

A GREMLIN Tutorial
for the
SUN Workstation

Mark Opperman, Jim Thompson, and Yih-Farn Chen

December 1986

Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley 94720

Table of Contents

1.	Introduction	1
	1.1. GREMLIN History	1
	1.2. Distribution of GREMLIN	1
	1.3. Purpose of This Tutorial	1
2.	Overview	2
	2.1. Getting Started	2
	2.2. Interactive and Non-interactive Modes	4
3.	Drawing Commands	4
	3.1. Box	4
	3.2. Vector	6
	3.3. Undo	6
	3.4. Arc	7
	3.5. Arrow	8
	3.6. Curve	8
	3.7. Poloygon	9
	3.8. Text	9
4.	Editing Commands	10
	4.1. Reference Points	10
	4.2. Defining the Current Set	10
	4.3. Transformations on the Current Set	11
	4.3.1. Move	12
	4.3.2. Copy	13
	4.3.3. Erase	13
	4.3.4. Scale	14
	4.3.5. Rotate	15
	4.3.6. Reflection	16
	4.3.7. Move Point	16
5.	Text and Line Styles	17
	5.1. Setting Current Styles	18
	5.1.1. Line Style	18
	5.1.2. Stipple Fill Pattern	18
	5.1.3. Font Style	19
	5.1.4. Font Size	20
	5.1.5. Text Justification	20

5.2. Modifying Styles	20
5.3. Modifying Text	21
5.4. Modifying Object Types	21
6. Placing Points Where You Want Them	21
6.1. Grid Lines	21
6.2. Alignment	22
6.3. Adjustment	23
6.4. Gravity	23
7. Writing and Reading GREMLIN Files	24
7.1. Writing GREMLIN Files	24
7.2. Writing the Current Set	24
7.3. Reading a GREMLIN File	24
7.4. Editing GREMLIN Files	25
7.5. The Path and Directory Mechanisms	25
8. Miscellaneous Functions	25
8.1. Buffer Operations	25
8.2. Panning	26
8.3. Symbolic Lines, Little Points	26
8.4. ETC Commands	27
8.5. Short Commands	27
9. Printing GREMLIN Pictures	29
9.1. The GRN DITROFF Pipe	29
9.2. Stipple Filled Polygons	29
10. GREMLIN Initialization	30
10.1. Command Line Options	30
10.2. The <i>.gremlinrc</i> File	31
Appendix A — Gremlin File Format	35
Appendix B — CIFPLOT, UNIGRAFIX and MAGIC Stipple Patterns	38
Appendix C — Examples of Curve Styles	41

1. Introduction

GREMLIN is an interactive graphics editor for UNIX systems. It permits various objects to be drawn and edited, including circles, rectangles, lines, curves, and polygons. Objects may be modified by moving, copying, resizing, or changing line and stipple styles. Text may be included in the picture in several sizes and fonts. GREMLIN pictures may be saved on disk for later editing, or included in typeset documents for printing. Versions of GREMLIN exist for the SUN workstation and the AED graphics terminal. This tutorial describes the version for the SUN running under the *suntools* environment.

1.1. GREMLIN History

GREMLIN's legacy encompasses more than five years and a half-dozen Berkeley graduate students. It all started in 1981 when Barry Roitblat built the first version of GREMLIN for his Master's project. That version ran only on AED color displays, and its output could be printed only on Versatec printers. In order to include figures in typeset documents, they had to be cut-and-pasted. In 1983, Dave Slattengren (another graduate student at UCB) acquired from AT&T the sources to Brian Kernighan's new DITROFF program. In addition to making the program work under 4.2 BSD and building drivers for several different printers, he wrote GRN, which reads files in GREMLIN format and generates DITROFF commands to print the pictures in-line in documents. In 1984, Mark Opperman, yet another UCB graduate student, rewrote GREMLIN to run on SUN workstations. He made massive improvements to the user interface and added stippling. In 1985 and 1986, Yih-Farn Chen maintained the system through several system upgrades and made several enhancements, including better stipples and curves. In 1986, Jim Thompson again made major improvements to the user interface, exploiting the capabilities of the SUN to make the commands more interactive. Peehong Chen contributed the variety of curve routines for GREMLIN and DITROFF. Great credit is also due to Professor John Ousterhout who sponsored and encouraged much of the development of GREMLIN.

1.2. Distribution of GREMLIN

GREMLIN is distributed free of charge by the University of California, Berkeley, along with a modified version of the DITROFF typesetting system which allows GREMLIN pictures to be printed in-line in documents. To find out more about the GREMLIN/DITROFF distribution, including the AT&T licenses required to receive it, write to:

Prof. John Ousterhout
Computer Science Division, Dept. of EECS
University of California
Berkeley, CA 94720

or electronically to ARPANET address: ouster@arpa.Berkeley.EDU

1.3. Purpose of This Tutorial

This tutorial is designed as an introduction to the GREMLIN picture editor for those with no previous exposure to it. However, it also describes the less frequently used commands of GREMLIN and as such, it may prove useful to those who already have some experience with GREMLIN, perhaps on the AED. Section 2 provides an overview of GREMLIN, its windows, and the use of the mouse. Sections 3-7 explain how to use GREMLIN to draw and edit pictures. Section 8 describes commands for reading and writing GREMLIN pictures to a file. A method of printing GREMLIN pictures is described in Section 9. Finally, Section 10 describes initialization and customization of GREMLIN.

This tutorial does not provide an introduction to the SUN workstation nor its window system. It is assumed that you are already familiar with *suntools*. Familiarity with DITR-OFF and any of its preprocessors would be helpful, though not necessary.

2. Overview

This section describes how to start GREMLIN and provides an overview of its structure. It is best if you read it while sitting in front of a SUN workstation so that you can try the functions as you read about them.

2.1. Getting Started

To begin GREMLIN you will first need to start up *suntools* and open a shell window[†]. Although you can start GREMLIN in the foreground, you should probably run it in the background since it will create its own window. Entering the command `gremlin&` should open a GREMLIN window. If it doesn't, `/usr/local` is probably not in your `PATH` environment variable and you will need to enter the full path name, i.e. `/usr/local/gremlin&`.

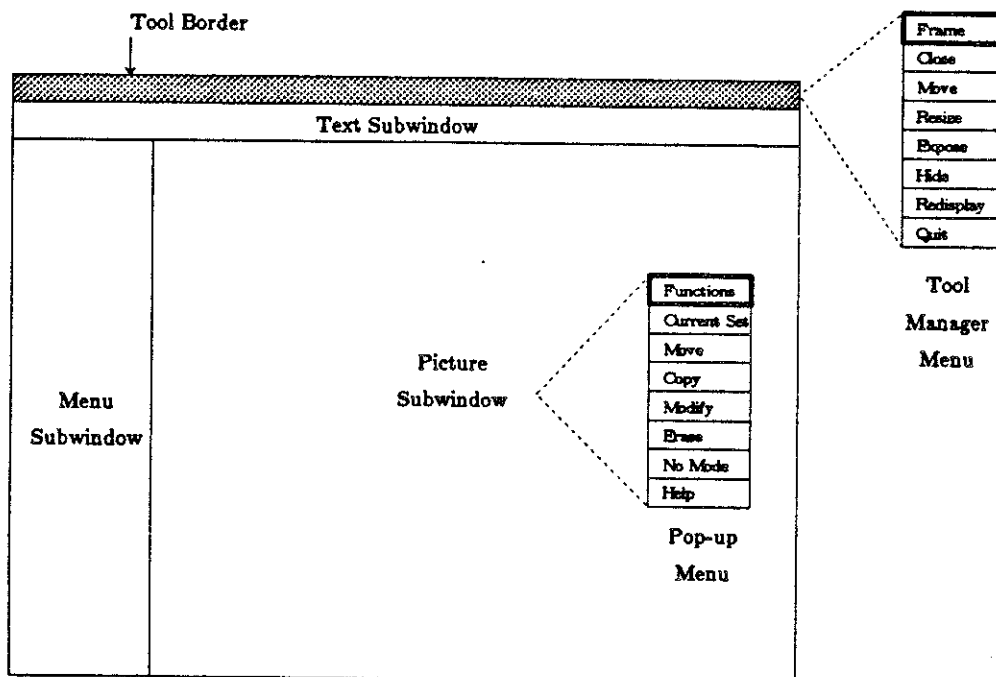


Figure 1: The Layout of GREMLIN Tool

GREMLIN provides the standard window manipulation functions in its tool manager pop-up menu. This menu is displayed by pressing the right mouse button with the cursor on the tool border (including subwindow boundaries). It includes commands like *close*, *move*, *resize*, *expose*, and *hide*. In particular, you will need to use this menu to *quit* GREMLIN or to *redisplay* the picture should GREMLIN become confused.

[†] If you have no idea what this means, see the *suntools* entry in the "User's Manual for the Sun Workstation,".

The GREMLIN window is divided into three subwindows: the *picture subwindow*, the *menu subwindow* and the *text subwindow*. The layout of these subwindows is shown in Figure 1. The tool border identifies the version of GREMLIN and the name of the file being edited. It should look like the following:

```
Sun Gremlin 2.7 (86.10.1) <> Editing
```

where 2.7 is the version of GREMLIN being run. If it indicates an earlier version then some features described in this tutorial will not be available, and you may want to acquire a current version.

The *picture subwindow* is where you will define and display picture objects. The *menu subwindow* is full of menu *icons* (symbols) which are used to select actions or functions affecting picture objects. The *text subwindow* stretches across the top of the GREMLIN window and is used to display messages and to enter text arguments for a few commands. The cursor will change shapes as you move the mouse across the various subwindows. You should experiment now with moving the cursor to each subwindow.

In the picture subwindow the cursor is displayed as a small arrow pointing up and to the left. When you are in the picture subwindow, try pressing the left mouse button a few times. With each button click, a point with a number next to it will be displayed - this is called *placing a point*. You can use the middle mouse button in the picture subwindow to erase points in the opposite order from which they were placed. Finally, pressing the right mouse button displays a pop-up menu of common commands. These will each be described later, but for now notice that the last menu item is *Help*. Selecting this option will cause a Help screen to be displayed over the picture subwindow. This particular help screen provides help on the use of the mouse in the picture subwindow. Try reading this help screen now; when you've finished, press any mouse button to bring back the picture subwindow. Table 1 summarizes the use of the mouse buttons in each of the subwindows.

Summary of Mouse Button and Key Usage			
Button	Menu Subwindow	Picture Subwindow	Text Subwindow
Left	Select function	Place point	Insert from current set or Move cursor
Middle	Modify current set	Remove point	Repeat last string
Right	Help	Pop-up menu	Help
Keyboard	Short command	Short command	Enter text

Table 1: Summary of mouse and keyboard usage

In the menu subwindow the cursor is displayed as a small diamond. As you move it near and across an icon, that icon will be highlighted. Pressing and releasing (*clicking*) the left mouse button with the cursor over a highlighted icon will invoke the function represented by the icon. This action is referred to as *selecting* the icon. Try this with the *Help* icon, which looks like this:

Help ?

Other icons will be similarly shown throughout this tutorial. Selecting this icon will display a Help screen describing the use of each of the GREMLIN subwindows. The

middle mouse button usually selects a modified version of the function, in most cases operating only on the *current set*, described later. In the case of the *Help* icon, the middle mouse button displays a help screen describing short commands, described in Section 8.5. **WARNING:** It would be best to wait until the other icons have been described before selecting them. For the moment, however, try using the right mouse button with the cursor over another icon. Doing so will cause a Help screen to be displayed describing the use of that particular icon.

The menu subwindow icons are grouped according to three categories: *style*, *drawing* and the catch-all *other*. The *style* icons are used to select font types and sizes, line styles, stipple patterns and text justification. The *drawing* icons are used to invoke functions that create or modify picture objects. The remaining icons represent a variety of *other* functions that facilitate drawing and provide file access. The progression of this tutorial will be from the drawing icons to the style icons and finally to the other icons.

In the text subwindow the cursor is displayed as a small open box. This indicates that GREMLIN is ready to receive input from the keyboard. In addition to the mouse cursor, there is a solid black text cursor. Typing something from the keyboard now will cause the characters to be displayed at the text cursor. There are a few special characters that GREMLIN recognizes. You can use your *backspace* and *line erase* (usually $\wedge U$) characters normally, as well as $\wedge W$ to delete a word.[†] You can also move the text cursor forward and backward in the text using $\wedge F$ and $\wedge B$, or the keyboard left and right arrows. The left mouse button will also move the cursor. However, all other control characters (including *tab*) will be ignored, with the exception of *return*. The *return* character is used when entering text objects into a picture and is discussed at the end of this section. Again, the right mouse button in the text subwindow will cause a Help screen to be displayed.

2.2. Interactive and Non-interactive Modes

Most drawing and editing functions can be done either in *interactive* mode or *non-interactive* mode. In *interactive* mode you select a function using the menu icon, then perform the function on the picture using the mouse. As the action is performed you see the result in the picture. On the other hand, in the *non-interactive* mode, you place points in the picture (using the left mouse button), then select the function to be performed using the points. Interactive mode is easier to use for most things, but is not available for all functions. It is also quicker to do some things non-interactively. Try both to see which you prefer.

