goes; we call this part the *template*. When a new item is created, the template is specified and an object is *bound* to that template. The way in which the template governs the object depends on the object type and is discussed in detail in a later section. A template is displayed as a set of *vertices*, shown as crosses, and a set of line segments connecting the vertices.

3. Command Entry

Grace commands are broken into two major categories; long commands and short commands.

A short command is specified by typing a single letter on the terminal or by *picking* the appropriate command icon from the PS300 screen. Picking is performed by placing the stylus over the desired icon and pressing the blue puck button. If the selection is registered the *mode box*, a box showing the current mode, will surround that icon, at least momentarily.

Long commands must be entered from the terminal, are preceded with a ':' and consist of one letter, possibly followed by other command parameters, followed by a carriage return.

Some commands require that a vertex be picked or a position digitized. To digitize, place the stylus at the desired location and depress the yellow puck button. If a vertex is to be picked, place the stylus above the vertex and press the blue button. If the pick is registered a blinking octagon will appear around the vertex, at least momentarily.

To keep the use of the puck simple and unconfusing the buttons always have the same meaning. The yellow button is always used to digitize, the blue button is always used to pick, and the white and green buttons always set the position of the box (the box is discussed in detail in a later section). One might imagine a scheme where the meaning of a puck button was determined dynamically, depending perhaps on the current mode. Such a scheme was deemed inappropriate for the class of users that are likely to comprise the Grace audience.

4. The PS300

The PS300 is a data-flow programmable graphics processor developed by the Evans and Sutherland Computer Corporation. The PS300 is programmed by specifying a data-flow graph using a moderately high-level syntax. Following strict data-flow concepts, all input and output devices appear as nodes in the data-flow graph. The output of these device nodes can be used as input to various input collection and reformatting networks, or can be used to drive object transformations; this a very powerful and innovative technique never before marketed. A display object can be defined in a hierarchical manner using the PS300 command language. This means that objects may be structured in much the same way as programs in a block structured programming language. Grace exploits the PS300 command repertoire to achieve a higher degree of realtime response than was previously possible. (In previous Picture Systems, realtime processing was possible, but demanded large amounts of host intervention. The PS300 host only needs to initiate the realtime behavior, but it need not monitor the transformations once started.)

5. The PS300 Screen

Grace divides the PS300 screen into several regions. The majority of the screen is devoted to the scratch pad area, that is, the area through which the Grace world is viewed. Digitization of new vertices and display of the scene is restricted to this region.

The area along the right side of the screen is devoted to short command icons (as explained above). The area along the bottom of the screen is for object icons. The current mode of the system is denoted by a box around the appropriate icon. If a long command is being processed, the mode box will appear around the icon labeled "long". At the command level the mode box will appear around the icon labeled "command". The default object type is surrounded by a box similar to the mode box (object types are explained later).

This format insures that all primary state information is visible on the PS300 screen while secondary information is relegated to the terminal screen. Except for entering long commands a beginner really never needs to use the terminal; the terminal is primarily for the convenience and speed of advanced users.

