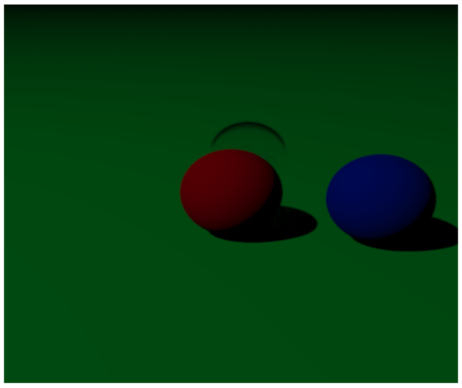
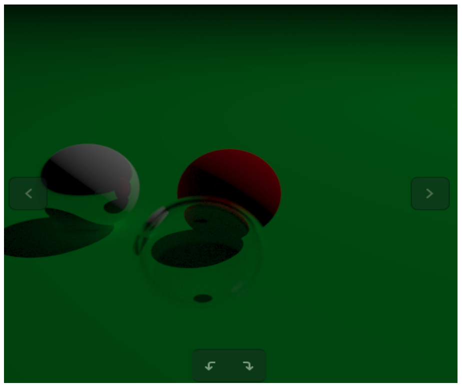
Assignment #1: Ray Tracing

Dustin Bolink

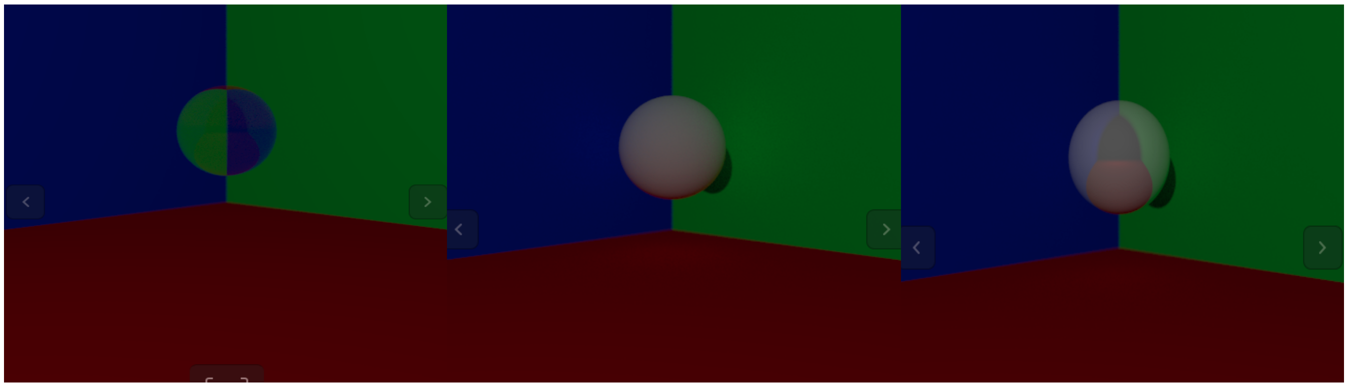
V00747883

The purpose of designing a ray tracer in an object oriented language is to better manage instances of objects in the image. Shapes that exist in the system can be stored easily and functions can be run from the shapes when they are interacted with. This is beneficial in comparision to not using object orientation as running a different light interactions when hitting a shape would be much more difficult. In my program I used the hitable, materials, hit records, and a camera. The purpose of the abstract hitable class is to make sure that all shapes that can be interacted with have a way of determining if a ray intersects it and where it is that that point is. Seeing as a plane and a sphere will have a different way of determining this there must be a way of calling the function to check for all hitable objects. Looping through the hitable list and calling the hit function will call a different function based on the type of the hitable object which allows us to achieve this. The material class is the abstract class to determine what each object is made of. When a shape is interacted with it uses the material type pointer stored in the class to decide how to scatter the light rays after hitting the object. In order to store information about light rays hitting objects the hit_record struct is used. The hit record includes the normal (n) of the object at the location where the light ray hits the object, the distance (t) along the ray, the point (p) in the space where the ray intersects the object, and a pointer to the material type of the object the ray collided with. The camera is used to transform rays created from the origin of the viewer to match the basis of the world that the scene is in. Transforming the rays of the camera to match the the coordinates of the row allows the objects to use their own basis and makes computations easier. It is also the job of the camera to adjust the lens of the photo to allow for different focuses and to store information of where the picture is being taken from.