GREMLIN always starts in the non-interactive mode. An interactive function is chosen by selecting the icon with the left mouse button before placing points in the picture. A box is drawn around the icon to show the current interactive function. To leave an interactive function simply select the same or another interactive icon. If another icon is selected, it becomes the interactive function; if the same icon is selected, GREMLIN returns to non-interactive operation. Selection of an icon which does not have an interactive mode, such as one of the *style* icons described later, will have no affect on the current interactive function. The next section will demonstrate both interactive and non-interactive operations using simple drawing commands.

[†] The caret (\wedge) preceding a character means to press and hold the CONTROL key while typing the character.

3. Drawing Commands

This section will show you how to draw objects both interactively and non-interactively, starting with a *box*. You should try each of the commands as they are described. Each description begins with a picture of the icon for the function, for example:

3.1. Drawing a Box

Draw Box 

All objects in the picture are drawn by specifying points which determine the location and shape of the object. A box is specified by two points at opposite corners. To draw a box non-interactively, place two points in the picture subwindow (with the left button). Now move the cursor to the menu and select the *draw box* icon with the left button. The points will disappear and be replaced by a box.

Notice that the box you just drew is flashing. This indicates that the box is in the *current set*, something that you need not be concerned with for the moment. As you experiment drawing more boxes, you will notice that as each new box is displayed it will begin flashing and the previously drawn box will stop flashing. Each time a new object is created, it becomes the *current set*.

To draw a box interactively, select the *draw box* icon before placing any points in the picture subwindow. (Don't worry if there are already other objects in the picture, as there should be if you are following the tutorial. As long as there are no numbered points placed in the picture, GREMLIN will begin an interactive function.) The *draw box* icon will be highlighted with a square to show the current interactive function. Now move the cursor to the place in the picture subwindow where you want to draw the box. Press and hold the left mouse button at one corner of the new box. With the button pressed, move the mouse and a box will appear with the opposite corner following the mouse cursor. This operation of moving the mouse with a button down is called *dragging*. When the box is the proper shape, release the button, and the new box becomes the current set. Figure 2 illustrates the operations involved. You can draw as many boxes as you like without having to go back to the menu. To stop drawing boxes, reselect the *draw box* icon.

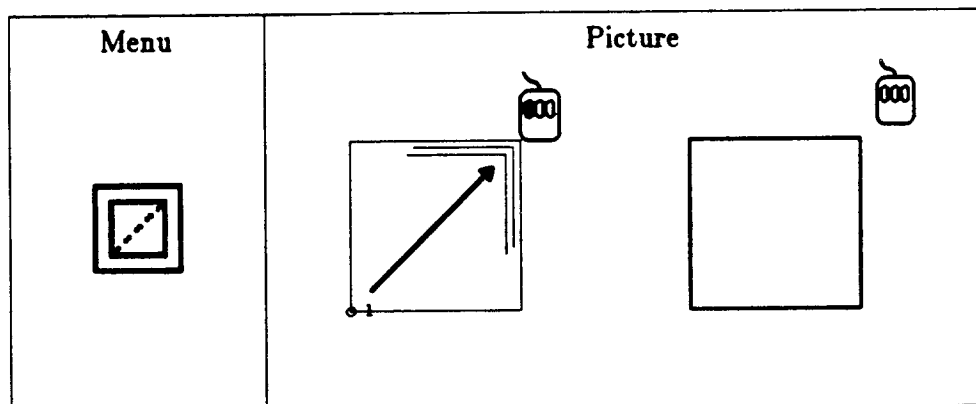



Figure 2: Interactively Drawing a Box

The interactive style of "press and drag" is used for all interactive functions which require only two points. In addition to boxes, this includes arrows, circles, move, and copy, as described later. A different style is used for functions requiring several points. This is illustrated with the *draw vector* function.

3.2. Drawing a Vector

Draw Vector 

A vector is a sequence of connected line segments specified by their end-points. To draw a vector non-interactively, place several points in the picture subwindow and select the *draw vector* icon. The points will be connected with lines, and the points will disappear. Again, the new object becomes the current set.

There are two possible styles of interactive drawing for vectors and other multi-point objects. Begin, as always, by selecting the *draw vector* icon before placing points. Using the first style, click the left mouse button at the first endpoint of the vector, then move the mouse to each succeeding point and click the button again. As you move the mouse, interactive feedback shows the vector so far, including a line from the previous point to the cursor. To end the vector, click the button twice on the final endpoint. The second style is similar, except you drag each point into position. Release the left button when the point is in the proper place, then press and drag the next point. Again, a click at the final point indicates the end of the vector object. Draw a few vectors using both styles to see which you prefer. Figure 3 illustrates the actions of interactively drawing a vector.

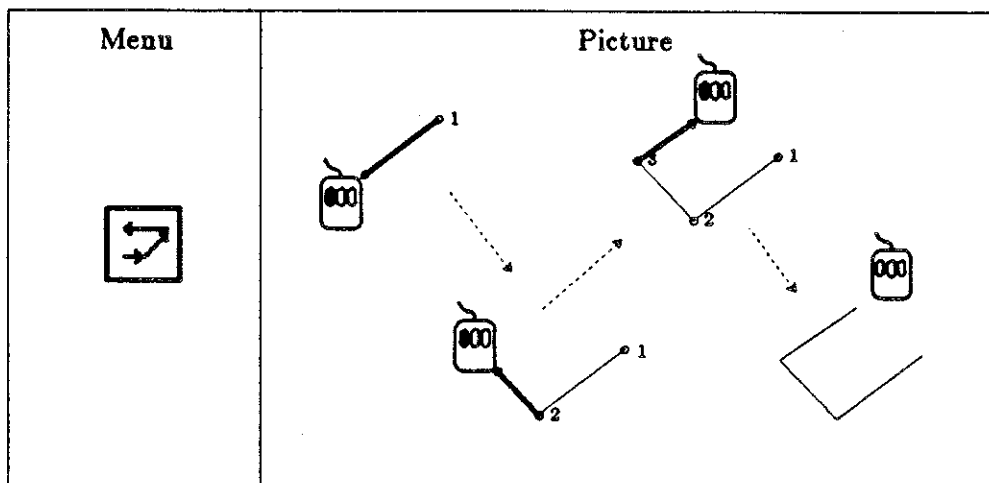


Figure 3: Interactively Drawing a Vector

The middle mouse button may be used to remove points from a vector any time before the final point is specified. Points are removed in reverse order, just as they are when placing points non-interactively. The interactive feedback will adjust itself to show the new vector with the prior point removed.

The box and vector illustrate the actions used for all other drawing functions. Before presenting the rest of the drawing commands, there is one important function to describe.


3.3. Undo

Undo Last Command 

The *undo* icon is worth mentioning early because it can save a great deal of time and effort. When you select this icon with the left mouse button, the last change to the picture subwindow will be *undone*. This is a one-level undo, so that if you select the icon twice, the picture will return to its state before the first undo. Get to know this icon now.

Undo is one of the commands in the pop-up menu available using the right mouse button in the picture subwindow. It is also available in most interactive functions using the middle mouse button. In most cases, the middle button removes points if you are in the process of drawing an object, or *undoes* the last change if you are not. Remember, if you undo something accidentally, just select *undo* again to recover the change.

3.4. Drawing an Arc

Draw Arc/Circle 

The *arc* icon is used to create arcs and circles, depending upon the number of points used. With two points, a circle will be drawn such that its center is at the first point and its radius is defined by the distance between the two points. If three points are placed, an arc will be created. Again, the first point defines the center of the arc and the second point defines a point on the arc, in particular the start of the arc. The arc will be drawn counterclockwise until a point is reached such that a line drawn from the first point to the third point would intersect the arc at that point. The examples in Figure 4 illustrate non-interactive circle and arc drawing.

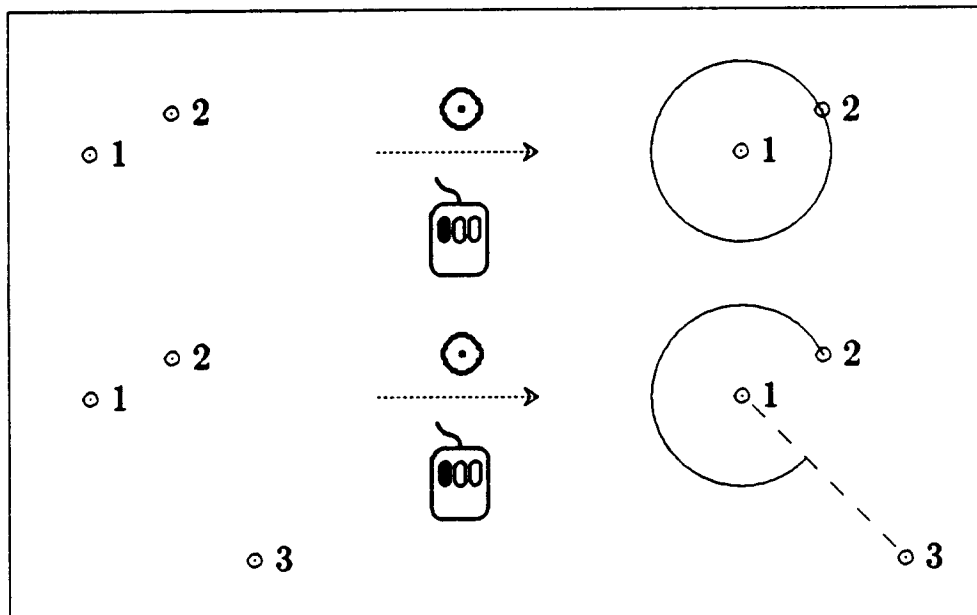


Figure 4: Drawing Circles and Arcs

There is currently no way to interactively draw an arc. Circles are drawn interactively by pressing the left button at the center and dragging the perimeter into position, similar to the way boxes are drawn.

3.5. Drawing an Arrow



There are three types of arrows available: arrowheads, an arrow with line attached, and a line with an arrowhead at both ends. The type is chosen by using the middle mouse button on the icon, which will change to show the current type. Two points are used to define an arrow. The arrow is drawn in the direction from the first point with the arrowhead located at the second. Interactively, press the left button at the tail and drag the head into position. NOTE: Some earlier versions of GREMLIN drew arrows in the opposite direction.

Arrows of all types, once created, are simply vectors. Therefore it is possible to do such things as replace the tail of the arrow by several points, creating a "bent" arrow, using functions described later. However, moving the tip of the arrow, or turning the arrow into a curve, although possible, rarely produce the expected result.

3.6. Drawing a Curve



The curve object is very similar to the vector object. Instead of connecting points with straight lines, however, the curve function connects the points with a smooth curve. You will need to use at least three points to get an interesting curve. There are three types of curves supported by GREMLIN: *Interpolated splines* (also called *GREMLIN splines*), *B-splines*, and *Bezier curves*[†]. The GREMLIN curve is guaranteed to pass through all of the points, whereas the other two types of curves are only constrained to lie "near" the points and within the area bounded by the points. Therefore the B-spline and Bezier curves are usually smoother curves than the GREMLIN curve. If the first and last points coincide, the GREMLIN and B-spline will produce a smooth, continuous curve. In all other cases, all three curve types will pass through the end points.

Currently at Berkeley, the Imagen printer with GRN can only print GREMLIN splines or B-splines. The Laserwriter can print all three types of curves.


The default curve type is the B-spline, represented by the center of the three icons shown above. It is the most reasonable curve to use for most situations. To change the type of curve to be drawn, hold down the **Shift** key and select the *curve* icon with the middle mouse button. The icon will change to the next type of curve, and the name of

[†]To be technical, the GREMLIN curve is drawn using the parametric spline curve on unit knot mesh described in "Spline Curve Techniques" by Baudelaire, Flegal, and Sproull, Xerox PARC. The B-Spline is a uniform cubic B-Spline as described in Chapters 4-5 of "An Introduction to the Use of Splines in Computer Graphics" by Bartels and Barsky, Tech. Report CS-83-136, Computer Science Division, University of California, Berkeley, 1984. The Bezier curve is a third-degree Bezier curve as described in "Fundamentals of Interactive Computer Graphics" by Foley and Van Dam.

the curve type will be displayed in the text window. The first and third icons shown above are for the Interpolated and Bezier curves, respectively. The default curve type can be set in the GREMLIN startup file, described in Section 10. The unshifted middle button has a different meaning, as described in Section 5.4.

Curves are drawn interactively just like vectors. When drawing a curve interactively, the feedback just connects the points with a straight line. The lines become a curve once the final point is specified.

3.7. Drawing a Polygon

Draw Polygon 

Draw Bordered Polygon 

Polygons are defined by three or more points and may be created with or without a border as depicted by the two icons above. If the first and last points are not the same, an implicit side will be created connecting those points. Whenever a polygon is displayed it will be filled with a stipple pattern. Selecting and changing stipple patterns is discussed in Section 5.

