Real-Time Spectral Scene Lighting on a Fragment Pipeline

BERNARDT DUVENHAGE
University of Pretoria and
Mathematical and Computational Modelling Research Group at the Council for Scientific and Industrial Research[1]

Real-time desktop computer graphics systems have historically been based on empirical lighting models, such as the Phong lighting model, designed to be perceptually appropriate, but computationally efficient. New hardware developments since early 2003 have resulted in an affordable fourth generation graphical processing unit technology. This technology is allowing desktop computer graphics systems, like NVIDIA's GeForce® series, to implement more and more complex computer graphics algorithms, and lighting models for real-time applications. This paper describes an innovative "physically based" spectral lighting, material and camera model that is based on radiometry theory and is an expansion of the historical fixed pipeline graphics system. There are two render target modes of which *Direct mode* is aimed at high spectral resolution solids rendering and *buffered mode* at including transparencies.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Color*; *Shading*; *Texture*; I.3.1 [**Computer Graphics**]: Hardware Architecture—*Graphics Processors*; I.3.3 [**Computer Graphics**]: Picture/Image Generation—*Framebuffer Operations*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Fragment pipeline, spectral lighting, spectral BRDF, spectral BTF, camera model, radiometry, physically based, Graphical Processing Unit (GPU)

1. INTRODUCTION

Real-time desktop computer graphics systems have historically been based on empirical lighting models, such as the Phong[Phong 1975] lighting model, designed to be perceptually appropriate, but computationally efficient. New hardware developments since early 2003 have resulted in an affordable fourth generation graphical processing unit (GPU) technology [Fernando and Kilgard 2003]. This technology is allowing desktop computer graphics systems, like NVIDIA's GeForce® series, to implement more and more complex computer graphics algorithms, and lighting models for real-time applications.

This paper reports on the research done towards a "physically based" spectral material, lighting and camera model suitable for efficient implementation on modern GPUs for a real-time application. Spectral scene rendering is being used, for example, in the CSIR's[1] Optronic Sensor Systems competency area to do Hardware In the Loop (HIL) infra-red missile and sensor analysis as discussed by Delport at al. in [Delport et al. 2005].

In the next section the historical fixed fragment pipeline is discussed in more detail followed by what already exists in the spectral shading literature. The new spectral material, camera and lighting model is proposed in section 4 followed by its implementation in section 5.

2. THE HISTORICAL FIXED PIPELINE COMPUTER GRAPHICS SYSTEM

Modern real-time computer graphics application programming interfaces (APIs) aimed at hardware acceleration such as OpenGL® and Direct3D® [Fernando and Kilgard 2003] describe scenes with polygonal primitives. The trend has been to implement these renderers as z-buffered rasterisers that render with fragments as discussed in [Fernando and Kilgard 2003]. A fragment can be seen as a 3D surface element on the parent triangle that

[1]The Council for Scientific and Industrial Research (CSIR) has been constituted by an Act of the South African Parliament in 1945. It is one of the leading scientific and technology research, development and implementation organisations in Africa. The organisation undertakes and applies directed research and innovation in science and technology to improve the quality of life of the country's people.

Author Addresses: B Duvenhage, Council for Scientific and Industrial Research, Meiring Naude Road, Pretoria 0001, bduvenhage@csir.co.za

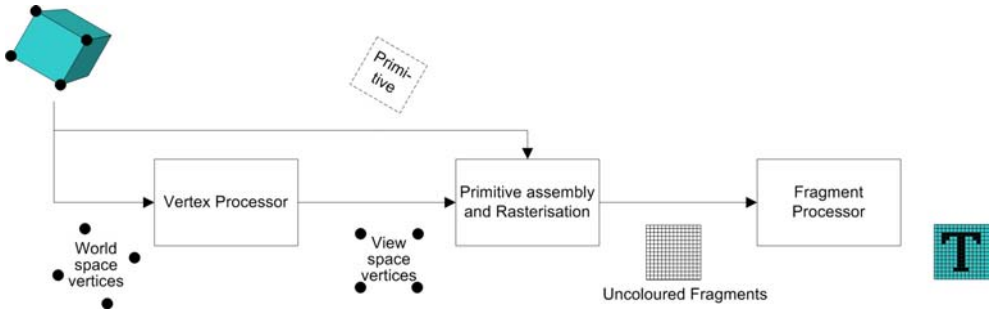

Figure 1. *The historical Fixed Pipeline Computer Graphics System*

projects onto a *single* potential picture element (pixel) as indicated in Figure 5. For a z-buffered rasterisation renderer to draw transparencies, the polygons have to be rendered in order from back to front according to the painter's algorithm as described in [Foley et al. 1997]. The processing steps of the renderer are structured into a fragment processing pipeline, shown in Figure 1, of three primary steps:

