UDC 621.397: 681.3

3D-CG System with Video Texturing for Personal Computers

●Toru Ozaki ●Tatsushi Otsuka ●Seiya Shimizu

(Manuscript received June 2,1997)

Three-dimensional computer graphics (3D-CG) is attracting attention as an Internet multimedia technology. We have developed a 3D-CG system with video texturing for personal computers.
First we developed a high-performance rendering LSI chip to reduce the size and cost of the system. This chip can handle 400 k polygons/s and 15 M texture cells/s. We then developed a personal computer accelerator board that incorporates the LSI rendering chip. In addition, we extended certain functions to enable video images to be entered as texture data and confirmed the effectiveness of the system through application trials.

1. Introduction

Three-dimensional computer graphics (3D-CG) is attracting much public attention as an Internet multimedia technology for movies, games, virtual malls, and other types of entertainment. In 1995, the advent of high-speed 3D-CG enabled middle-range UNIX workstations to process high-end 3D-CG. By 1996, the combination of Windows NT machines and OpenGL[1] cards could also run 3D-CG applications. These technological changes established a personal computer environment in a 3D-CG market previously monopolized by workstations. Since the end of 1996, the availability of OpenGL has been extended to Windows 95. There are now many graphics libraries, for example, PEX, HEIDI, Direct3D, and OpenGL. OpenGL has become available for most OSs run on equipment ranging from workstations to personal computers, and has become established as the de-facto standard 3D-CG library. Also, the 3D-CG description methods for the Internet have been standardized into the virtual reality modeling language (VRML).

Because of the above developments, the 3D-CG system market will not be limited to the CAD/CAM field but will also expand into the field of image generation. However, 3D-CG systems must be improved in terms of cost and performance. Many companies are developing 3D-CG products. For example, Sun Microsystems has developed extended instructions for 3D-CG that are implemented using RISC processors,[2] and Mitsubishi Electric has developed 3D RAM,[3] which are special integrated functions for 3D-CG. These new products are effective for increasing performance but they do not reduce the cost of 3D-CG systems and they are not applicable to personal computer systems.

This paper describes a 3D-CG system that can achieve both high-performance and low-cost.

2. Rendering chip

In general, 3D-CG systems used on workstations and personal computers are identical. As **Fig.1** shows, a 3D-CG system consists of an API layer, a geometry layer for coordinate conversion and lighting, and a rendering layer for shading and texture mapping. The rendering layer, which is the final stage, needs to be accelerated to increase the hardware processing speed.

Ray-tracing and radiosity are high-grade shading technologies; however, because they involve an enormous number of calculations, it is not possible to produce a rendering LSI for personal computers using current technology. Therefore, we decided to use Gouraud shading and tex-

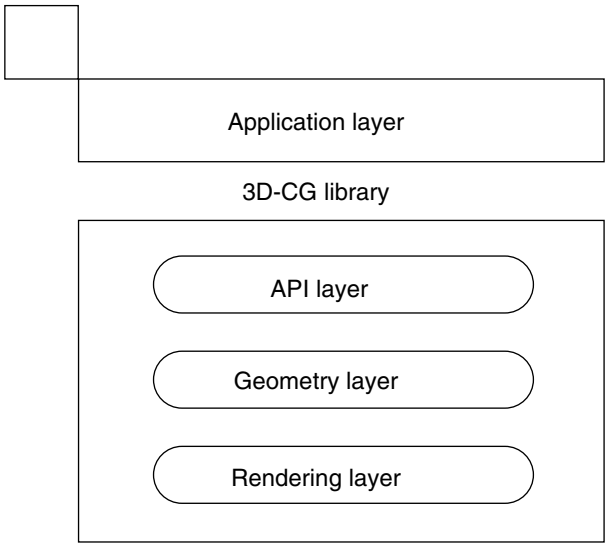

Fig.1– Hierarchical structure of 3D-CG system.

ture mapping instead. Gouraud shading provides satisfactory shading with fewer calculations. Texture mapping allows images generated by ray-tracing and other high-grade shading methods to be cut and pasted.