Polygons are interactively drawn just like vectors. The feedback is a closed, unfilled polygon which becomes filled when the final point is specified.

3.8. Drawing Text

Display Text **TEXT**

No picture is complete without some text sprinkled about here and there. Text is entered from the keyboard with the cursor in the text subwindow, as described earlier, then moved into the picture at specified points. There are two ways to do this.

In most cases you will want to enter several text strings at one time to reduce the amount of switching from mouse to keyboard. To do this, you first place points in the picture for as many text strings as you would like to add to the picture. Then you move the cursor to the text subwindow and begin entering the strings, each terminated by a *return* character. The first time you hit return, the text string will disappear from the text subwindow and appear in the picture at the location of the first point placed. That point will also disappear, and all remaining points will be renumbered. Similarly, each time you hit return the string will be displayed relative to the point currently numbered "1" and that point will be removed from the display. This method relieves the tedium of moving back and forth between subwindows and switching from mouse to keyboard input. Notice that the *text* icon is not used at all in this method.

The second method does require the use of the icon and is only useful when you want to center text in an object, a box for example. After typing the text into the text subwindow, place two points in the picture subwindow at opposite corners of the box. Then select the *text* icon. The text will be displayed at the midpoint of the line connecting the two points (i.e. the center of the box). Repeat this for each string. You can also enter text at a single point using this method, but there is no need to bother. There is currently no interactive method of entering text.

When you display text, a small dot will appear slightly below and to the left of the text string while the text remains in the current set. This dot indicates that the string was entered with "bottom left" justification. There are eight other text justification modes in GREMLIN; these will be explained further in Section 5.1.

4. Editing Commands

You now know how to draw every type of object that can be created with GREMLIN. However, GREMLIN also provides many more tools that aid in creating and editing pictures. Integral to these tools is the notion of the *current set*.

The *current set* is simply a collection of objects in the picture upon which transformations and modifications can be applied. Each time you create a new object it becomes the single element of the current set. In addition, there are commands that allow you to define the current set or add objects to the current set.

4.1. Reference Points

Before exploring the commands that select the current set, another concept must be introduced: picture object *reference points*. When each object is created or modified, GREMLIN associates certain *reference points* with it internally. These points are used when selecting objects to be included in the current set. For example, when you create a vector object or a curve object, its reference points are exactly those points that you placed when defining the object. Boxes and arrows, once created, are also vectors, with reference points at each vertex. For circles, however, GREMLIN adds some points that will help in selecting it later (you may have forgotten where the point is that was used to define the circle's radius).

To make this clear, you should add a circle to your picture now. With the circle flashing (indicating that it is the current set), select the *etc* icon (shown below).

Miscellaneous Commands **ETC**

Selecting this icon will cause a pop-up menu to be displayed. By choosing the *Show Points* item in the pop-up menu, you will cause the reference points of all objects in the current set to be displayed, in this case for the circle only. This is illustrated in Figure 5. Notice that, in addition to the two points used to define the circle, GREMLIN has added four points at 3, 6, 9, and 12 o'clock. You will probably want to do this same experiment with the other types of GREMLIN objects as well. If you do, you will find that GREMLIN only adds points, other than those that were used to define the object, to text and to arc objects.

4.2. Defining the Current Set

You can select objects other than the one just created for inclusion in the current set by specifying a reference point of the object, or by defining an area containing the object. Both way can be done either interactively or non-interactively.

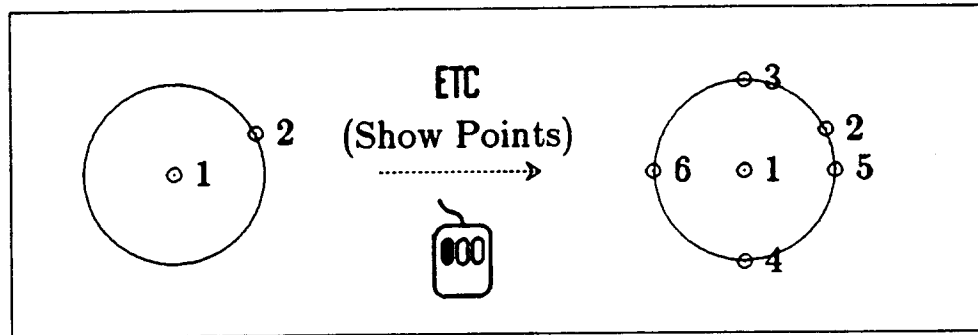


Figure 5: Reference Points for the Circle Object

Define Set



With the non-interactive *define set* method, points are placed near a reference point of those picture objects that you would like to include in the current set. After selecting the *define set* icon (shown above) with the left mouse button, those objects will begin flashing to indicate that they now constitute the current set. The *define set* icon can also be used to include the entire picture in the current set by selecting it with no points placed in the picture. This option requires confirmation by a second click of the left button.

Box Include



Alternatively, you may include objects in the current set by placing two points which define a rectangular area (as you would when creating a box) and selecting the *box include* icon (shown above) with the left mouse button. This will select all those objects whose reference points are entirely included in the box to be in the current set.

Using either of the above two non-interactive methods results in redefining the current set to be just those objects selected. Often you would like to simply *add* one or more objects to the current set. You can do this by specifying the elements exactly as before and then pressing the middle mouse button on the appropriate icon. This is one example of how the middle mouse button is used to represent *modifications* of the function performed by the left button, in this case modifying the current set by adding elements to it instead of redefining it as with the left mouse button. Each of these techniques for defining the current set is illustrated in Figure 6. NOTE: in each example of Figure 6, the box is assumed to be the only object in the current set before the selections are made.

The current set may also be defined interactively by selecting the *box include* icon with no points placed. This interactive command is a combination of the two non-interactive defining operations. The left mouse button selects objects to become the current set, while the middle button adds objects to the current set. In both cases, clicking the mouse button selects the object nearest the mouse cursor. Pressing and dragging produces a rectangle on the screen. When the button is released, all objects entirely within the rectangle are selected. This function is also available as the *Current Set* option on the pop-up menu in the picture subwindow.

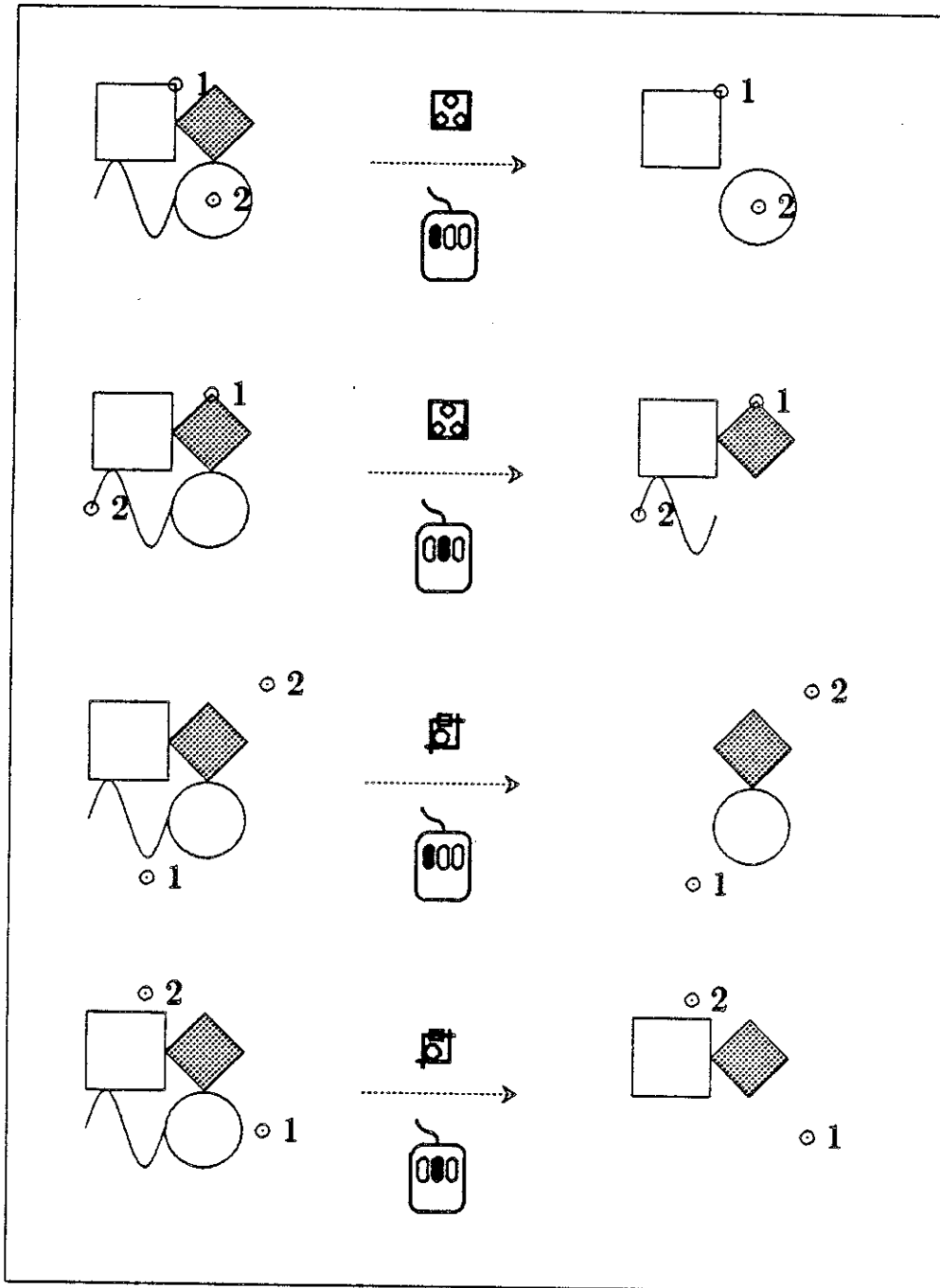


Figure 6: Defining the Current Set (Assuming the box is currently in the current set)

4.3. Transformations on the Current Set

Once the current set has been defined, you may invoke a variety of commands that will transform it one way or another: *move*, *scale*, *rotate*, *horizontal reflection*, *vertical reflection*, *copy*, *erase* and something called *move point*. These will each be discussed in turn.

4.3.1. Move

Move Current Set 

To move the current set to another place in the picture subwindow, place two points and then select the *move* icon. The current set will be moved (translated) by an amount equal to the relative distance between the first and second points. See the example in Figure 7.

The interactive move operates on the current set, which must have been previously defined. Press the left button at a point anywhere on the screen. While the button is down, the current set is dragged in relation to the mouse motion. When the button is released, the current set is redrawn in its new location. The image while moving may be "clipped" where it used to abut the edge of the picture, but will be redrawn correctly when the mouse button is released. If the cursor moves out of the picture while moving, the current set returns to its location before the move began. *Move* is another of the pop-up menu options in the picture window.

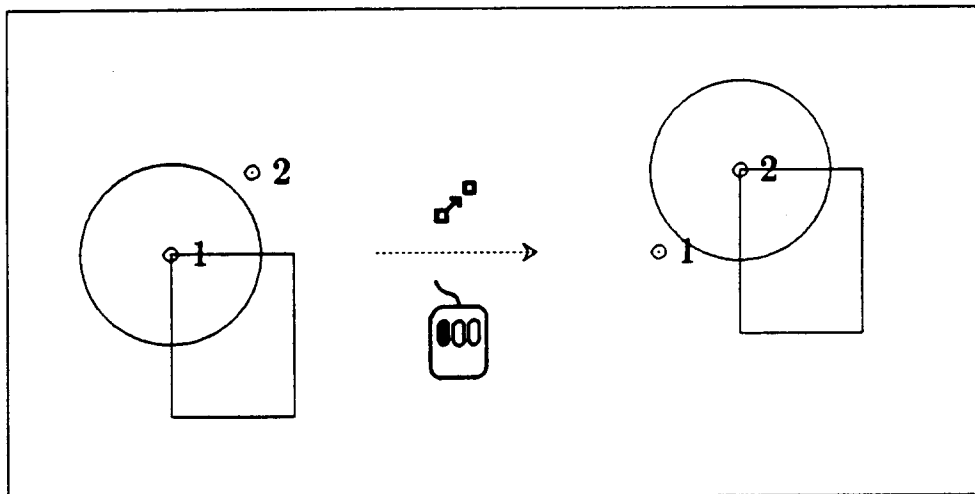


Figure 7: Moving the Current Set

4.3.2. Copy

Copy Current Set 

The copy command will create duplicate instances of the current set and position them according to the points specified. This command works very much like the move command except that the current set can be moved to more than one place at a time and a copy is left at each place, including the starting position. Use two points to make one copy, three points to make two copies, etc. See the example in Figure 8.

Interactive copy works just like Move, except that the original current set is left in place and a single copy is dragged. The new copy becomes the current set.