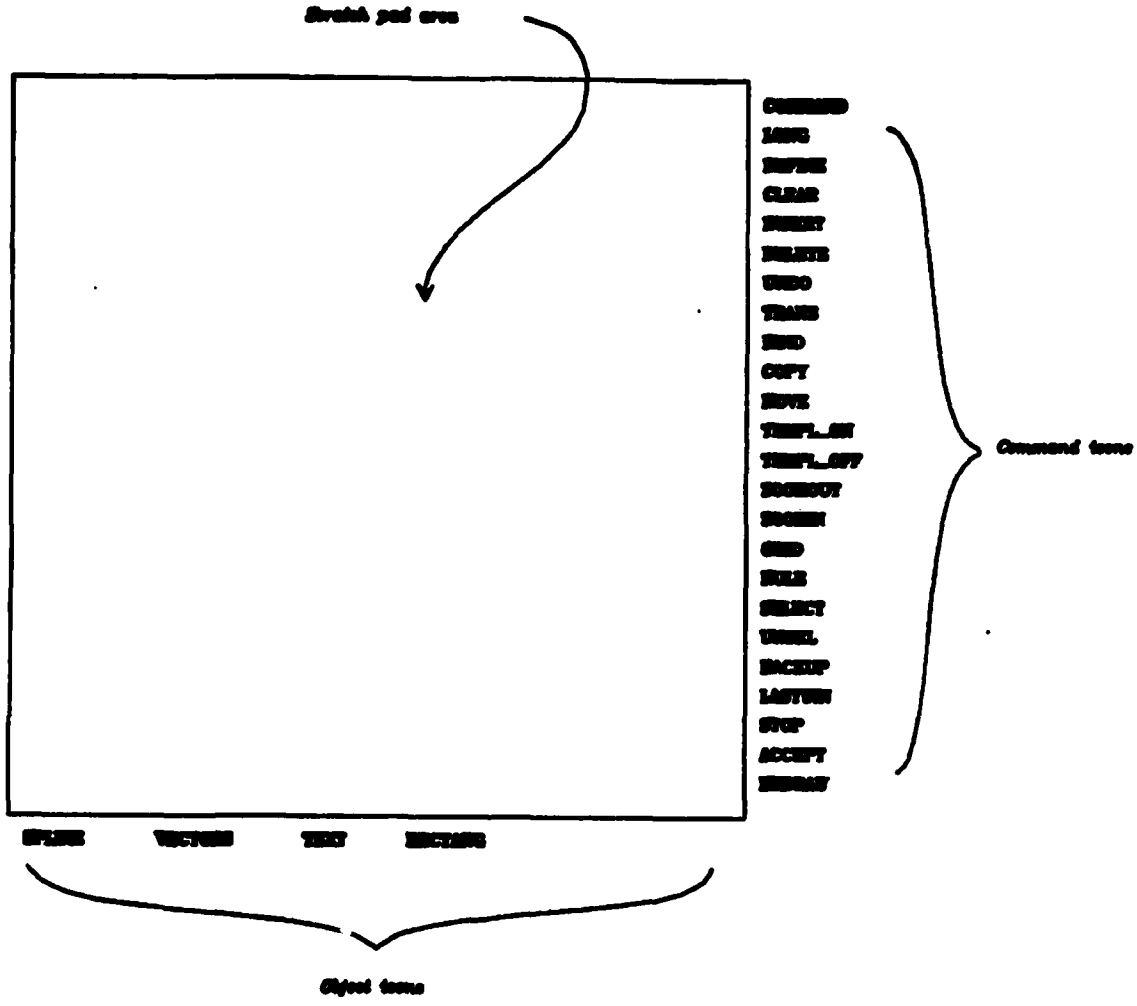


Figure 1. *PS300 screen format*

6. The Terminal Screen

The terminal screen is divided into four regions. The upper left region displays all status information. Status information includes the file being edited, the current mode, the current object type, etc. The upper right region of the screen is a command menu. The command menu is divided into two parts: short commands and long commands. The next region, located on the second to the last line of the screen, is used as the user input line. When a colon is typed to begin a long command, it, and the remainder of the line, are echoed here. The last region, located on the bottom line of the terminal, is for Grace diagnostic messages.

7. The Basic Commands

```
                                    Grace V2.0
-------------------------------------------------------------------------------
        Status:                 |                    Commands:
-------------------------------------------------------------------------------
                                |                        Short
Mode      :                     |  ' '- Stop      |  ' ' - Accept   |  y - Bind
File      :                     |  #; - Gridtog   |  #/ - Ruletog   |  ^t - Translate
                                |  #. - Templ off |  #, - Templ on  |  #b - Last win
Object    :                     |  #n - Spline    |  #o - Text      |  #p - Vectors
Grid      :                     |   h - Backup    |  ^i - Select    |  ^u - Unsel
Gridspace:                      |   [ - Delete    |   \\ - insert   |   m - Move
Distance  :                     |   ] - Define    |  ^c - Clear     |   - - Undo
Window    :                     |  #j - Zoom in   |  #k - Zoom out  |  ^= - Copy
                                |  #{ - Rectangle |  ^L - Redraw    |
                                |-----------------------------------------------
                                |                         Long
                                |   :beta n n     |  :font n        |  :adjust (clr)
                                |   :edit <file>  |  :write <file>  |  :incl <file>
Beta1:          Beta2:          |   :quit         |  :gridspace n   |  :hard (params)
Font:           Adjust:         |
-------------------------------------------------------------------------------
```

Figure 2. *Terminal screen format*

7.1. Creating a Scene

This section describes the options necessary to create, save and load scenes in the simplest possible manner.

