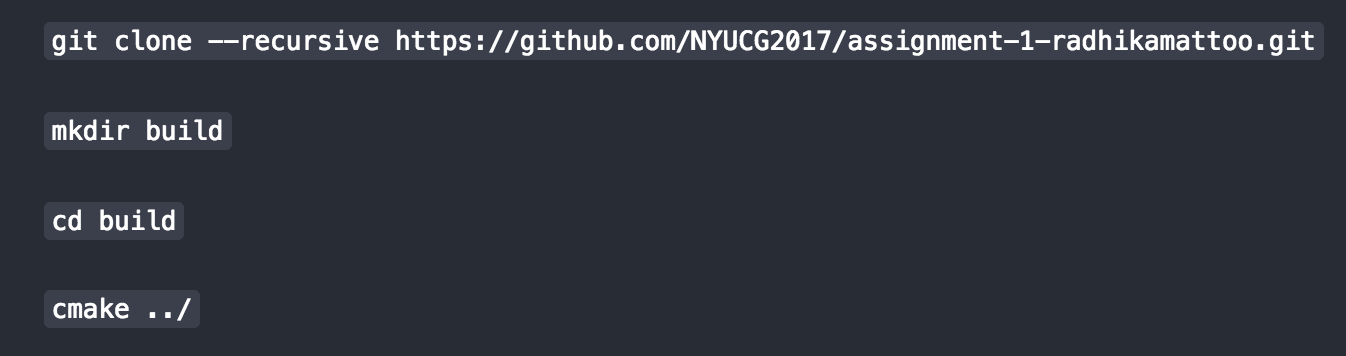
**Ray Tracing**

Radhika Mattoo, rm3485@nyu.edu

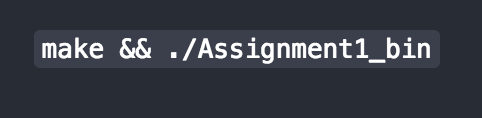
**Overview**

* I have split up some portions of the homework (sections 1.1 to 1.3) into separate functions in my *main.cpp* file, and concatenated sections 1.4 - 1.6 into one function.
* General setup & running instructions are below. Instructions for running a specific section of the code will be detailed in each section.
* The images created by when running my code are saved to the *build/* directory.
* For comparison, the images presented in this file are located in the *screenshots/* directory.

**Setup**



**Running**

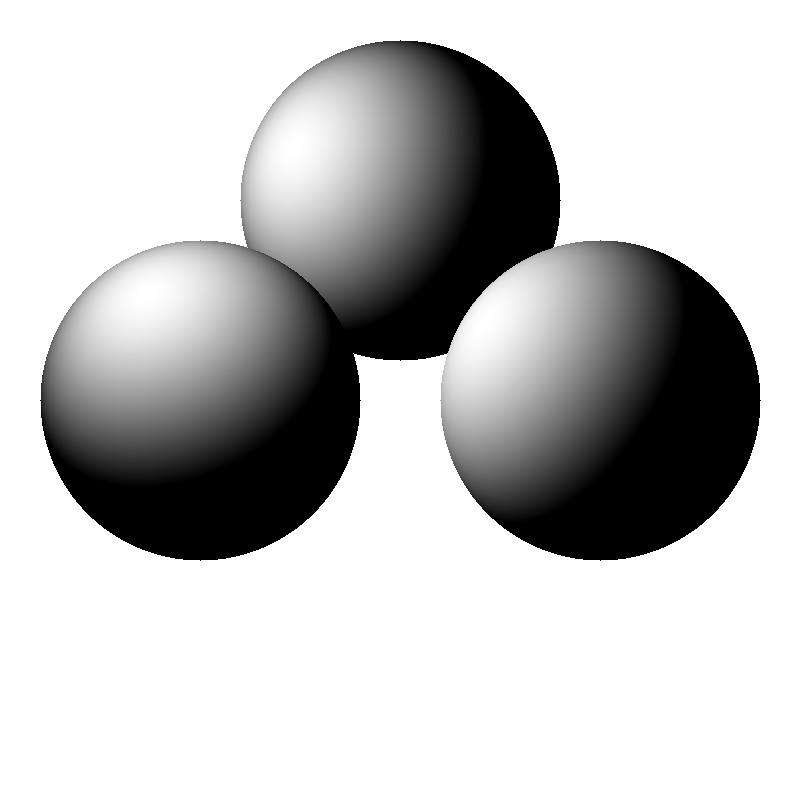


**1.1 Ray Tracing Spheres**

* This code is in the function called *part1()*
* I decided to render 3 spheres with *radius = 0.4* located at:
  + (-0.5, 0, 0)
  + (0.5, 0, 0)
  + (0, 0.5, 0)

with the light positioned at:

* (-1, 1, 1)
* Now you can see how the position of a given sphere affects its diffuse lighting
* To run this code, uncomment the call to *part1()* in *main()* at the bottom of the *main.cpp* file and in the *build/* directory run:
  + *make && ./Assignment1\_bin*
* The output image will be saved to the *build/* directory as *part1.png*

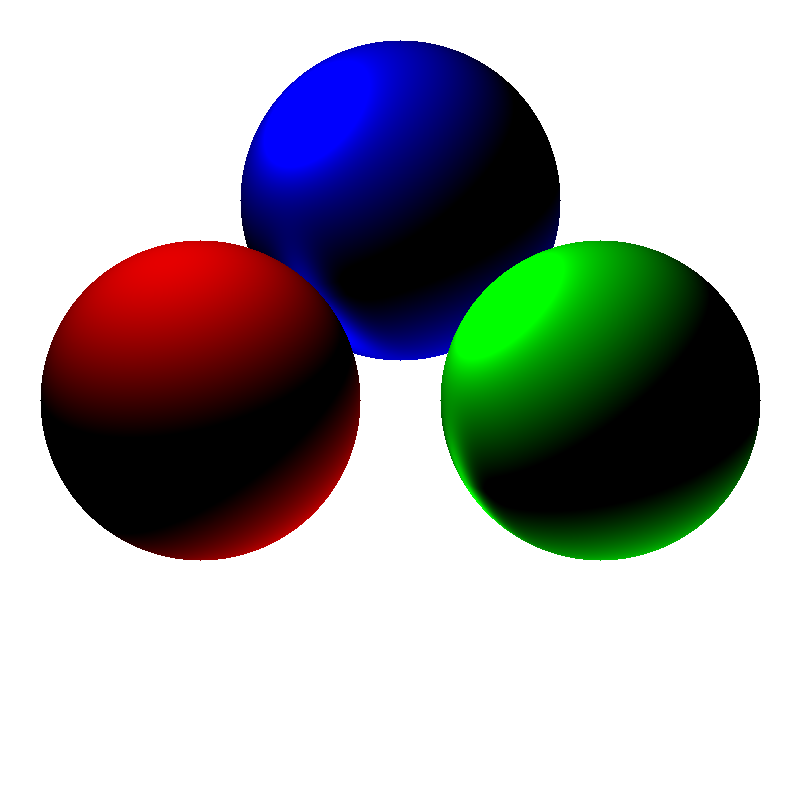


**1.2 Shading**

* This code is in the function called *part2()*
* I decided to render 3 spheres with *radius = 0.4* located at:
  + (-0.5, 0, 0)
  + (0.5, 0, 0)
  + (0, 0.5, 0)