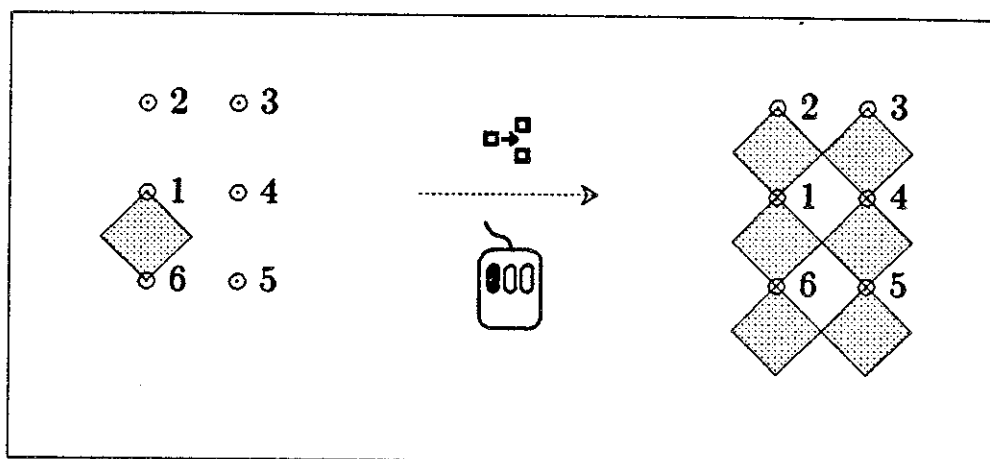




Figure 8: Copying the Current Set

4.3.3. Erase

Erase Current Set 

Selecting this icon will cause all objects in the current set to be erased. If you select it by mistake, the undo command will restore what was just erased. To erase the entire picture, select all objects into the current set, then erase it.

4.3.4. Scale

Scale Current Set 

Sometimes it is easier to initially draw an object larger than its intended size. Later, you can scale it to fit a specific area in the picture. The scale command has three options. You can scale a current set by modifying its reference points' X coordinates, Y coordinates, or both coordinates. A pop-up menu is invoked when you select the *scale* icon, from which you can select one of the three scale modes: (1) *Scale X*, (2) *Scale Y*, and (3) *Scale XY*.

The scale command uses three points to specify the scaling factor. The current set will be scaled by the ratio of the distance between the first and third points and the distance between the first and second points. Thus, the operations can be viewed as defining the current size of the object with points one and two and then defining the desired size of the object with a third point, relative to the first. This easily allows making the current set larger or smaller, as illustrated in Figure 9 (all objects are assumed to be in the current set). There is currently no interactive scaling.

Try scaling a rectangle, a vector, a curve, or any picture element using the three scale modes. The *Scale X* and *Scale Y* modes work for picture elements of any shape except a circle. You can't scale a circle in X or Y direction and expect it to become an ellipse. This is because only two reference points are used to define a circle, and you cannot make an ellipse out of two reference points.

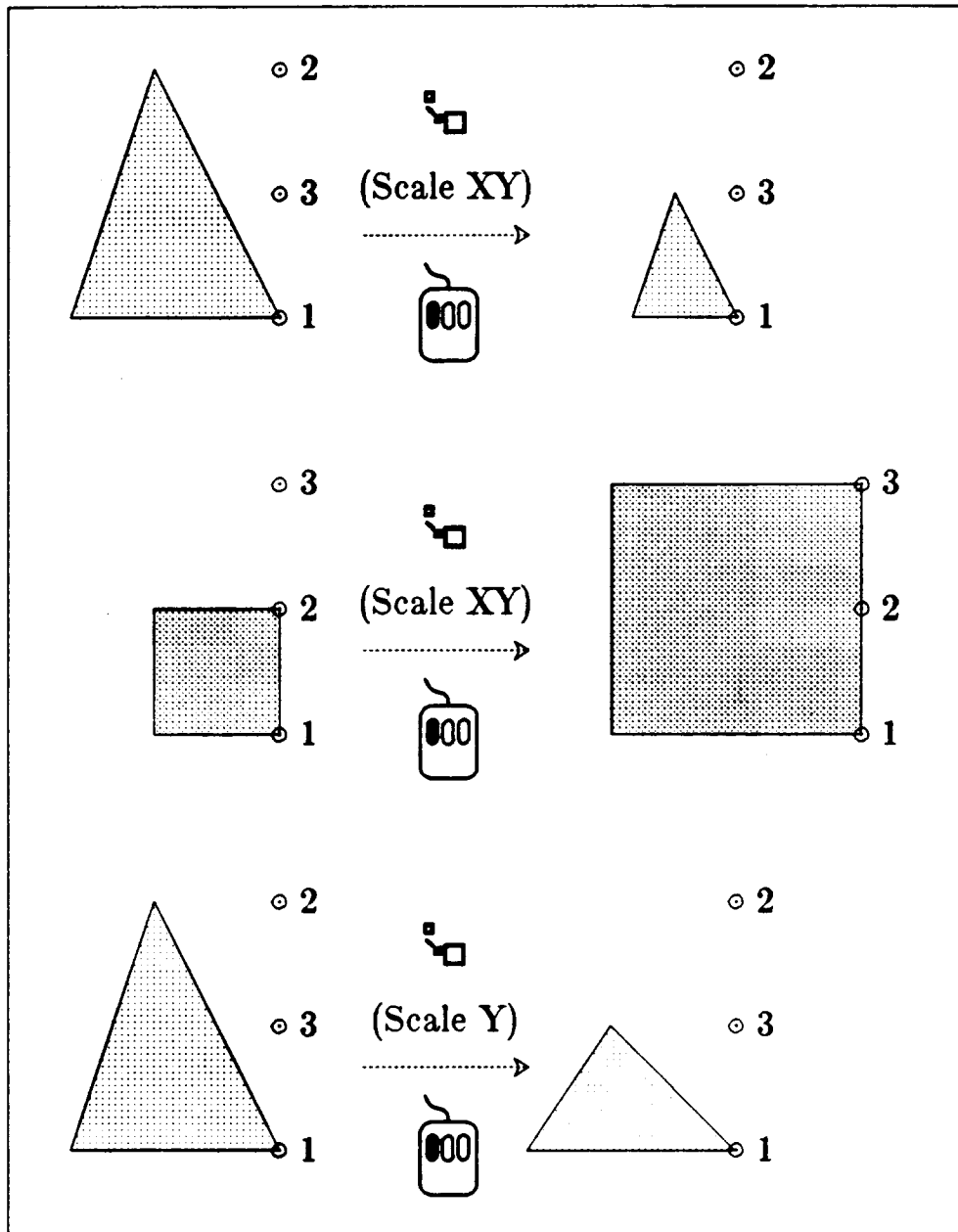



Figure 9: Scaling the Current Set

4.3.5. Rotate

Rotate Current Set 

To rotate the current set, you will need to place three points. The angle of rotation will be defined as the angle formed by the imaginary lines between the first and second points, and the first and third points. This is illustrated in Figure 10. There is currently no interactive rotate.

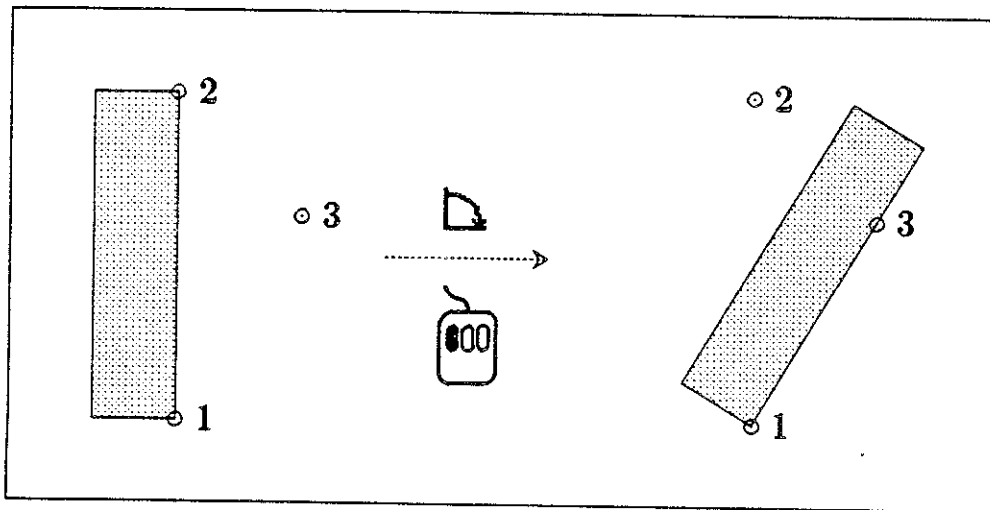




Figure 10: Rotating the Current Set

4.3.6. Reflection

Horizontal Reflection 

Vertical Reflection 

To complete the group of geometric transformations allowed on the current set, there are two commands for performing reflections (NOTE: these commands supersede the *mirror* command in the AED version). To perform a reflection you will need to place just one point. Depending upon the type of reflection selected, the current set will be reflected about either a horizontal or vertical axis containing the point. See Figure 11 for two examples. There is currently no interactive mirror function.

Note that for the scale, rotate and reflection operations, text objects are treated somewhat differently. These operations will affect only the **reference points** of text objects and not the size of the text nor the direction in which it is displayed. Text will always be displayed horizontally even though these operations will affect its *position* within the picture.

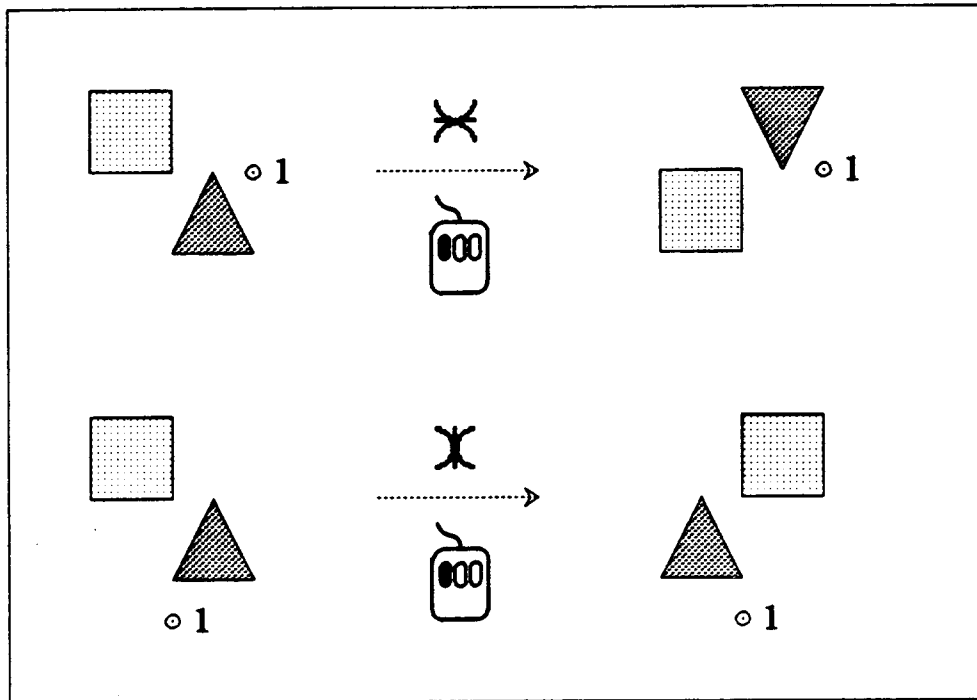


Figure 11: Horizontal and Vertical Reflection of the Current Set

4.3.7. Move Point

Move Reference Point



The *move point* command is used to change the position of a reference point of an object in the current set (except text objects). It has different effects depending upon whether you select it with one or more points placed. In each case, however, the reference point nearest the first point placed is removed from that object's list of reference points. If two or more points are placed, the remaining points will be added to the object's reference points in place of the one deleted. These two uses are shown in Figure 12.

The interactive form of this function moves or deletes points in the current set, which must have been previously defined. If the left button is clicked near a point in the current set, the point is deleted. Alternatively, a point can be moved by pressing the left button near the point and dragging it to its new location. In this case, the current set vanishes and feedback appears which is appropriate to the element containing the point. Note: if there are several elements in the current set, the feedback only includes the one containing the point being moved. This function is the *Modify* item on the pop-up menu in the picture subwindow.

5. Text and Line Styles

So far you have been drawing objects with the default styles set by GREMLIN when it starts: narrow lines, small text bottom-left-justified with a roman font, and always the same stipple pattern for polygons. In this section, you will learn to change the current styles and to modify the styles of existing objects.