To develop an LSI rendering chip for a personal-computer 3D-CG system, we had to meet the following three technical requirements:

1) The LSI chip must be flexible enough to recognize new library functions and support libraries other than OpenGL.
2) The basic level of performance should be higher than that of a dedicated hardware device.
3) To reduce system costs, all the circuits should be integrated into a single chip.

To meet the first requirement, a microprocessor was adopted that extends the number of functions and provides program flexibility. However, if we use only a microprocessor it will be difficult to meet the second requirement because microprocessor performance generally lags behind the performance of dedicated hardware. 3D-CG processing requires not only color values (RGB) but also a depth value (Z), blending parameter (α), and texture values (RGB and α) for each pixel. These values are stored in memory linked to the LSI. Therefore, memory access rather than processing operations may cause a bottleneck. One way to avoid this problem is to add an ASIC, but this would make it impossible to meet our third requirement. Therefore, we designed a device architecture that combined a microprocessor core with a CG pipeline.

2.1 Outline of the MB86271

Figure 2 shows the architecture of our MB86271 LSI rendering chip.[4] The MB86271 consists of a microprocessor core and a CG pipeline. The microprocessor core operates on microcode stored in local memory to provide flexibility and function extendibility. The CG pipeline consists of a digital differential analyzer (DDA), a drawing control block, and three memory controllers. The DDA calculates the position data, color data, and texture coordinates needed to draw each pixel. The drawing control block reads texture data from memory and blends texture values with the color values according to their coordinates. To determine depth, the drawing control block also reads the Z (depth) values from local memory. The connected memory is divided into three blocks to prevent access-contentions from lowering the performance. To increase the processing speed, we used SDRAM devices for local and texture memory and a VRAM device for frame memory, which needs drawing and display ports. However, since VRAM is relatively slow, we created a configura-

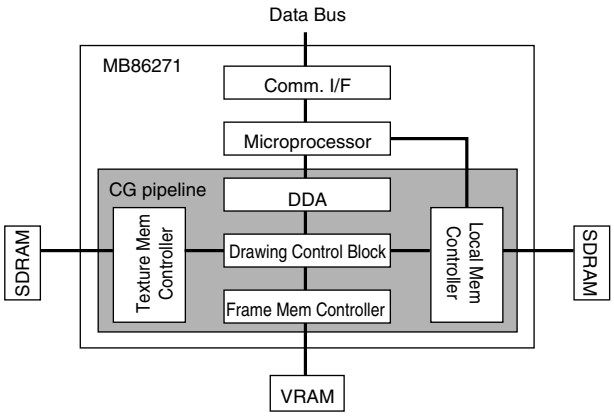

Fig.2– Architecture of MB86271.

tion that requires only write accesses for drawing. The RGB values in local memory are used as display data. The required RGB values in local memory are read, combined with α, and re-written. The MB86271 writes data to the frame memory but does not read from it.

2.2 Performance of the MB86271

The MB86271 is manufactured using 0.5 μm CMOS technology and is packaged in a 304-pin ceramic QFP. Its power consumption at 50 MHz is 4 W.

This LSI can perform all the rendering functions of the OpenGL Library. The performance for Gouraud shading is 400 k polygons/s at 25 pixels, and the performance for texture mapping for a 200-pixel Gouraud polygon is 15 M texture cells/s.

3. 3D-CG system

We built an experimental 3D-CG system for personal computers based on a 3D-CG accelerator board containing the MB86271 rendering LSI. When creating the system, we tried to fully exploit the features of the MB86271.

If we use an accelerator board to do a large part of the geometric processing, the system can achieve a higher geometric processing speed, but it would need large-scale hardware. Therefore, we decided to apply the accelerator board only to part of the geometric processing. To process a triangular polygon, the MB86271 requires three color values, three slope values, and vertex information — which amounts to 128 bytes. If all this data is transferred through a PCI bus, a bottleneck may occur. To prevent this, we included a slope value calculation section in the PCI board. This eases the load on the CPU and reduces the data transfer rate through the PCI bus to 40 bytes/polygon.

