GPU Based Pathtracer with physically accurate features

Gundeep Singh*
University of Pennsylvania

Abstract

As we know ray tracing and path tracing are both embarrassingly parallel problems. So taking advantage of their parallelism i implemented a path tracer on GPU. Parallel path tracing is not something novel but with project the intent was to implement a simple path tracer and then add extended features like the subsurface scattering, BRDF models, Depth of Field and Motion Blur etc. to get more photorealistic results.

Keywords: Subsurface scattering, global illumination, cook Torrance, BRDF's, Fresnel

Links: DL PDF

1 Introduction

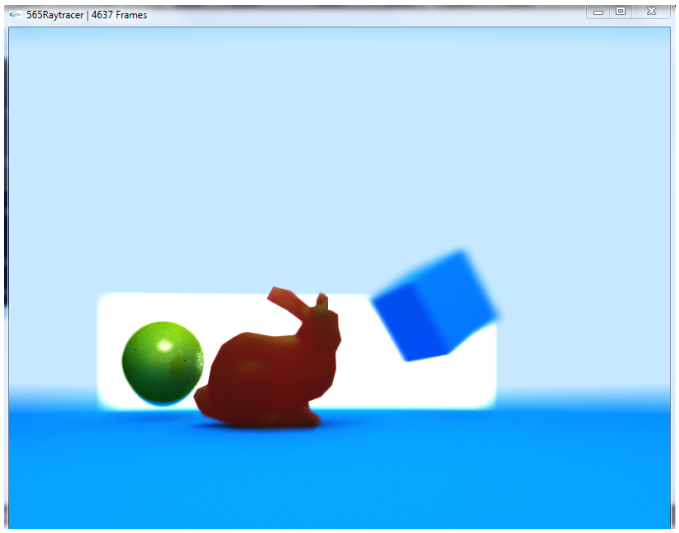

Figure 1: *Result from my path tracer.*

The Rendering Equation as first presented in [**?**] described all the illumination at any given point in the scene via and Integral including all the reflections in the scene. Path tracing is since then considered as a stochastic method to solve the rendering equation. Path tracing does require a lot more samples than Ray tracing but delivers an unbiased result.

*e-mail:gundeep@seas.upenn.edu

In this paper i present a novel approach to implement a Path Tracer and additional features that i added to the basic Path Tracer, as part of my final project.

2 Related Work

To generate images that are close to reality has become more and more important in several different applications.Path Tracing was introduced then as an algorithm to find a numerical solution to the integral of the rendering equation. However,the efficiency ofMonte Carlo methods is still the main concern when applying it in practice. A decade later, Lafortune suggested many refinements, including bidirectional path tracing.

Other techniques cache and interpolate portions of the light transport that are similar between pixels, reducing variance at the expense of biasing the solution. Irradiance caching [Ward et al. 1988], density estimation [Shirley et al. 1995] and photon mapping [Jensen 1996] all take this approach.

3 Details

The basic path tracer implementation was done as a part of my class project with a basic little base code available. Here the main emphasis was to take the path tracer i had and exploit its unbiased nature to add some more physically accurate features to it.

Exploiting the great compute that the recent GPU's have to offer i started of by creating a Ray tracer. The ray tracer simply just accounted for reflections shadows and point lights and from there i implemented the path tracing algorithm using the Monte Carlo's Solution.

The implementation of this project was done in C++ using the CUDA. It offers great online support and is constantly improving including the important features available in CPU programming.I used "Thrust" Library for stream compaction and some few other tasks on GPU and "GLM" library for other Vector and Matrix operations.

After finishing up the basic path tracing i focussed my attention to the more visually appealing features like motion blur, depth of field, subsurface scattering, Cook Torrance Brdf etc. each of which is explained later-:

3.1 Path Tracing

Here i start of by shooting rays into the scene and jitter around pixel for antialiansing. On hitting the diffuse surface, i generate a random rays based on a cosine distribution around a hemisphere. If the ray hits a specular surface it reflects back and keeps bouncing in the scene until they hit a light source or the background.

