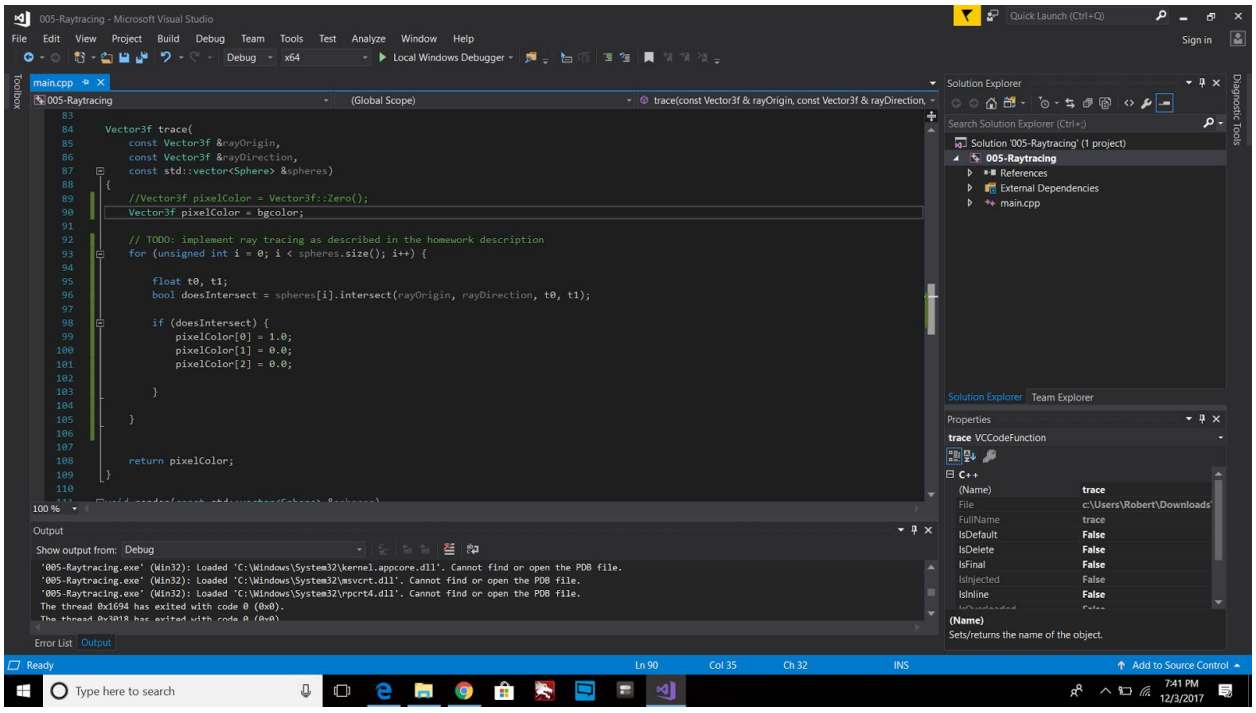
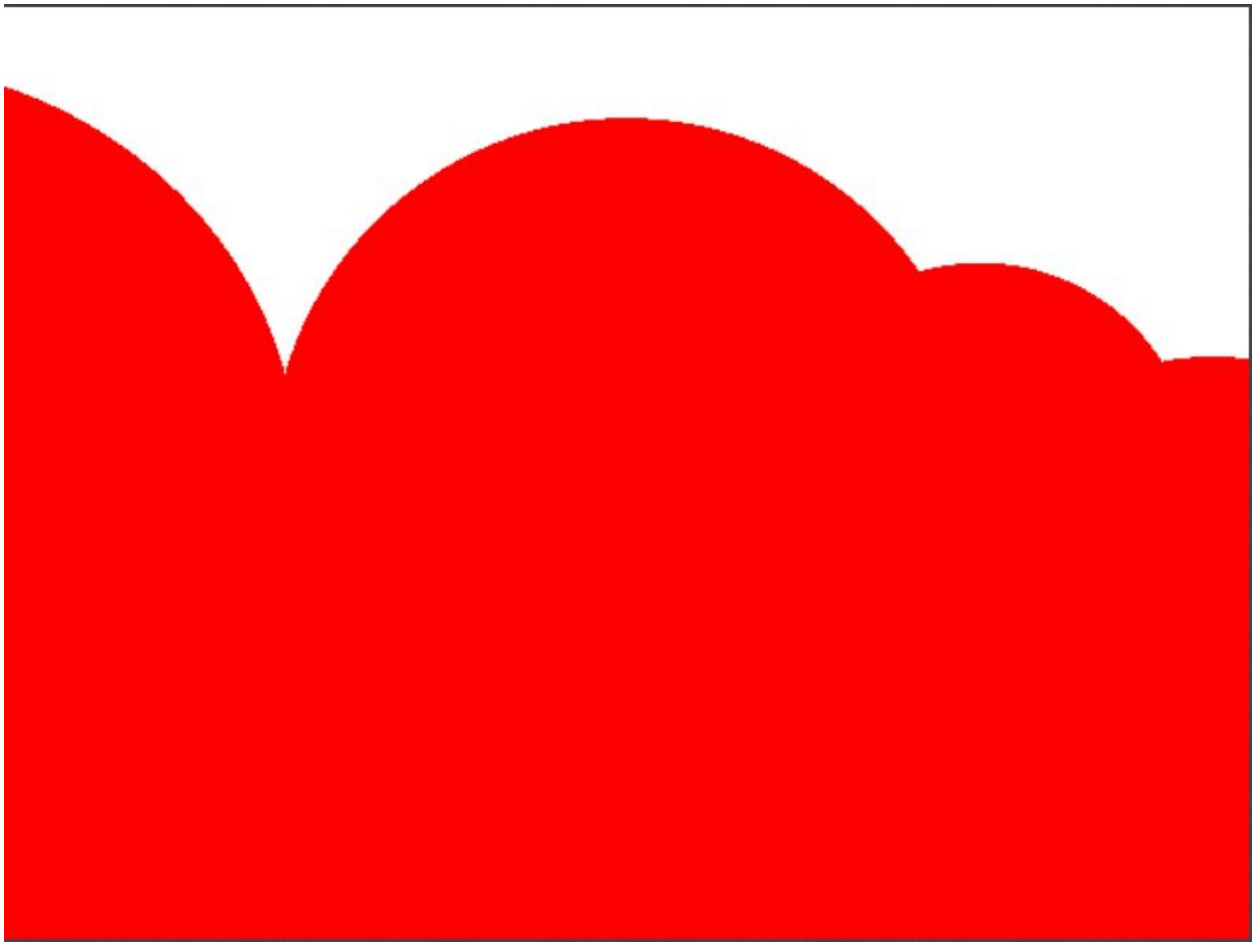
Robert Weischedel
CS 4600
12/4/17