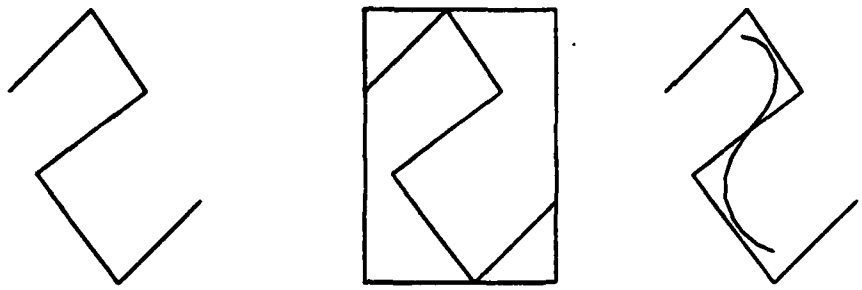
A new item is created by chosing the *define* option. The program will enter the define mode waiting for a set of vertices to be digitized. These vertices will become the item's template. When the vertices have been entered the *accept* option is chosen to complete the definition. Note that vertices are constrained to lie on grid locations if grid adjustment is on. Grid adjustment is in effect by default but can be toggled by chosing the *grid* option.

When *accept* is entered the newly created template is bound to the current object type. The current object type is shown with a box around it on the PS300 screen and can can be changed at any time by picking the icon corresponding to the new object type. If the object type is *spline*, the template acts as the control polygon governing a Beta-spline curve (the values of beta1 and beta2 are shown in the status area of the terminal screen). If the object is of type *vector*, the segments of the template trace out the vector list. A *rectangular* object is defined by the bounding box of the template. The way in which the template guides a text object is slightly more complex so only the default binding will be described in this section. The first two vertices of the template are the only ones significant in defining text. The lower of the two vertices (i.e. the one having the smallest y value) is used as the bottom center of the text string. The difference in height of the two vertices determines the text size. When the accept option is given, the user is prompted for the text string on the user input line of the terminal screen.

If the user misplaces a vertex while digitizing, he can back up to the last vertex digitized by choosing the *backup* option. Backup is the graphical analog of the backspace character in text editing.

All commands and modes can be exited immediately, and without change by chosing the *stop* option.

An item can be removed from the scene by chosing the *clear* option, then picking a vertex of the item to be cleared.



VECTORS RECTANGLE SPLINE

TEXT

Figure 3. *The four object types, together with their defining templates. The templates corresponding to the vectors, rectangle, and spline objects are identical (shown as a backward S). Note that the template is coincident with the vector object. The template defining the text object is shown as two crosses near the X in TEXT.*

7.2. Saving and Loading

The current scene can be saved for later editing by using the long command *:w <filename>*, in the style of vi [3]. A similar command can be given to load an existing file into Grace for editing. This command is *:e <filename>*.

7.3. Exiting the Program

Grace is exited by giving the long command *:q*. The scene, if any, will be removed from the PS300 screen and control returned to the shell.

7.4. Getting Hardcopy

Getting Varian hardcopy is very easy in the Grace system. Save the current scene in a file (using the :w command), then use the long command *:h*. The whole process takes a few minutes (the actual time naturally depends on the current system load) so Grace will beep when ready for more processing.

6. Intermediate Level Use

6.1. Binding

It sometimes happens that a template is bound to the wrong type of object. This situation can be corrected by using the *bind* option. To rebind an object to a template choose the bind option then pick a vertex of the template to be rebound. The current object is bound to the template, and the old object should disappear.

6.2. Undoing

Commands that alter the scene can be undone by chosing the *undo* option. This is a one level undo allowing undos of undos, etc. Commands that change default settings, etc., cannot be undone using this facility.

6.3. Structural Modification

An existing template can be structurally modified in one of three ways; moving a vertex, inserting new vertices between two existing vertices, or by deleting one or more vertices. In general, when a structural change is made to a template the associated object will also be affected. The object is automatically updated when the template is changed.

6.3.1. Vertex Movement

To move a vertex of an existing template choose the *move* option then pick the vertex to be moved. The stylus will be connected to the vertex position as will the segments adjacent to the vertex. (While the stylus is connected to an object, moving the stylus causes the object to move also.) When the vertex is in the proper place, digitize its new position by pressing the yellow puck button. If the digitization is registered the stylus will be disconnected, the object updated and Grace will return to command mode.

