SEGINI

With the Model One/10 and Model One/80 Display List Firmware, the parameter of this command specifies the number of bytes per data block. In addition, this parameter is a 16-bit quantity. With the Display List Firmware of the Model One/25, Model One/40, and Model One/60, however, this command's parameter specifies the number of words and it is an 8-bit quantity.

SEGINQ

In addition to returning segment attributes, this command with the new firmware also indicates whether or not the segment exists.

SEGPID

SEGPID in the earlier product is called "PICKID" in the new Display List Firmware, having the same opcode. The PICKID parameter is a 32-bit integer instead of the 16-bit integer with SEGPID.

SEGREF

With the Model One/10 and Model One/80 Display List Firmware, this enhanced command is not used exclusively to nest segment executions as is the case with the previous Display List Firmware. Instead, the updated version always invokes the execution of the specified segment. In this manner, a segment tree may be executed (drawn, picked, highlighted) by merely SEGREFing the top segment in the tree while the display controller is in the appropriate execution mode (NORMAL, PICK, (UN)HILITE). In the newer version, the argument to this command is a 32-bit integer; in the earlier version it is a 16-bit integer.

SEGREN

This command is unchanged except that the segment numbers are 32-bit integers in the Model One/10 and Model One/80 firmware.

SET

SET in the Model One/25, Model One/40, Model One/60 Display List Firmware is called SETGL on the Model One/10 and Model One/80, having the same opcode. With both products, the global parameter is the same, although some new specifiers are available with SETGL.

In addition, segment IDs and pick IDs are 32-bit quantities with SETGL instead of SET's 16-bit quantities. Another difference is that the pick aperture has independent X and Y specifiers with the new SETGL command.

SETATR

This command is unchanged except that the segment number is a 32-bit integer in the Model One/10 and Model One/80 firmware.

SYSTAT

The Model One/10, Model One/80 version of this command has two added parameters. With this enhanced version, you can specify for read back:

- The number of free memory blocks available.
- An array of all defined segments.
- The number of memory blocks used by a particular segment.

The remaining portion of this appendix lists the new Display List Firmware commands not yet discussed.

IGNORE

This is a new command for the Model One/10 and Model One/80 Display List Firmware Product. It is for use with host-based display lists only. With flag ON, the display controller ignores incoming commands.

INCPID

This is a new command for the Model One/10 and Model One/80 Display List Firmware Product. It increments the current pick ID by 1.

POP

This is a new command for the Model One/10 and Model One/80 Display List Firmware Product. It restores a PUSHed state or item by popping it off the current stack.

PUSH

This is a new command for the Model One/10 and Model One/80 Display List Firmware Product. It saves a specified state or item by placing it on top of the current stack.

Raster Technologies

MODEL ONE/80

Shading and Depth-Buffering Reference, Rev. 2.0

July 29, 1985

Raster Technologies, Inc.
9 Executive Park Drive
North Billerica, MA 01862
(617) 667-8900

RASTER TECHNOLOGIES
MODEL ONE

NOTICE:

The information contained in this document is subject to change without notice.

RASTER TECHNOLOGIES DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS MATERIAL (INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE), EITHER EXPRESS OR IMPLIED. RASTER TECHNOLOGIES SHALL NOT BE LIABLE FOR DAMAGES RESULTING FROM ANY ERROR CONTAINED HEREIN, INCLUDING, BUT NOT LIMITED TO, FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS MATERIAL.

WARNING: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual may cause interference with to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Table of Contents

1.0 Introduction

The Model One/80-S Enhanced Rendering option provides solid modeling capabilities as an expansion of the Model One/80 which is used as a hardware base. In addition to the full complement of features of the standard Model One/80, the Model One/80-S contains additional microcode and firmware to allow local generation of smooth shaded polygons and to perform local hidden-surface removal.

A unique feature of the Model One/80's Enhanced Rendering option is the ability to use the same physical image memory in either a high resolution (1280 x 1024) 8 bit mode or in a medium (640 x 512) resolution mode with shading and hidden-surface removal. This modal switch is under software control and allows the same system to serve in a variety of application environments without modification.

With most CAD systems, shading and hidden-surface processing is done on the host, with the resulting image being sent to the display controller in pixel or scan-line format. This process typically takes many minutes on a VAX 11/780 class minicomputer.