Homework 5: Ray Tracing

Ray Casting
First Portion - out001.ppm

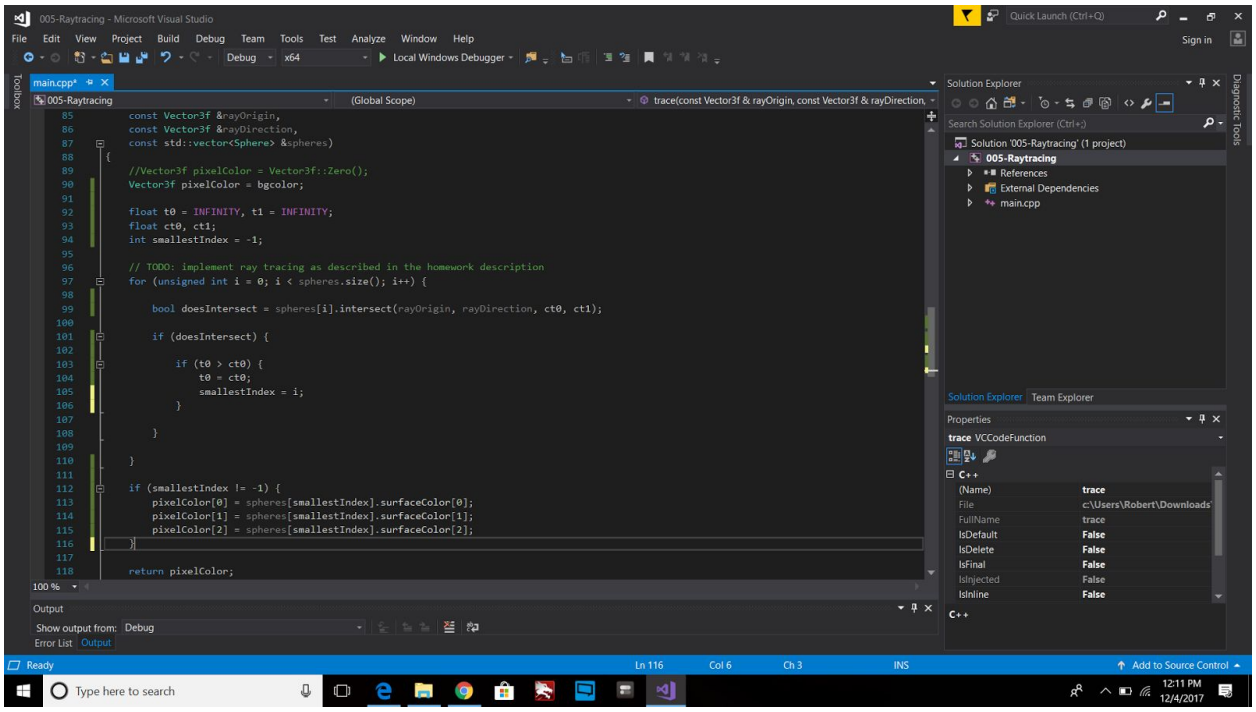
For this section I simply just iterated through all the spheres and if they intersected or if the doesIntersect variable I use is true, I set the pixelColor value to be red. I changed the original pixelColor value to be white, so that if that particular ray doesn't intersect with anything that when we return the pixelColor, the pixel will be the background color, white.