Simple path tracing runs fairly quick on the GPU giving a speed of roughly 15 iterations per second on my NVIDIA GTX660. and the image converges in roughly 500-600 iterations (depending upon the scene).

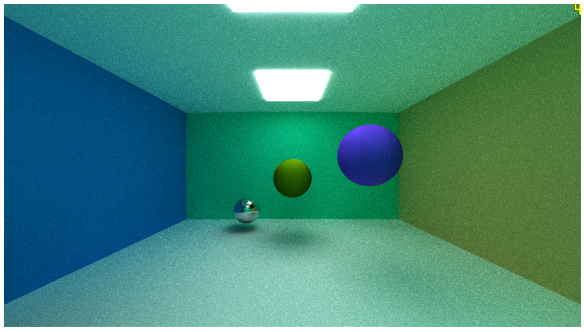

Figure 2: *Simple path tracer*

3.2 Obj Mesh loader

Here i added a mesh loader to my path tracer so that i can load arbitrary shapes. The mesh loader is simply .Obj file parser which stores the vertex and face and normal information for the geometry. Results:

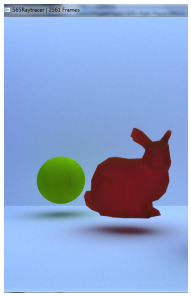

Figure 3: *Mesh loader in action.*

3.3 Fresnel Reflections

When light strikes a material boundary, for example, crossing from air into glass, only some of the light transmits into the new material; some light reflects at the boundary. The name for this effect is Fresnel reflection. Fresnel reflection is most visible when viewing semi-transparent material such as water, window-glass, skin, or car-paint. Fresnel reflection also occurs when viewing opaque materials such as metal or paper.

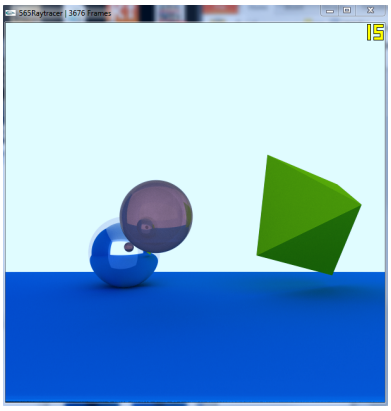

Figure 4: *Fresnel Reflections.*

3.4 Subsurface Scattering

Subsurface scattering (or SSS) is a mechanism of light transport in which light penetrates the surface of a translucent object, is scattered by interacting with the material, and exits the surface at a different point. The light will generally penetrate the surface and be reflected a number of times at irregular angles inside the material, before passing back out of the material at an angle other than the angle it would have if it had been reflected directly off the surface. Here besides scattering of light i am also considering the absorption of light when it travels from one medium to another. This makes the results look more pretty.

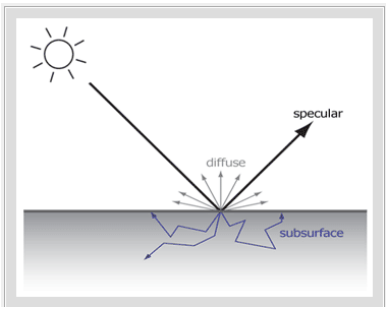

Figure 5: *Single Scattering.*

This is my most favorite effects and i wanted to do it nicely. Below is an image showing single scattering.

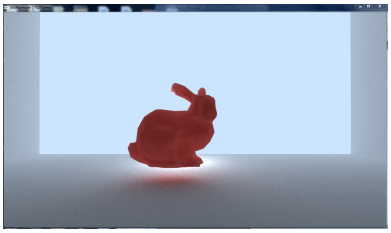

Figure 6: *SSS in my path tracer.*

3.5 Depth of Field

Earlier we are using pinhole camera now to implement depth of field we assume some size of camera. Simple and most important concept for DOF is to jitter the location of ray start (i.e. camera position) and construct a ray using that camera position.