(1) The polygon vertices are fed into the vertex processor to be transformed into screen space and lit or coloured.
(2) The vertices are then structured into primitives which are fed into a rasteriser that generates the fragments (potential pixels) that fill the triangle on screen while interpolating per fragment attributes like texture coordinates, depth(z) value and colour for each fragment in surface space.
(3) Each fragment is then passed into the fragment processor to compute the final depth and colour information of the pixel that is written to the screen (render target). The depth information is used for the z-buffer visible surface determination algorithm [Foley et al. 1997].

OpenGL® and Direct3D® both use three wide spectral bands which are usually Red, Green and Blue (RGB) and implement the Phong[Phong 1975] lighting model. The Phong lighting model is a wide band empirical model to get an efficient algorithm that looks approximately correct. Wide band calculations model the light-scene interactions by doing all calculations for an *average* wavelength such as red, green or blue, slightly modifying the mathematics to make the result as accurate (or efficient) as possible for the chosen spectral band. Lorenzo et al.[Lorenzo et al. 1999] and Le Roux [le Roux 2001] discuss the classical fixed OpenGL® pipeline and how it may be directly applied to wide band IR scene modelling. Compared to current technology though, the pre-2003 generations of desktop CG hardware was severely limited in terms of performance and functionality.

3. PREVIOUS WORK DONE ON PHYSICALLY BASED AND SPECTRAL LIGHTING

Considerable work has been done on creating algorithms and models for "physically based" and spectral lighting. Some of the previous models and the research direction followed for a real-time lighting model to be implementable on modern desktop CG hardware are discussed.

The *classic lighting models*, covering empirical and "physically based" algorithms, are discussed in Foley, et al.[Foley et al. 1997]. The implementation of some of these algorithms on modern desktop CG hardware may be found in [Fernando 2004] and [Pharr 2005]. Kolb et al.[Kolb et al. 1995] divides the challenge of producing realistic scenes into three sub-problems: modelling reflection and transmission to account for the interaction of light with different materials, deriving illumination algorithms to model the transport of light throughout the environment, and modelling a camera that simulates the process of image formation and recording.

To address sub-problem one, modelling reflection and transmission at a high fidelity, we must model the light in spectral space because, according to Devlin et al.[Devlin et al. 2002], Evans and McCool[Evans and McCool 1999] and Gondek et al.[Gondek et al. 1994], material-light interactions are hugely dependent on wavelength. To do this the spectral space has to be sampled or covered by the model in some way. The fidelity of the model depends on the effectiveness of the sample coverage as described in [Chalmers et al. 2000]. Cabral et al.[Cabral et al. 1987] discusses "physically based" lighting and the use of Bidirectional Reflectance Distribution Functions (BRDFs) which is a concept from radiometry theory to represent or record a material's local reflection properties under different lighting and viewing directions. BRDF theory is also discussed in detail in [Williers 2003],[Westin et al. 2004] and [Cook and Torrance 1982]. Spatial BRDFs are discussed in [Fernando 2004] and BRDF texture functions (BTFs) in [Dana et al. 1999] and [Pharr 2005]. Both spatial BRDFs and BTFs approximate a material

surface as having a BRDF that varies with surface position.

Williers[Williers 2003] gives an in depth technical background on radiometry and atmospheric radiometry as a well established mathematical tool to help address sub-problem two, deriving an illumination algorithm, but there seems to be a gap in the literature on the application of radiometry theory to the fragment pipeline. This paper attempts to address that gap with a physically based spectral illumination expansion to the fragment pipeline for a real-time application.

In the normal OpenGL® pipeline the camera exposure calculation is arbitrary and Kolb et al.[Kolb et al. 1995] is one of a rare few resources found by the author that discusses realistic modelling of the camera's film exposure on a pixel level to address sub-problem three. Williers[Williers 2003] also gives an in depth technical background on optical systems.

4. PHYSICALLY BASED SPECTRAL LIGHTING ON A FRAGMENT PIPELINE

The proposed lighting model is a spectral expansion of the tri-colour wideband lighting model implemented on the fragment processor (last processing block in figure 1). In this section the radiometry theory, adapted from Williers in [Williers 2003], required for doing the spectral calculations is introduced. Following on this the new spectral material and light source models are discussed. Finally the camera model and the expanded spectral lighting fragment pipeline are discussed.