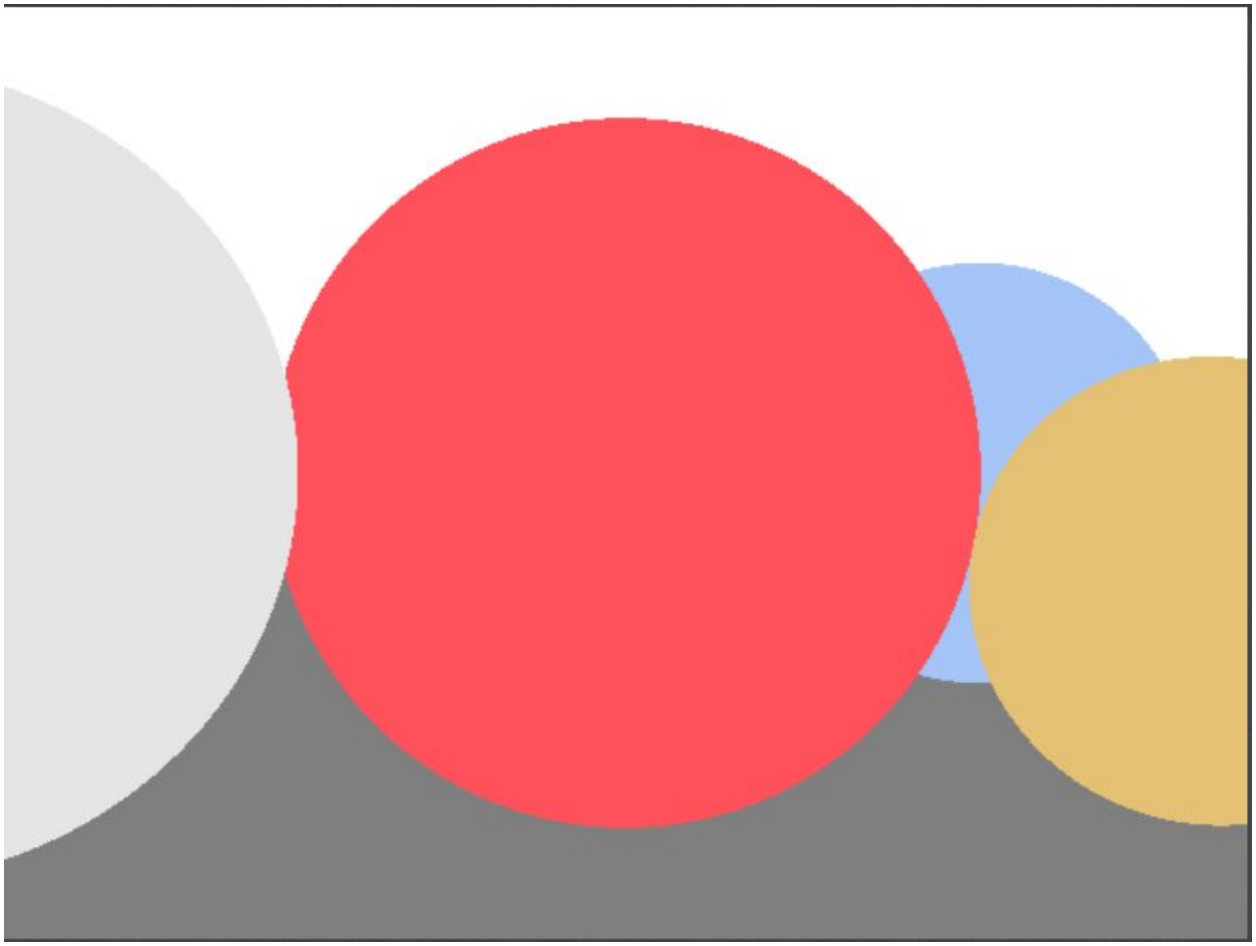




Second Portion - out002.ppm

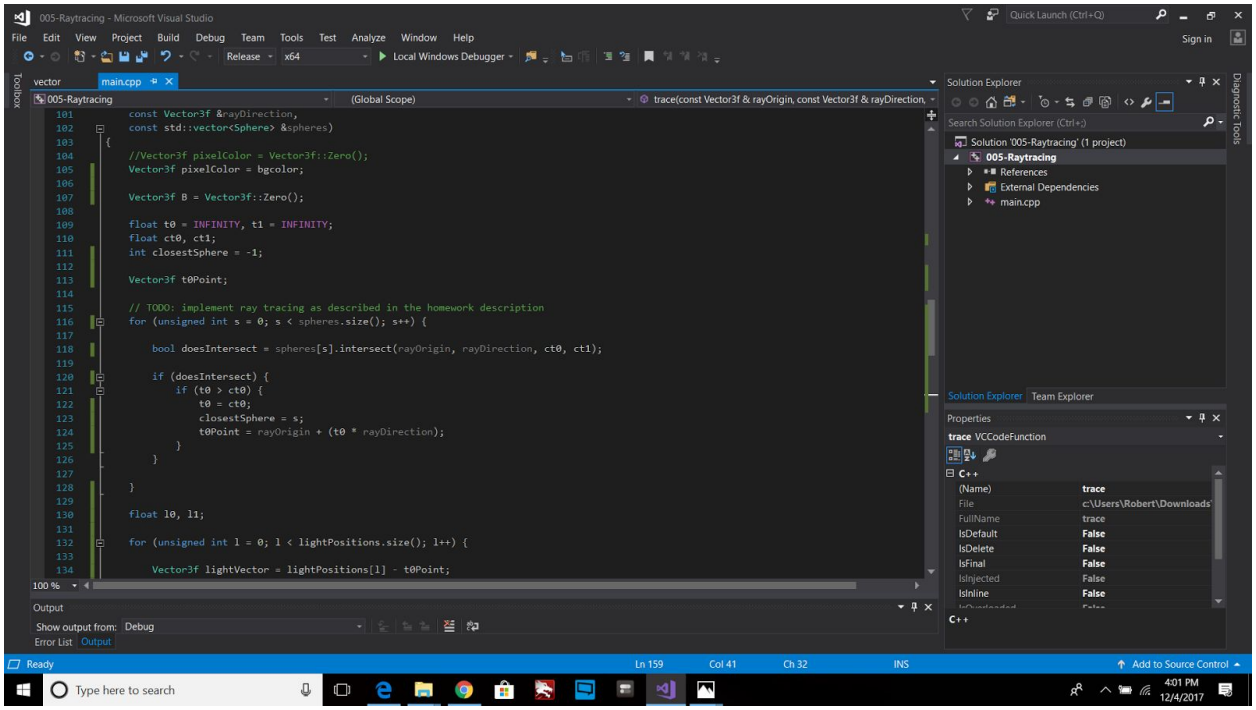
For this portion I made a few small modifications to the original code. First I started interacting with the t0 value that is returned from the intersect method. Since we need to return the color of the pixel of the sphere that the ray first intersects. I capture and store the index of the sphere with the smallest t0 value. After I finish looping through all the spheres and find that value, I then check and make sure I actually found a sphere. If I did, I set pixelColor to the sphere color, otherwise I return the default white color.