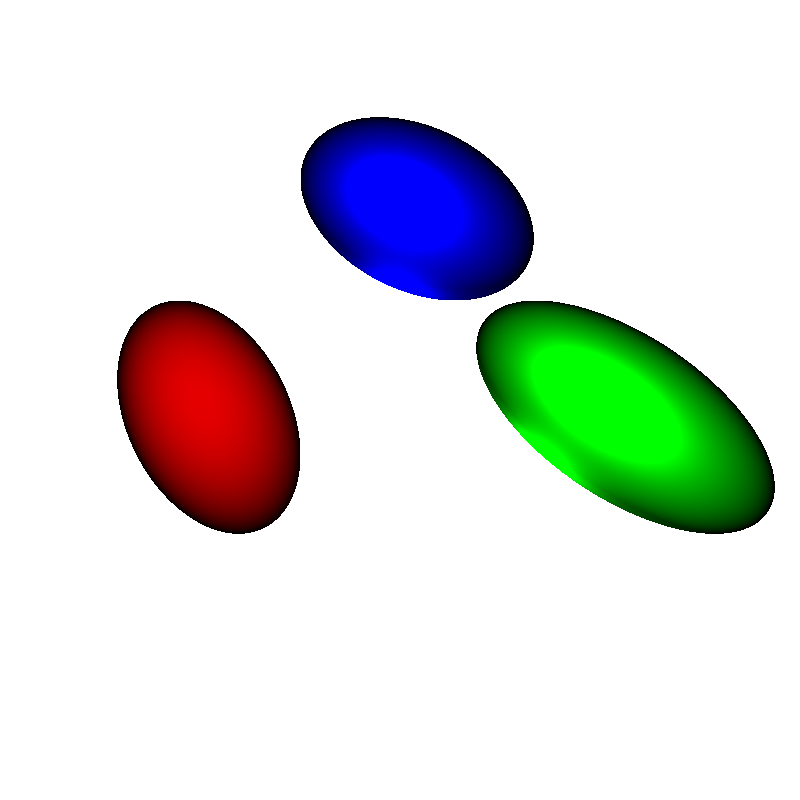
with 2 light sources positioned at:

* (-1, 2, 1)
* (1, -2, -1)
* I set the pixel value for each sphere in its own color matrix to render them as separate colors.
* The left red sphere has only diffuse lighting.
* The middle blue sphere and right green sphere have specular lighting, with the *Phong exponent* set to *10*, and *ambient lighting* set to *0.01*.
* To run this code, uncomment the call to *part2()* in *main()* at the bottom of the *main.cpp* file and in the *build/* directory run:
  + *make && ./Assignment1\_bin*
* The output image will be saved to the *build/* directory as *part2.png*



**1.3 Perspective Projection**

* This code is in the function called *part3()*
* The spheres and lighting setup is exactly the same as part 1.2
* The only difference with perspective projection in ray casting is how you construct the ray; thus, I simply changed the ray construction within my nested for loop, and the rest of the code stayed the same.
* I used a *focal length* of *1.0*
* The output image makes sense, as perspective projection renders spheres as ellipses.
* To run this code, uncomment the call to *part3()* in *main()* at the bottom of the *main.cpp* file and in the *build/* directory run:
  + *make && ./Assignment1\_bin*
* The output image will be saved to the *build/* directory as *part3.png*



**1.4 Ray Tracing Triangle Meshes**

* This code is in the function called *part4()*
* The only difference in the ray tracing algorithm from previous sections is how we determine an intersection; instead of using a discriminant, we use the vertices of the given triangle along with the ray data to construct matrix *A* and vector *b*, solve for *A x = b*, where *x* is a vector that determines whether an intersection has occurred. After this, the shading is the same as before.
* To run this code, uncomment the call to *part4(false, false)* in *main()* at the bottom of the *main.cpp* file and in the *build/* directory run:
  + *make && ./Assignment1\_bin*
* The output image will be saved to the *build/* directory as *part4.png*



**1.5 Shadows**

* This code is in the function called *part4()*
* When an intersection with one of the meshes occurs, there’s a possibility that it is in the shadow of itself or another object. Thus, when an intersection occurs, I cast a ray from the intersection point towards the light, and if that ray intersects with any object, I set the pixel value at the intersection point to *0*.
* I have included 2 screenshots with the light positioned at different locations to showcase shadows on the bumpy cube. The first image below has a light positioned at: *(-1, 1, 1)*. The second image below has a light positioned at *(1, -1, 1)*.
  + By default, the light is positioned at *(-1, 1, 1)*. To produce the second image, simply comment out the first light position and uncomment the second light position within the *part4* function.
* To run this code, uncomment the call to *part4(true, false)* in *main()* at the bottom of the *main.cpp* file and in the *build/* directory run:
  + *make && ./Assignment1\_bin*
* The output image will be saved to the *build/* directory as *part4-shadows.png*
* Light positioned at *(-1, 1, 1)* (Default) :



* Light positioned at *(1, -1, 1* (Need to uncomment line to render):



**1.6 Reflections on the floor**