# CSC 473
# Program 3 – reflection, refraction and triangles

# Due Sunday May 1, 2013 at 11:55pm

Overview:
Throughout this quarter you will implement the basic functions of a distributed ray tracer. *This software will be enhanced throughout the quarter, thus this second assignment builds on your prior code and will serve as the base for following assignments*. Since you will be adding and reusing your code, it is advised that you write your code in a clean, structured object-oriented fashion. All code must be written in C++.

**Goal for assignment 3:**
In this assignment you will implement reflection and refraction on mirrored and translucent surfaces. You will also need to compute intersections with triangles. Please work on these features incrementally using the appropriate files. For example:

- start with adding parsing and intersections for triangles and use the file simple_tri.pov to test your triangle intersections.
- add reflections and test with simple_reflect1.pov (reflective plane) and simple_reflect2.pov (reflective spheres) and simple_reflect3.pov ( spheres and planes)
- add refractions (use simple_refract.pov) <and additional refraction file tests to be determined>
- test them all with recurse_simp2.pov <and additional files to be determined>

Reflection and refraction should be implemented via recursion. Limit the recursion to 5 bounces. You will be building on to your previous code, and thus, your program should include shading and shadows.

---

What you should hand in via polylearn:

- Your code, include all files necessary to compile and run your ray tracer, including a Makefile or cmakeLists.txt
- A README.txt file with any information about what is working or not working with your implementation to assist the grader in determining what is causing potential errors in your output and help in assigning partial credit.
- ***Your own .pov file and rendered image of a reflective and refractive scene. Be creative and create an interesting scene. Choose colors and an arrangement of geometry that you find pleasing.***

You need to handin your code and images generated using poly learn. Look for the assignment directory.

**Grading breakdown:**
27 points working triangle intersections
27 points working reflection
37 points working refraction
9 general sanity

---

**Program execution:**
Your program should have the following syntax:
**raytrace <width> <height> <input_filename> <BRDF>**
where the options are:
width = the image width
height = the image height
input_filename = the name of the povray file to read and render
BRDF: value of 0 or null uses Blinn-Phong, values of 1 use your alternative BRDF
Thus:
**raytrace 640 480 sample.pov**
or
**raytrace 640 480 sample.pov 0**

will render a 640x480 image file, "sample.tga" consisting of the scene defined in "sample.pov" using the Blinn-Phong BRDF for shading.

Sample input files and images are given on the class webpage. Some of these pictures may be generated by Povray and will not look identical to your output (due to differences in the shading model, etc.).