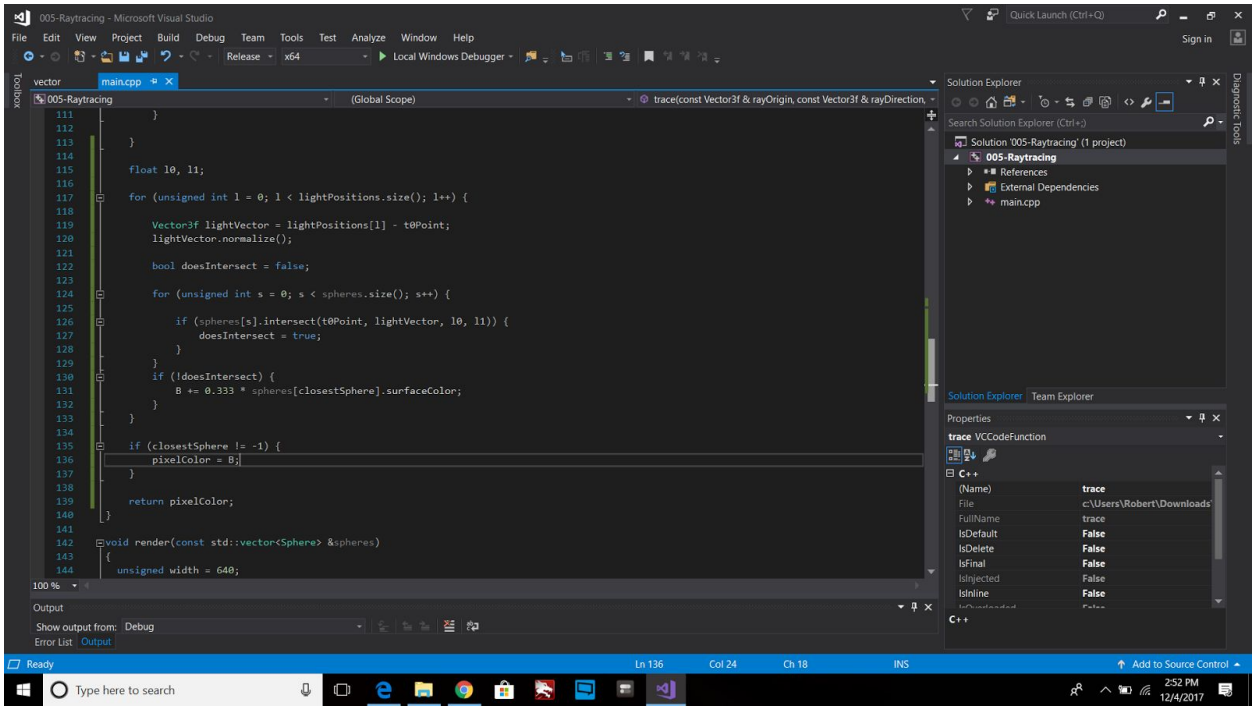


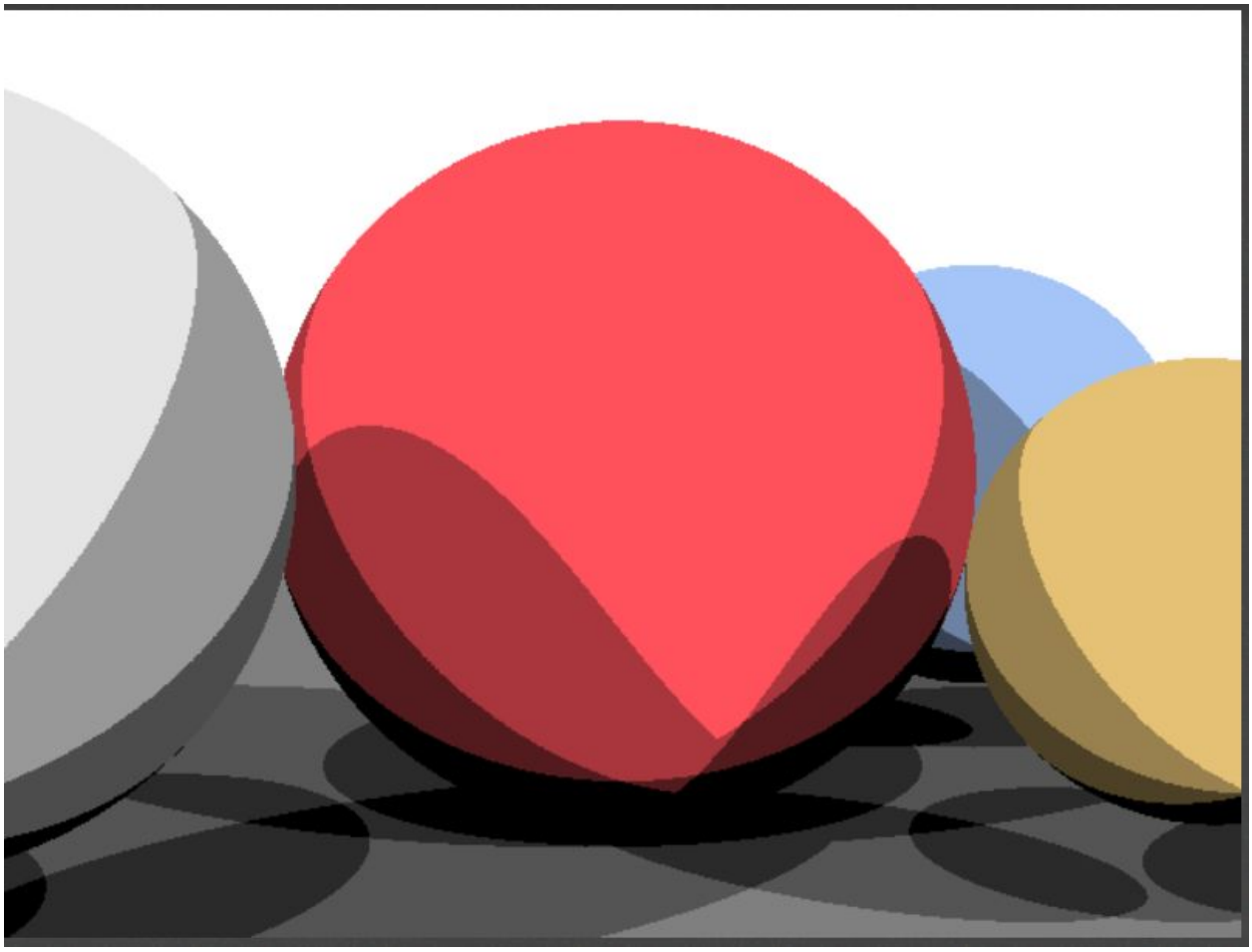


Shadow Rays
Third Portion - out003.ppm

For this portion I did a significant redesign of my original code in order to make it easier to implement the shadow rays. First I created a new pixelColor vector which I call B, and I set it to the color black in order to represent the shadows. Then I renamed my index variable to closestSphere as it made more sense than smallestIndex. Then I also added to my smallest t0 check if statement another line that would calculate the actual point of t0 and stores it for later use. Then outside of that loop I create a double nested for loop where I iterate not only through all the light sources, but also all the spheres again. This is to ensure that the particular point we are looking at doesn't have any spheres intersecting along the path to the light source. I calculate the vector from the t0 point (the one we calculated above) to the light source by subtracting the lightPosition from the t0 point. As long as there are no intersections I take and perform the color summation. Then at the end, as long as a sphere was found, then we set the B vector to the original pixelColor and return it.