In contrast, the Model One/80 enables the host to send entire polygonal patches, locally performs the shading interpolation, and performs the hidden-surface removal -- all using a 16 bit depth-buffer. The resulting image is displayed as either a full color 24 bit true color image or as a patterned dithered 8 bit image yielding the visible effect of approximately 12 bits. Display mode selection is specified by the user based on the physical memory configuration in the Model One/80 and the format of the incoming data. This local processing allows construction of complex images in seconds rather than minutes and dramatically reduces the host computational and communications overhead.

By allowing the use of a full 24 bits color model, the Model One/80 provides almost unlimited use of color, shading, and specular reflection with the Gouraud algorithm. The automatic dithering algorithm allows the same application code to run unaltered on either an 8 bit, 24 bit, or 40 bit Model One/80 configuration. In addition, the Model One/80's use of a 16 bit depth-buffer supports a depth range of +32767 to -32768, allowing superior resolution of depth features.

2.0 Technical Overview

The Model One/80-S performs two of the major functions associated with generating realistic three-dimensional images -- shading and hidden-surface removal. These functions are extremely time-consuming to do on a host and porting them to the local graphics system results in both higher performance and lower host overhead. These operations are applied to polygonal patches sent down from the host with x,y,z, and r,g,b information for each vertex of the polygon. In contrast, host-based hidden-surface removal and shading systems usually send down scan-line data on a pixel-by-pixel basis, consuming both host computational resources and communications bandwidth.

2.1 Shading

The process of generating a polygon on the screen with smoothly varying color is referred to as shading. Because the colors are sent to the Model One/80-S for only the vertices of a polygon, the colors for the intermediate pixels are interpolated to fill that polygon. The colors are interpolated not only along a scan line but also from one scan line to another. This process of interpolating both along scan lines (x direction) and between them (y direction) is known as bilinear interpolated shading. This type of shading is also called Gouraud shading, named after the French computer scientist who first described the method.

The Model One/80-S provides an additional method of shading -- constant, or grenade, shading. This is simply drawing the polygons in a constant color, for speed in previewing.

2.2 Hidden-Surface Removal

Hidden-surface removal is the process of drawing only the visible surfaces of a "solid" object. This is typically a time-consuming operation on the host; offloading it to the local graphics system improves total system performance even more than does the shading. The hidden-surface removal method used on the Model One/80-S is the depth-buffer algorithm, or depth-buffering.

Depth-buffering works as follows: Consider the visible screen of the Model One/80 as a long, narrow rectangular box that is 640 x 512 pixels in the x and y dimensions respectively, with a depth, or z dimension, extending back into the screen. This z dimension is up to 65,535 units. Assume that you are looking into this long narrow box from an infinite distance, so that there is no perspective; things at the back ("farthest" from the screen) of the box look as large as things in the front ("nearest" to the screen). Imagine a memory area, separate from the image memory, which contains the depth, or z value, for each pixel on the screen. With these basic assumptions, do the following:

1. First, set the z value of all pixels to the maximum value. The depth memory now "thinks" that there is a solid surface across the farthest end of the visible box. Since the image memory has nothing in it, the screen shows nothing.

2. Next, write a polygon into the image memory. Each pixel that makes up the polygon depth is compared with the value already in the depth buffer for that location. If the new pixel's value is less than the existing value, it is written into image memory and its z value is substituted for the old pixel's z value in the depth-buffer. Because all locations of this buffer have been preset to the maximum value, all pixels of the polygon are written.

Z values for pixels other than at the vertices of the polygon are determined by bilinear interpolation of the vertex values. In fact, the algorithm is almost identical to that used for color interpolation.

3. Now draw a second polygon. As the z values of its pixels are calculated, each is compared with the value in that corresponding x,y location in image memory. If a z value is greater, the new location is "farther away" than the existing location. Because it is "farther" from the screen than the existing pixel at that x,y coordinate, it should be hidden, and is therefore not written. If, on the other hand, the z value is less than the existing pixel, it is "closer" to the screen and should hide the previous pixel. Therefore, it is written and its z value replaces the previous value.

The effect of this process is that you see on the screen only the visible surfaces of the polygons that have been sent, with the hidden surfaces properly removed. When this process is combined with smooth shading, you can display realistic representations of three-dimensional objects with minimal host involvement after the polygonal data base has been constructed.