4.1 Radiometry Theory

Flux is the amount of watt (energy per second, unit is W) that flows between a light source and a fragment or between a fragment and the camera aperture shown in Figure 5. *Irradiance* is the surface density of flux per receiving surface area measured in W/m^2. The *Solid Angle* of an object measured from an observer's point of view is defined as the area covered by a projection of the object on to the unit radius sphere around the observer. The dimensional unit is sr (ste-radian) and the maximum value of solid angle is 4π being the entire unit sphere.

Radiance is what humans perceive as brightness and it is measured in $W/(m^2*sr)$. If two surfaces irrespective of distance from viewer, ignoring atmospheric attenuation for now, appear equally bright (their composing pixels have the same value) then the surfaces have the same radiance. *Brightness* may be defined as pixel value for computer graphics. *Intensity*(pointance), measured in W/sr, is used to describe the output power of a point light.

4.2 Spectral BRDF Material Component Model

The material component model splits the material's $\mathrm{BTF}(\lambda, x, y, l, v)$ into a texture function or spatial modifier, $\mathrm{TF}(x, y)$, and a spectral BRDF, $\mathrm{BRDF}(\lambda, l, v)$ where λ is wavelength, x and y the surface position, and l and v the light and view directions respectively. This allows the material BTF to be represented by two lower dimensional functions, although at the cost of restricting the material approximation to having a TF that's independent of the BRDF parameters and vice versa. The advantage of the component model is that when used as a multidimensional structure to record material properties it uses orders of magnitude less memory and it may be implemented on current hardware which only support low dimensional (up to three dimensional) table lookups. A complete material model is then described by a set of material components and it is important to note that an object's material will be specified as a certain combination of reflection, transmission and emission components that are simultaneously applied according to the different component modifiers.

A BRDF is usually used for the modelling of material reflection, shown in Figure 3 and as discussed by Cabral et al. in [Cabral et al. 1987], but the author uses the BRDF structure to represent the pre-computed material property distribution functions for transmission and emission as well. The scalar value returned by the BRDF represents $W/(m^2*sr)$ for emission, and (fraction of flux)/(m^2*sr) for reflection and transmission. Figure 2 shows three views of a transparent disc that selectively transmits portions of the system spectral bandwidth according to viewing angle.

The material component modifier is a 2D single channel float texture that is used to create greater surface detail by spatially modulating the material component spectrum as shown in Figure 4. Per vertex texture co-ordinates, exactly as for the fixed pipeline system, are used. Each texture element (texel) is populated with a scalar that the material component spectrum is multiplied by to get the final surface element spectrum. An

Figure 2. *Wavelength dependent transmission at different incident angles due to a spectral BRDF material component*

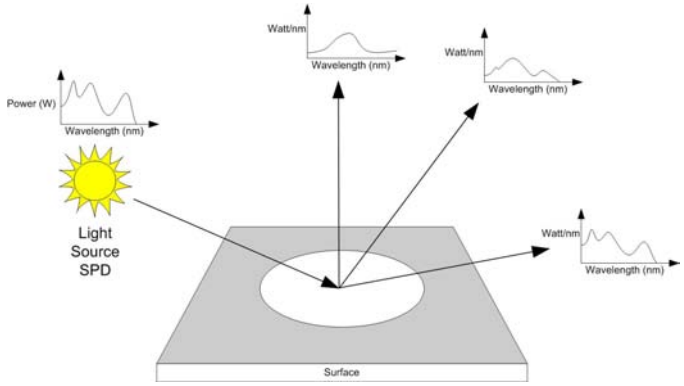

Figure 3. *Spectral BRDF*

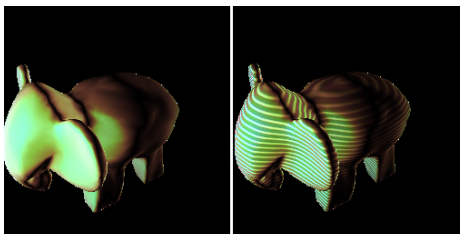

Figure 4. *Spectral Material (left) and Spatially Modulated Spectral Material (right).*

emission spectrum modifier would be a surface texture of values between 0.0 (no emission areas) and 1.0 (full power emission areas) which get multiplied with the emission spectrum to calculate the actual surface element emission. The reflection and transmission spectrum modifiers work similar to emission, but a 1.0 would indicate full reflection or full transmission and 0.0 would indicate no reflection or no transmission.