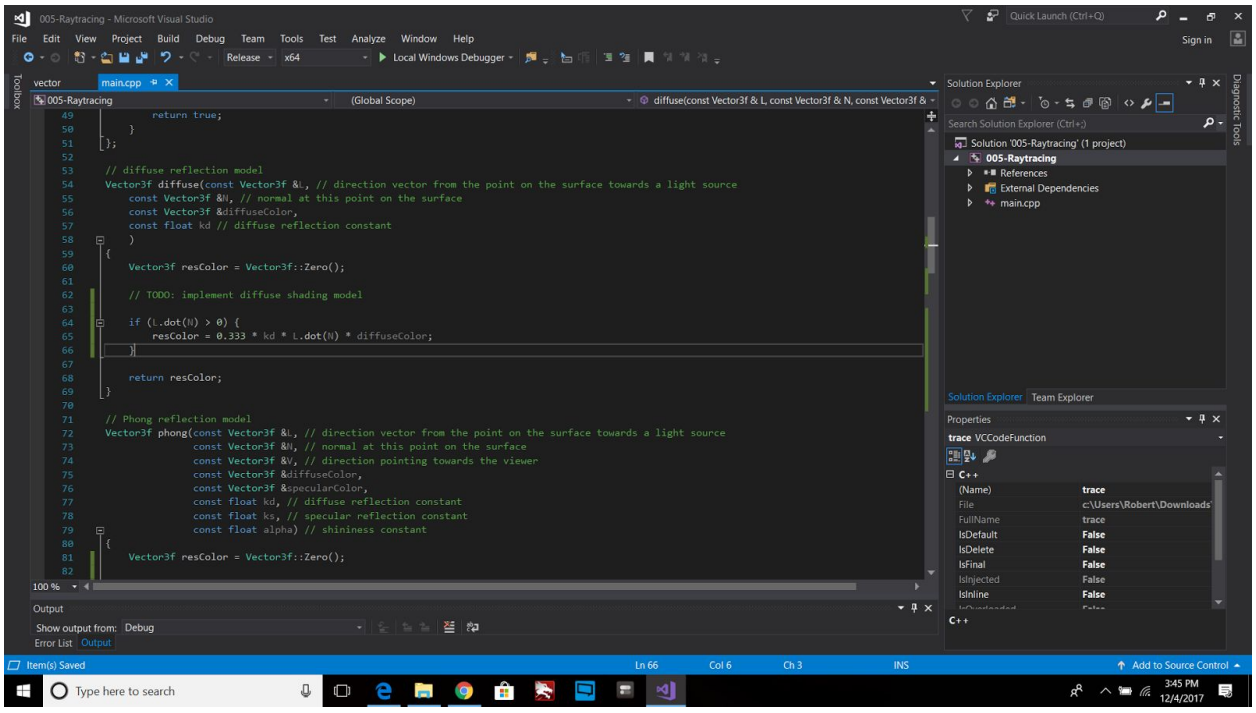


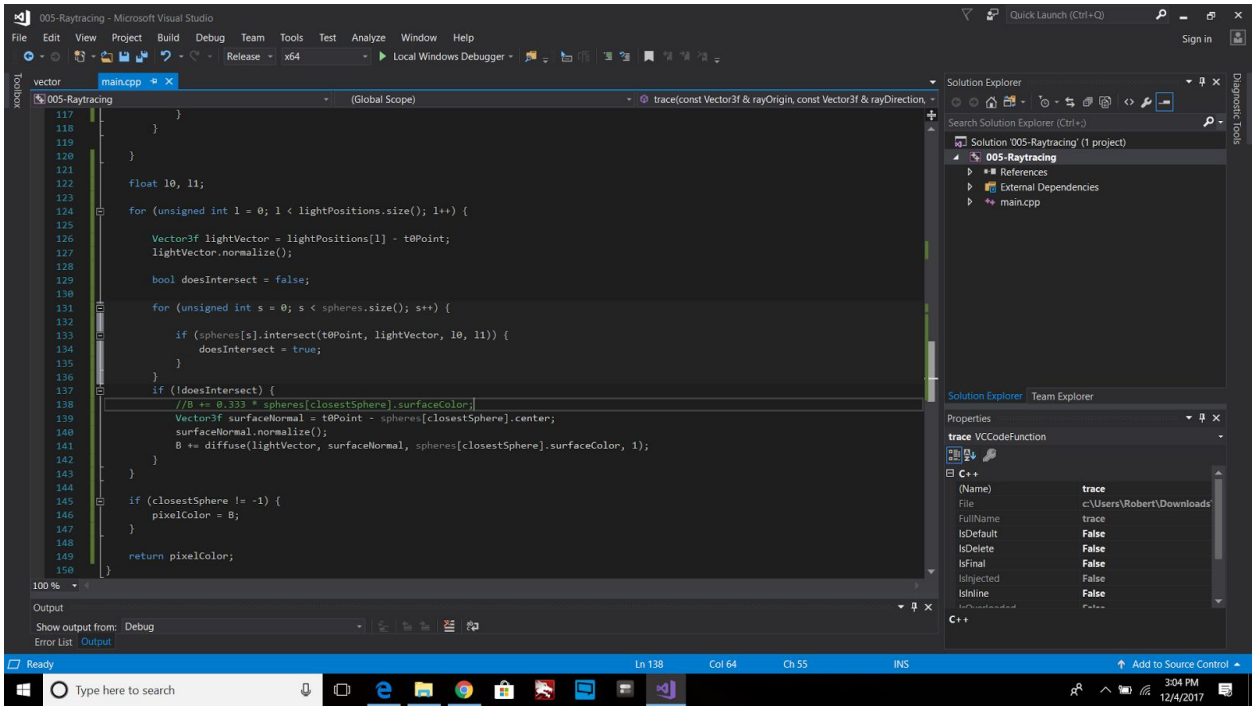

005-Raytracing - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Release x64 Local Windows Debugger
vector main.cpp
005-Raytracing (Global Scope) trace(const Vector3f & rayOrigin, const Vector3f & rayDirection,
float l0, l1;
for (unsigned int l = 0; l < lightPositions.size(); l++) {
Vector3f lightVector = lightPositions[l] - t0Point;
lightVector.normalize();
bool doesIntersect = false;
for (unsigned int s = 0; s < spheres.size(); s++) {
if (spheres[s].intersect(t0Point, lightVector, l0, l1)) {
doesIntersect = true;
}
}
if (!doesIntersect) {
B += 0.333 * spheres[closestSphere].surfaceColor;
}
}
if (closestSphere != -1) {
pixelColor = B;
}
return pixelColor;
}
void render(const std::vector<Sphere> &spheres)
{
unsigned width = 640;
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution '005-Raytracing' (1 project)
005-Raytracing
References
External Dependencies
main.cpp
Solution Explorer Team Explorer
Properties
trace VCCodeFunction
C++
(Name) trace
File c:\Users\Robert\Downloads
FullName trace
IsDefault False
IsDelete False
IsFinal False
IsInjected False
IsInline False
Output
Show output from: Debug
Error List Output
Ready
Ln 136 Col 24 Ch 18 INS
Add to Source Control
Type here to search
2:52 PM
12/4/2017

