CSC 473
Lab Assignment 1
Due: Tuesday, April 9th

Overview
In this assignment you will implement the initial stage of your ray tracer, recall that there are four sections to your ray tracer:

- Parsing the scene description file.
- Computing ray-object intersections
- Shading
- Recursive Tracing (reflection, refraction and shadows)

For today's lab, you will start by developing the abstract object(s) to represent geometry in your scene and write the parsing code to read in the scene description file (see below). Please read this entire assignment before you start.

Scene Description Language
The scene description language that we will be using will be based (loosely) on a subset of the Povray format.
The big difference is that we will be using a right handed coordinate system for our world, object and camera coordinates.

Your raytracer must be able to parse the following types:

- // comments
- camera
- light_source
- translate, scale, rotate
- box
- sphere
- cone
- plane
- triangle
- pigment
    - color
        - rgb
        - rgbf
- finish
    - ambient
    - diffuse
    - specular
    - roughness
    - reflection
    - refraction

- ior

You should create an abstract object from which all of the ray tracer objects will be derived. Each derived object should have its own parse function (read) that takes the filestream in, processes the data for that object, and returns the altered file pointer. (Unless you are developing for CUDA in which case you can organize your objects how you see fit).

Your program should have the following syntax:
raytrace [options]
where the options are:
imageWidth
imageHeight
-I inputFilename.pov
So raytrace 640 480 -Isample.pov will render a 640x480 image file, sample.ppm, consisting of dummy information at this point.
Image files should be output named inputFilename.ppm or .tga or whatever image format you are using.

Deliverables: You will need to demo that you have working code in place to parse the povray file format.
If you are developing with an OO focus, you will need to provide a UML class diagram (showing classes and relationships). If you are developing for CUDA, you will need to show that your code compiles with nvcc (with objects passed to the GPU with no kernel processing on them).