Normal Image

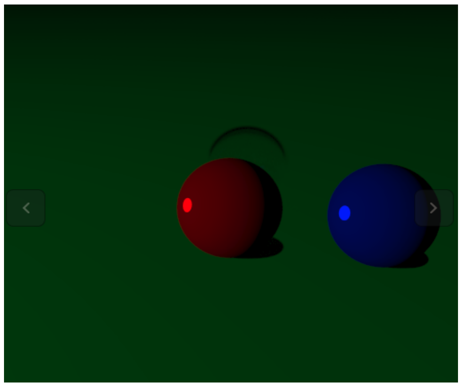

Rotated Image

The transport of light in my ray tracer starts at the camera. A ray is created by the camera for each pixel in the image and is sent into the world space. In order to get a gradual change of colors, a form of anti aliasing is appied. This is done by casting multiple rays within each pixel and averaging the results of each one. The program then loops through each object in the world space and watches for any collision. If no collision occurs the background color is returned, in this case it is black. If the ray from the camera hits an object two different things happen. The first is a ray is sent from the collision point to each of the lights that are in the world. If the ray reaches the light then shading is calculated based on the dot product of the normal of the object the ray is sent from and the ray direction to the light. The more direct the light the less shading, and vice versa. After the shading level of pixel has been determined any residule color from adjacent objects is calculated. This is done by looking at the material of the object the ray collided with. There are currently three materials in my ray tracer: metal, lambertian, and dielectric. The main properties of each material is that, metal will show reflections of adjacent objects in the scene, dielectric will show objects that are behind the object with some reflections and some refractions based on where the ray hits the object, and lambertian objects will appear the color of the attenuation of the object. After bouncing up to fifty times the color of the pixel is calculated with diminishing returns from each bounce.

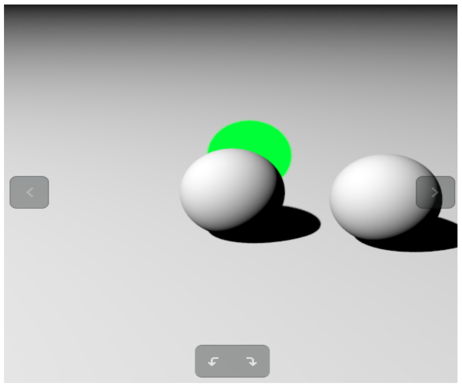


Dielectric Material Lambertian Material Metal Material

In a lambertian material the light is diffused randomly, no matter how light hits the object the scattered ray will be in the opposite direction at a random angle. By scattering the rays randomly the object gets a matte appearance that reflects light equally in all directions. In metal objects the scattered light ray is the reflected ray from the normal of the angle of incidence. If a ray hits a metal object at a 30 degree angle to the normal of the object, the reflected ray is 30 degrees from the other side of the normal. There are also abilities within the metal to fuzz the angle that the scattered ray returns at. The larger the fuzz value more random the scattered rays are, which makes it more difficult to know what objects are around based on the reflection. The light rays that hit a dielectric material are never absorbed into the object, all light passes through at either a reflected or refracted angle and observes the objects behinds it. When tracing a ray through a dielectric object the angle at which the light travels through it is based on the composition of the objects. Ifthe refractive indexes of the objects determines the refractive angle of the light within the object. When the refractive index is large inside the object the light ray will be reflected. The reflected angle is a complex equation that is not covered by snells law so I used Schlicks approximation that was recommended in the book. This allows for simple variable reflection within the material based on its reflection index. I found that this is where a fair bit of my troubles came in when doing lighting, the scattering of these rays from the light source caused some weird looking dots on other objects due to the random values in the scattering.



Specular Image



Shading Image

In order to achieve shading, I used a hybrid model. When an object is collided with a ray is traced from the object to the light to see if it is in direct light. This makes certain parts of the image light up much more than is typical for ray tracers. The only exception to this rule is the dielectic materials. When a dielectric material is colided with on tracing a ray to the light source it is not detected. This was a design choice as I ran out of time to calculate the ray out the other side to get a proper lighting value. This is shown by the color of the dielectric material in the shading image. The other lighting effect I attemped to put in my ray tracer was specular lighting. I did not have time to change the angles and weighting of the speculars but I was able to generate a proof of concept that could let the camera know the color of the camera and if a specular should be returned. This involved reusing the hit_record struct to return lighting ray information for coloring.

The last note is that I have learned a lot about the lighting in ray tracing that I would have missed by following the book. I spent a long time on my project and although the ray tracer could have had more features that looked cleaner I would not have understood what I do now by mindlessly following the text. This is why I reccomend changing this assignment in the future to not give a textbook that has written code in it. I feel it deters from the learning experience and can effect the learning in this project negitvely. If you have any other questions about what I learned during this project feel free to ask as i have much more that I could not fit into three pages.