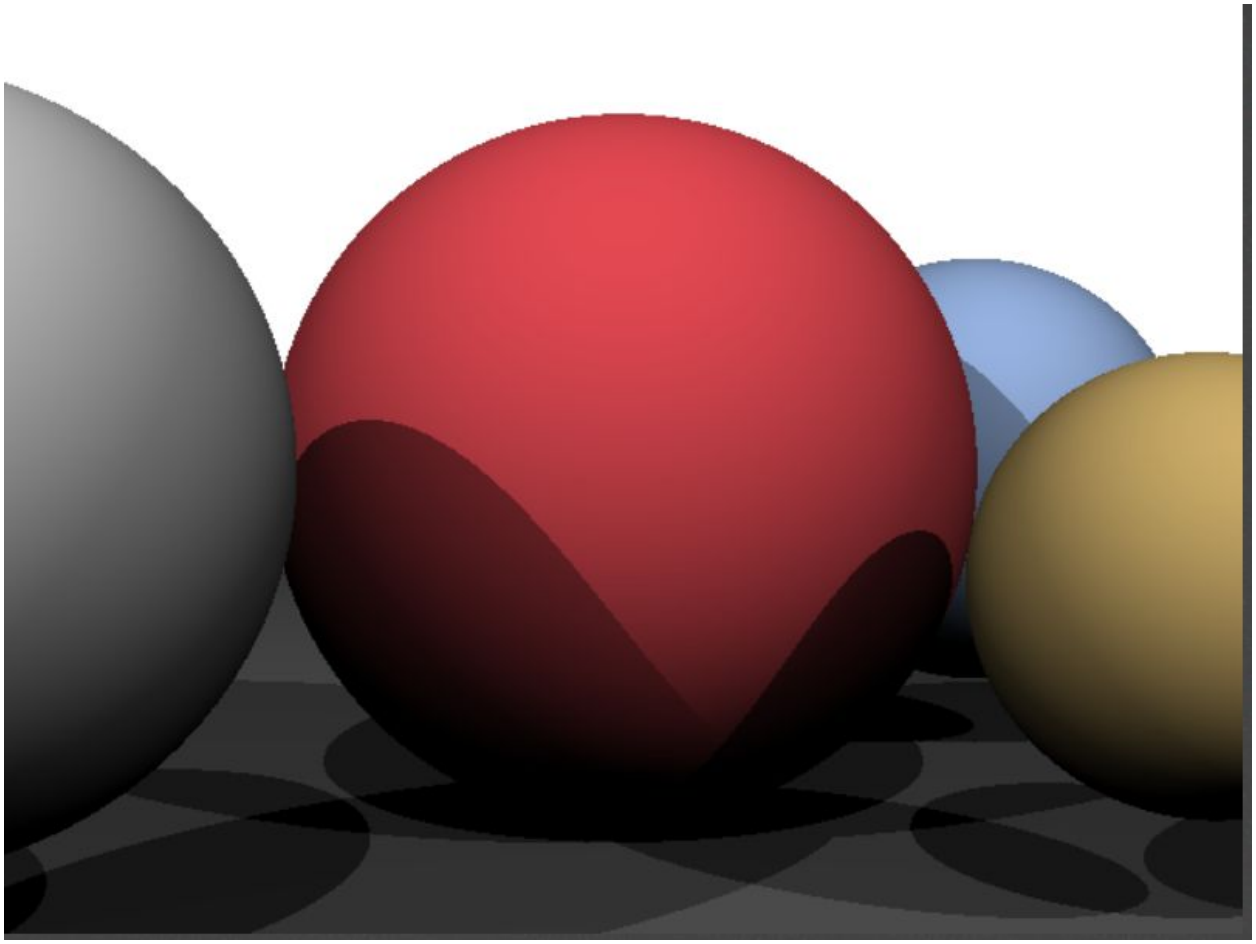


Illumination Models
Fourth Portion - out004.ppm

For this portion I made very little changes to my render method. I simply commented out the old way I was calculating the color and set B to the summation of values from the new diffuse method. Since the diffuse method also required the calculation of the surface normal, I also calculated it by taking the t0Point we are looking at minus the center of the sphere that the t0 originates from. Then I normalized it. The major changes come from the diffuse method, where I first checked if the direction vector dotted with the normal at that point was equal to zero. If it wasn't I take and set the resColor to the result of 0.333 * kd * L.dot(N) * diffuseColor. Otherwise, I leave the resColor vector as all zeros.

