Assignment 3 – Ray Tracing (Part 2 – Triangles & Recursion)
Due: October 23rd

**Getting Started:**

For this assignment you will be making a full-featured ray tracer. You can find several sample scenes posted on the course website as sample files.

**Requirements:**

Your ray tracer should ideally build off your submission to HW2. It will be worth 85/100 points to get these basics features:

- Arbitrary camera placement, film resolution, and aspect ratio
- Arbitrary scenes with spheres, triangles (possible with vertex normals), and arbitrary background colors
- Arbitrary materials, including diffuse and specular shading, reflections, and refractions
- Point and directional lights
- Ambient lighting
- Shadows
- Recursion to a bounded depth

To get the full 100 points you need to add additional features from the list below (ask me if you have something in mind not on the list). Its okay if you need to extend the scene format in some way, just make sure you include a sample scene that shows off your new, cool features!

**Additional Features:**

The number in front is how many points a feature is worth. Their will be partial credit for features that "sort of" work.

*Scene specifications / Primitives:*
(5) Cones and Cylinders
(5) Boxes and Planes
(5) Constructive Solid Geometry (union, difference, and intersection of primitives)
(10) Transformations on primitives (support 4×4 transformations or procedural ones! http://bit.ly/1vO4tnl)
(20) Procedurally generated terrain/heightfields

*Complex Lighting*
(5) Area lights that produce soft shadows
(10) Ambient Occlusion
(20) Image-based lighting (e.g., environment maps)

*Sampling*
(5) Jittered supersampling
(5) Adaptive supersampling
(5) Motion Blur
(5) Depth of Field
(15) Physically-based camera lens simulation (match the assortment of lens commonly found in real cameras, e.g., https://en.wikipedia.org/wiki/Zoom_lens)

*Materials*
(5) Texture mapping
(5) Bump mapping
(5) Procedural texturing or bump mapping (checkerboard, wood, marble, mandelbrot set, etc..)

*Miscellaneous*
(5) User interface that shows the raytraced image being updated
(10) An acceleration structure: BVH, OctTree, etc. (measure the performance impact on different scenes!)
(15) Parallelize the raytracer (and analyze the performance gains as you add more processors!)
(50) Real-time SIMD/GPU Implementation using e.g., CUDA - http://bit.ly/1EplJpd

**Submission Instructions:**

Create a sample webpage with:

- A zip file of all your code
- Several output images from your raytracer (including at least one of the sample scenes)
- Brief description of your implementation, any issues you saw, and a list of any extra credit tasks you attempted
- Submission to Art Contest (optional)

**Hints:**

- You should be able to leverage your previous raytracer. Any extra credit work you did there should roll easily into this assignment. Try to be strategic about what extra features you choose to implement.
- A BVH is an easy way to render large, impressive scenes in a reasonable amount of time
- If your running out of time focus on the easier options such as boxes, planes, CSG and supersampling. That way you get some easy things completed well rather than rushing through something you don't have time to finish.