CSC 473
Program 4 – geometric transforms, anti-aliasing and spatial data structures
(revise into two parts A & B)

Part A Due Sunday May 14th, at 11:55pm
Part B Due Thursday May 18th, 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++.

Goals for assignment 4:
For this portion of your ray tracer, your program needs to:

Part A:

- handle all previous specifications with modifications to the way reflection and refraction are weighted by using the Schlick approximation to weight reflection and refraction
- handle geometric transforms on all geometric types
- handle multiple lights
- use anti-aliasing (9 stratified super samples per pixel) *<be able to turn this on and off with command line>*

Part B:

- compute intersections and appropriate shading for boxes
- handle large files (e.g. balls.pov and bunny.pov) in reasonable times by including either a bounding volume hierarchy (BVH), binary space partitioning tree (BSP tree) or an oct-tree (ie a spatial data structure) in your code to optimize ray intersection testing

Example files and results will be posted and announced on email. You will need to create your own visually interesting scene that you render and submit the .pov file.

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 an interesting scene***

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

Grading breakdown (will be applied to each part appropriately):
25 spatial data structure
20 handle geometric transforms
15 anti-aliasing *<note that anti-aliasing should be able to be turned on an off via command line see below>*
15 points working box intersections
10 Schlick for weighting reflection and refraction
5 multiple lights
10 general sanity

For anti-aliasing, please assume that we will now default to one BRDF (Blinn-Phong), thus change your command line arguments such that the 4th command line argument could be used to turn on and off anti-aliasing. Please use 1 to enable anti-aliasing and 0 to not use anti-aliasing. For example:

Your program should have the following syntax:
raytrace <width> <height> <input_filename> <anti-aliasing on or off>
where the options are:
width = the image width
height = the image height
input_filename = the name of the povray file to read and render uses anti-alising
Thus:
raytrace 640 480 sample.pov 1

will render a 640x480 image file, "sample.tga" consisting of the scene defined in "sample.pov" using the Blinn-Phong *and anti-aliasing!*

For Part B, to get full credit for your spatial data structure - your code just has to be as fast or faster then my very old code with still many printfs:

times are:
gnarly: 44 seconds

balls2: 31 seconds

Image size is 640 x 480 and I did _not_ have anti-aliasing in place for those timings.

Don't worry if you are slower then these, you can still get credit for your spatial data structure, if you can show that your raytracer performance is enhanced by at least 20% using a spatial data structure.

Program execution – same as before:
Your program should have the following syntax:
raytrace <width> <height> <input_filename> <BRDF>

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.).