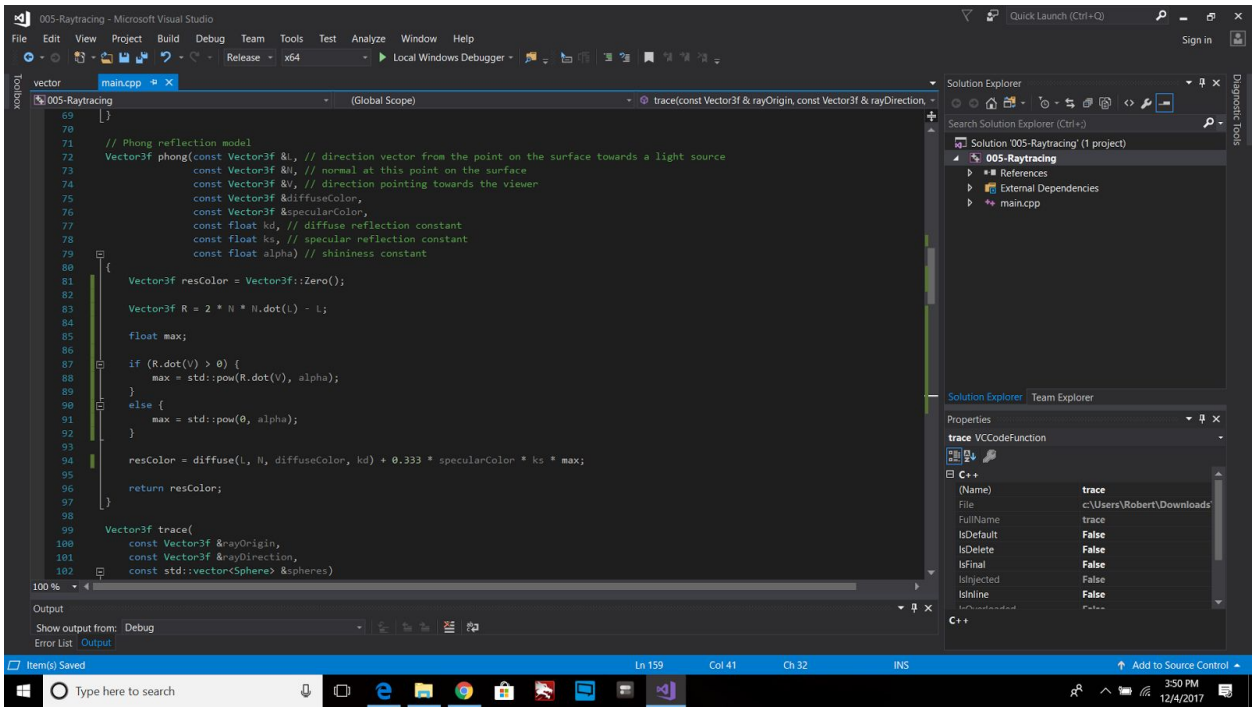


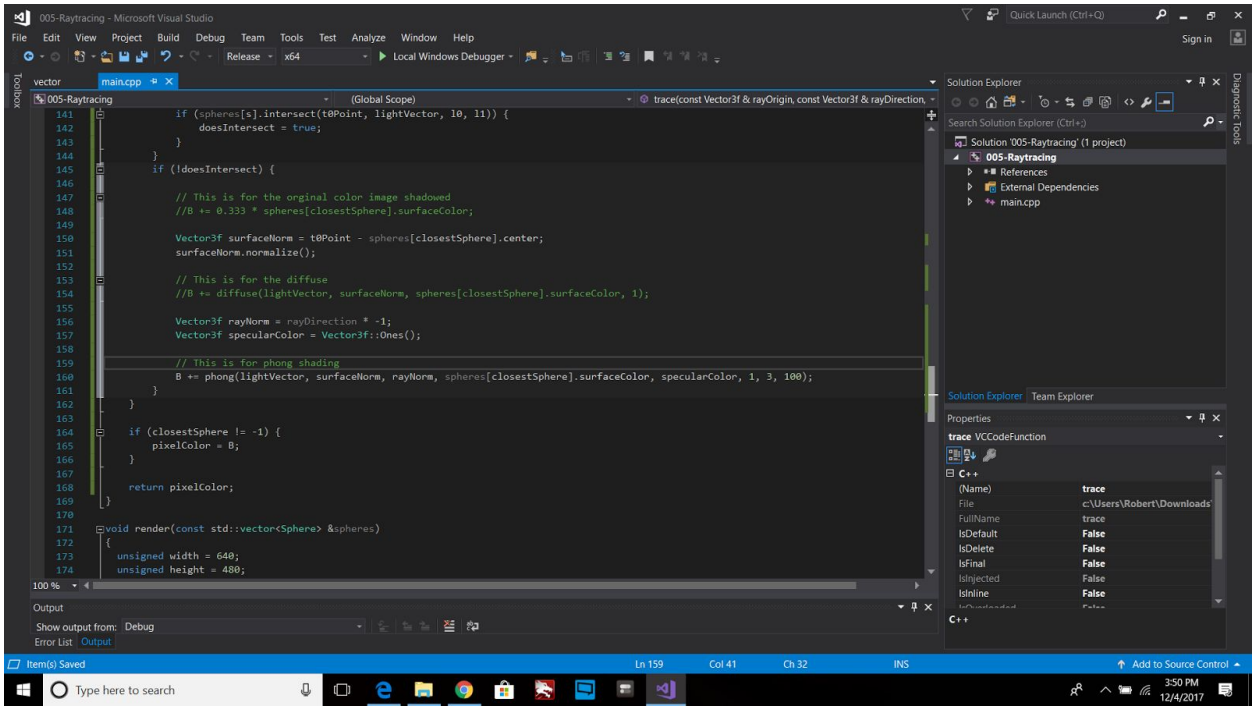

005-Raytracing - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Release x64 Local Windows Debugger
vector main.cpp
005-Raytracing (Global Scope) trace(const Vector3f & rayOrigin, const Vector3f & rayDirection,
}
}
}
float l0, l1;
for (unsigned int l = 0; l < lightPositions.size(); l++) {
Vector3f lightVector = lightPositions[l] - t0Point;
lightVector.normalize();
bool doesIntersect = false;
for (unsigned int s = 0; s < spheres.size(); s++) {
if (spheres[s].intersect(t0Point, lightVector, l0, l1)) {
doesIntersect = true;
}
}
if (!doesIntersect) {
//B += 0.333 * spheres[closestSphere].surfaceColor;
Vector3f surfaceNormal = t0Point - spheres[closestSphere].center;
surfaceNormal.normalize();
B += diffuse(lightVector, surfaceNormal, spheres[closestSphere].surfaceColor, 1);
}
}
if (closestSphere != -1) {
pixelColor = B;
}
return pixelColor;
}
100 %
Output
Show output from: Debug
Error List Output
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution '005-Raytracing' (1 project)
005-Raytracing
References
External Dependencies
main.cpp
Solution Explorer Team Explorer
Properties
trace VCCodeFunction
C++
(Name) trace
File c:\Users\Robert\Downloads
FullName trace
IsDefault False
IsDelete False
IsFinal False
IsInjected False
IsInline False
C++
Ready
Ln 138 Col 64 Ch 55 INS
Add to Source Control
Type here to search
3:04 PM
12/4/2017



Fifth Portion - out005.ppm

For this portion I once again made very little changes to render. I calculated the ray normal vector that the phong method needs, buy taking and looking at the opposite direction of the original rayDirection. I did this by simply taking and multiplying the rayDirection vector by -1. I also created a new vector called specularColor, which is a vector filled with ones. The major changes come from the phong method. I followed the algorithm found on