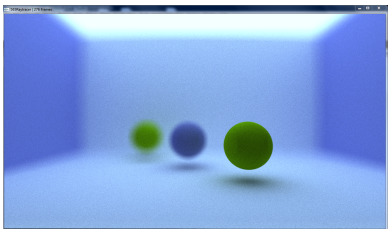

Figure 7: *Depth of Field*

3.6 Motion Blur

The motion blur effects is trivial to implement in a path tracer due to its recursive nature. The easiest way to do the same is to move

the object in the scene over the iterations and keep accumulating the results, it gives you very realistic motion blur result.

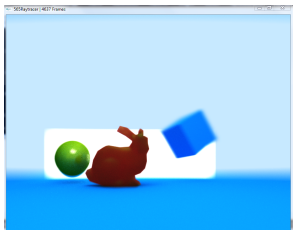

Figure 8: *Motion Blur*

3.7 Cook Torrance BRDF

What makes the Cook-Torrance shading model so favorable? First, it computes its specular lobe using the half-angle vector Second, it accurately models the Fresnel effect, increasing the specular reflectivity as the surface turns away from the viewing direction. Finally, it models micro imperfections in the surface.

$$k_{spec} = \frac{DFG}{4(E.N)(N.L)} \tag{1}$$

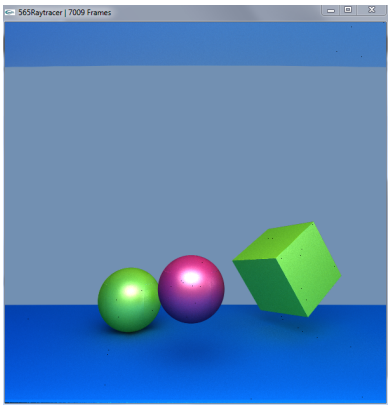

Figure 9: *Gauss Constant= 100, Roughness=0.2*

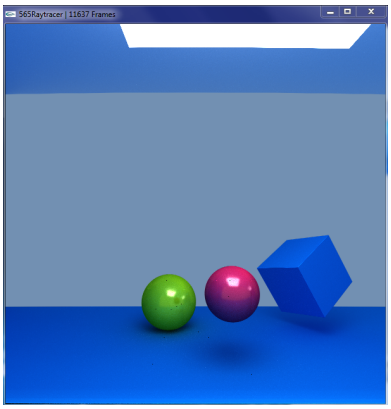

Figure 10: *Gauss Constant= 50, Reduction coeficient=0.02*

3.8 Post processing

Just for experimental purpose i also added some post processing effects in the fragment shader. These effects were trivial to implement but gave cool effects. For now i had the contrast and the and grayscale filter post processed but i can definitely add more features like Screen space ambient occlusion(SSAO) etc.

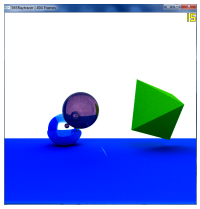

Figure 11: *Contrast*

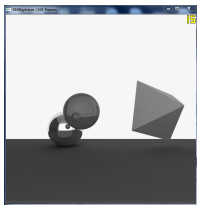

Figure 12: *GrayScale*

4 Future Work

There is still a lot of scope in terms of features that my renderer can have. I eventually want to make it a complete robust renderer with all possible features that a modern renderer has. Most importantly i would like to integrate an KD tree based intersection test, in order to load more complex geometry and also texture mapping would be an important next step.

5 Conclusion

I was able to accomplish my task of adding physically accurate features in my path tracer and get more realistic results. Right now it is not a very high performance solution, some parts can be optimized further to take the advantage of GPU parallelism.

Acknowledgements

To Patrick cozzi, for sharing excellent knowledge on CUDA programming and overall GPU programming and Karl for providing the basecode for path tracing.

References

The Reatlime Rendering book, chapter 7 p223-251.

The Realtime Rendering book, chapter 10 p486-490.

Writing a Cook Torrance surface shader, Pixar Animations.

Subsurface scattering, renderman.pixar.com and wikipedia.

Peter and karl's blog, http://gpupathtracer.blogspot.com/