6.3.2. Vertex Insertion

To insert a set of new vertices between two existing ones the user must first choose the *insert* option, then specify the region to have insertion performed on it. This is done by picking two adjacent vertices. The segment of the template between the two will disappear signaling that Grace is ready to accept the digitization of new vertices. The order that the vertices are picked is crucial. The new vertices are inserted by starting at the first picked vertex, moving toward the second vertex. As the new vertices are input they will be connected with segments between them but the connections to the picked vertices will not be explicitly shown until the accept option is specified.

To add vertices to either end of an existing template pick the endpoint of interest twice, then digitize the new set of vertices, chosing the accept option to end the mode.

When accept is received the connections to the existing template are shown and the object bound to the template is updated.

6.3.3. Vertex Deletion

To delete a vertex or set of adjacent vertices choose the *delete* option, then specify the region to be deleted. A region is specified by picking two vertices. These vertices, and all between them are removed from the template. Note that one vertex can be removed by picking it twice. As soon as the region has been specified the template and object are automatically updated after which Grace

will return to command mode.

8.4. The Box

The *box* is a graphical entity used to specify an area of the world to be operated on. It is displayed as a rectangle with only the corners showing. The lower left corner of the box is positioned by using the white puck button, the upper right corner of the box is set by using the green puck button. When the lower left corner is set neither the size nor the shape of the box changes, only the position of the box within the world. When the upper right corner is set the position does not change, only the shape changes.

8.5. Windowing

It is possible, and very useful, to move around in the Grace world. The Grace world extends from -10 to 10 in both directions. When the program is started, the scratch pad area is set to view the entire world but this can be changed at any time by using one of the windowing commands. A new window is specified by positioning the *box* within the current scratch pad area. The *zoomin* option is used to zoom into the current box. The *zoomout* is used to do a times two zoom out, keeping the same window center. The last command dealing with the window is the *lastwin* option. Lastwin will return the window to the position it last had. The position of the current window is given in the status area of the terminal screen, the four numbers shown in the window field of the status area represent the window boundary planes.

9. Advanced Use

9.1. Current Set

The options so far have operated on one item at a time. It is possible for some commands to operate on a group of items. Commands of this type operate on the items contained in the current set. The current set is highlighted by making items in the set blink. Items are added to the current set by *selecting* them and removed from the set by *unselecting* them. A single item can be selected by picking a vertex of the item while in command mode. If the pick is registered the item will begin to blink, signaling entry into the current set. A single item can be unselected by picking a vertex of a currently selected item. One can select/unselect a group of items by placing the box around them and chosing the *select/unselect* option. The box must enclose the bounding box of an item's template for the item to be considered a candidate for selection/unselection.

Before explaining the remaining commands some explanation on how commands are categorized may be useful. As was stated above, commands are broken into two major categories, long and short. The short commands are further broken into *primitive* commands and commands that operate on the *current set*.

A primitive command is a command that can be issued in any mode. For example, the windowing commands are primitives since it is possible to give a zoomin or zoomout command while in insert or define mode. Short commands preceded by a '*f*' in the command menu are primitive commands.

Commands that operate on the current set are preceded by '*' in the command menu and behave as follows. If there are no items in the current set the command waits for a single item to be picked, this operation has been explained in previous sections. However if the current set is non-null, all items in the set are operated on. For instance, consider the *clear* command. If the current set is

non-null all items in the current set are cleared as soon as the command is invoked.

9.2. Translation

An item or items in the current set can be translated by chosing the *translate* option. If the current set is null the command waits for a vertex to be picked. The stylus is connected to this vertex making it possible to translate the corresponding template and object in realtime. The final position is specified by a digitization. If the current set is not empty the command waits for a vertex to be picked. This vertex must belong to an element of the current set. The stylus is connect to this vertex and the current set is translated as a unit in realtime. Translation is terminated by digitizing the new location for the set.

9.3. Copying

One of the most powerful features that **Grace** provides is the ability to copy one or more items in the scene.

If the current set is null, the command waits for a vertex to be picked. An exact copy of the item that the vertex belongs to is created and the stylus is connected to the vertex. Perform a digitization at the place where the structure is to stay to disconnect the stylus.