4.3 Spectral Point and Ambient Light Sources

OpenGL® and Direct3D® traditionally use point light sources and the new spectral lighting model makes no modification to this to keep the mathematics and implementation simple. A point light is dimensionless, omni-directional and radiates a certain amount of energy per second given by its spectral power distribution (SPD), shown in Figure 3. The flux flowing through a surface fragment, ignoring atmospheric effects, is calculated by $(\omega/4\pi)$ * SPD where ω is the fragment solid angel from the light source.

Ambient light as discussed by Foley et al. in [Foley et al. 1997] has no position and is meant to, in local illumination models, approximate sky-dome emissions and indirect lighting which is light reflected from surrounding surfaces on to the object. Computing the reflection of ambient light according to the BRDF requires an integration of the incoming light across the sky dome. To avoid doing the runtime BRDF-over-sky-dome-integration, the ambient light may be approximated or pre-reflected from the material and added as a pseudo emission material

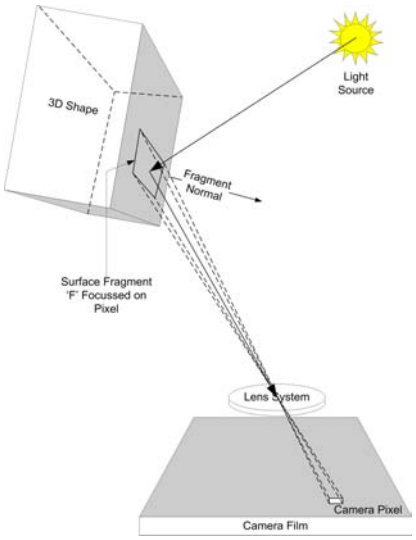

Figure 5. *Fragment Pipeline's Camera Model*

component.

4.4 Atmospheric Effects

The atmosphere is described by an extinction coefficient distribution across the system spectral bandwidth and an atmospheric radiance spectrum as discussed by Williers in [Williers 2003]. The atmosphere attenuates the flux from the light source to the surface with $e^{-R\gamma}$, where R is the distance between the light source and the fragment, and γ is the atmosphere's extinction coefficient and a function of wavelength. The same happens for the reflected flux between the surface fragment and the camera aperture.

The atmosphere also adds to the flux from light source to surface fragment (and similarly for the reflected flux from surface fragment to camera aperture) with $L_m\left(1-e^{-R\gamma}\right)$, where R and γ is as above and L_m is the atmospheric radiance and a function of wavelength.

4.5 Camera Model

The camera model is an important part of the fragment pipeline that matches the rasteriser to the radiometry calculations by computing each fragment's position, orientation and area. These properties may be found by reverse projecting each camera pixel onto its parent surface as shown in Figure 5.

The fixed pipeline camera is a pinhole camera which simplifies the scene to film transformation to a linear projection. The camera model used for the spectral pipeline uses the same linear projection, but to model an "always in focus" optical system for the reason that the radiometry calculations includes a bigger than pinhole camera aperture size for energy collection calculations. The camera film model is constructed from one or more sensor types arranged in pixel arrays on the film. Each sensor type has a certain spectral sensitivity which is represented by a 1D spectral lookup. The spectral sensitivity is integrated with the spectrum of the incoming flux of each fragment to get the absorbed flux that contributes to the corresponding pixel's brightness.

Each pixel's flux value is exposed, multiplied by a shutter time variable, to become an energy value in Joule. The sensor spectral sensitivities and exposure may alternatively be calibrated to expose the pixel flux values into any measurement units such as candle power, photon count or camera hardware pixel voltage.

4.6 Spectral Render Target

The expansion of the render target was the most important change made to the fixed pipeline graphics system because it allows the hardware to have a spectral image buffer instead of the usual RGB image buffer. There are two render target modes and each mode has advantages and disadvantages.

4.6.1 *Direct Mode*

When the lighting model is in direct mode the render target is used to store the pixel flux values which are the results of directly integrating the pixel fragment spectra with the sensor sensitivities. Each sensor type is allocated to a unique pixel channel allowing as many sensor types as pixel channels. Direct mode thus never stores the spectral image which means that the pixel spectral resolution is not limited by the render target's bits per pixel.

Direct Mode is not capable of doing transparencies because the spectral pixels are never stored to apply the painter's algorithm. Another disadvantage of Direct Mode is that the spectral image, before sensor spectral integration, is never available for download to the CPU or further processing.

