Real Time Phong Shading

Ute Claussen

ABSTRACT Nowadays, hardware support for Gouraud shading is state-of-the-art. Most hardware implementations of the Phong shading algorithm lack flexibility, for example are restricted in the number of light sources. In this paper, we will present a concept of shading processors that has been developed in a rendering system called PROOF. Two types of processors have been designed, one performing the normalization of vectors. The other one is designed for faster and cheaper shading. The capabilities of the processors are demonstrated.

1 Introduction

Hardware implementations of real-time shading algorithms are aiming at Phong shading today. Several approaches have been made.

- The 'normal vector shader' developed at Schlumberger [5]. It is restricted to five directional light sources and its performance led to 24784 triangles rendered in less than 50 ms. The resolution of the screen is 1280 × 1024 pixels.
- The 'fast Phong shading' approach [1]. In principle, the computations of the illumination model and of the shading algorithms are combined into one formula. This term then is evaluated as a Taylor series of second order. Hence, rendering can be done with two adds per pixel. Unfortunately, a Taylor series should be developed for every light source. Furthermore, it is restricted to directional light sources.
- The 'faster Phong shading' by Kuijk and Blake [6]. They propose to interpolate angularly along great circles. This is an algorithmic approach that seems to be promising to be built in hardware.
- Another approach has been designed to render bicubic patches [11]. The hardware consists of adaptive forward differencing units and is also restricted to a single directional light source.

Our approach stands in the framework of the PROOF rendering system that has been presented before [3,10]. It is a more expensive but more flexible approach to real-time Phong shading.

First, the concept of the processor will be described. Two instantiations of the processor are developed. Their dataflow and datapaths as well as the algorithms and commands that are implemented will be described. Capabilities, the ways and the cost of its use is described. An outlook to the future and critical discussion will close the paper.

Table 1. Naming Conventions of this Paper

I_a	ambient intensity
I_d	intensity of directional light source
I_p	intensity of positional light source
k_a	ambient reflection
k_d	diffuse reflection
k_s	specular reflection
$\vec{N}$	surface normal
$\vec{L}$	light source vector
$\vec{H}$	highlight vector
$\vec{E}$	eyevector
$\vec{P}$	point on a surface
$\vec{P}_p$	position of positional light source
C	colour
P	number of positional light sources
R	number of directional light sources

2 The Processor Concept

The rendering system PROOF consists of three stages. The object processor pipeline is a distributed z-buffer system that solves the hidden surface problem. A second stage, called shading stage, calculates the illumination model, if an interpolation of normal vectors takes place in the object processor pipeline. The last stage, a filter stage, performs an oversampling and filtering of the resulting picture. The object processor pipeline and the filter stage have been described before, e.g. in [9,8].
A standard illumination model for a Phong shading is defined as:

$$C_{amb} = k_a I_a \tag{1}$$

for an ambient light source,

$$C_{dir} = \sum_{l=1}^{R} [k_d I_{d_l} (\vec{N} \cdot \vec{L}_{d_l}) + k_s I_{d_l} (\vec{N} \cdot \vec{H}_{d_l})^m] \tag{2}$$

for all directional light sources, and

$$C_{pos} = \sum_{l=1}^{P} [k_d I_{p_l} (\vec{N} \cdot \frac{\vec{P}_{p_l} - \vec{P}}{|\vec{P}_{p_l} - \vec{P}|}) + k_s I_{p_l} (\vec{N} \cdot \frac{\vec{E} + \vec{P}_{p_l} - \vec{P}}{|\vec{E} + \vec{P}_{p_l} - \vec{P}|})^m] \tag{3}$$

for all positional light sources [12]. The naming conventions of this paper are described in Table 1. Regarding these formulae, it is obvious that a common part of all computations is

$$C = k_x I_y (\vec{X} \cdot \vec{Y})^m. \tag{4}$$

Additionally, these terms have to be summed up.

The basic idea now is to built a processor that computes terms of the form

$$I_{out} = I_{in} + kI(\vec{N} \cdot \vec{A})^x. \tag{5}$$

Table 2. Instantiations of k, I, $\vec{A}$ and x for different types of light source

light source	ambient term k	I	$\vec{A}$	x	diffuse term k	I	$\vec{A}$	x	specular term k	I	$\vec{A}$	x
ambient	k_a	I_a	d	0	—				—			
directional light source	—				k_d	I_{d_l}	$\vec{L}_{d_l}$	1	k_s	I_{d_l}	$\vec{H}_{d_l}$	m
positional light source	—				k_d	I_{p_l}	d	1	k_s	I_{p_l}	d	m

Table 3. Capabilities of the two versions of the shading processor

version	light source model	illumination model
without normalization	ambient directional	ambient reflection diffuse reflection specular reflection
including normalization	ambient directional positional	ambient reflection diffuse reflection specular reflection

The instantiations of k, I, $\vec{A}$ and x for the different terms of Equations 1-3 are shown in Table 2. In addition, the values $\vec{N}$, $\vec{L}$, $\vec{H}$ or $\vec{P}_p$ respectively, and $\vec{P}$ have to be provided.

