Project 3: part a, RAY-CASTER: Out of 50 possible points. Due 10:59AM Thursday Nov 1; You get BONUS if you turn in by 11:59pm Tuesday Oct 30.

Purpose: is to understand how ray casting and initial ray tracing work, as well as being able to implement the concept and pixmaps in practice.

Overall: Your project should have the following elements within a 3D world coordinate system:

1) Environment: Your program should assume that the camera is located one meter from the origin at (0, 0, 1) and is aimed along the negative z axis. Assume that the view screen (window size 640 by 480) is centered at the origin, parallel with the x-y plane, and with sides parallel to the x and y axes. The width of the viewscreen is 0.5 m, its viewing angle is 30 degrees and its aspect ratio is 5/4. Implement your object list such that you have the following transformed objects: a red sphere with radius 0.125 m at (0.125, -0.25, - 1), a green sphere of radius 0.375 meters at (0.5, 0.5, -1.75) and a blue sphere of radius 0.75 m at (-0.5, 0, -2.5).

2) Implement RayCast class. Follow pseudo-code from lecture slides. Implement the part that loops through each pixel row and column and computes the start of each rcth ray and rcth direction based on perspective projection.

3) Using the start point and direction, implement the component of the ray class that loops through your object list, determines if there is a hit, and identifies the lowest hit time and corresponding object index that was hit for that rcth ray. To do this follow the pseudo code for finding the implicit form of a sphere, compute the inverse transformed ray, and determine if this new ray intersects with the implicit form of the generic sphere. If it does intersect with this object, then you will need to color the pixel the same color as that object has stored. Hint: Don’t forget to use the shortcuts for determining if there is no intersection at all, if there is only one intersection, and if there is 2 intersections, so you do not have to compute the entire formula each time. There will not be a need compute the 3d hit point at this time, only hit time to determine the closest object and hit intersection.

4) Use a pixmap (ie glDrawPixels and other functions that are used with this) to set the color of each rcth pixel to be the color of the object that was hit or the background color. If you use glRecti, you will gain half credit but not full credit for element 4.

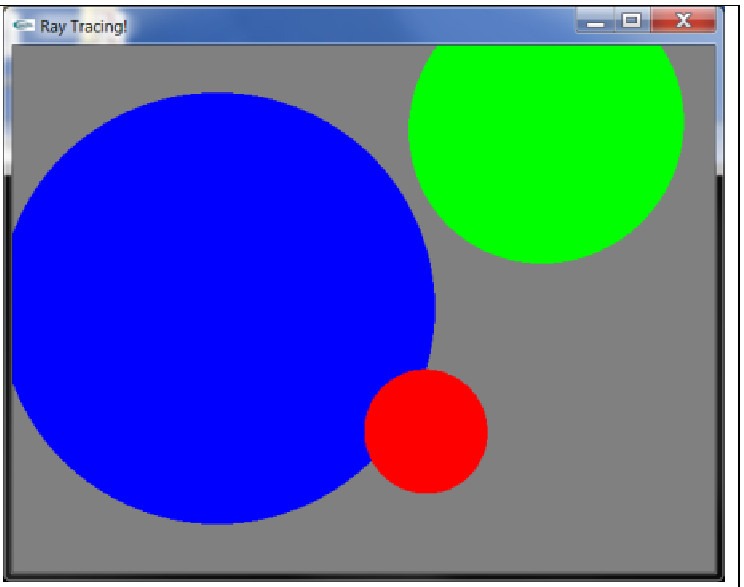
Additional Tasks for 5450:

5) Add in your object list: 1 transformed plane or 1 transformed cylinder.

Choose the size, orientation and location so they are somewhere in the viewing area (perform at least 1+ transformations (remember you are using matrices not OpenGL for these transformations). They can be behind or in front of other objects. As a result of adding these objects to your list, you have to ADD to your code to determine intersection of a generic plane and a generic cylinder.

REMEMBER: You should not be using OpenGL at all for transforming objects, defining objects, or displaying objects (other than pixmaps).

BONUS for both 4550 and 5450 (2): Allow for a dynamic adjustment of resolution for your pixmap. Hint: blocksize affects this. You will be graded on elegance of code and the completion of implementation details for elements specified. Elegant code should include programming concepts and formation of classes and functions that you have learned in Data Structures and Algorithms course.

Point values are distributed among the categories as follows: 4450: Element 1 – 5 points Element 2 – 15 points Element 3- 20 points Element 4- 10 points 5450: Element 1 – 5 points Element 2 – 10 points Element 3- 15 points Element 4- 10 points Element 5- 10 points	 Figure 1: Result from tasks 1-4

Start Early! Ask Questions! Don't forget to comment your code!