4.6.2 *Buffered Mode*

When the lighting model is in buffered mode the render target is used to store the spectral pixels, binning the wavelength samples into the available render target channels. Once rendering is finished the sensor spectral integration will be done as a separate rendering pass. Buffered mode is capable of doing transparencies because the render target is used as a temporary image buffer for the painter's algorithm. The disadvantage of this is that the pixel's spectral resolution is limited to the hardware render target channel depth.

4.7 Spectral Radiance Fragment Pipeline

The spectral pipeline is similar to the fixed pipeline in Figure 1 except for the fragment processor that has been significantly expanded. Considering only a reflection material component as an example, the fragment processor's basic algorithm is (symbols as defined in Section 4.2; k defined below):

(1) calculate the fragment's area, A, by back projecting the pixel onto its parent surface,

(2) calculate the light to fragment solid angle, ω_1, and the fragment to camera aperture solid angle, ω_2,

(3) calculate the flux, Φ_l, that flows from the light source onto the fragment using $\Phi_l(\lambda) = SPD(\lambda)\frac{\omega_1}{4\pi}$,

(4) calculate the modifier texture value, m_{refl}, using $m_{refl} = TF_{refl}(x, y)$,

(5) calculate the flux, Φ_{refl}, that's reflected from the fragment and flows from the fragment into the camera aperture using $\Phi_{refl}(\lambda) = \Phi_l(\lambda) \cdot m_{refl} \cdot BRDF(\lambda, l, v) \cdot \omega_2$,

(6) calculate the pixel flux, Φ_{pixel}, by numerically integrating $SensorSensitivity(\lambda) \cdot k(\lambda) \cdot \Phi_{refl}(\lambda)\,\delta\lambda$ over the system spectral bandwidth, and

(7) calculate the pixel exposure, Q_{pixel}, using $Q_{pixel} = \Phi_{pixel} \cdot t_{exposure}$.

$k(\lambda)$ is the calibration constant, being, for example, 1 for pixel values exposed to energy in Joule or $\frac{\lambda}{hc}$, where h is Planck's constant and c the speed of light, for pixel values exposed to photon counts.

5. MODEL IMPLEMENTATION

In this section the current desktop CG hardware architecture and programming model is shortly discussed followed by the implementation details of the spectral lighting model. The expanded fragment processor is shown in Figure 6.

5.1 Current Hardware

The Graphical Processing Unit (GPU) used was the GeForce® 6 series which has a fourth generation programmable pipeline architecture and is computationally very powerful [Pharr 2005]:

(1) The GPU to Graphics memory bandwidth is 35 GB/sec while the typical CPU to System memory bandwidth is 6.4 GB/sec,

(2) the GPU processing power is 100+ gigaflops compared to 12 gigaflops for high end CPUs,

(3) the GPU to CPU bandwidth available is a maximum of 4GB/sec on PCI-Express,

(4) to support efficient vertex and fragment processor texture lookups a texture cache is used to hide the latency of consecutive and localised texture accesses,

(5) the internal processor, texture units and render target support full 32bit floating point precision,

(6) the GPU natively supports vector and matrix instructions,

(7) the GPU uses a stream programming model with multiple vertex and fragment processors to support it.