3.1 System configuration

3.1.1 Hardware configuration

We installed the 3D-CG accelerator board in an FM V6200, which is a Fujitsu personal computer with a Pentium Pro CPU. **Figure 3** shows the hardware configuration of the accelerator board. Packet data for drawing and control data for the MB86235, MB86271, and RAM DAC are transferred through the PCI bus. The control data is received by the PCI controller, interpreted by the FPGA data control circuit, and then sent to the MB86235, MB86271, and RAM DAC. The PCI controller also receives the packet data and then sends it to the MB86235 for slope calculations. The MB86235 is a Fujitsu DSP that performs geometric calculations at 80 MFLOPS and sends the resulting data to the MB86271 for rendering. The MB86271 accesses Memory 1 (SDRAM) for the

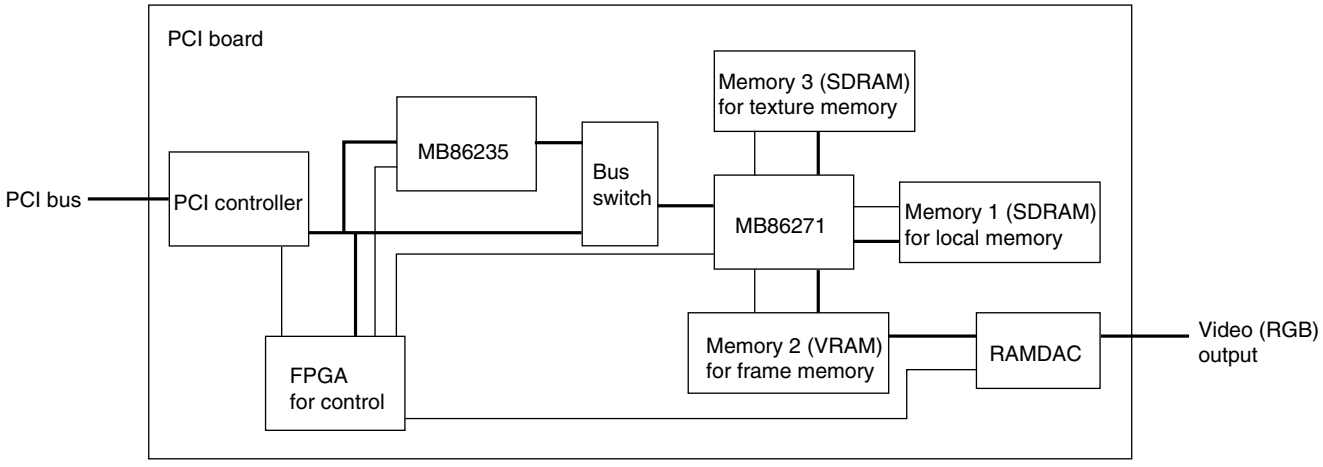

Fig.3– Hardware configuration of 3D-CG accelerator board.

RGB and Z values of pixel data and accesses Memory 3 (SDRAM) for the RGB values of texture data. Then, the MB86271 processes the data and stores the final results in Memory 2 (VRAM). The VRAM serial port outputs the display signals through the RAM DAC.

3.1.2 Software configuration

Figure 4 shows the software configuration of the system. Once linked, the OpenGL driver loads the microcode into the MB86235 and MB86271 and initializes the accelerator. When the application software calls for an OpenGL function, the OpenGL API receives the function, processes all of the geometric operations except the slope, and sends the results to the PCI driver. After accumulating the appropriate amount of data in main memory, the PCI driver sends it in a packet to the accelerator. The packet data is first interpreted by the MB86235. Then, after slope operations, the data is sent as packet data to the MB86271 for rendering.

