A Real Time Rendering System with Normal Vector Shading

D. Jackèl
University of Rostock, 18051 Rostock

H. Rüsseler
GMD-FIRST*) at the Technical University of Berlin
Rudower Chaussee 5, 12489 Berlin

Abstract

This paper presents a graphics workstation for real-time display of Phong shaded polygons. The applied shading method is based on normal vector interpolation using a precalculated reflectance map. Nearly 40 million pixels and more than 2 million 3-sided polygons per second can be displayed. After giving an overview of the system, we describe the implementation details of the architecture. The normal vector-based shader unit is the focus of this description. In addition, we present a new method for the reduction of Z-buffer initialization time by means of an additional frame counter. Finally, we summarize the performance data of the rendering system and give an outlook of our future activities in this field.

Keywords: real-time image display, parallel processors, rendering processor, reflectance map, normal vector interpolation, visual realism, Phong-shading hardware.

1 Introduction

The main objectives in the development of real-time graphics workstations are, firstly, to increase the rendering performance and, secondly, to improve the visualization quality. Contemporary "high-end" graphics workstations have rendering performances of more than one million polygons per second in conjunction with hardware supported transparency, anti-aliasing, perspective texture mapping, image compositing, etc. In spite of the fact that great strides have been made to improve the realistic display of very complex objects and scenes, the shading process of all commercial real-time graphics workstations is based on the Gouraud-interpolation which is only suitable to render dull surfaces. The reason for this can be found in the fact that a rendering process which uses more efficient shading methods, for example Phong-shading, needs approximately 10 times more computation time per pixel than for simple Goraud-shading.

In order to render objects with shiny surfaces by means of a Gouraud-interpolation hardware, so-called *dirty Phong shading* is often used. For this shading method a closed-mesh polygon net is needed to achieve sufficient visualization results and to avoid "highlight jumps". A considerable disadvantage of *dirty Phong shading* is the extreme increase of the number of polygons that are necessary to define the surface of an object.

In the past decade, many attempts have been made to make Phong shading applicable for hardware implementations.

*) *GMD: Gesellschaft für Mathematik und Datenverarbeitung mbH (German NationalResearch Center for Computer Science)*
FIRST: Forschungsinstitut für Rechnerarchitektur und Softwaretechnik (Research Institute for Computer Architecture and Software Technology)

M. Deering et al. presented a simulated VLSI-approach of a *normal vector shader* (NVS) chip suitable for Phong shading [DEE88]. The estimated performance of a rendering system based on 16 NVS-chips is given by 220,000 triangle shaped polygons and 25 million pixels per second.

The objective of other approaches is to make the implementation of Phong more efficient by a combination of the interpolation and the light reflection equations, which are evaluated by forward differences [DUF79], or the application of an appropriate approximation technique.

Bishop and Weimar [BIS86] suggested a method which approximates Phong's normal interpolation and light reflection equation by a Taylor series expansion. Bishop Weimar's *fast Phong shading* reduces the amount of computation steps per pixel to only 5 additions and one memory access. Unfortunately, the additional computation of the Taylor series that requires about 500 floating point operations per polygon vertex diminishes its efficiency [APG88].

An alternative, suggested by U. Clausen [CLA89], interpolates polar angles instead of normal vectors, which has the advantage that the normal vector unit length remains unchanged during the interpolation. The disadvantage of the polar angle method is the occurrence of interpolation errors in the case of large polygons and the need of additional computation steps for changing the coordinate systems. Our approach is based on the approximation of the illumination model geometry by using the reflection map method that was originally suggested by B. Horn for shading analysis [HOR77]. The rendering processor architecture of the VISA system (VISA = VISualization Accelerator), which was developed in close collaboration between the GMD-FIRST and the Technical University of Berlin, was based on this method.

2 Architectural overview

The aim of the VISA project was to design a real-time graphics engine for the MANNA Supercomputer, which was constructed during the last four years at GMD-FIRST. Furthermore VISA was also designed to operate in connection with commercial platforms, e.g. SUNs SparcStation or Motorola Series 900.

The task of the VISA platform is the execution of the geometry process, which is divided into geometry transformations, clipping process, perspective and viewing transformations. Further-more, a parameter set needed to initialize the rendering processor must also be computed by this platform. To achieve VISA's maximum rendering performance of up to 2.4 million polygons per second (15 pixels per polygon) a specialized geometry processor is under development. Currently, in combination with the MANNA computer, we obtain an overall rendering performance of 520K polygons per second, which is limited by MANNA/VISA communication bandwidth of 50 Mbytes per second.

The MANNA system. MANNA is a scaleable parallel architecture. The computation units of this system are called MANNA nodes. Each node is based on a dual-i860XP processor shared memory architecture.