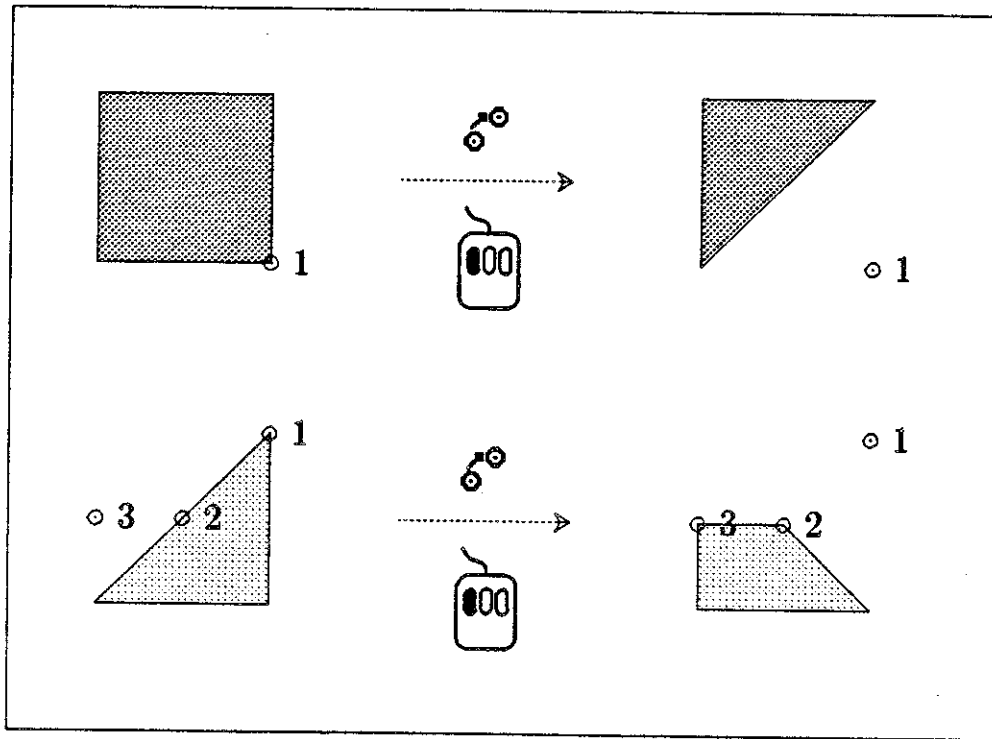


Figure 12: Moving a Reference Point

5.1. Setting Current Styles

Current styles are all set using the group of icons at the top of the menu subwindow. You can choose from among four font types, four font sizes, nine text justification modes, six line styles and eight stipple patterns. When selecting current styles, use the left mouse button. The middle mouse button changes the style of the current set, as described in Section 5.2.

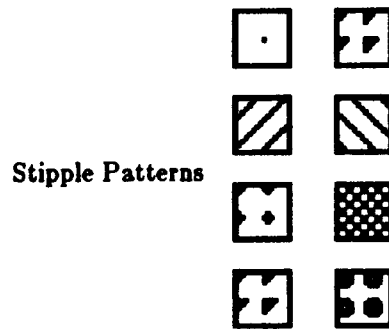
5.1.1. Line Style



Lines can be drawn in any of six line styles (also called *brushes*: narrow, medium, thick, dotted, dashed and broken. Select the current line style by pressing the left mouse button with the cursor located on the appropriate, highlighted icon. The current line style is shown with a box around the icon. Vectors, curves, circles, and polygon borders are always created with the current line style.

When using the interactive drawing commands, the feedback is always drawn using a narrow line style. The object is then redrawn with the proper line style when the final point is specified. Similarly, polygons are filled with the proper stipple only after the final point is given.

5.1.2. Stipple Fill Pattern

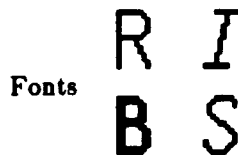


Three types of stipple fonts are currently available: the Magic font (*mg*), the Cifplot font (*cf*), and the Unigrafix font (*ug*). The default stipple font used by GREMLIN is the Cifplot font (*cf*). Try reading in *cf.g* in `/usr/local/lib/gremlin†`; you will see the 32 stipple patterns simultaneously. Note that the stipple patterns displayed on SUN workstations are just approximations to what would appear on your Imagen output.

The current stipple pattern is selected from among eight choices shown in the *stipple* icons just like selecting the current line style. Although there are only eight choices shown, you can actually use up to 256 stipple patterns in the picture window. You can redefine the mapping of any *stipple* icon as follows. Type the stipple pattern index of the stipple you want (which is shown in the *cf.g* display) into the text subwindow, then invoke a pop-up menu by clicking the middle mouse button on the *ETC* menu icon. The pop-up menu displays the eight stipple patterns currently used. Select the one you want to replace, and you'll see that the corresponding menu icon changes to display the new stipple.

If you want to use the Unigrafix or Magic stipple font, then set *stipplettype* to *ug* in your *.gremlinrc*, described in Section 10.2. Then read in *ug.g* in `/usr/local/lib/gremlin`, and you'll see the 64 stipple patterns of Unigrafix font displayed. Similarly, *mg.g* displays the Magic fonts. These fonts are also shown in Appendix B for easy reference.

5.1.3. Font Style



There are four fonts available in GREMLIN: roman, italics, bold and special. These are represented by the four icons shown above. Set the current font by pressing the left mouse button as above. Section 10.2 will explain how different fonts may be associated with these four font names (e.g., constant width for *special*). The selected font is actually just the starting font for the string. The string can contain CIFPLOT formatting sequences such as “\f” or “\d” to change the font or include other special characters. However, since GREMLIN is unaware of the meaning of these sequences they are displayed intact in the picture.

† Instructions on reading GREMLIN files is given in Section 7.3. Note that `/usr/local/lib/gremlin` is included in the default search path for your GREMLIN files, so you can just type the file name *cf.g*.

5.1.4. Font Size

Font Sizes

1	2
3	4

Four font sizes are available that by default correspond to point sizes 10, 14, 24 and 36 on hard copy. As with font types, these associations may be modified, but in general do not need to be (see Section 10.2).

5.1.5. Text Justification

JUSTIFY

Text justification is an aid to use in placing text in a picture. When you entered text as described in Section 2.3, the text always appeared above and to the right of the point which you placed to indicate its location. Text justification always refers to the relative position of the *point* (and not the text), therefore this justification mode is called "bottom left". This mode is fine for lining up text along the left side of a page, for example, but what if you wanted to right justify several lines of text? To do that, you would choose either bottom right, center right, or top right justification, depending upon the composition of the picture. Notice that when text is in the current set, a small dot will also be displayed indicating its justification. The dot will appear in the same relative positions as the dot displayed in the *text justification* icon.

The *text justification* icon (shown above) is a special purpose icon in the menu subwindow which is used somewhat differently from most. As you move the cursor across it, a small dot will move to one of nine positions representing the text justification modes: top left, top center, top right, center left, center center, center right, bottom left, bottom center and bottom right. To select a new justification mode, simply press the left mouse button after moving the dot to the position representing the desired mode. The dot will move to the new position indicating that the current justification mode has been changed.

5.2. Modifying Styles

The default settings described in the previous section only affect objects drawn after the style is selected; it has no affect on objects already in the picture. Sometimes though, you would like to change an object's style after it has been drawn. This section describes how to do that.

The most important point to remember is that you can not change an object's style unless it is in the current set. By the same token, when you make a change, it can affect all objects in the current set. With this in mind, you should be able to modify styles just as easily as setting the current styles.

Modifying styles is done in the same way as setting the current styles, with the exception that the **middle** mouse button is used, not the left button. For example, to make all lines in the current set dotted, select the *dotted* icon with the **middle** mouse button. The current set will be redrawn with dotted lines. When you change line styles, you will affect not only vector objects, but curves, arcs and polygon borders as well. See

the example in Figure 13. Stipple patterns are modified in the same way, however these will affect only polygon objects.

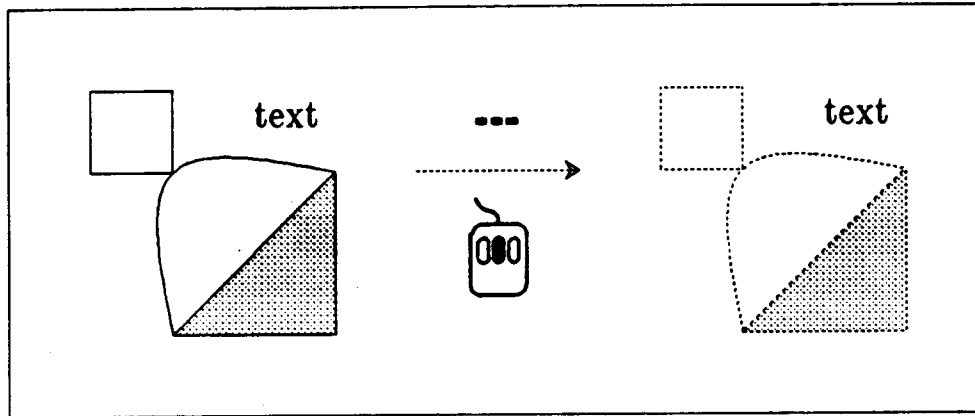


Figure 13: Modifying Line Style

5.3. Modifying Text

Text objects can be modified in four different ways: font, font size, justification, and the text string. To modify the font, font size, or justification, select the icon as if you were setting the current style, but press the middle button instead. To modify the text string, enter or edit the new string in the text subwindow and then select the *text* icon with the middle mouse button. Modifying text attributes is illustrated in Figure 14.

A new text string can be entered in any of several ways. First, it can be typed from scratch. Second, if there is no text displayed in the text window, you can use the left mouse button to recall text from the current set for editing. If there is text present, the left button moves the text cursor. Third, the middle mouse button will recall the last string typed. Both of these only work if the text input area is empty. Once the string is in the text subwindow it may be edited using the backspace, word erase, forward or backward control characters given in Section 2.1. Finally, replace the text in the picture by selecting the *text* icon with the middle mouse button. NOTE: If the current set contains several text strings, all of the strings will be replaced with the new text.

5.4. Modifying Object Types

In addition to modifying styles as described above, GREMLIN allows one group of objects to have their basic *type* changed: vectors, curves and polygons (bordered and unbordered). This facility allows you to change an object of one type into another simply by selecting the desired icon with the middle mouse button. Remember that the objects that you want to change must be in the current set, and that all objects in this group will be modified. See Figure 15 for some examples.

One special case of this facility can be used to change the type of a curve, for example, from a GREMLIN spline to a B-spline. When the *curve* icon is selected with the middle mouse button, the objects in the current set are redrawn using the current type of curve, as indicated in the *curve* icon. Therefore, if the object in the current set is a GREMLIN spline curve, and the current curve type is B-spline, the curve will become a B-spline.

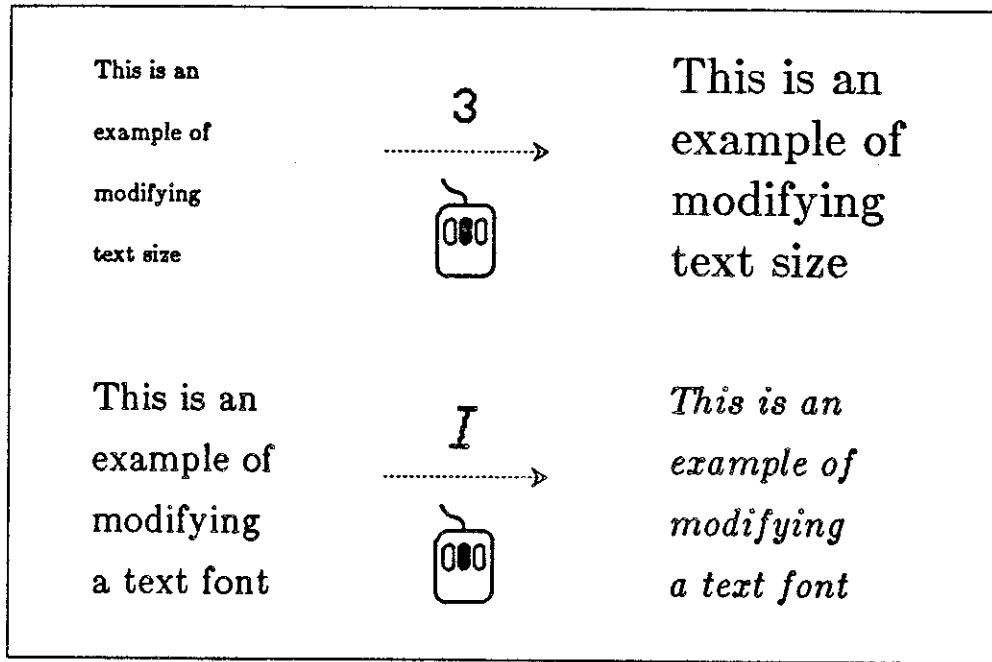


Figure 14: Modifying Text Attributes

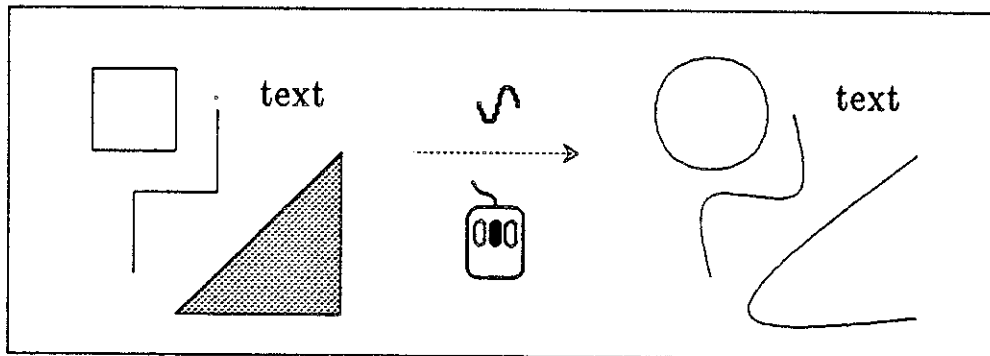



Figure 15: Modifying Object Type

6. Placing Points Where You Want Them

It can sometimes be difficult to place points *exactly* where you want them using the mouse. However, there are several aids in GREMLIN that will help you do this.