If the current set is non-null, the command waits for a vertex belonging to the set to be picked. All items in the current set are copied and the stylus is connected to the picked vertex. Digitize the final position of the group to disconnect the stylus. If any other type of input is received the program will return to command mode - no other action will be taken.

9.4. Turning Off Templates

Since it is the objects which are of interest, it is often desirable to turn off some of the templates. Templates can be turned off by putting the box around the corresponding items and chosing the *Templ off* option. Templates can be turned back on by chosing the *Templ on* option.

9.5. More on Text Objects

The behavior of the text objects explained in section 7.1 is the behavior of the default *text adjustment mode*. The default adjustment is centered text, meaning that text string is horizontally centered about the lower of the first two vertices of the template. The current default adjustment is displayed in the *adjust:* field of the status area of the terminal screen. To change this adjustment to *left*, i.e. left adjust the text string, use the long command :a *l*. The command :a *r* can be used for setting right adjustment. The current default adjustment can be temporarily overriden by prefacing the text string with a two character sequence of the form <adjustment char>:. For example, suppose the default adjustment is centered text. This adjustment can temporarily made left adjusted by typing:

l:<remaining text string>

when prompted for the text string.

There is also the notion of a font associated with each text object. The font is displayed in smaller text near one of the first two vertices of the template. The default font number can be changed by giving the command f *n*, where n is

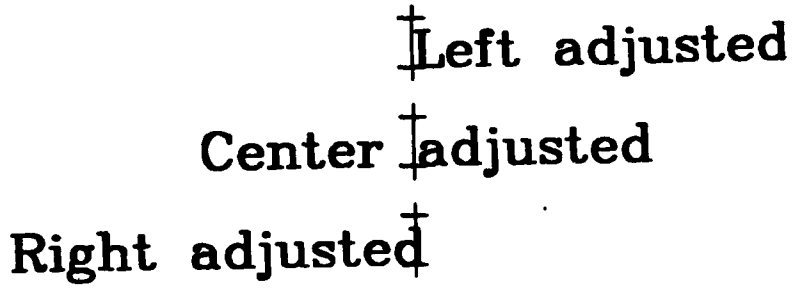


Figure 4. ***The three types of text adjustment. The templates are shown as thin lines with crosses at the endpoints.***

the new default font number (1-9). The default font can be temporarily overriden by prefixing the text string with a two character sequence of the form *n:*, where *n* is the new font number. If both the default font and the default adjustment are overriden the order is irrelevant.

9.6. More on Digitization Aids

9.6.1. The Grid

It was mentioned above that grid adjustment of vertices can be useful for creating symmetrical templates. The grid spacing can be changed by using the *.g f*, where *f* is the new grid spacing, given in world units. Due to a bug in Grace's data flow n.. ork, grid adjustment must be toggled for the new grid spacing to be registed.

9.6.2. The Ruler

If grid adjustment is too confining but some help is needed for vertex placement the *ruler* is useful. The ruler is toggled on by chosing the *Ruletog* option. The ruler can be thought of as a piece of graph paper that is laid down over the scratch pad area. Unlike the grid, the ruler is not part of the world, it is part of the display screen only. The ruler provides no explicit adjustment; it is meant merely as a visual aid in vertex placement.

9.6.3. Distance

As measuring aid, the *distance:* field of the terminal screen gives the distance, in units of grid spaces, between the last digitized position and the digitization before it.

9.7. More on Beta Splines

A spline curve is a piecewise function satisfying some continuity constraints. A common form of splines, the B-splines, have continuity of the first and second derivatives where the pieces meet (often called the join, or joint). Grace uses a new spline technique called Beta-splines[4]. Beta-splines are a generalized form of B-splines, allowing the specification of *bias* and *tension* at

joint. The bias factor is called *beta1* and tension is also known as *beta2*. ing tension to a Beta-spline (i.e. increasing the beta2 value at each joint) s the curve to follow the template more closely. In the limit of infinite , the curve would follow the template exactly. Bias alters the shape of the by causing the curve to follow the direction of the unit tangent vector r on one side of the joint than the other.

Globally Changing Beta1 and Beta2

When a spline is bound to a template, either at the *accept* stage of a ition or as a result of an explicit bind, the values of the default beta1 and become part of the spline's definition, i.e., the default beta1 and beta2 me the shape parameters for all of the curve's joints. To change the default gs use the :b option. This command requires two floating point numbers, and beta2, respectively. The current default shape parameters are shown e *Beta1* and *Beta2* fields of the status area of the terminal screen. When e is initiated, beta1 is set to 1 and beta2 is set to 0, indicating a bias of 1 o tension, the values which generate the special case of a B-spline curve.