Both processors are operating totally symmetrical on a four-fold interleaved 32 Mbytes DRAM with an access bandwidth of 400 Mbytes per second. To connect the MANNA node with an interconnection network, each node has a bi-directional communication link. The

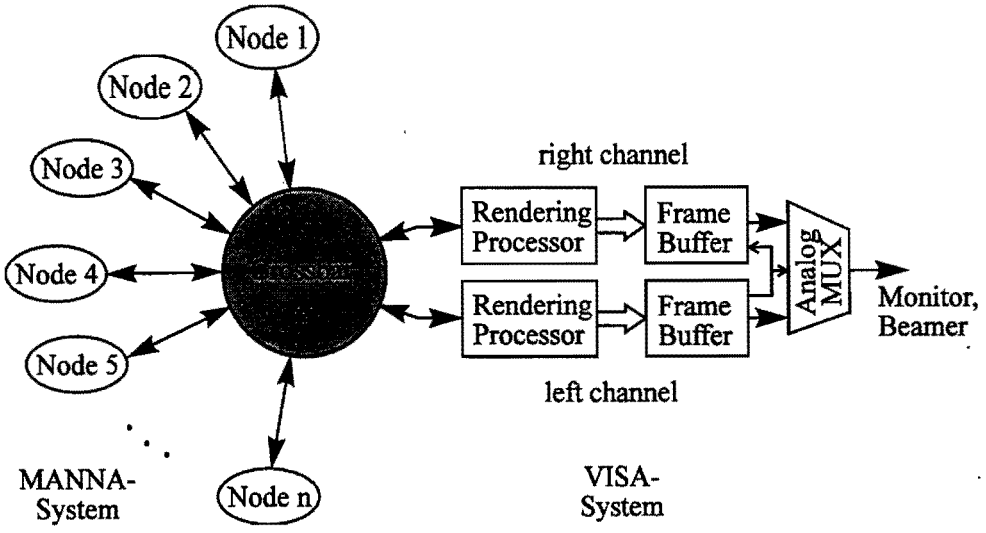


Figure 1. MANNA/ VISA system for stereoscopic display

communication bandwidth per node amounts to 2 x 50 Mbytes per second. The basic building block of the MANNA interconnection network is a byte-wide 16x16 crossbar switch. Any larger configuration can be composed by a hierarchy of crossbar switches. For example, a MANNA configuration with 160 nodes is divided into 10 clusters. Each of these clusters is formed by a 16x16 crossbar switch which interconnects 10 computation units. Its remaining six free links are needed for the cluster-interconnection by using six additional crossbar switch units. Detailed information of the MANNA system can be found in [GIL94].

For graphics applications one crossbar link is connected with the VISA real-time rendering system (rendering node). For stereoscopic or multi-channel (look around) displays it is easy to attach up to four VISA systems by means of free crossbar links to MANNA (figure 1).

VISA system. The main tasks of the VISA system, shown in figure 2, are *scan converting, normal vector based shading* and *z-buffering*. This process includes the interpolation of the coordinates and the normal vectors of all pixels of the polygon surfaces. Moreover, this unit uses the two polar-coordinates of the normals to determine the pixel-lightness by a simple memory access to a precalculated reflectance-map. The pixel-lightness will be combined with the hue- and saturation-values of the polygon and converted from HLS to RGB values by means of a lookup table. After z-filtering, the RGB values of the visible pixels are written into the image buffer.

3 Architecture and Function of the Rendering-Processor

The rendering processor is divided into parallel pipeline units, which are designated as XYZ and RGB pipeline. The task of these units is the execution of the scan-converting process including the determination of the pixel-colours for triangle-shaped polygons. The function of the XYZ pipeline is similar to many other well known real time systems [JAC92]. Therefore, we will put the function of the RGB pipeline and especially the normal vector shader operation into the focus of this section.

In order to render a polygon the initialising register of the XYZ and RGB rendering pipelines are loaded with the parameter set mentioned above.

3.1 Initialising

The parameter set is a hardware-adjusted geometry description of the projected polygon (figure 3). It comprises the number of polygon scanlines (n_1, n_2), the start co-ordinate P_{start}, the edge slope increments (Δru, Δlu, Δrd, Δld) the surface slopes for incremental steps in X- and Y-direction (Δzx, Δzy) as well as start vector and incremental vectors, needed for normal vector shading. Moreover, the parameter set embodies the shading mode, the hue and saturation value of the