The provision of $\vec{N}$ and $\vec{P}$ is done by the object processor pipeline. This pipeline acts as a distributed z-buffer system and linearly interpolates the normal vector $\vec{N}$ for triangles. Consequently, vectors provided from this pipeline and passed on to the shading stage are not normalized.

3 Datapaths and Algorithms

It has been shown before that the computation of an illumination model without normalization of the interpolated normal can be accepted as a 'cheap' Phong shading [2]. Therefore, two versions of the shading processor are proposed: one that includes normalization in its algorithms as well as in its datapaths, and one that doesn't. An overview of the resulting capabilities of the processor versions is given in Table 3.

3.1 Datapaths

A design for the datapath of the simpler processor can be derived from equation (5). Basic functional blocks are dot product computation, vector multiplication, vector addition, and a look-up-table for exponentiation (see Figure 1). The intensities as well as the reflection and the highlight coefficient are represented as integers whereas vectors are represented as 2's-complement fix-point numbers. One of the characteristics of this processor is the high compression rate from input to output data.

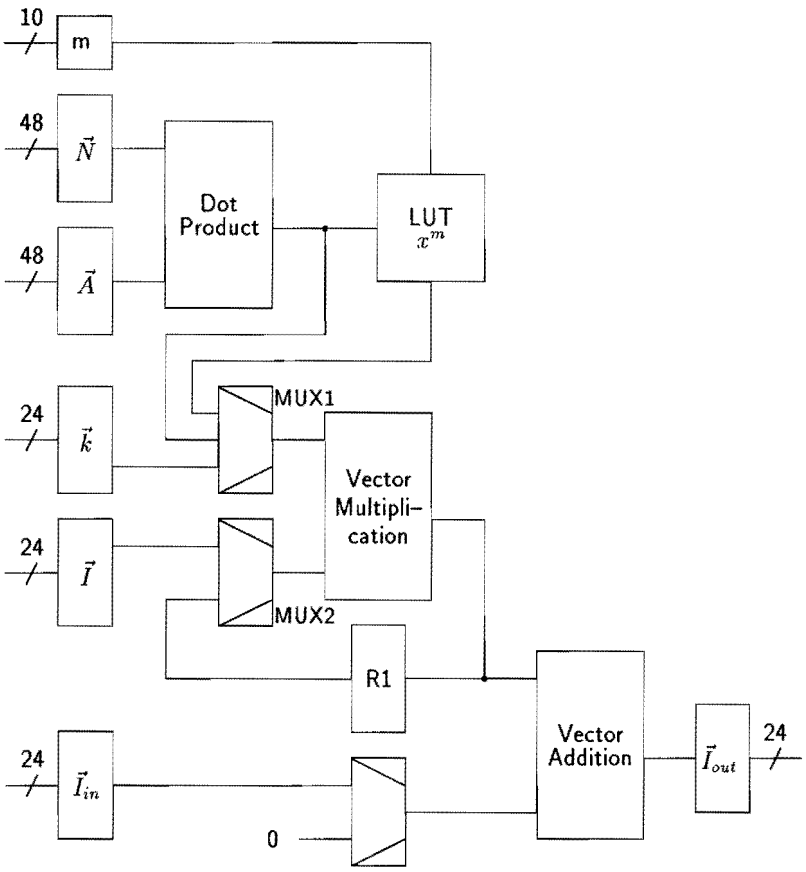

Fig. 1. Datapath of the simpler processor. One of the characteristics of the processors is the high compression rate from input to output data

3.2 Commands

Though the processor can be implemented in a lot of hardware environments, the following descriptions are restricted to its application in PROOF. It can be shown that a pipeline of shading processors is optimal concerning this implementation [4]. The first processor computes the ambient intensity, hence $I_{in} = 0$ and $I_{out} = C_{amb}$. Afterwards, each couple of processors computes the diffuse and the specular colour component for one light source. Consequently, for a scene with P positional and R directional light sources, $2(P + R) + 1$ shading processors are needed.

The processor can act in two modes, a loading mode and a pixel mode. In the loading mode, light source data is loaded into the registers of the processor where they remain constant during pixel mode. During the pixel mode, object data are passed through the pipeline to compute the colour of the object. Loading can be done with the two commands *define* and *redefine*. Light source data can also be *deleted*.

During pixel mode, *new pixel*, *new scanline*, and *new frame* indicate the respective change. A fourth command indicates a change of object without changing the pixel.