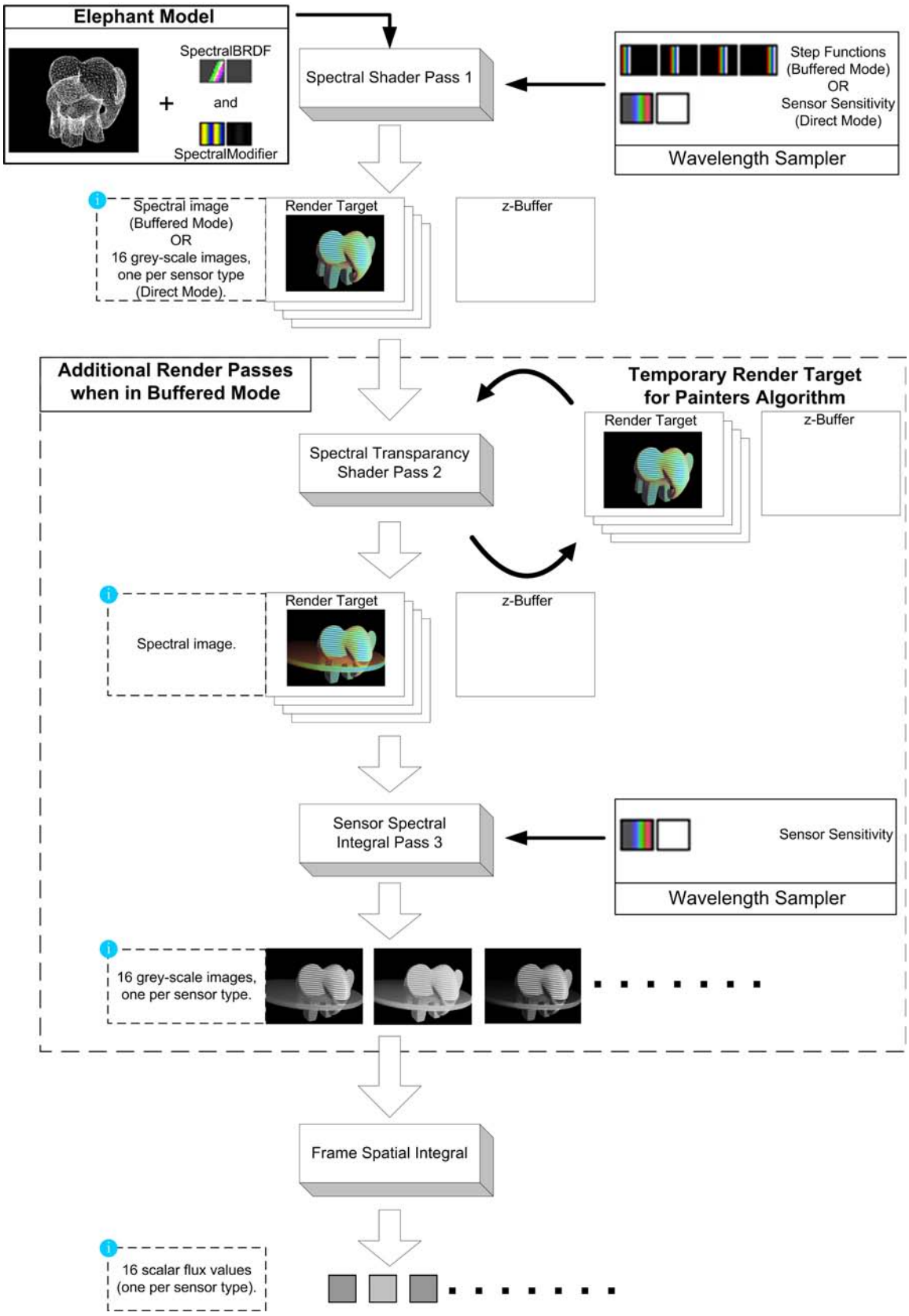

Figure 6. *Expanded Fragment Processor*

The stream programming model, discussed in more detail in [Fernando 2004], promotes data parallelism and arithmetic intensity which is respectively, the processing of a lot of data in the same way and the minimisation of moving data to and from the system. The stream programming model is an efficient way of expressing programs for the GPU.

The render target is constructed from multiple frame buffers to get the required colour channel resolution per render target pixel. The maximum number of frame buffers per render pass supported by the GeForce® 6 series hardware is four and a depth buffer. Each frame buffer has four 32bit floating point colour channels giving a maximum of sixteen 32bit floating point colour channels (512 bits) per render target pixel. For buffered mode more frame buffers would allow for better spectral resolution and more sensors. For direct mode the spectral resolution is independent of the number of frame buffers, but more frame buffers would allow for more sensors.

5.2 BRDF Implementation

Each material component BRDF is a single channel 1D, 2D, 3D, etc. lookup table where the first dimension encodes the wavelength and the other dimensions are used for encoding material BRDF angles. A table lookup BRDF model is more feasible on the GPU than on the CPU due to the fairly large amount of memory, high memory bandwidth and latency hiding caching available in modern desktop CG hardware. The highest dimensional lookup tables, implemented as textures, supported by the hardware is three, which is enough to directly implement the spectral dimension and two BRDF angles. Higher dimensional lookups can be implemented by packing the data into multiple 3D (or 1D or 2D) textures.

Each spectral material component (spectrum and modifier pair) uses only one channel, but the hardware supports four-channel textures efficiently. Four material components may thus be stored in a four channel spectrum and modifier pair. The two spectrum type combinations currently implemented are (R1,R2,R3,E) and (R1,R2,R3,T) where R is a reflection, E is an emission and T is a transmission material component type. In other words the material may be constructed from three individually spatially modulated reflection spectra and either a spatially modulated emission or a spatially modulated transmission spectrum.