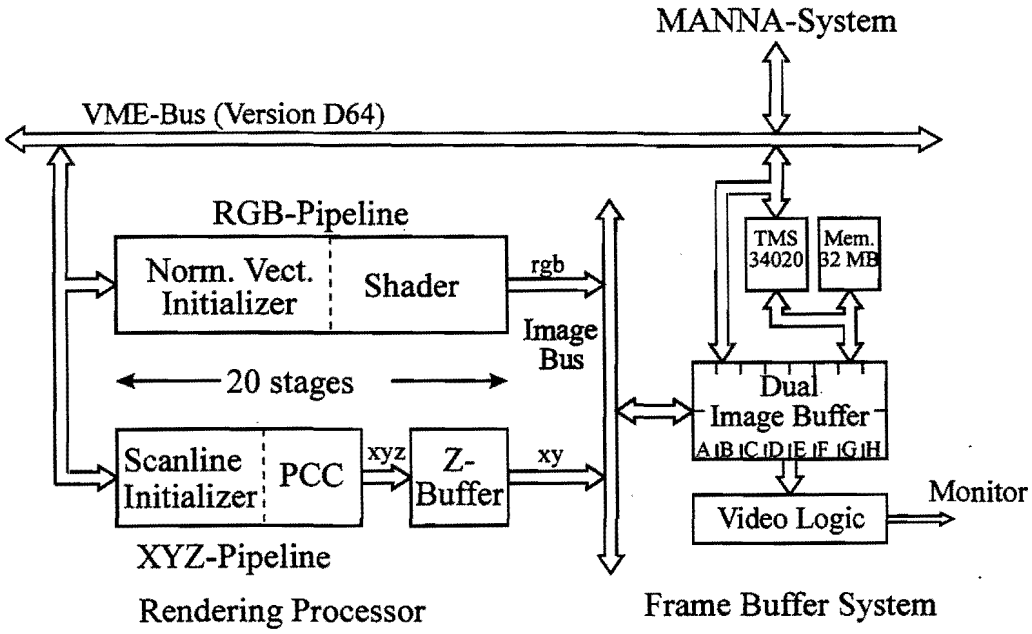


Figure 2. Block diagram of the VISA system

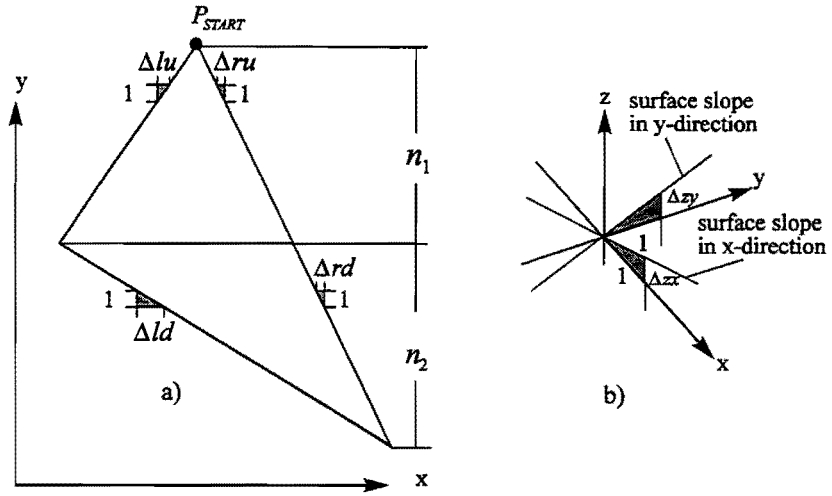


Figure 3. *Parameter set of the projected polygon: a) Start coordinate and edge slope increments b) surface slope increment in x- and y-direction. The normal vector and the incremental vectors are not shown*

polygon's colour, the transparency values, and the frame buffer control bits.

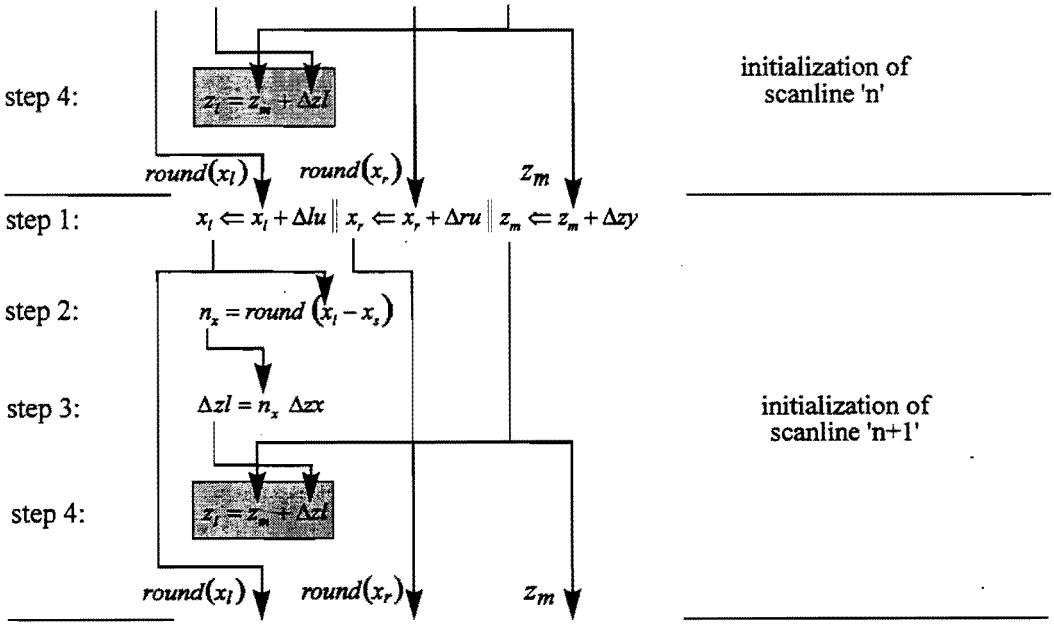
3.2 XYZ pipeline

As figure 2 shows, the pipeline unit is divided into a *scanline initializer* (SI) and *a pixel coordinate calculator* (PCC). The task of the initalizer unit is the computation of the pixel coordinates at the start and stop points of each new scanline. Four computation steps, illustrated by algorithm 1, are required to determine the initialising coordinates by linear interpolation of the precalculated edge and slope paramer values mentioned above.

The task of the pixel coordinate calculator is the linear interpolation of all pixel coordinates of a scanline. The PCC has only to interpolate the z-coordinates of the scanline through the surface slope increment for the X-direction because the y-coordinates of each scanline are constant and the x-coordinates are incremented by '1'.

