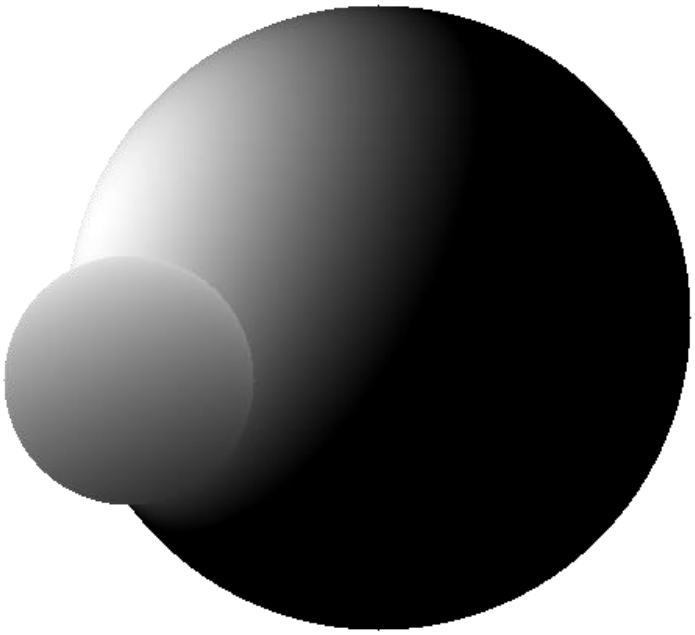
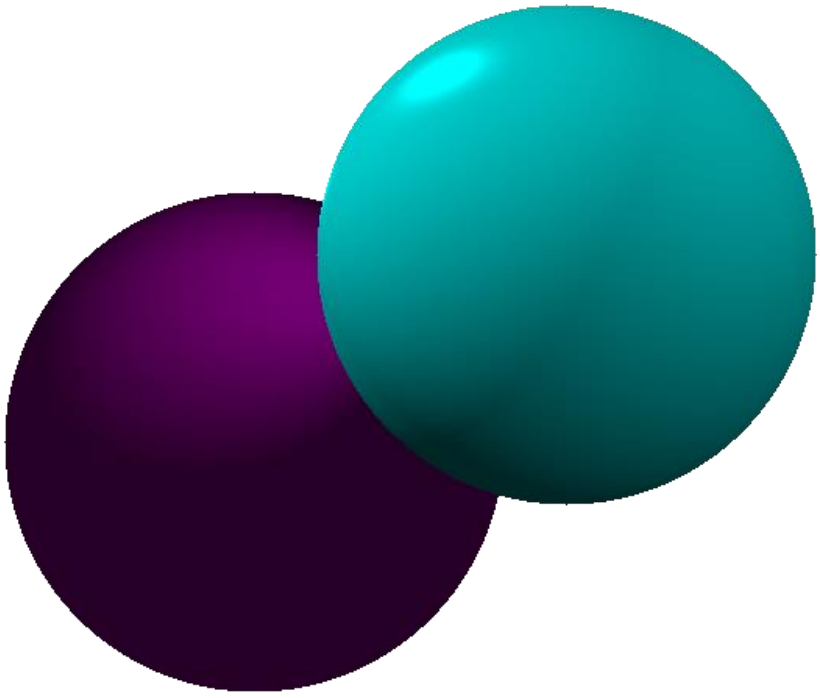
Compile code using normal instructions each part is in its own function, so inside main you can comment out any functions you don't wish to build at that time. Otherwise, each function makes its own png image that is named to match which task I was working on.

The source code is very long this is because I have so many if functions, one for ever sphere I have made, otherwise the code is very well organized. I made a sphere class to track the radiuses and origins of new spheres and functions to calculate the ray intersection point along with the diffuse and specular shading.

Ray Tracing Spheres – created two spheres slightly off origin, did this by creating a sphere class so that it was easy for me to change radiuses and make new origin points.