As an example, suppose there is a spline curve that was defined with the lt parameter settings and it is desired to increase the tension of the curve. s done by first changing the default setting by typing:

```
:b 1 10
```

sets beta1 (bias) to 1 and beta2 (tension) to 10. Now we can re-bind a spline these shape parameters by chosing the bind option, then picking a vertex ging to the spline's template. As the binding proceeds you should be able e the curve "hug" the template more strongly than before. It is instructive e the undo facility to "ping-pong" between the two sets of shape parame- (see figure 5).

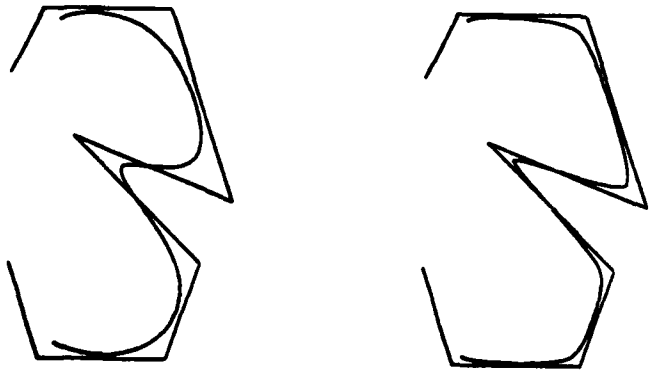


Figure 5. ***The effect of globally changing the tension. The Beta-spline on the left has a tension value of 0, the one on the right has a tension of 10.***

9.7.2. Local Changes in Beta1 and Beta2

The basic theory of Beta-splines[4] was expressed in terms of a beta1 and beta2 which could either be global or local. The latter case is particularly useful for the purposes of computer aided geometric design, and for this reason this is being studied and extended in [5].

Using these techniques, Beta-splines with local shape parameters have been implemented in **Grace**. To change the shape parameter at a single joint choose the *vary* option, then pick the vertex corresponding to the desired joint. The current shape parameters are displayed on the terminal screen along with a prompt for the new values. Local changes affect only the segments on either side of the joint, the rest of the curve remains invariant.

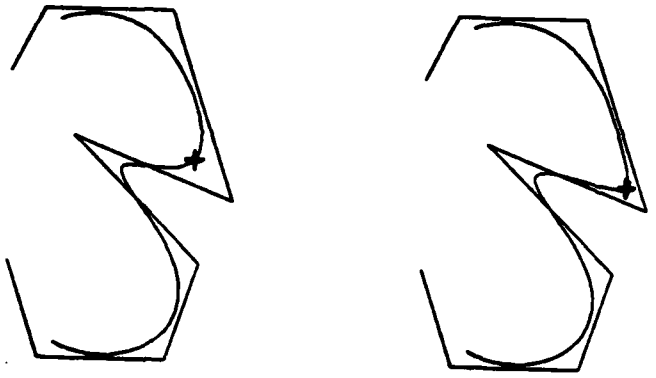


Figure 6. *The effect of locally changing the tension. The tension has been changed at the joint marked with a cross from 0, on the left, to 10 on the right.*

10. Future Enhancements

Rotation and scaling of items and the creation of a circle object type are the main short term enhancements currently planned. Following the lead of **Gremlin**, various line/curve thicknesses may be implemented in the near future, depending on user response, etc. In the not too distant future we hope to develop a three-dimensional version of **Grace**. The extension to three dimensions introduces many new problems, but many of the concepts used in two dimensions can be utilized.

References

[1] Ousterhout, John C., *Editing VLSI Circuits with Caesar*, EECS Department, U.C. Berkeley.

[2] Roitblat, Barry, *The Gremlin Picture Editor*, EECS Department, U.C. Berkeley.

[3] Joy, William, *Introduction to Display Editing with Vi*, EECS Department, U.C. Berkeley.

[4] Barsky, Brian A., *The Beta-spline: A Local Representation Based on Shape Parameters and Fundamental Geometric Measures*, PhD Thesis, University of Utah, December 1981.

[5] Barsky, Brian A., and Beatty, John C., *On Varying the Betas in Beta-Splines*, to appear.