$z := z_l$; **for** $x := \text{round}(x_l)$ **to** $\text{round}(x_r)$ **do** $z := z + \Delta zy$;

To achieve a sufficient load balance between the SI- and the PCC -units a FIFO-buffer is used.



Algorithm 1. *Determination of thr start- and stop-coordinates*

3.3 RGB pipeline

Function of the normal vector initializer. The computation of the pixel vector components at the starting point of a scanline is the task of the normal vector. This computation process is similar to the determination of z_f-coordinates. For each new scanline $\Delta N_y = [\Delta xn_y\ \Delta yn_y\ \Delta zn_y]$ (ΔN_y: change of the normal vector if x=const; y:=y+1) is added to the intermediate vector $N_m = [xn_m\ yn_m\ zn_m]$. We get

$$N_m := N_m + \Delta N_y.$$

In the next steps we get the normal vector at the starting point of the scanline by:

$$N_l = N_m + round(x_l - x_i)\ \Delta N_x.$$

(ΔN_x: change of the normal vector if y=const; x:=x+1). Algorithm 2 clarifies the computation steps of the normal vector initializer.

Principle of the normal vector shader. In order to understand the function of the normal vector shader, the following basic considerations are necessary.

If the light source and the viewer are far from the rendered object, we obtain a good approximation of the specular light reflectance values only as a function of the surface normal direction (figure 4).

The direction of the normal $N = [x_n\ y_n\ z_n]$ is given by the angles a_1 and a_2:

$$\alpha_1 = \arctan(x_n / z_n) \text{ and } \alpha_2 = \arctan(y_n / z_n).$$

Considering the simplified geometry of the lighting model (figure 4), we are able to determine the amount of light reflected by an object's surface 'r' as a function of the angles a_1 and a_2:

$$r = f(\alpha_1, \alpha_2)$$

The light reflectance values are precalculated for all α_1, α_2-combinations and then stored into a so-called reflectance map (RM). The reflectance map that can be considered as a matrix being addressed by the a_1 and a_2-values. For a given normal vector N only the computation of its direction angles and an additional memory access to the precalculated reflectance map are necessary in order to obtain the corresponding light reflectance value (figure 5a,b).

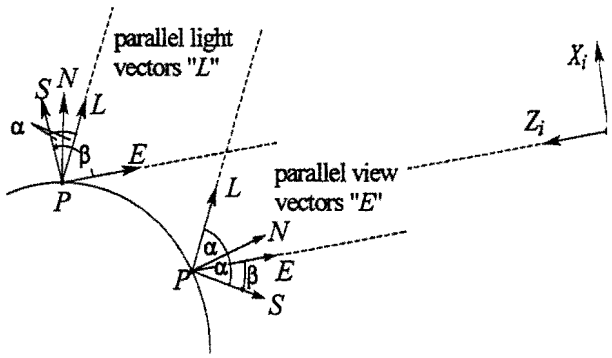
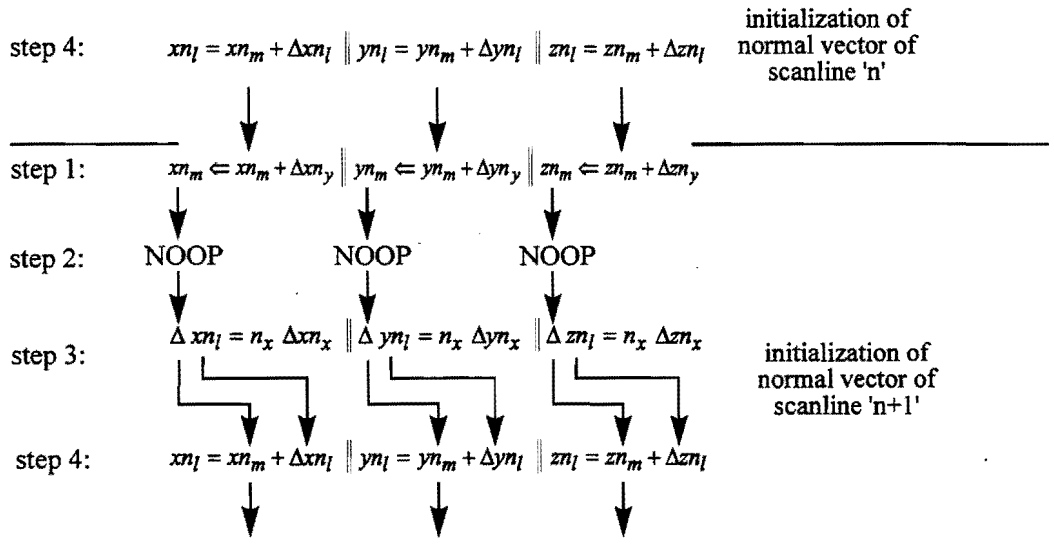


***Figure 4.** Simplification of the light model geometry by using parallel view (E) and light vectors (L). (S: reflecti vector)*



***Algorithm 2.** Initializing of the normal vector*

Function of the normal vector shader. Analogously to the determination of the z-coordinates, the shader unit, shown in figure 6, computes the pixel normal vectors $N=[x_n\ y_n\ z_n]$ of a scanline. The normal vector that corresponds to the first scanline pixel is initialized by N_i.

