**CMSC427 Fall 2020**
**Lab 1 – Ray tracing exercises**

Due by midnight Thursday, Nov. 12th

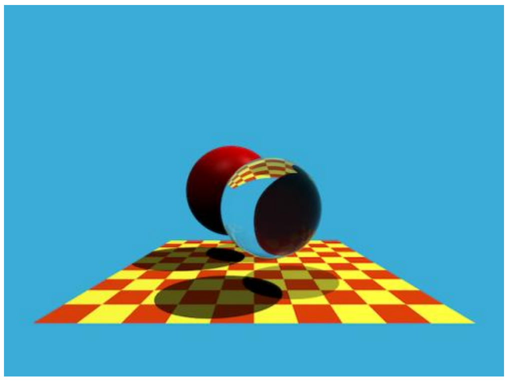
***Objectives of lab:***

- Get started with ray tracing
- Work with a procedural texture
- Use a basic form of antialiasing

***Requirements:***
This lab will start with the file whitted.cpp from the web site Scratchpixel.com.

https://www.scratchapixel.com/code.php?id=8&origin=/lessons/3d-basic-rendering/ray-tracing-overview

You should start by downloading and running it using the instructions in the file itself. The file says it saves in png format, but it saves in .ppm (Portable Bitmap Format). That has image dimensions in plain text and the pixels in binary format. You should end up with an "out.ppm" image like this:



This labs works through exercises to understand and modify the code. As you work through these exercises, keep a short record of the changes you made.

**Step 1: Play with the scene options.** These are all set in main().

A. Double the size of the image to make it easier to see. The aliasing may be less apparent but is still there.

B. Expand the objects in the image by finding the field of view. It starts at 90 degrees, change it to 70 or so.

C. Change the color of the background and the chessboard.

D. Set the maxDepth variable to different values to see what happens.

Write all these up, with a particular addition to describing what happened with (D). What is bias for?

**Step 2: Play with the diffuse object material (type DIFFUSE_AND_GLOSSY).** The color is set in main(), surface properties set in class Object and applied to the first sphere (sph1 in main()).

A. Change the color of sphere 1.

B. Modify the other diffuse properties so you can control the material. See class Object and this line.

```
materialType(DIFFUSE_AND_GLOSSY),
        ior(1.3), Kd(0.8), Ks(0.2), diffuseColor(0.2), specularExponent(25) {}
```

See if you can change this for sphere 1 with new values, and make them part of the constructor.

C. Practice moving and resizing the sphere so you understand the coordinate system.

Write these up.

**Step 3: Play the transparent object material (type REFLECTION_AND_REFRACTION).** This is sphere 2 (sph2 in main()).

A. In main() the property ior is set to 1.5. Figure out what that is, and practice changing it. Note we only have air to object interfaces and the code follows that.

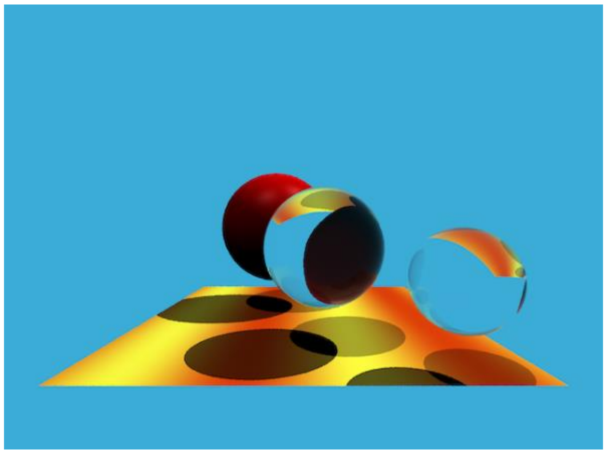
**Step 4: Play with the colors and pattern for the checkboard plane.** The plane is actually only two triangles and the color (texture) is set as a *procedural pattern*.

A. Change the colors – pick ones you like.

B. Find the function evalDiffuseColor. That takes the x,y coordinates of the intersection and computes the variable `pattern` as percentage 0 to 1 for mixing the two checkerboard colors. First change the scale variable and see what happens.

C. Now try changing the pattern equation to something other than a checkerboard – a smooth gradient, a sine or cosine gradient, any that works. Just need to set `pattern` to a value You can compute any texture as long as `pattern` stays in the range 0 to 1.

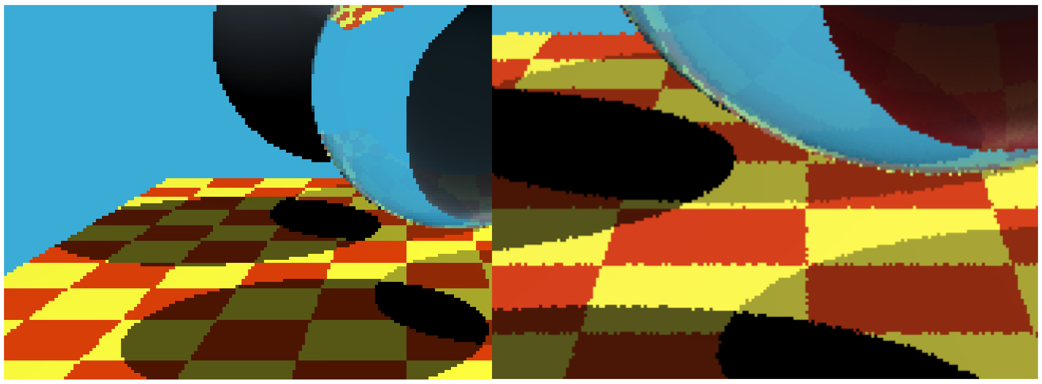
A procedural pattern is an alternative to a stored image texture – rather than retrieve it, you compute it on the fly.



Report on the pattern you created. The example here uses the sine function with an additional sphere.

**Step 5: Add antialiasing.** This isn't too hard as long as you can manage the C++ code. For each pixel you'll be taking multiple samples and averaging them.

The starting code gives systematic aliasing along edges of shapes – see the left image below.



A. First randomize the position of the sample inside the pixel area. The original code sets the sample in the middle at (0.5,0.5) (offset from 0,0). Random choose x and y displacements in range 0 to 1.You should get something with random aliasing artifacts like the right image above.

B. Now add a loop to sample at random positions N times and average the resulting color. Hint: you may want to add to the Vec3 type an overloaded divide operator (/). (I did this lab in an hour but it took an extra hour to get the C++ code right for

Play with N to understand the tradeoff between time and quality of result. Give an N you believe works for reasonable anti-aliasing. Report on your results.

## Submission

On Elms submit your report and your final code. Your final report should be at least a page, but need not be wordy.

You can make other modifications if you'd like – move the lights, move the shapes, add shapes, and so on. In the next lab we will extend the code to add more patterns and implicit shapes.

Links for procedural texture:
https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/procedural-texturing
http://www.upvector.com/?section=Tutorials&subsection=Intro%20to%20Procedural%20Textures
https://www.mdpi.com/1424-8220/20/4/1135/pdf