Exploring New Approaches to Interactive Rendering Utilizing Pixel Synchronization

Bryan Pawlowski, Professor Mike Bailey
Oregon State University

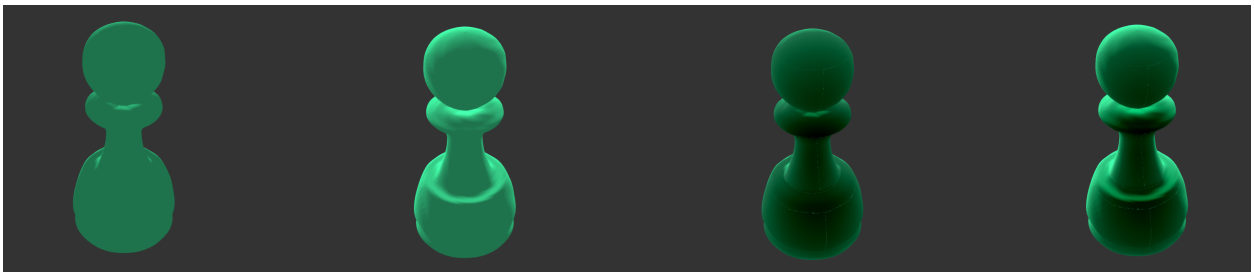

Figure 1: *From left to right, Phong Smooth Shading, Phong with Normal Wrapping, Phong with Pixel Sync Depth Map, then Phong with a combination of Smooth Shading and Pixel Sync Depth Map.*

Introduction

In the early stages of graphics programming, we had a very basic job to do in the eyes of the GPU. We were to give the GPU information on the 3D object, set various contexts and modes, then told it to draw. Once these fixed-function GPUs began to draw, our influence on the GPU was over. Later on, we were given the ability to program the GPU's pipeline stages; starting first with a programmable vertex and fragment/pixel shader, then getting increasingly complex. The advent of programmable shaders gave graphics programmers the chance to unlock the GPU's potential to render a large number of what the programmer could think of. However, there remained the issue that we couldn't program the GPU to take advantage of any types of syncronization constructs. Since the main advantage of a GPU is to produce spectacular images quickly, it makes sense why synchronization was never really factored into a Graphics Processor's architecture, until now.

With the advent of hardware support of Pixel Synchronization, programmers are given the ability to put a barrier into shader code which orders pixels being processed. Given this new capability, we are now able to build knowledge of entire objects with fewer passes, when coupled with the use of Unordered Access Views. Since UAVs are shared between all pixels in flight, a programmer used to have no way of guaranteeing against data races. With the addition of Pixel Synchronization, we are now able to guarantee that parts of the UAV will be accessed in a deterministic order. When pixel ordering is invoked within the pixel shader, a barrier based on screen space is created. What this means is that each pixel in flight at the same screen position hits a barrier, is ordered based upon primitive, then executed in-order.

My Work

The main purpose of my work is to answer the question of how this Pixel Synchronization hardware capability is can be utilized to enhance existing render techniques. This work is important to the exploration of this hardware, as other GPU companies such as nVidia, are introducing similar hardware into their architectures.

Subsurface Scattering Approximation

Using a two-pass method, a depth map of the object is created with respect to the light source. Using this depth info, on the third and final pass of this rendering technique, we modify the diffuse color based on the depth of the model with respect to the light. I compare a few characteristics of the performance of different algorithms, including regular Phong smoothing, Diffuse Wrapping[1], the Pixel Ordering Subsurface scattering approximation, a combination of the diffuse wrapping and the Pixel Ordering technique, and a smarter utilization of the Pixel Ordering technique.

Accurate Complex Model Refraction

For this application, I use pixel ordering to create a voxelization of the model, then trace through the interior of the model based on the vector returned from the *refract* intrinsic function. Being able to traverse through to the other side of a model gives a more accurate refraction effect than just using the front side of the model. In the talk, a characterization will be made between the normal refraction approach and the one with improved quality using Pixel Synchronization. Timing benchmarks will also be presented.

[1] Randima Fernando. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Addison-Wesley Professional, Apr. 2004. Chap. 16.