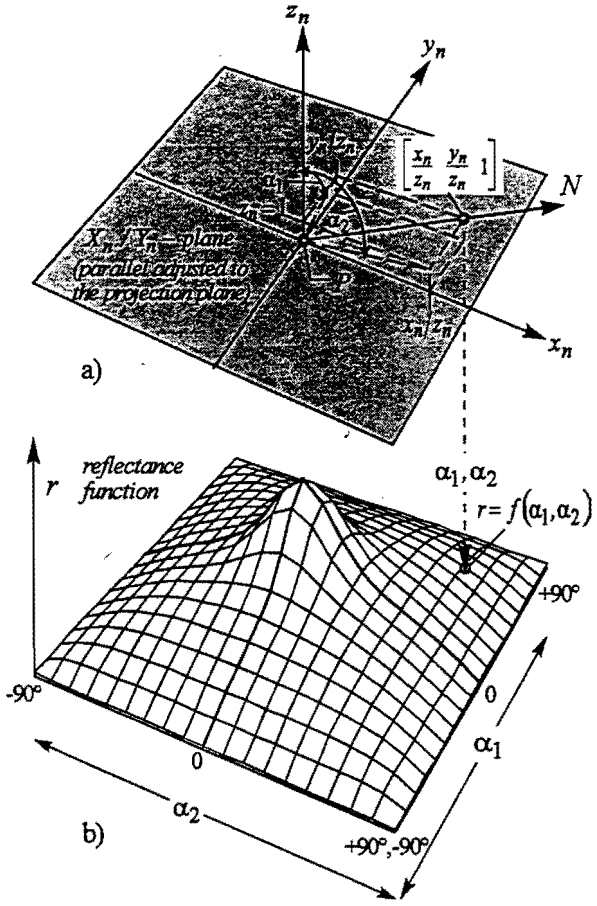


Figure 5a,b. Determination of the r-values by using a reflectance map: a) determination of the x_n/z_n- and y_n/z_n-quotients; b) Determination of. The computation of the reflectance map based on the Phong illumination model.

The normals of all succeeding pixels are determined by stepwise incrementation of $N := N + \Delta N_x$.

In the next processing steps the pixel normal components are converted from $[x_n\ y_n\ z_n]$ to $[x_n/z_n\ y_n/z_n\ 1]$ and subsequently, by using *arctan*-tables, from $[x_n/z_n\ y_n/z_n\ 1]$ to the angles a_1 and a_2, representing the normal direction.

As described above, we consider the combination of both angles as a RM-address. As the RM is realized as a memory matrix we get the corresponding light reflectance value r in a very simple way by only one RM-memory access. In order to render "walking through" image sequences or objects with different surface reflection factors, the VISA shader hardware contains up to eight RM's.

In the last processing steps the r-value is combined with the hue H and color saturation values S, which is allocated to each polygon. Assuming identity between the light reflectance value r and the lightness value L, we obtain the RGB output by means of a HLS/ RGB convertion table. In parallel to the HLS/RGB conversion, the Z-filtering will be executed.

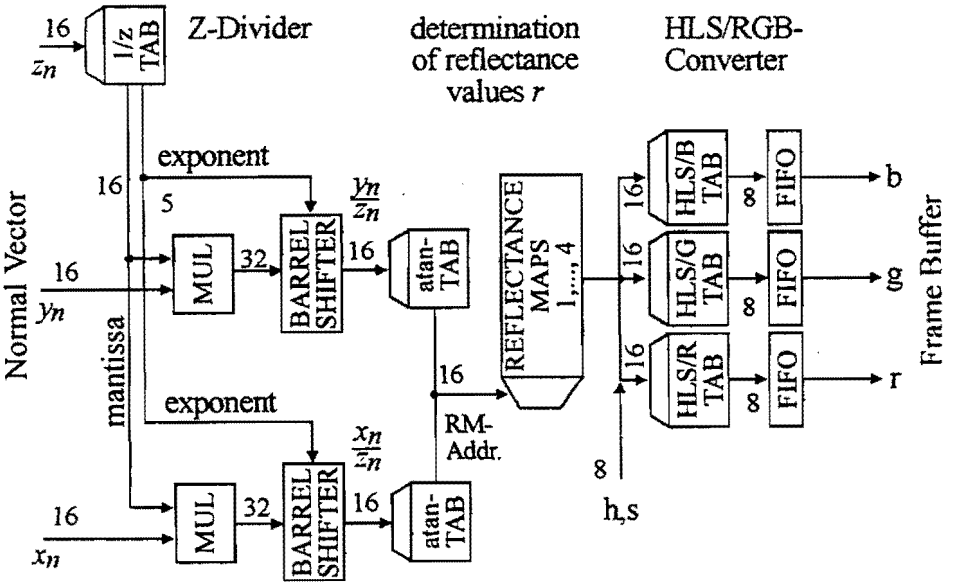
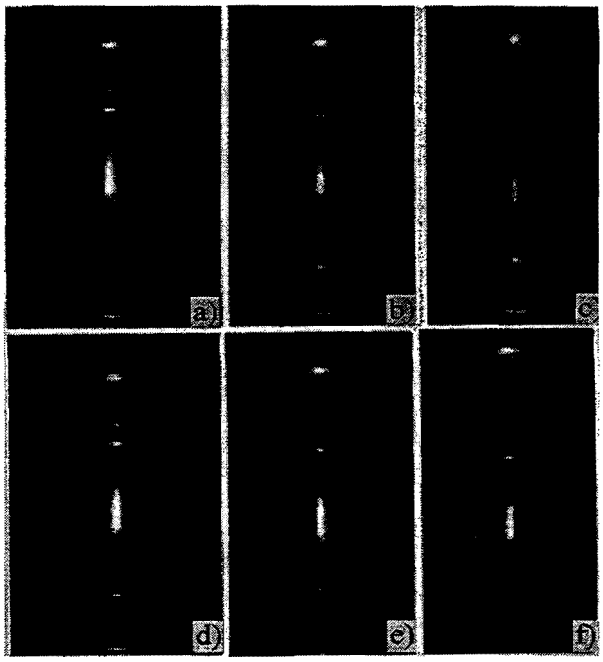