6.1. Grid Lines

Grid 

Selecting the *grid* icon will toggle the display of a grid in the picture subwindow. The grid is displayed on 32-pixel boundaries.

6.2. Alignment

Point Alignment 

Points are placed at multiple intervals of the alignment factor. This can be quite useful for lining up points both horizontally and vertically. For example, if you set the alignment to 32, all points will be placed at grid intersections. The alignment can be set from 1 to 512 in powers of two (i.e., 1, 2, 4, ..., 256, 512). You can move to the next higher alignment value by clicking the left mouse button when the alignment icon is highlighted; move to the next smaller value by clicking the middle mouse button.

6.3. Adjustment

Horizontal Adjust 

This command can be used to force all points to line up horizontally. Once the first point is placed in the picture subwindow, subsequent points will lie on the same horizontal line. Horizontal adjustment is toggled with the left mouse button; when ON, the icon is drawn with a box around it.

Vertical Adjust 

This command is similar to the preceding one except that points are forced to line up vertically. Vertical adjustment is toggled with the left mouse button; when ON, the icon is drawn with a box around it.

Manhattan Adjust 

This command combines horizontal and vertical adjustment. Points are adjusted to line up with the previous point either horizontally or vertically, depending upon the closest distance: horizontal or vertical. The best way to see how this works is to experiment. Again, the left mouse button is used to toggle this mode, and when ON, the icon is drawn with a box around it.

6.4. Gravity

Gravity 

This drawing aid will force points placed to *gravitate* to the nearest reference point. Gravity will only take effect, however, when the point is near enough to an object to be gravitated to it (about 32 pixels). As with the adjustment commands, gravity mode is toggled with each left mouse button click, and when ON, the icon is drawn with a box around it. NOTE: if multiple modifiers are in effect for point positioning, the precedence is as follows — gravity will override alignment and then adjustment will be applied.

Alignment, adjustment, and gravity, when in effect, apply to points placed non-interactively and to points positioned interactively. Interactive feedback will always show the effect of alignment and adjustment, so the feedback may not exactly follow the mouse

cursor. For example, if horizontal adjustment is in effect the feedback is constrained to a horizontal line, although the cursor is not. Gravity will affect the final position of a point, but is not visible unless the mouse is stopped. Therefore, to see the effect of gravity simply stop the mouse and the feedback will *gravitate* to its proper location.

Some forms of adjustment are illogical for certain interactive modes. For example, Horizontal and Vertical adjustment for polygons, and any adjustment for boxes. In these cases, the current adjustment is overridden while the particular interactive mode is in affect. The adjustment is reinstated when a different function is selected.

7. Writing and Reading GREMLIN Files

You now have the tools to create and modify pictures to achieve exactly the effect desired. The next step will be to incorporate these pictures into a document. To begin, you will need to understand something about the files created by GREMLIN and how to write and read them.

File Access Commands



All of GREMLIN's file-access commands are selected through the pop-up menu that is displayed when you press the left mouse button on the *file cabinet* icon (shown above). You will need to hold down the left button until you have made your selection. Those menu items are *Write*, *Save Set*, *Read*, *Edit*, *Path*, and *Directory*.

7.1. Writing GREMLIN Files

When you have completed your picture, you will want to save it in a file to be included in a document later. The format of GREMLIN files is described in Appendix A, but the details are not important to most users. To write the picture to a file, you will first need to enter the file name in the text subwindow. An optional point can be placed in the picture subwindow that will be used if the file is later read into another picture. Next, move the cursor to the *file cabinet* icon and invoke its pop-up menu with the left mouse button as described above. After selecting the *Write* menu item, the file will be written. For a file that already exists, you will be prompted before overwriting it. If the write fails, an appropriate message will be displayed.

7.2. Writing the Current Set

The other menu items in the file cabinet pop-up menu are used in a similar fashion by first entering the file name in the text subwindow and then selecting the item from the file cabinet icon's pop-up menu. The *Save Set* menu item is similar to the *Write* item, the difference being that *Save Set* will write only the *current set* to the specified file. Again, an optional point can be specified for positioning.

7.3. Reading a GREMLIN File

To read a GREMLIN file into a picture you must first enter the file name in the text subwindow. Next, select the *Read* menu item in the file cabinet icon's pop-up menu. The file read is added to the picture and becomes the current set. If a point is placed in the picture subwindow, the file will be read into the picture at that location, using the reference point stored with the file when it was written.

7.4. Editing GREMLIN Files

Similar to read, *Edit* causes the named file to be displayed for editing. If it exists, the file will be read into the picture subwindow, replacing any existing picture; if not, the message *creating new file* will be displayed. When you are editing a file, you can invoke the *Write* command without entering a file name in the text subwindow, and the picture will be written to the edit file. The edit file name is always displayed in the tool border. When this item is selected before saving the current picture, a prompt will be displayed before destroying the current picture.

7.5. The Path and Directory Mechanisms

GREMLIN's *path* facility allows you to specify certain directories that will be searched when *reading* or *editing* GREMLIN files. GREMLIN will examine the directories in the search path in order each time a file is read. If no directory in the path contains the file, one more attempt will be made to read the file from the GREMLIN library, */usr/local/lib/gremlin*. If the file name begins with a '~' or '/' then the path mechanism is bypassed and GREMLIN searches only in the home or root directory, respectively.

Paths are specified in the text subwindow by separating the directory names with colons. Paths may be specified using the '~' notation, and "::" is equivalent to ":", the current directory. When *Path* is selected with no string specified, the current path is displayed as a message in the text subwindow. The search path mechanism is used only for reading files. You might set the path with a string like this:

```
~/pictures/lib1:~/pictures/lib2::~
```

This will cause GREMLIN to look first in my picture libraries, then in my home directory, and finally in the current directory for any GREMLIN file read.


The *directory* command allows you to change the current working directory for reads and writes. This is useful when you are creating pictures in other than your home directory. Any path which does not begin with ".", "/", or "~" begins at the current directory. Therefore it is recommended that you always include "." in your path. With no argument, the *Directory* command displays the full path name of the current working directory.

8. Miscellaneous Functions

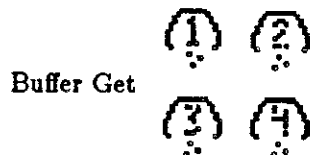
This section describes the rest of the function icons, and describes a short-hand method of invoking many of the functions.

8.1. Buffer Operations

GREMLIN has four internal buffers that can be used to save different parts of a picture. These buffers are preserved across edits so that their contents can be read into other pictures. They are most useful when several pictures share common objects because these objects can be saved in the buffers rather than creating temporary files for them.

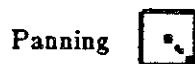
Buffer Put 

Only the current set can be saved in a GREMLIN buffer. As when writing GREMLIN files, you can (and probably should) place a point that will be used as a reference when reading the buffer back into the picture. You can save the current set in a buffer by selecting one of the *buffer put* icons with the left mouse button. After saving the current set in a buffer, a box will be drawn around the icon indicating that the buffer contains something.



Retrieving picture objects stored in a buffer is much like reading a GREMLIN file into a picture. You will first need to place a point indicating where the buffer contents should be placed in the picture. Next, simply select the *buffer get* icon for the appropriate buffer.

8.2. Panning



Panning allows you to edit a picture with dimensions larger than those of GREMLIN's picture subwindow. This command normally requires that one point be placed before selecting the *pan* icon. When selected with the left mouse button, the picture will be redrawn such that the point placed will be located at the center of the picture subwindow. An alternative use of the icon does not require that a point be placed. When the icon is selected with the middle mouse button, the picture will be redrawn such that its center is located at the center of the picture subwindow. This can be useful if you ever get "lost" panning around a picture.

Do not forget that you are not limited to the size of the picture subwindow displayed when GREMLIN starts. The *resize* menu item in the tool manager menu works the same as with other tools. The GREMLIN window can be moved and stretched to cover the entire display, if need be.

8.3. Symbolic Lines, Little Points



The *symbolic lines* command is used to speed the display of picture objects by drawing all lines in the "narrow" style, eliminating the expensive computation associated with the other line styles. Although this will increase performance, the interpretation of line styles in the picture will be left to your imagination. Selecting this icon with the left mouse button will toggle this display mode, and cause the picture to be redisplayed in the new mode. When symbolic lines are NOT being drawn, the icon will be highlighted with a box around it.

Little Point

The *littlepoint* command toggles the style of points displayed in the picture subwindow. By default, GREMLIN will display numbered points, drawn as a small circle with a dot in the center. This point style can be difficult to use for fine work. In this case, the alternative point style, a small unnumbered diamond, can be more effective in placing points. Selecting this icon with the left mouse button will toggle the point style displayed, and cause points in the picture subwindow to be redisplayed in the new style.

8.4. ETC Commands

Miscellaneous Commands

The final group of commands to be discussed are available through the *etc* icon pop-up menu. These five commands, *Clear Points*, *Show Points*, *Flash On/Off*, *Gripe*, and *Point*, are the least frequently-used in GREMLIN.

The *Clear Points* command is simply a shorthand way to erase all the points placed in the picture subwindow. The same effect could be achieved by repeatedly clicking the middle mouse button in the picture subwindow.

The *Show Points* command was mentioned in Section 4.1 where it was used to display a circle's reference points. When selected, this command will toggle the display of all reference points of objects in the current set. If left displayed, the reference points will automatically be erased the next time the current set changes.

The *Flash On/Off* option turns on or off the flashing of the current set. Flashing is normally on, but can be turned off with this option, or with a command in the *.gremlinrc* file, described in Section 10.2. If flashing is off, it can be turned on again with this option. The flash times (described in Section 10.2) are set to the defaults; On for 800ms, Off for 400ms.

In extreme cases, you may want to place points in the picture subwindow by specifying their absolute coordinates. The *Point* command allows you to do this, albeit in a roundabout manner. To place a point this way, you will need to enter the coordinates in the text subwindow first (x-coordinate, blank, y-coordinate), and then select the *Point* menu item. The point will be displayed in the picture subwindow. We have found this command useful when we needed to translate the current set by one pixel, in which case using the mouse is difficult at best.

GREMLIN's least frequently-used command is *Gripe*. However, if you should find that you need to comment upon the integrity of the software, select this menu item and follow the instructions.

The use of the middle mouse button on the icon to change stipple patterns has been described in Section 5.1.

8.5. Short Commands

The most common drawing commands and transformations can be invoked from the keyboard with single keystrokes when the cursor is in either the picture subwindow or the menu subwindow. Most of these commands are located under the left hand to allow the most efficient interaction with the editor. The short commands are usually faster than selecting the menu icon, and often faster than using the picture pop-up menu. You should experiment with their use. See Table 2 for a list of short commands.

The short commands are entirely equivalent to selecting the corresponding menu icon with the left button. This means that the short commands may be used to enter or leave an interactive mode, or to draw non-interactively if points have already been placed. Some short commands, such as *Make Iconic* have no menu equivalent.

The **SHIFT** key is a special short command used to define the current set. When the **SHIFT** key is pressed, GREMLIN enters the interactive Define Set mode; when it is released, GREMLIN returns to the previous mode. This makes it easy to select and move several objects, for example. This short command only works in the picture subwindow.

If you ever need to know the short command equivalent of a menu icon, click the **right** mouse button on the icon to display the help screen; the short command, if one exists, will be displayed next to the icon in the help screen. A full list of short commands can be seen by selecting the *Help* icon with the middle button.

Short Command	Menu Icon	Description	Short Command	Menu Icon	Description
1		Put Buffer 1	n		Non-Interactive
2		Put Buffer 2	q		Grid
3		Put Buffer 3	r		Rotate
4		Put Buffer 4	s		Scale (XY)
a		Draw Circle/Arc	t		Move
b		Draw Curve	u		Undo
c		Copy	v		Draw Vector
d		Define Set by Points	w		Draw Arrow
e		Erase	x		Draw Box
f		Define Current Set	z		Manhattan Adjust
g		Gravity	?		Help
i		Make iconic	^L		Redisplay
o		Expose Window	.		Repeat Last Cmd
h		Hide Window			
SHIFT		Define Set (current mode resumes when SHIFT released)			

Table 2: Short Commands

9. Printing GREMLIN Pictures

The final step in preparing a document that includes GREMLIN pictures is to include *GRN* commands in the paper where each picture is to occur. *GRN* is the DITROFF preprocessor that translates GREMLIN files into DITROFF commands. There is also a program called *GR2PS* which converts a GREMLIN file into PostScript for the Laserwriter. The manual entries for these programs are the best source of documentation on them. This section will describe only the new *GRN* commands specific to the SUN version of GREMLIN, and give some hints on producing the results you expect from *GRN*.