3.2 Performance

The 3D-CG system has all the functions of the OpenGL library. **Table 1** compares performance figures for Gouraud polygon shading, and **Table 2** compares performance figures for texture mapping. For the performance measurements, we used an FM V6200 200 MHz Pentium Pro running under Windows NT 3.51. First, OpenGL with no accelerator was coupled to Windows NT 3.51. "1 light" in Table 1 refers to a parallel light source. The geometric primitive in both tables is a 24-bit color triangular strip. Table 1 shows that the system with the accelerator board has equal performances for polygons of 5 and 25 pixels, but a lower performance for a polygon of 50 pixels. Since the amount of calculations for geometry depends on the number of polygons and the amount of calculations for rendering depends on the number of pixels to be drawn, the bottleneck occurs in the CPU for polygons up to 25 pixels and in the accelerator for polygons over 50 pixels. Table 1 shows that compared to the system with no accelerator, the system with the accelerator is about four or five times faster. Table 2 shows that for easy point sampling, the system with the accelerator is more than three times faster. For bilinear sampling of a higher grade, the performance is more than 42

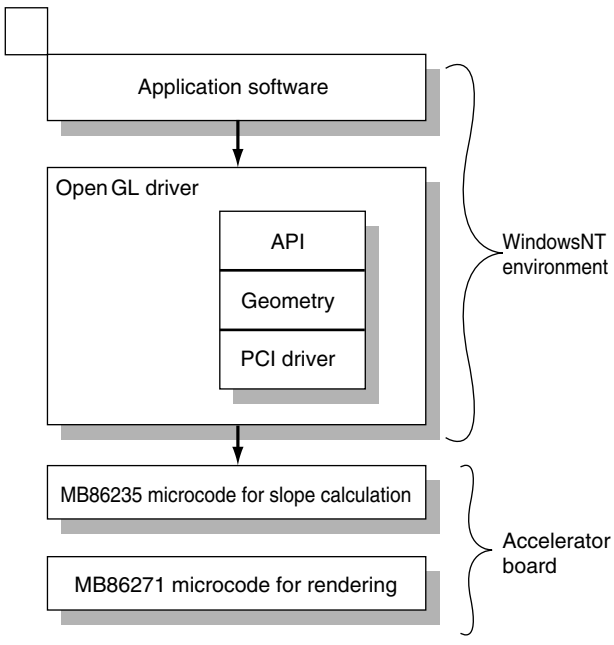

Fig.4– Software configuration of 3D-CG PC system.

Table 1. Performance for Gouraud shading.

Conditioning No.	1	2	3	4	5	6
Lighting	no	no	no	1 light	1 light	1 light
Polygon size [pixel]	5	25	50	5	25	50
Accelerator [K polygons/s]	221	221	172	203	203	171
No accelerator [K polygons/s]	77.4	45.9	36.7	80.6	53.8	46.3

Table 2. Performance for texture mapping.

Unit [M texels]

Sampling condition	Point sampling	Bi-Linear sampling	Mipmap point sampling	Mipmap bi-linear sampling	Mipmap tri-linear sampling
Accelerator	15	5.5	3.0	2.9	1.4
No Accelerator	4.9	0.13	0.18	0.12	0.065

times better. The performance difference for the highest grade of MIP map sampling is not remarkable because the texture calculation of the MIP map sampling is performed mainly by microcode operations of the MB86271. **Figure 5** shows a sample image. The frame rate (the most important factor for practical use) was 49 frames/s in the system with the accelerator and 0.82 frames/s in the system without the accelerator.

4. Video texture system

Video images can be improved by adding 3D-CG texture, but an enormous number of calculations are required to smoothly generate a complex background by 3D-CG. The required number of calculations can be reduced however by pasting video images of an actual locale — which also has the advantage of producing a more realistic image.

To add video texture, we must achieve the following:

1) Live video input, and
2) transfer of 3D-CG data and video data to the accelerator in a personal computer system.