Figure 6. Architecture of the normal vector shader

4 Advantages and restrictions of the RM-method

DBecause of the simplification of the lighting model the positions of the highlights are incorrect. This leads to the question whether the rendering result is tolerable if we use such a simplified geometry. In Figure 7 a Phong shaded object is considered, which is rendered



Figures 7a-f. *Comparison between a traditional and RM-based Phong-shading: a-c) traditional Phong-shading (visual angles from left to right: 11°,30°, 50°); e-f) RM-based Phong-shading (visual angles: 11°,30°,50°)*

by means of a correct lighting model geometry Figures 7a,b,c and a simplified geometry Figures 7d,e,f, that is used in VISA. As this example shows, only if the distance of the projection center is very close to an object any differences of the highlight's positions are noticeable. Nevertheless, even in the case of an extremely perspective view, shown in figures 6c and 6f, the wrongly placed highlight leads neither to any visual misinterpretations of the object's geometry nor to an incorrect appearance of the rendered object.

The precalculated RM-entries are only valid for fixed directions of viewing and light vectors. The entries of the reflectance map must be updated from frame to frame if the position of the light source is changed dynamically or if we have to render "walking through" sequences. In these special cases we also achieve a considerable speed-up S of our RM-method compared to commonly used vector normal based shading techniques.

S is given by: $S = \frac{N_P}{N_E}$.

N_p: number of shaded pixels per frame.
N_E: number of RM-entries per frame.

To update the VISA reflectance map the computation of 180 by 180 = 32400 RM-entries (α_1,α_2-combinations) is necessary. If VISA is used in a MANNA environment, a MANNA node is responsible for the real-time RM-update. N_p depends on the object's complexity. Provided the number of the Phong shaded pixels differs from 0.3 to 40 million pixels per frame we get a speed-up from 10 to 120.

It is obvious that the RM method is not restricted by the number of light sources. In case of multiple light sources only the time to update the reflectance map will be increased. It is also possible to apply multiple colored light sources. In this case three separate reflectance maps for the primary colors red, green and blue are needed.

Another advantage of our normal vector shader approach is its low-cost hardware realization. According to our estimations the over-all cost of the VISA system is less than one percent higher than a possible VISA realization with a Gouraud shader hardware of similar performance.

5 Architecture and function of the Z-filter

The origin of the term Z-filter can be found in the function of this unit, which "strains" only the xy-coordinates and the RGB values of the visible pixel. For comparison with a conventional Z-buffer, which operates in parallel to the image buffer, this unit is integrated into the pipeline of the rendering processor. The special feature of the Z-filter is a 8-bit frame counter memory (FCM) and an additional frame counter (FC) which are used to reduce the initializing time to 0.1 ms.

In the following section the interaction between FC and FCM will be explained by a simple example (Figures 8a-h). Figures 8a-h shows eight snap shots of a 2-bit FCM state when a ball is moved across the frame buffer. Before the first image is generated, all FCM entries are initialized by 0 and the FC-state by 1. It is important that the FC-state is different from all FCM-entries at any start of a frame generation cycle. With each write access to the z-memory, in the course of the image generation, the FC-state is written into the FCM. As result we obtain a "foot print" of the generated ball (Figure 8a). At the end of this process the first of three FCM-segments will be initialized by the FC-state (Figure 8b) followed by the incrementation of the frame counter. The next and all further image generation cycles function in the same way (Figures 8a-h).

can be considerably reduced. The other advantage, the reduction of the initializing time, has already been mentioned above. By using the conventional Z-buffer technique, the entire 24-bit z-memory must be preloaded by the maximum z-value (2^{24}-1). Contrary to this, by using this method only a relatively small FCM-segment must be initialized before starting a new image generation cycle. As shown in Figures 8a-h, the initializing time of the Z-filter is in inverse proportion to the number of FCM-segments. A more detailed description of the Z-filter can be found in [COB92].