A typical material component contains a modifier texture of 128x128 floating point texels and a BRDF 3D lookup of 16x32x32 floating point samples. The modifier texture thus has a size of 128*128*4= 2^{16}= 64kBytes and the BRDF a size of 16*32*32*4= 2^{16}= 64kBytes making the size of a typical material component 128kBytes. A four component material would then have a size of 4*128kBytes which is 512kBytes.

5.3 Rendering Passes

Rendering a scene may be split into multiple render passes. Each render pass would add to or modify the render target, as shown graphically in Figure 6, until the render target contains the final image. Direct mode, for example, requires only one render pass to process the scene geometry, to do the spectral lighting and to do the sensor integrals.

Buffered mode on the other hand requires two to three render passes. The first pass processes and lights the solid geometry after which the render target contains a spectral image of the solid geometry. The second pass processes, lights and blends the transparent geometry into the render target after which the render target contains a spectral image of the final scene. The spectral image may be used directly or an optional third render pass may be applied that does the sensor spectral integration.

5.3.1 *Solid Geometry Render Pass*

The spectral integral is implemented as a for-loop over the system spectral bandwidth. The spectral integral algorithm is designed such that it may be used for direct mode and buffered mode with the only modification being the wavelength sampler textures. If the wavelength sampler textures represent sensor sensitivities (one to sixteen sensors) then the renderer is in direct mode and neither a transparency nor a spectral integral pass needs to be done. The render target would already contain the spectral integral results. If buffered mode is required, then the wavelength sampler textures are special step functions designed to bin the system spectral bandwidth into the sixteen channels of the render target. A buffered solids render pass populates the frame buffer with spectral pixels and may be followed by a transparency render pass and a spectral integration render pass as required.

5.3.2 *Transparent Geometry Render Pass*

The transparent geometry render pass looks similar to the solids render pass except that the fragment processor first samples the appropriate render target pixel, then generates the new spectral pixel value and finally outputs the new blended pixel to the render target. According to the painter's algorithm a scene containing transparencies must be rendered in order from back to front, but when combined with a z-buffer this restriction is relaxed to rendering only the transparencies in order from back to front as a second render pass after the solid geometry. Doing the transparencies in a second pass thus only requires sorting of the transparencies and not all polygons.

To get access to the render target pixels the render target is also used as an input texture which is in general undefined in the stream programming model. An exception to the rule, which was discussed in the General Purpose GPU (GPGPU) programming forum [Duvenhage 2005] topic *Programmable blend unit*, is when the fragment processor's input texture pixel is the exact render target pixel that it is rendering to. To find the fragment position in the input texture the VPOS fragment processor argument(semantic) is used. Doing transparencies without making use of this special case would require an entire frame buffer blend-copy (called ping-ponging[Pharr 2005]) per transparent polygon to combine the fragment processor output with the render target.

6. RESULTS

All screen shots in this paper were done with an eye camera containing red, green and blue sensor types. The CIE's eye sensitivity table was used to construct three sensor sensitivities, one for each of the x, y, and z colour components. The CIE sensitivities are then converted to red, green and blue by means of a matrix transformation to create three spectral sensors, sampler texture shown in the top right corner of Figure 6, that map any spectrum to the screen's tri-colour phosphors according to the eye's spectral sensitivity. A more detailed discussion of modelling the eye may be found in [Chalmers et al. 2000]. The BRDF data used to create the screen shots was procedurally generated from a physically based spectral modification of the Phong model.

The measure generally used for quoting fragment processor performance is render target fill rate, measured in millions of pixels per second. The performance of the fragment processor for a specific scene may then be estimated by taking into account the frame resolution and average overdraw[Pharr 2005], measured as the number of pixels passing the hardware early culling z-test[Pharr 2005] divided by the screen area. The spectral fragment pipeline implementation was tested on an Intel® P4 3GHz and a GeForce® 6600GT and was measured to have a direct mode render target fill rate of 5.2 million pixels per second and a buffered mode fill rate of 4.6 million pixels per second. The fragment processor unfortunately seems to be texture bandwidth limited because an increase in BRDF dimension, leading to less localised texture accesses and increased cache misses, results in a decrease of the render target fill rate.

The actual fill rate figures may be multiplied by up to sixteen for the reason that sixteen sensor types (sixteen images) can be rendered simultaneously which results in fill rates of 83.2 million pixels per second and 73.6 million pixels per second for direct mode and buffered mode respectively. Taking buffered mode as an example, the fill rate is high enough to render sixteen simultaneous sensor views of a 256x256 frame at 70 frames/s for an overdraw of one or ten frames/s for a worst case overdraw of seven times in a complex scene.