9.1. The *GRN* | *DITROFF* Pipe

The following *GRN* commands demonstrate the simplest way to include a GREMLIN picture in a document:

```
.
.
.
.GS
file gremlin_file_name
.GE
.
.
.
```

The *.GS* and *.GE* macros surround *GRN* commands. *GRN* recognizes these macros and replaces what they surround with *DITROFF* commands that produce the GREMLIN picture. The final step is to pipe this output through *DITROFF*. For example, to produce output for the Imagen for a document using the *-me* macros, you might execute this pipe:

```
% grn -Pip myfile.t | ditroff -me -Pip
```

It is also possible to preview your Imagen output on the SUN using the *DSUN* program. Since you generally would like to be able to print the output without having to run it through *DITROFF* a second time, use the following commands:

```
% grn -Pip myfile.t | ditroff -me -Pip -t > myfile.dit
% dsun myfile.dit
% lpr -Pip -n myfile.dit
```

The *-t* command to *ditroff* sends the *DITROFF* output to *stdout*. The *-n* command to *lpr* indicates that the input is from *DITROFF*. There is no restriction on GREMLIN file names, however, by convention they contain a "g" suffix. You can preview and print Varian output using the above command sequence with "Pip" replaced by "Pva".

9.2. Stipple Filled Polygons

If your picture includes polygons, you should specify the stipple type (*CIFPLOT*, *UNIGRAFIX*, or *MAGIC*) to *GRN* by including one of the following commands between the *.GS* and *.GE*:

```
st cf      (CIFPLOT)
st ug      (UNIGRAFIX)
st mg      (MAGIC)
```

The default stipple patterns displayed in GREMLIN are *CIFPLOT* stipples.

In some prior versions of GREMLIN only eight stipple patterns were permitted, so there was a mapping from the GREMLIN stipples to an index in the stipple font. If you have GREMLIN files created under versions prior to 2.0, then you should use the command:

```
oldstipplemap
```

between the *.GS* and *.GE* commands. In this case, you can also change the assignment of the GREMLIN stipples by including *l1—l8* commands. The stipple pattern is denoted by its index in the stipple font file (see Appendix B). The default associations, which you need not specify, are as follows:

```
l1 1
l2 3
l3 12
l4 14
l5 16
l6 19
l7 21
l8 23
```

NOTE: These commands are **only** required if you are using old GREMLIN files. The current GREMLIN file format uses absolute stipple numbers, as explained in Appendix A.

10. GREMLIN Initialization

This section tells you how you can customize the GREMLIN interface and initial environment in some ways.

10.1. Command Line Options

There are two command line options to GREMLIN: *-s* and *-o*. The *-s* option must be followed by a file name indicating a file of GREMLIN startup commands. These will be fully explained in the next section. Briefly though, a startup file can be used to set the default styles (stipple pattern, line thickness, etc.), drawing aids (alignment, gravity, etc.), and the rest of the editor environment. The default startup file is named *.gremlinrc* and must be located in either your home directory or the current working directory. The *-s* option lets you specify an alternate startup file not constrained by name or location.

The *-o* command line option indicates that GREMLIN files should be written in the *old* (AED) format. When GREMLIN was moved from the AED to the SUNs, the GREMLIN file format was modified to be more readable and to allow negative coordinates (see Appendix A). If you would like your files to be maintained in the old format, use the *-o* option (watch out if you use negative coordinates!).

GREMLIN also accepts generic tool arguments. You can specify things like the initial size and location of GREMLIN tool, your own version of the GREMLIN icon, the location of the GREMLIN icon, etc. For example, typing

```
gremlin -Ws 812 700 -Wp 64 64 -WI ghost.icon -WP 5 5 &
```

in a shell tool window will invoke your GREMLIN tool of size 812 × 700 pixels at position (64, 64). When you close the GREMLIN tool, its icon image will be read from file *ghost.icon*, and placed at position (5, 5). The other GREMLIN command line options can be mixed with the generic tool arguments in any order. For a list of the generic tool arguments, see *SUNTOOLS(1)*. Note that the GREMLIN tool header is reserved for displaying

the filename of the edited file, so the *-W* tool argument is ignored.

10.2. The *.gremlinrc* File

Before the GREMLIN window is displayed, the drawing environment is initialized from commands found in one or more *.gremlinrc* startup files. The search for *.gremlinrc* startup files begins in the home directory, followed by the current directory. Additionally, the *-s* command line argument can be used to indicate the path name of a third startup file (which need not be called *required for pictures maintained in the same directory*). In some cases then, GREMLIN will process three separate files on startup: the in the current directory and the user specified file.

The GREMLIN startup file contains lines consisting of either commands or comments. A sharp sign ('#') in the first column of a line introduces a comment that continues to the end of the line. Lines containing only the newline character are ignored. Each command requires an argument, separated from the command by white space. Upper and lower cases are significant when indicating commands, although only the leading characters that disambiguate the command need be entered. For example, *b dotted* is equivalent to *brush dotted* since no other startup command begins with the letter *b*. A table of all startup commands can be found on the next page.

Text and Stipple Font Directories

The directory path name where GREMLIN looks for font files to be used when displaying text is set with the *fontdir* command. All fonts must be in the *vfont* format. The font used for displaying stipples may be located in a directory other than that for the text fonts. This directory path name is set with the *stippledირ* command. There are currently three stipple types available — *ug* for UNIGRAFIX, *cf* for CIFPLOT, and *mg* for MAGIC. These are set with the *stipplettype* command. CIFPLOT fonts are used by default, so if you want to start with the UNIGRAFIX or MAGIC stipples use the command "stipplettype ug" or "stipplettype mg".

Font Types

The *R*, *I*, *B* and *S* commands are used to alter the font type associated with the roman, bold, italics and special icons in GREMLIN. For example, GREMLIN will display the constant width font in place of the roman font with the command *R CW*. The user must insure that the font files of all the font/size combinations indicated in the startup file actually exist in the text font directory.

Font Point Sizes

The *1*, *2*, *3* and *4* commands are used to alter the font sizes associated with the text size icons in GREMLIN. An integer argument indicates the point size of the font to be displayed. Again, the user must ensure that these files exist in the text font directory.

Stipple Patterns

Eight stipple patterns may be displayed in GREMLIN. The pattern associated with each stipple icon can be set with the *stipple1*, *stipple2*, ..., *stipple8* commands, or equivalently, *11*, *12*, ..., *18*. Each command requires an argument indicating the index in the stipple font file to be associated with that icon. Unigrafix fonts are indexed from 1 to 64, while cifplot and magic fonts are indexed from 1 to 32.

Current Brush

The *brush* command sets the current brush style upon starting GREMLIN. One argument among *narrow*, *medium*, *thick*, *dotted*, *dashed* or *broken* is required. The argument may be specified in either upper or lower case.

Current Font

The current font is set with the *font* command. One argument among *roman*, *bold*, *italics* or *special* is required. The argument may be specified in either upper or lower case.

Current Text Justification

The current text justification mode is set with the *justify* command. One argument among *tl*, *tc*, *tr*, *cl*, *cc*, *cr*, *bl*, *bc* or *br* (top left, top center, top right, center left, etc) is required. The argument may be specified in either upper or lower case.

Current Font Size

The current font size is set with the *size* command. This command requires an integer argument between 1 and 4.

Current Stipple Pattern

The current stipple pattern is set with the *stipple* command. This command requires an integer argument between 1 and 8.

Search Path Name

The search path of directories checked when reading a GREMLIN file is set with the *path* command. Directory path names must be separated by a colon.

Point Alignment

The *align* command is used to set the pixel alignment used during GREMLIN when placing points. An integer argument is required that must be a power of two between 1 and 512.

Point Adjustment

Points placed by the user may have certain adjustments applied to them based on gravitation to the nearest element or simply horizontal or vertical adjustment relative to the previous point. These commands require a string argument of either *on* or *off* (upper or lower case).

Gravity

The *gravity* command sets the adjustment mode where points are gravitated to the nearest reference point of an existing element.

Horizontal Adjustment

The *hadjust* command sets the horizontal adjustment mode. When *on*, points are placed at the same horizontal level as the previous point.

GREMLIN Startup Commands (SUN version)			
Command	Argument Type	Default	Purpose/Comments
1	#	7	point size of gremlin size 1 font
2	#	10	point size of gremlin size 2 font
3	#	14	point size of gremlin size 3 font
4	#	24	point size of gremlin size 4 font
R	string	R	font type of gremlin R font
I	string	I	font type of gremlin I font
B	string	B	font type of gremlin B font
S	string	S	font type of gremlin S font
stipple1	#	1	stipple 1 font index
stipple2	#	3	stipple 2 font index
stipple3	#	12	stipple 3 font index
stipple4	#	14	stipple 4 font index
stipple5	#	16	stipple 5 font index
stipple6	#	19	stipple 6 font index
stipple7	#	21	stipple 7 font index
stipple8	#	23	stipple 8 font index
brush	string	narrow	current brush style [narrow, medium, thick, dotted, dashed, broken]
font	string	roman	current font [roman, bold, italics or special]
justify	string	bl	current text justification [tl, tc, tr, cl, cc, cr, bl, bc,br]
size	#	1	current text size [1, 2, 3, 4]
stipple	#	1	current stipple pattern [1, 2, ..., 7, 8]
curve	string	inter	current curve type [inter, bspline, bezier]
fontdir	string	/usr/lib/font/devsun/	font directory path name
stippledირ	string	/usr/lib/font/devsun/	stipple font directory
stipplettype	string	cf	type of stipple [ug or cf]
path	string	::	file search path
align	#	4	alignment of user points
flashoff	#	400	current set off period (ms)
flashon	#	800	current set on period (ms)
gravity	string	off	turn gravity mode on/off
grid	string	off	turn grid display on/off
hadjust	string	off	turn horizontal adjustment on/off
madjust	string	off	turn manhattan adjustment on/off
vadjust	string	off	turn vertical adjustment on/off
littlepoint	string	off	turn point style on/off
symboliclines	string	off	turn symbolic line mode on/off

Vertical Adjustment

The *vadjust* command sets the vertical adjustment mode. Analogous to *hadjust* except applied vertically.

Manhattan Adjustment

The *madjust* command sets the manhattan adjustment mode. In this mode, points may be adjusted either horizontally or vertically depending upon the proximity of the previous point.

Current Set Flashing Period

The *flashon* and *flashoff* commands set the flashing period for the current set. Both commands require an argument (in ms) indicating either the time on or off for the current set. Flashing can be turned off by typing "flashoff 0". No other times smaller than 200ms can be specified, since it takes excessive CPU resources to flash any faster than that.

Grid

The *grid* command sets the display of the grid. One argument is required (*on* or *off*).

Point Style

The *littlepoint* command sets the style of the points displayed. One argument is required (*on* or *off*). When off, points are numbered and indicated by a small circle. When on, points are represented by a small diamond.

Symbolic Lines

The *symbolicl原因* command sets the drawing mode of lines. One argument is required (*on* or *off*). When off, lines are drawn with their true style and thickness. When on, all lines are drawn with the *narrow* style.

Appendix A — Gremlin File Format

This appendix was lifted from a description of the AED GREMLIN file format — differences in the SUN version are noted where appropriate.

There now exist two distinct GREMLIN file formats, the original format from the AED version, and a new format from the SUN version. An extension to the SUN version allowing reference points with negative coordinates is NOT compatible with the AED version. As long as a GREMLIN file does not contain negative coordinates, either format will be read correctly by either version of GREMLIN. The other change to the SUN format is the use of names for picture objects (e.g., POLYGON, CURVE) instead of numbers. Files representing the same picture are shown below in each format.

0	240.00 128.00	0	240.00 128.00
CENTCENT		2	
240.00 128.00		240.00 128.00	
185.00 120.00		185.00 120.00	
240.00 120.00		240.00 120.00	
296.00 120.00		296.00 120.00	
*		-1.00 -1.00	
2 3		2 3	
10 A Triangle		10 A Triangle	
POLYGON		6	
224.00 416.00		224.00 416.00	
96.00 160.00		96.00 160.00	
384.00 160.00		384.00 160.00	
*		-1.00 -1.00	
5 1		5 1	
0		0	
-1		-1	

- The first line of each GREMLIN file contains either the string "gremlinfile" (AED version) or "sungremlinfile" (SUN version).
- The second line of the file contains an orientation, and x and y values for a positioning point, separated by spaces. The orientation, either '0' or '1', is ignored by the SUN version. 0 means that Gremlin will display things in horizontal format (drawing area wider than it is tall, with menu across top). 1 means that Gremlin will display things in vertical format (drawing area taller than it is wide, with menu on left side). X and y are floating point values giving a positioning point to be used when this file is read into another file. The stuff on this line really isn't all that important; We suggest using "1 0.00 0.00".
- The rest of the file consists of zero or more element specifications. After the last element specification is a line containing the string "-1".