6 Frame buffer system

Figure 9 shows the block diagram of the frame buffer system. It consists of a TMS 34020 graphics processor,

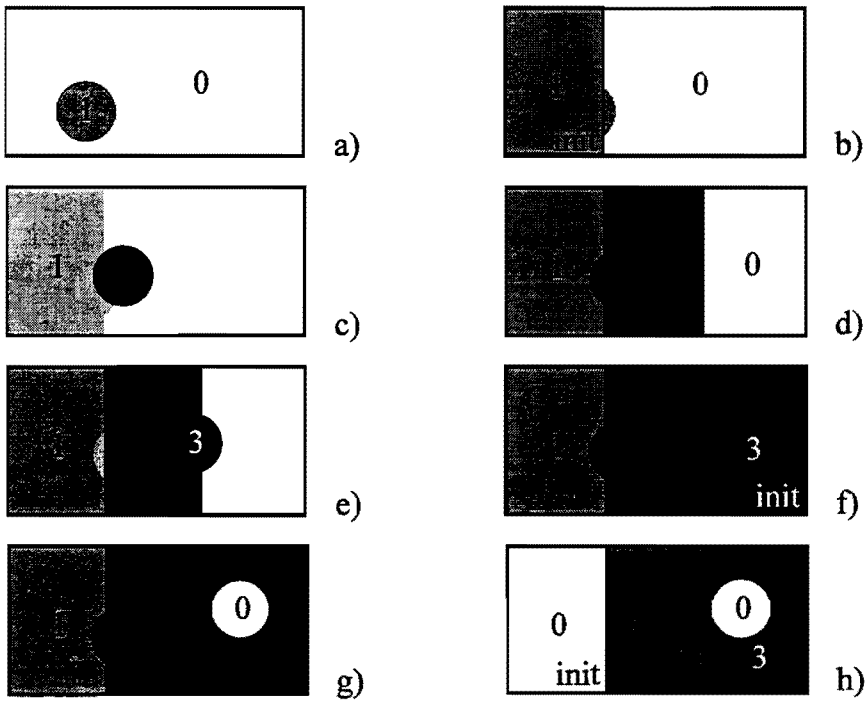


Figure 8a-h. *States of a 4-bit frame counter memory (a) after 1st cycle; (b) after initializing, before 2nd cycle; (c) after 2nd cycle; (d) after initializing, before 3rd cycle; (e) after 3rd cycle; (f) after initializing, before 4th cycle; (g) after 4th cycle; (h) after initializing, before 5th cycle.*

In case the FC-state and the FCM-entries are of unequal values, the z-value comparison can be substituted by a much faster comparison of the FC-state with the FCM-entries. The update of the z-memory is carried out directly. In this way the number of z-comparisons needed for the hidden pixel removal

up to 32 Mbytes working storage, 16 Mbytes image storage and the video logic.

The communication with the rendering system is carried out by an image bus or via an VME-bus interface. The following passage will discuss the main units of the frame buffer system in more detail.

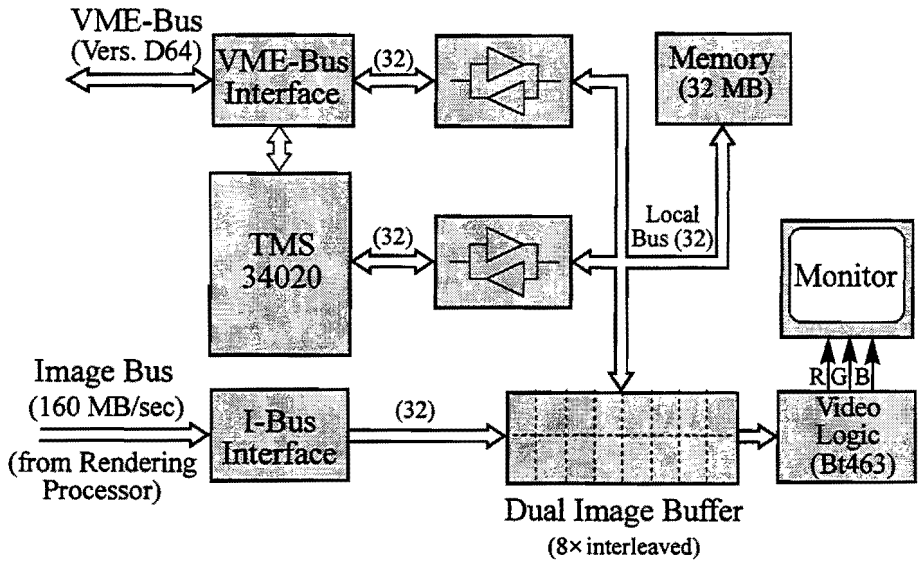
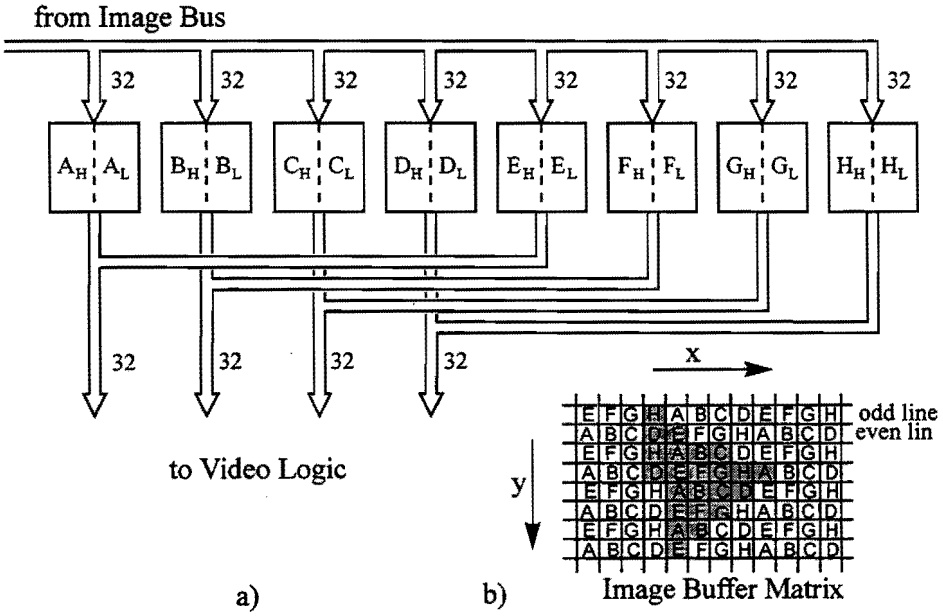