lecture14-shading slides. In this method I first calculate the R value which I use to perform that dot product of on V. If that value is greater than zero, I take that value and raise it to the power of the alpha value and set that value to max. Or if it is zero I set max to zero. Then as suggested in the homework description, I called the diffuse method and add that value to the product of specularColor * ks * max.




005-Raytracing - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Release x64 Local Windows Debugger
vector main.cpp
005-Raytracing (Global Scope) trace(const Vector3f & rayOrigin, const Vector3f & rayDirection,
if (spheres[s].intersect(t0Point, lightVector, l0, l1)) {
doesIntersect = true;
}
}
if (!doesIntersect) {
// This is for the orginal color image shadowed
//B += 0.333 * spheres[closestSphere].surfaceColor;
Vector3f surfaceNorm = t0Point - spheres[closestSphere].center;
surfaceNorm.normalize();
// This is for the diffuse
//B += diffuse(lightVector, surfaceNorm, spheres[closestSphere].surfaceColor, 1);
Vector3f rayNorm = rayDirection * -1;
Vector3f specularColor = Vector3f::Ones();
// This is for phong shading
B += phong(lightVector, surfaceNorm, rayNorm, spheres[closestSphere].surfaceColor, specularColor, 1, 3, 100);
}
}
if (closestSphere != -1) {
pixelColor = B;
}
return pixelColor;
}
void render(const std::vector<Sphere> &spheres)
{
unsigned width = 640;
unsigned height = 480;
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution '005-Raytracing' (1 project)
005-Raytracing
References
External Dependencies
main.cpp
Solution Explorer Team Explorer
Properties
trace VCCodeFunction
C++
(Name) trace
File c:\Users\Robert\Downloads
FullName trace
IsDefault False
IsDelete False
IsFinal False
IsInjected False
IsInline False
Output
Show output from: Debug
Error List Output
Item(s) Saved
Ln 159 Col 41 Ch 32 INS
Add to Source Control
Type here to search
3:50 PM
12/4/2017