7. CONCLUSION

A spectral lighting, material and camera model was developed as an expansion to the historical RGB fixed pipeline CG system. This paper showed that current desktop CG hardware is capable of brute force "physically based" spectral sampling which opens the door to developing algorithms to sample the spectral space more efficiently, but still lend themselves to efficient hardware implementation.

The spectral lighting model was derived from radiometry theory and may also be verified using radiometry theory, but it is still a local illumination model and only an approximation of the global illumination system which is reality. The below topics are proposed ideas for future research directions:

(1) How to calculate or measure BRDF data to populate this specific component BRDF model realistically,
(2) a smarter sampling strategy most suitable to hardware acceleration should be investigated,
(3) a BRDF might also be a combination of some mathematical model and a BRDF lookup, and
(4) a global illumination model suitable for efficient hardware implementation.

It was also found that a hardware programmable blend unit is required in future hardware generations to enable the fragment processor to blend its output with the render target without being limited to the RGBA hardware alpha blending. A programmable blend unit could be implemented as simply as providing a fragment program input semantic to read the current render target pixel information.

ACKNOWLEDGMENTS

I would like to thank Nelis Franken from the University of Pretoria for his guidance during my BSc(honours) project in Computer Graphics. I would also like to thank the CSIR's Optronic Sensor Systems competency area for their support and inputs. Finally I would like to express my gratitude for the valuable comments from the SAICSIT reviewers and everyone else that read draft versions of this paper.

REFERENCES

Cabral, B., Max, N., and Springmeyer, R. 1987. Bidirectional reflection functions from surface bump maps. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. ACM Press, New York, NY, USA, 273–281.

Chalmers, A., McNamara, A., Daly, S., Myszkowski, K., and Troscianko, T. 2000. *Image Quality Metrics*. ACM SIGGRAPH.

Cook, R. L. and Torrance, K. E. 1982. A reflectance model for computer graphics. *ACM Trans. Graph. 1,* 1, 7–24.

Dana, K. J., van Ginneken, B., Nayar, S. K., and Koenderink, J. J. 1999. Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18,* 1, 1–34.

Delport, J. P., le Roux, F. P. J., du Plooy, M. J. U., Theron, H. J., and Annamalai, L. 2005. Ir scene generation and missile signal injection. In *Defence & Security Symposium*. International Society for Optical Engineering, Florida, USA.

Devlin, K., Chalmers, A., Wilkie, A., and Purgathofer, W. 2002. Tone Reproduction and Physically Based Spectral Rendering. In *STAR Proceedings of Eurographics 2002*. Eurographics Association, Geneva, Switzerland.

Duvenhage, B. 2005. Programmable blend unit. www.gpgpu.org Forum, Hardware Features.

Evans, G. F. and McCool, M. D. 1999. Stratified wavelength clusters for efficient spectral monte carlo rendering. In *Graphics Interface*. 42–49.

Fernando, R. 2004. *GPU Gems*. Addison-Wesley, USA.

Fernando, R. and Kilgard, M. J. 2003. *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Addison-Wesley, USA.

Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. 1997. *Computer Graphics: Principles and Practice*, Second ed. Addison-Wesley, USA.

Gondek, J. S., Meyer, G. W., and Newman, J. G. 1994. Wavelength dependent reflectance functions. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM Press, New York, NY, USA, 213–220.

Kolb, C., Mitchell, D., and Hanrahan, P. 1995. A realistic camera model for computer graphics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM Press, New York, NY, USA, 317–324.

le Roux, F. P. J. 2001. An investigation into the use of opengl as a library for the simulation of infrared scenarios. M.S. thesis, The University of Pretoria.

Lorenzo, M., Jacobs, E., Moulton, R., and Liu, J. 1999. Optimised mapping of radiometric quantities into opengl. In *SPIE Conference on Modeling, Simulation and Visualization for Real and Virtual Environments*. Vol. 3694. 173–182.

Pharr, M. 2005. *GPU Gems 2*. Addison-Wesley, Massachusetts, USA.

Phong, B. T. 1975. Illumination for computer generated pictures. *Commun. ACM 18,* 6, 311–317.

Westin, S. H., Li, H., and Torrence, K. E. 2004. A comparison of four brdf models. Research Note PCG-04-02, Cornell University Program of Computer Graphics.

Williers, C. J. 2003. *Electro-Optical System Design*. Wiltru Trust, Brummeria, South Africa.