3.3 Shadow Polygons

One sort of objects that can be handled by PROOF are shadow polygons [7]. They don't have to be rendered by the shading processor but they influence the rendering of other objects. Furthermore, shadow polygons will be removed from the list of objects that are passed through the shading stage.

An object will not be illuminated by a certain light source, if it is inside the shadow polyhedron that is generated by the light source. Hence, in each shading processor, it has to be determined, if a shadow polygon 'opens' or 'closes' a polyhedron. Therefore, an additional counter has been implemented to do this determination. If the counter equals zero, the respective object will be illuminated. Otherwise, the current intensity will be passed on.

The extended datapath including this counter and the components for the normalization are shown in Figure 2. An additional vector adder and a look-up-table for the computation of $1/\sqrt{x}$ have been included and the dot product unit is split to make intermediate results available.

4 Capabilities of the Processors

The complexity of operations lead to the assumption that the clock frequency of the shading processor will be about 20 MHz. The number of cycles that is needed to compute the illumination for a certain configuration is listed in Table 4.

To support real time shading, a huge number of shading processors will be needed. This number depends on the number of pixels on the screen (M), the number of light sources (P and R), the average number of objects per pixel (o), and the cycle time of the shading processors (see Table 5). Here, it is assumed that to preserve the clock rate needed in a real time system, parallel pipelines are used.

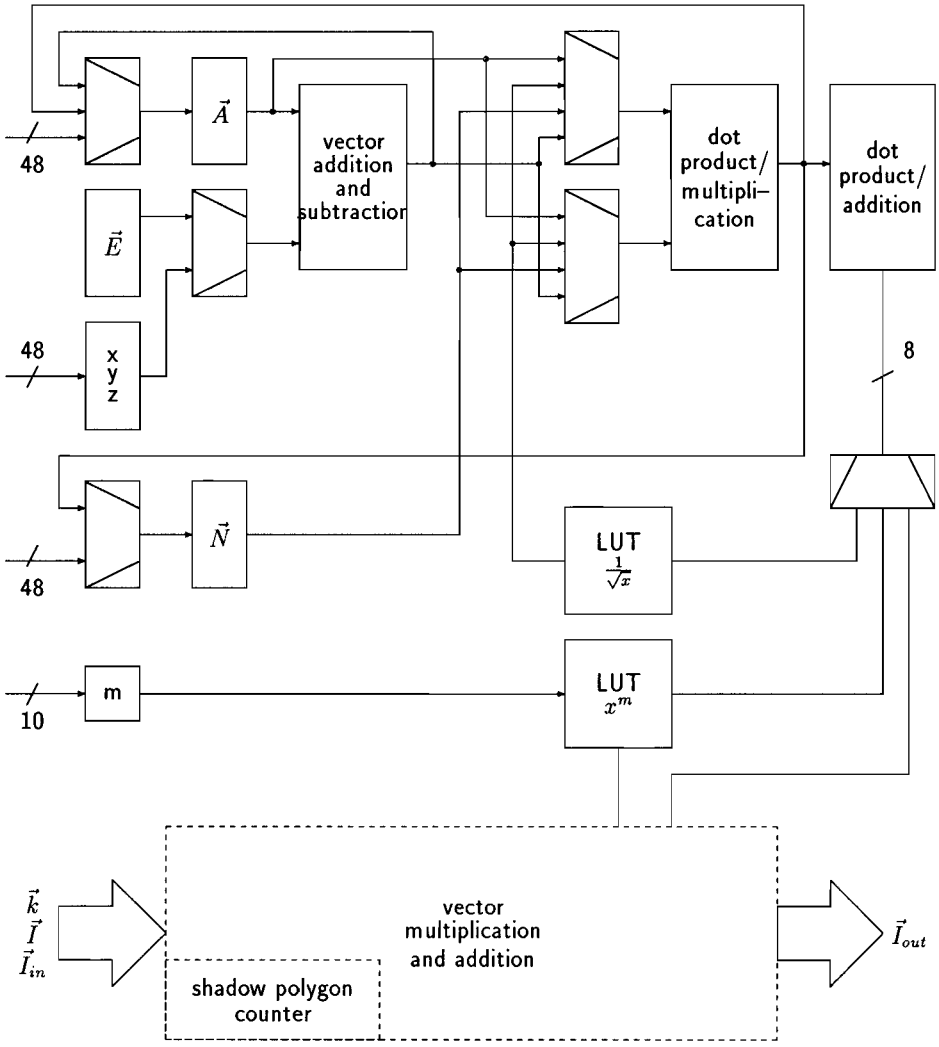

Fig. 2. The extended datapath for the shading processor with normalization and the treatment of shadow polygons