Figure 9. *Block diagram of the frame buffer system*

Graphics processor. The main task of the TMS34020 graphics processor is the generation of 2d-primitives such as vectors, circles, 2d-polygons as well as text and symbols. Special mail box registers which are mapped into the address space of the graphics processor are used for communication with the host system. The working storage consists of two interleaved operating memory banks, with a size of up to 16 Mbytes each.

Image buffer. The image buffer is designed as a multiport-RAM and permits the direct data access from the graphics processor and the access via the image bus and via VME-bus. Figure 10a shows the image buffer arrangement in eight identical memory banks (A-Bank,..., H-Bank) which are regulated by individual bank controllers. For operating in double buffering mode each memory bank is divided into an upper and a lower partition.



Figures 10a,b. *Image buffer: (a) logical organization of the memory banks; (b) logical assignment of pixel addresses to memory banks.*

The write access is carried out by means of the well known memory interleaving method by using eight address busses. In order to improve the performance for the write accesses different addressing schemes for even and odd lines are applied. For even lines the memory banks will be addressed by the sequence cycle of the memory banks: A, B, C, D, E, F, G, H; A, B, C, ... etc. For odd lines the sequence cycle is given by: E, F, G, H, A, B, C, D, E, F; E, F, ... etc. (Figure 10b). With an access time-displacement of 25ns a maximum write-input bandwidth of 40 Mpixel/sec will be achieved. The pixel data transfer to the video logic takes place with eight 32-bit data busses (A-Bus, ..., H-Bus), which are multiplexed by the following video logic with a frequency of 135 Mhz.

7 Conclusions and future work

We have described a rendering system for fast image generation. One highlight of this system is a shading unit, based on a light reflection map, which allows the application of the Phong-shading technique in a relatively simple way. Another important feature is the improvement of the well known Z-buffer. By using additional memory planes and a frame counter a considerable reduction of the initializing time can be achieved.

The bottleneck of the system is the insufficient processing power for the computation of the parameter set for initializing the rendering system. Presently this task is carried out by the MANNA system, which is able to transfer about 530,000 parameter sets per second. In order to achieve the maximum throughput rate of about 2,4 10^6 parameter sets per second a specialized processor is necessary. The development of such a dedicated processing unit is one of our next main objectives in the near future.

Another project assignment concerns the generation of textured objects. For this purpose, a texture and bump mapping pipeline which operates in parallel to the existing XYZ and RGB units is under development.

Acknowledgments

We thank the following people for their suggestions and contributions to this work: M. Cobernuss (Z-filter design), S. Budianto, T. Le Vin (simulation and gate array design), P. Wippich, K. Tehrani, W.-D. Plath (design of frame buffer and video-multiplexer), I. Ernst, R. Rasche, T. Stickdorn, M. Fröhlich (PEX- and driver implementation), T. Jung (application software) and the MANNA and MARCOS teams.

The funding of this research was provided by the German Federal Ministry of Research and Technology (BMFT)

References

[APG88] B. Apgar, B. Bersack, A. Mammen: *A Display System for the Stellars Graphics Super-computer Model GS1000*; Computer Graphics, Vol. 22, No. 4; pp. 255-262, 1988.

[COB92] M. Cobernuss, H. Rüsseler: *Verfahren und Schaltungsanordnung zur Unterdrückung verdeckter Bildpunkte*, patent specification: DE 41 34 576 A1, 1993 (in German).

[BIS86] G. Bishop, D.M. Weimar: *Fast Phong Shading*; Computer Graphics, Vol. 20, No. 4; pp. 103-106, 1986.

[CLA89] U. Clausen: Reducing the Phong Shading Method; Proc. Eurographics'89; Eds. W. Hansmann, F.R.A. Hopgood, W. Straßer; pp. 333-344, North-Holland, 1989.

[DEE88] M. Deering, S. Winner, B. Schediwy, C. Duffy, N. Hunt: *The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics;* Computer Graphics, Vol. 22, No. 4; pp. 21-30, 1988.

[DUF79] T. Duff: *Smoothly shaded renderings of polyhedral objects on raster displays*; Computer Graphics, Vol. 13, No. 2; pp. 270-275, 1979.

[GIL94] W.K. Giloi: *From Supremum to MANNA and META-Parallel Computer Development at GMD-FIRST*; Proc. Mannheim Supercomputing Seminar 1994, Sauer-Verlag.

[HOR77] B.K.P. Horn: *Understanding Image Intensities*; Artificial Intelligence Vol. 8, No. 2; pp. 201-231, 1977.

[JAC92] D. Jackèl: *Grafik Computer,* Springer-Verlag 1992, (in German).