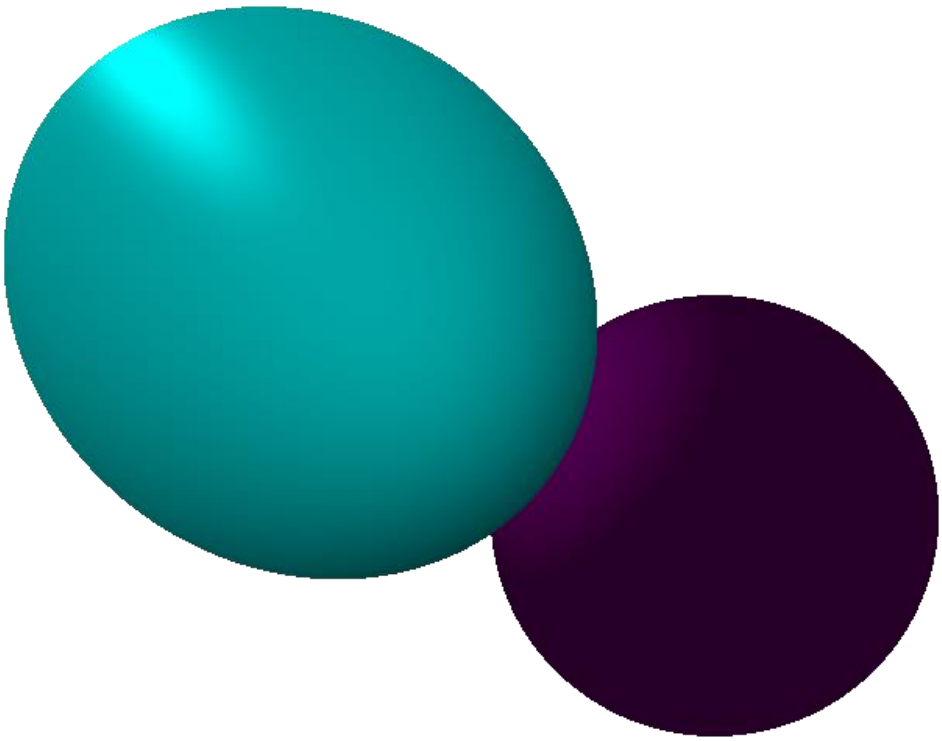


Shading – have two spheres of different origins, with different colors, and different textures, also two light sources, at this point I created the functions for diffuse and specular shading. Also, I made R,G,B matrixes in order to create new colors. I followed the book instructions where I add all the different lights and then multiple each matrix by how intense I want the color to be. And calculated the same for the second light source adding the two together.

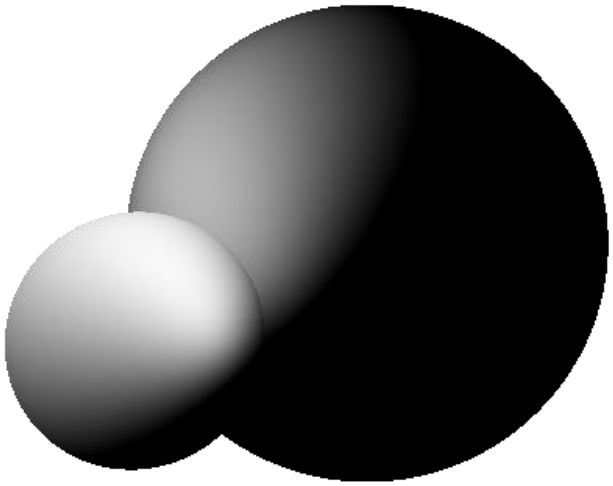


Perspective Projection – shown for both previous images with the origin coordinates slightly changed for the camera and spheres. This is where I had to start over and create a function to calculate the ray intersection at every point. It made future challenges much easier and allowed me to implement the perspective equation in a more readable fashion.