Element Specifications

- The first line of each element contains a single decimal number giving the type of the element (AED version) or its ASCII name (SUN version). See Table 3. Notice that curves other than GREMLIN splines are given by a two-word name.

GREMLIN File Format — Object Type Specification		
AED Number	SUN Name	Description
0	BOTLEFT	bottom-left-justified text
1	BOTRIGHT	bottom-right-justified text
2	CENTCENT	center-justified text
3	VECTOR	vector
4	ARC	arc
5	CURVE	GREMLIN curve
6	POLYGON	polygon
7	CURVE BSPLINE	B-spline curve
8	CURVE BEZIER	Bezier curve
10	TOPLEFT	top-left-justified text
11	TOPCENT	top-center-justified text
12	TOPRIGHT	top-right-justified text
13	CENTLEFT	left-center-justified text
14	CENTRIGHT	right-center-justified text
15	BOTCENT	bottom-center-justified text

Table 3: Type Specification in GREMLIN Files

- After the object type comes a variable number of lines, each specifying a reference point used to display the element. Each line contains an x-coordinate and a y-coordinate in floating point format, separated by spaces. The list of points is terminated by a line containing the string "-1.0 -1.0" (AED version) or a single asterisk, "*" (SUN version).
- The number of points varies with the type of object, since the points are the reference points of the object. For VECTOR, CURVE, and POLYGON objects, these are simply the vertices of the figure. For ARC objects the first and second points are the center and radius-defining points, respectively. If the arc is a circle, four additional points designate the 12, 6, 3, and 9 o'clock points on the circle. If the arc is incomplete, the third point is the opposite endpoint of the arc, and there are no other positioning points. For all types of text, the first point is the reference location point, and three other points specify the bottom left, bottom center, and bottom right of the text string in the picture. Since these are complicated to compute, they can be omitted and will be regenerated by GREMLIN when it reads the file.
- After the points comes a line containing two decimal values, giving the brush and size for the element. The brush determines the style in which things are drawn. For vectors, arcs, and curves there are six legal brush values:
 - 1 - thin dotted lines
 - 2 - thin dot-dashed lines
 - 3 - thick solid lines
 - 4 - thin dashed lines
 - 5 - thin solid lines
 - 6 - medium solid lines

For polygons one more value, 0, is legal. It specifies a polygon with an invisible border. For text, the brush selects a font as follows:

- 1 - roman (R font in troff)
- 2 - italics (I font in troff)
- 3 - bold (B font in troff)
- 4 - special (S font in troff?)

If you're using GRN to run your pictures through DITROFF, the font is really just a starting font: the text string can contain formatting sequences like "\fI" or "\d" which may change the font (as well as do many other things).

For text, the size field is a decimal value between 1 and 4. It selects the size of the font in which the text will be drawn. For polygons, this size field is interpreted as a stipple number with which to fill the polygon. The number is an index into a stipple font file. NOTE: GREMLIN files produced prior to version 2.0 contain a stipple index between 1 and 8, which is mapped to an index in the stipple font file at print time, using the *l1—l8* commands.

- The last line of each element contains a decimal number and a string of characters, separated by a single space. The number is a count of the number of characters in the string. This information is only used for text elements, and contains the text string. There can be spaces inside the text. For arcs, curves, and vectors, this line of the element contains the string "0 ".

Notes on Coordinates

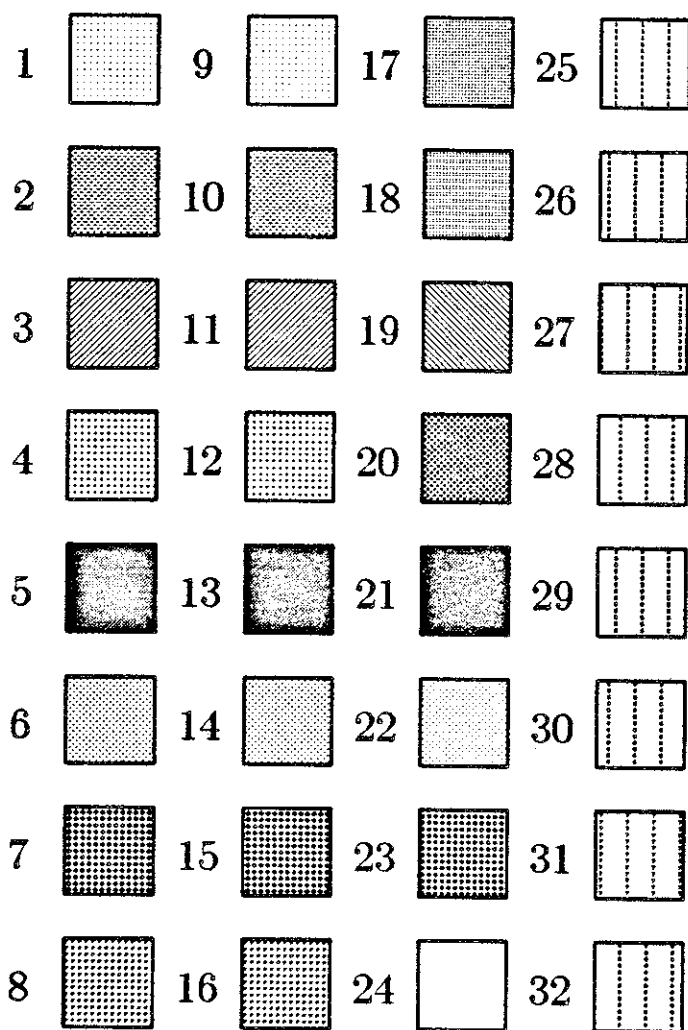
Gremlin was designed for AEDs, and its coordinates reflect the AED coordinate space. For vertical pictures, x-values range 116 to 511, and y-values range from 0 to 483. For horizontal pictures, x-values range from 0 to 511 and y-values range from 0 to 367. Although you needn't absolutely stick to this range, you'll get best results if you at least stay in this vicinity. This is because some of the plotting programs do scaling on the assumption that the range above is to just fit on an 8.5 x 11-inch piece of paper. Also, point lists are terminated by a point of (-1, -1), so you shouldn't ever use negative coordinates. Gremlin writes out coordinates using format "%f1.2"; it's probably a good idea to use the same format if you want to be absolutely safe.

Notes on SUN Coordinates

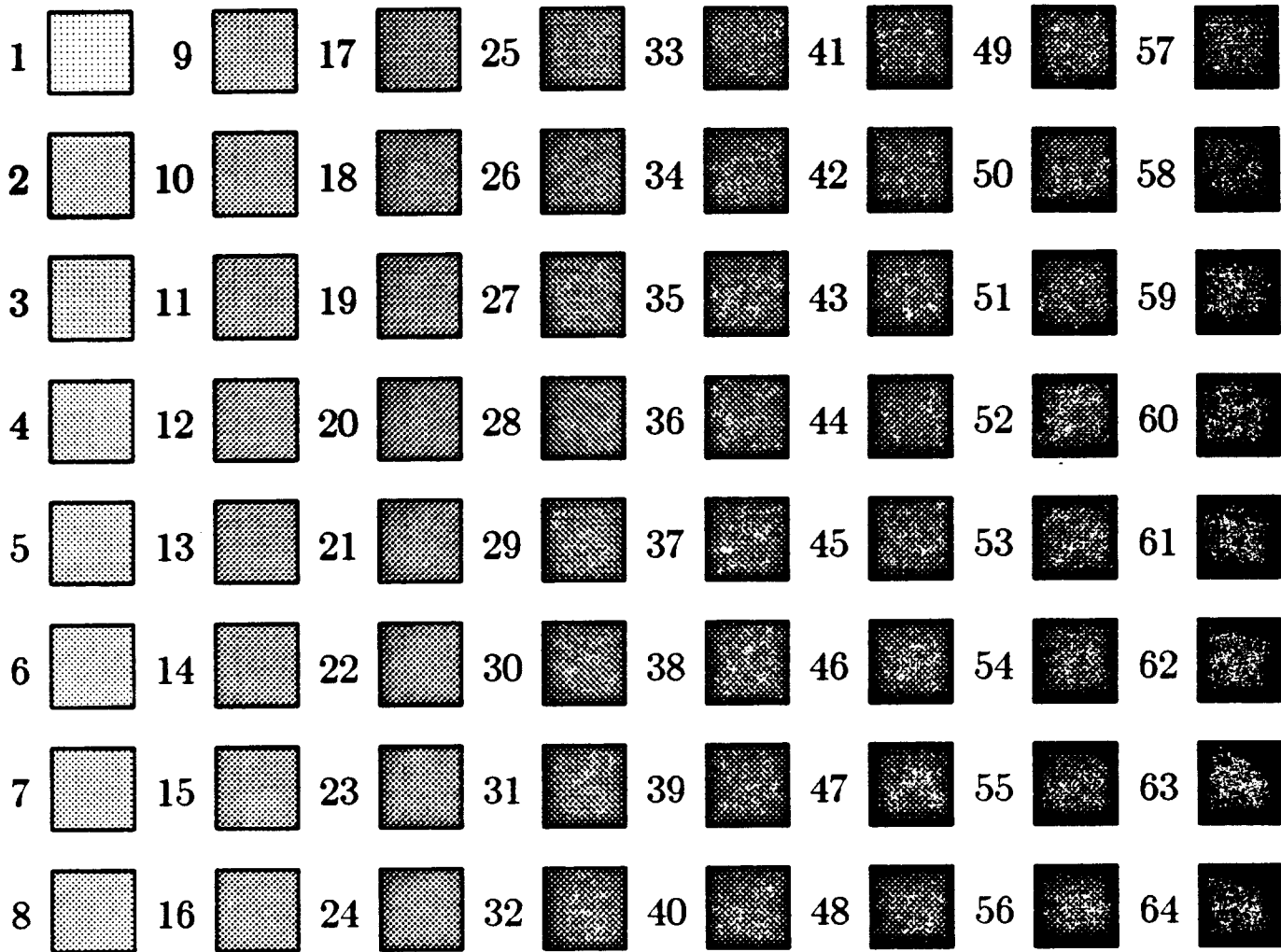
There is no longer a restriction on the range of coordinates used to create objects in the SUN version of GREMLIN. However, files with negative coordinates WILL cause problems if displayed on the AED.

Appendix B — CIFPLOT, UNIGRAFIX and MAGIC Stipple Patterns

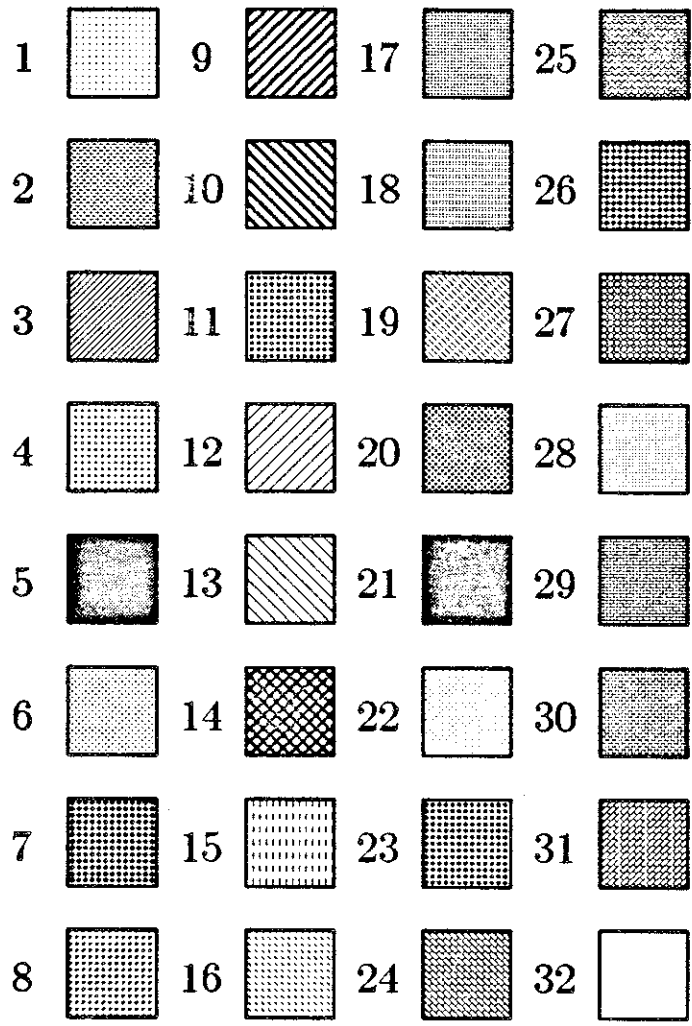
Cifplot Stipples



Unigrafix Stipples



Magic Stipples



Appendix C — Examples of Curve Styles

Points	Interpolated Curves	B-Spline	Bezier Curves
<ul style="list-style-type: none"> • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 2 • 3 • 1 			
<ul style="list-style-type: none"> • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			
<ul style="list-style-type: none"> • 4 • 3 • 2 • 1 			