A video capture board is a printed circuit board added to a personal computer. It accepts live video data and makes it available to the personal computer. However, for live video, an enormous amount of image data must be captured and transferred to the accelerator through the PCI bus. In addition, to produce high-quality images, a large amount of 3D-CG control data must be transferred through the PCI bus. It is difficult to transfer these two types of data through the PCI bus simultaneously.

Therefore, we added a video-input daughterboard to the motherboard to handle captured data as texture memory of the MB86271. This system therefore enables video texturing without the need to transfer data through the bus.

4.1 System configuration

Figure 6 shows the hardware configuration of the video texture system. **Figure 7** shows a photograph of the FM V6200 personal computer into which the system was installed. **Figure 8** shows the motherboard and daughterboard. The motherboard is the PCI board explained in Chapter 3 minus the MB86235 and Memory 3 but with an added daughterboard connector. This system does not need the MB86235 because the texture mapping performance is more important than the polygon performance. The daughterboard accepts NTSC video and stores it in memory as texture

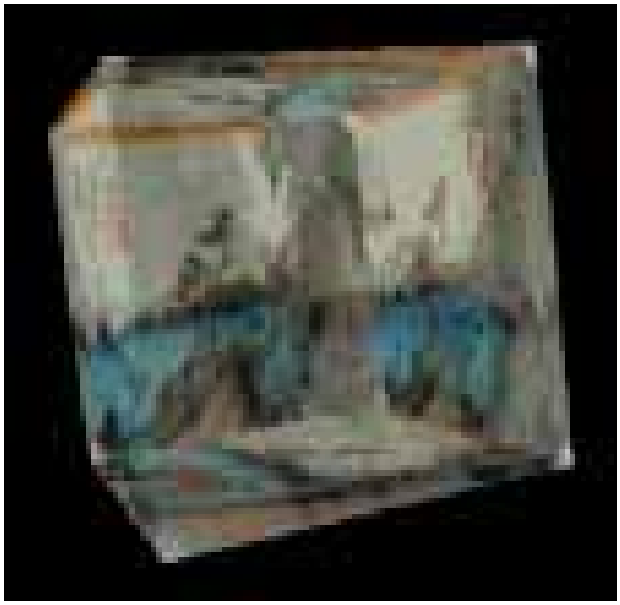

Fig.5– Sample image of texture mapping.

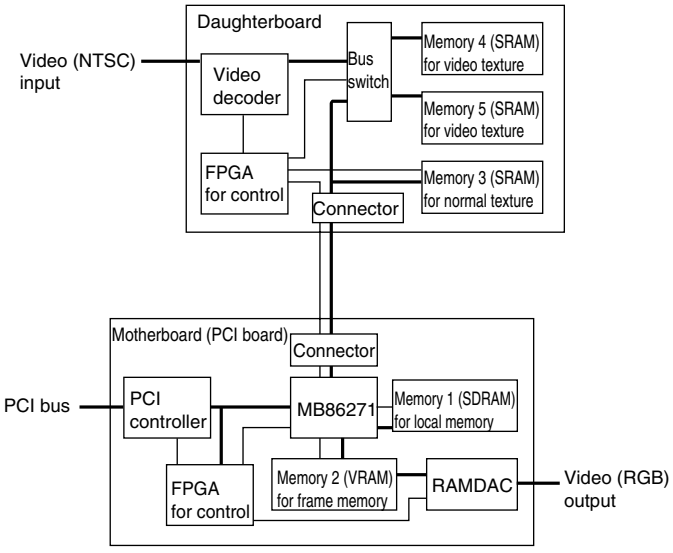

Fig.6– Hardware configuration of video texture system.

Fig.7– Video texture system.

Fig.9– Example 1: Live video reflection mapping.