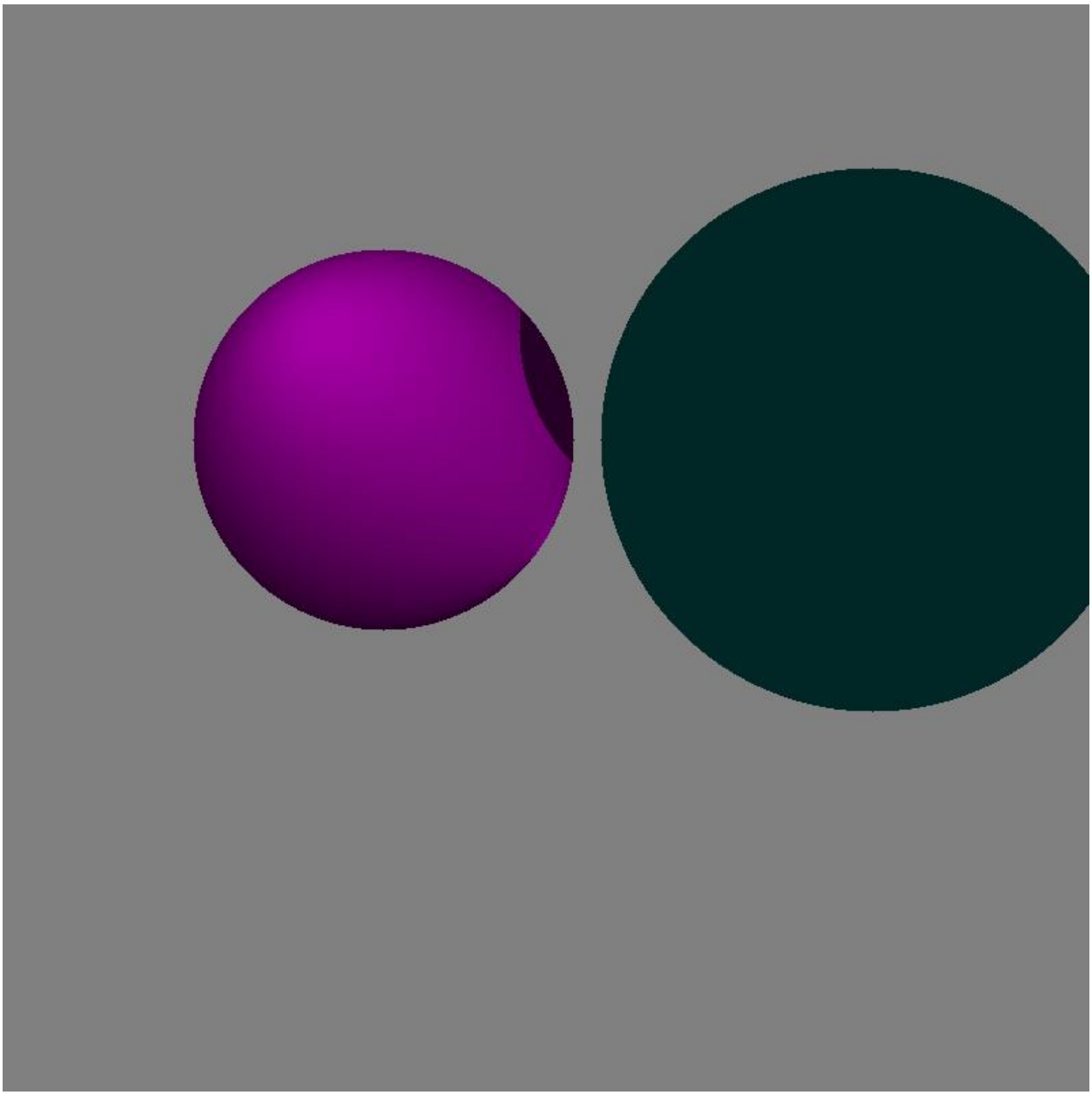
This is perspective of 1.2 note how the blue ball is clearly closer to the viewers eye than the pink ball.