Table 4. Number of cycles that are needed to compute the different components of the illumination equation including loading

processor		without normalization	including normalization
ambient light source	computation of the ambient component	4	4
	computation of the ambient component and normalization of the interpolated normal	—	6
directional light source	diffuse component	6	6
	specular component	7	7
positional light source	diffuse component	—	12
	specular component	—	13
all	treatment of shadow polygons	2	2

Table 5. Computation of the number of shading processors needed for a certain configuration. Δt denotes the cycle time of a processor, R the number of directional light sources, and P the number of positional light sources. o denotes the average number of objects per pixel and M the number of pixels per frame. The number of frames per second is assumed to be 30.

		processor without normalization	processor including normalization
maximum time per pixel	Δt_{Pixel}	$7 \cdot o \cdot \Delta t$	$13 \cdot o \cdot \Delta t$
maximum throughput time per pipeline	Δt_{tp}	$(14R + 4) \cdot o \cdot \Delta t$	$(14R + 26P + 6) \cdot o \cdot \Delta t$
due time per pixel	Δt_{due}	$\frac{1}{30 \cdot M} s$	$\frac{1}{30 \cdot M} s$
number of parallel pipelines	$p = \frac{\Delta t_{Pixel}}{\Delta t_{due}}$	$\frac{30 \cdot M \cdot 7 \cdot o \cdot \Delta t}{s}$	$\frac{30 \cdot M \cdot 13 \cdot o \cdot \Delta t}{s}$
number of shading processors needed	L	$p(2R + 1)$	$p(2R + 2P + 1)$

An example demonstrates the number of processors that are needed to perform a real-time Phong shading. Assume 30 frames per second and 1 M pixels, hence

$$\Delta t_{due} = 33 \text{ ns}$$

which is less than the cycle time of a processor! Compared with the time that is needed to render $R = P = 5$ light sources for one pixel ($o = 3$),

$$\Delta t_{pixel} = 1950 \text{ ns},$$

the result is that $p \approx 60$ parallel pipelines, or 1260 processors would be needed.

5 Discussion and Conclusion

Though the concept of the shading processor developed in this paper seems to be promising, for example with regard to the hardware support of PHIGS PLUS, its implementation in the PROOF environment is quite expensive. Beyond the fact that spot light sources cannot be treated, perspective transformation is not handled correctly either.

Furthermore, it is questionable if the amount of hardware is justifiable if preprocessing methods like the highlight shading method could lead to a similar quality without the need for any shading stage in the system.

Last, but not least, in commercially available processors like the Intel i860, most of the parts that are needed for the illumination computation are already implemented. Furthermore, this processor already is available with a clock frequency of 40 MHz. Consequently, the shading processor can only be seen as a case study, that hopefully leads to new concepts for shading hardware.

Acknowledgements

This work was partly supported by the Commission of the European Communities through the ESPRIT II project 2484, SPIRIT.

References

[1] Bishop, G. and Weimer, D. M.: Fast Phong shading. *ACM Computer Graphics*, 20(4):103–106, 1986.

[2] Claussen, U.: On reducing the Phong shading method. In F.R.A. Hopgood and W. Straßer, editors, *Proceedings of the Eurographics'89*, Elsevier (North-Holland), Amsterdam, 1989.

[3] Claussen, U.: Parallel subpixel scanconversion. In A.A.M. Kuijk and W. Straßer, editors, *Advances in Graphics Hardware II*, Springer, Berlin, 1988.

[4] Claussen, U.: *Verfahren zur schnellen Beleuchtungs- und Schattierungsberechnung.* PhD thesis, Universität Tübingen, 1990.

[5] Deering, M., Winner, S., Schediwy, B., Duffy, C. and Hunt, N.: The triangle processor and normal vector shader: a VLSI system for high performance graphics. *ACM Computer Graphics*, 22(4):21–30, 1988.

[6] Kuijk, A. A. M. and Blake, E. H.: Faster Phong shading via angular interpolation. *Computer Graphics Forum*, 8(4):315–324, 1989.

[7] Newman, W. M. and Sproull, R. F.: *Principles of Interactive Computer Graphics.* McGraw-Hill, 1981.

[8] Romanova, C. and Wagner, U.: A VLSI architecture for anti-aliasing. 1989. Presentation at the Eurographics Hardware Workshop 1989 in Hamburg.

[9] Schneider, B.: A processor for an object-oriented rendering system. *Computer Graphics Forum*, 7(4):301–310, 1988.

[10] Schneider, B. and Claussen, U.: PROOF: an architecture for rendering in object space. In A. A. M. Kuijk, editor, *Advances in Graphics Hardware III*, Eurographics, Springer, Berlin, 1989.

[11] Shantz, M. and Lien, S.: Shading bicubic patches. *ACM Computer Graphics*, 21(4):189–196, July 1987.

[12] van Dam, A.: PHIGS+ functional description revision 3.0. *ACM Computer Graphics*, 22(3):125–218, 1988.