In addition to the elimination of host overhead for depth-buffering and scan-line conversion, the local depth-buffer technique offers a very important advantage for interactive applications -- the surfaces need not be in any predetermined order. This fact again saves host time by eliminating a time-consuming sort phase and allows objects on the screen to be incrementally constructed. Incremental viewing of an object is not possible with scan-line-based algorithms which must paint in scan line order.

3.0 Model One/80-S Command Set

Below is a description of the Model One/80-S commands used for shading and depth-buffering. These commands are divided into three functional groups:

1. Depth-buffer commands: ZCLEAR, ZPATCH, ZFUNCT, ZRANGE, ZGRID, and ZCLIP.

2. Shading commands: SHMODE

3. Configuration commands: FOLD

It should be noted that these functions operate independently. You can, for example, draw shaded polygons without depth-buffering. Setting the shading mode to constant makes possible depth-buffering without interpolated shading. This allows you a great deal of flexibility in using the Model One/80.

4.0 Differences Between the Model One/25-S and the Model One/80-S Command Set

The Model One/80-S command set is designed to be completely compatible with the Model One/25-S command set except in those details enumerated in this section. The only differences are the absence of anti-aliasing on the Model One/80-S; the addition of the depth buffer configuration command FOLD; and additional value formats for ZPATCH, dithering, translucency, hidden-line mode, and edge-highlight mode. The Model One/80-S also offers additional parameters for the SHMODE, ZPATCH, and ZGRID commands.

The Model One/80-S supports value (color) formats and dithering, which are not available on the Model One/25-S. The Model One/80-S allows an additional color value format which is 16-bit. Also, there are three Model One/80-S configurations, as opposed to the single One/25-S configuration.

5.0 Available Model One/80-S Configurations

The three Model One/80 available configurations are:

- 8-bit, folded configuration, consisting of one GPU and one image-memory card. (Optional extra depth memory card for displaying one 8-bit, unfolded image using four folded pieces.)

- 24-bit, unfolded configuration consisting of one GPU, three image memory cards, and video-adding circuitry to allow either a 24-bit, no depth buffer or an 8-bit, depth buffer -- both software-selectable.

- 40-bit, unfolded configuration consisting of one GPU, three image memory cards, and two depth memory cards.

The manner in which the input color values are applied to the image memory is done so that the maximum information possible is displayed. The table on the following page shows the way that is done.

Table 5.1 Methods of Applying Input Values to Image Memory

Value Format	Memory Format		
	8-bit folded	24-bit unfolded	40-bit unfolded
24-bit values	True-color dithering. Depth is in image memory.	True-color dithering Depth is in depth memory.	Direct true color. Depth is in depth memory.
8-bit values	Direct pseudo-color. Depth is in image memory.	Direct pseudo-color. Depth is in depth memory.	Replicated pseudo-color. Depth is in depth memory.
16-bit values	Pseudo-color dithering. Depth is in image memory.	Pseudo-color dithering. Depth is in depth memory.	Replicated pseudo-color dithering. Depth is in depth memory.

Translucency is supported as a texturing operation, working through the use of the area pattern register. This allows for a variety of "translucent" surfaces in the sense that one can then see other surfaces showing through. The area pattern register is set to the pattern of translucency which is desired and the shading commands are then sent. The parts of the pattern which are to be drawn in foreground value are drawn in the current shaded color, while parts which would be drawn in the background value during a normal pattern fill operation are inhibited in the depth buffer operation.

Hidden-line removal consists of drawing surfaces in a background color and highlighting the edges (lines). Edge highlighting, supported through the SHMODE command, consists of drawing the edges highlighted and the rest of the surface normally.

Note on Use of True-Color Dithering

The true-color dithering process assumes that the look-up tables are set in a certain way. Each bit in the image memory value maps directly to a part of the color. The three high-order (MS) bits always indicate the red component, the next three bits always indicate the green component, and the two low bits (LS) indicate the blue component. For example, a value of "255" is full red, full green, and full blue; a value of "0" is no red, no green , and no blue; a value of "3" is full blue, no red, and no green; and a value of "28" is full green, no red, and no blue. A simple FORTRAN program can be written to set the look-up tables according to the bits of the index, as described above.