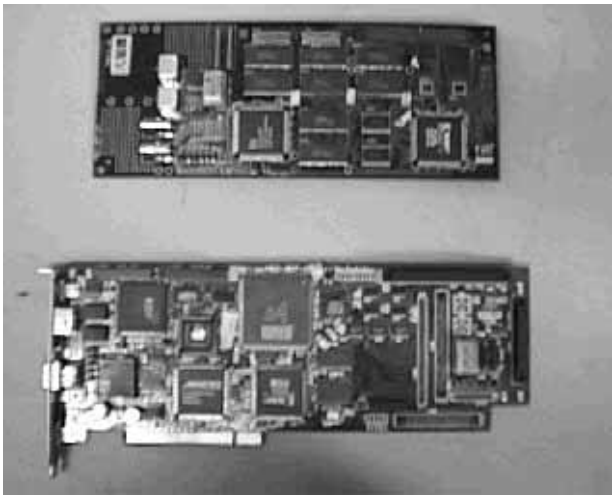

Fig.8– Video texture boards.

Fig.10– Example 2: Live video background scene.

data. The video decoder converts the input video signals into digital RGB signals and outputs them with synchronizing signals. The FPGA receives the synchronizing signals and generates address and control signals to store the video data in Memory 4 or 5. Memories 4 and 5 are for video texture data and have double buffers to enable texture data updating at the normal video rate. In other words, the MB86271 reads from Memory 4 or 5 and writes digital video signals into the other memory (Memory 5 or 4, respectively) after texture mapping. Control signals from the FPGA are sent to the bus switch for data bus input and output control. Memory 3 is for ordinary texture display.

4.2 Application trials

This section first explains how live images from a monitor camera are mapped into texture data. A texture mapping technique called reflection mapping allows the video texture system to express an image as if it were reflected off a curved object. **Figure 9** gives an example of a person reflected off the surface of a metal teapot. The reflected image changes when the person moves or the teapot is moved to a different position on the screen.

Recorded video images can also be mapped into texture data. **Figure 10** shows an example of a 3D-CG vehicle moving down a road. The road

and background were generated by texture mapping from images recorded from a moving vehicle. The video images make the 3D-CG vehicle appear to be moving.

5. Conclusion

We have developed an accelerator board for the generation and manipulation of video for 3D-CG on personal computers. The board contains a specially developed, high-performance rendering LSI chip. In addition, we extended certain functions to enable video images to be entered as texture data and confirmed the effectiveness of the system through application trials.

Further improvements in the cost and performance of 3D-CG systems are required. There are two key directions in which to develop future systems. One is to design one-chip solutions, for example, a chip that processes both geometry and rendering and a chip that contains a rendering engine and frame buffer memories. Another is to achieve high-speed transfer of processed data including texture data from main memory. We believe that success in these two directions will lead to an expanded role for 3D-CG in the field of image generation.

References

1) Neider, J. et al. : OpenGL Programming Guide. Addison-Wesley Publishing Company, 1993.
2) Chamas, A. et al. : A 64b Microprocessor with Multimedia Support. Proc. of ISSCC'95, pp. 178-179, 1995.
3) Inoue, K. et al. : A 10Mb 3D Frame Buffer Memory with Z-Compare and Alpha-Blend Units. Proc. of ISSCC'95, pp. 302-303, 1995.
4) Yoshizawa, H., Otsuka, T. and Sasaki, S.: High-Performance Architecture of a Next-Generation 3D-CG Rendering Processor. AGP. Proc. of ICSPAT'95, 1, pp. 195-199, 1995.

Toru Ozaki received the Bachelor and Master degrees in Information Engineering from Tokyo Institute of Technology, Tokyo, Japan in 1981 and 1983, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1983. He is currently engaged in research and development of image processing systems and computer graphics systems. He is a member of the IPSJ.

Seiya Shimizu received the Bachelor and Master degrees in Precision Engineering from Hokkaido University, Sapporo, Japan in 1987 and 1989, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1989. He is currently engaged in research and development of image processing systems and computer graphics systems.

Tatsushi Otsuka received the Bachelor and Master degrees in Electrical Engineering from Nagoya University, Nagoya, Japan in 1986 and 1988, respectively. He joined Fujitsu Ltd., Kawasaki, Japan in 1988. He is currently engaged in research and development of computer graphics systems. He is a member of the IPSJ.