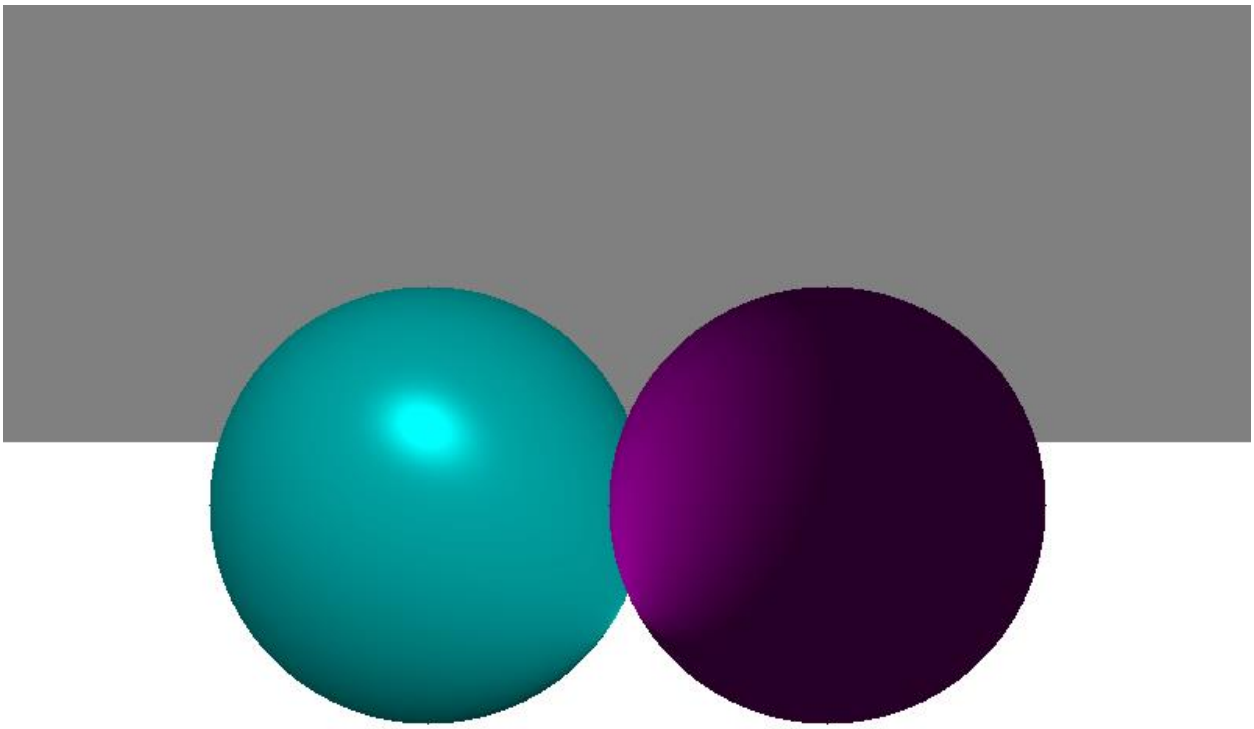
This is perspective of 1.1 they may not look very different from the original due to not having very different depths however I feel like it is easier to see the smaller ball is in front of the larger one.



Shadows – used ambient shading in order to create realistic shadows, the giant one is just isn't in the light source path but the shadow is shown against the pink one. Used orthographic over perspective view.



Reflection – was able to distinguish floor from there I would've added the specular shading for the mirror and then implemented a recursive function to apply the color, however I ran out of time



***Note that 1.4 the triangle mesh is missing I also ran out of time to work on this part. I understand what was supposed to happen though. When you load the meshes there will be one for the object shape which can be stored in V and one for the object “face”, shading/color properties, that can be stored in F. From there you use the existing loop and add this new object into the picture applying shading from the light source as you did with the others.