322COM Coursework 1

Contents

[Introduction 2](#_Toc22897547)

[Project Progression 3](#_Toc22897548)

[Hard Shadows and Phong’s Shading 3](#_Toc22897549)

[Soft Shadows 5](#_Toc22897550)

[Reflection 6](#_Toc22897551)

[Object Loading 7](#_Toc22897552)

[Optimisation 8](#_Toc22897553)

[Reflective Review 9](#_Toc22897554)

[References 11](#_Toc22897555)

# Introduction

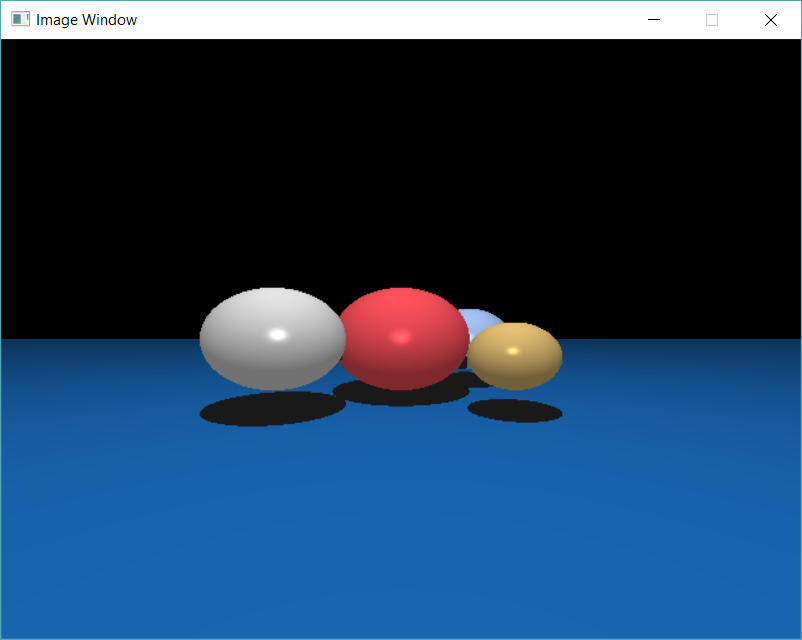
The task that I was given for this coursework involved the need to write, from scratch an interactive ray tracer that will run close to real-time performance. Ray tracing is a powerful method to perform global illumination. Per the specification of the coursework I was free to create any scene I wished if it included the use of primitive shapes and materials that could best demonstrate the ray tracing features that I implemented.

# Project Progression

## Hard Shadows and Phong’s Shading

In this section I will be covering the steps that I took to complete the project, I will be including a brief overview of what is involved in each step.

After the initial setup phase and I had enabled the project to build directly to the screen via SDL2, I wanted to incorporate Phong’s shading so that the project would hopefully become more aesthetically pleasing to viewers and it would also add to the realism of the project. I also wanted to try and incorporate, at the very minimum, hard shadows so that more realism could be included in the project. Figure 1 shows the result of both Phong’s shading and hard shadows onto the primitives that I included in my scene.



Figure

To create the shadows of the spheres onto the plane I needed to work out where the plane was being intersected by the ray, from this intersection point I created a new “shadow ray” that traced back to the light source, if this ray intersected with any object along the way then the point of intersection would be coloured by ambient light.

To work out the shadow ray origin I used the equation

***L = s – p***

where L is the direction of the ray, S is the light source and P is the point of intersection with the object.

To work out Phongs shading I first had to work out the ambient colour of the object using the equation

***Ambient = colour \* intensity***

This equation could then be used in the equation of Diffuse colour

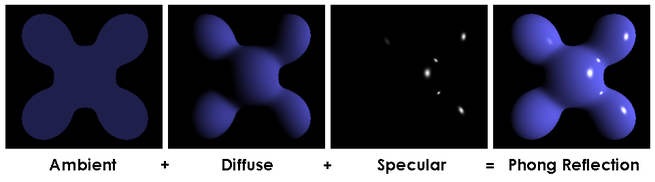
***AmbientColour \* max(0.0f, lightDirNormal.dot(norIntersect))***

And then again in the equation of Specular colour

***AmbientColour \* pow(max(0.0f, -norR.dot(normalize(direction))), 128.0f);***

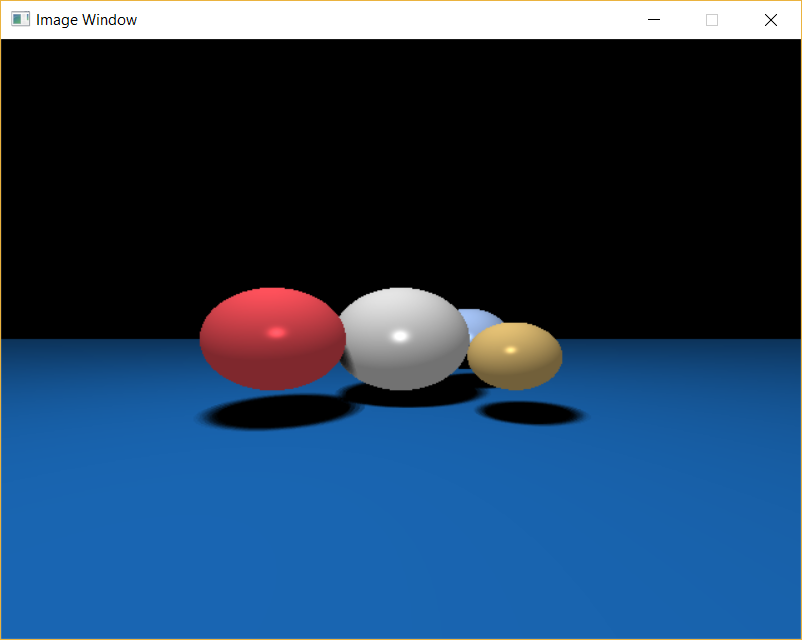
Then to get the Phong’s shading for the object you combine Ambient with Diffuse along with Specular, these three lightings combined creates Phong’s shading.

This can be seen in Figure 2



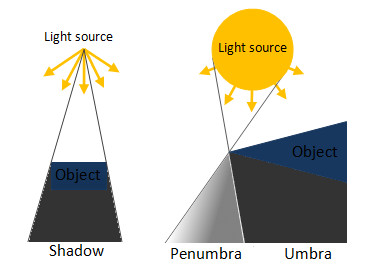
Figure

## Soft Shadows



Figure

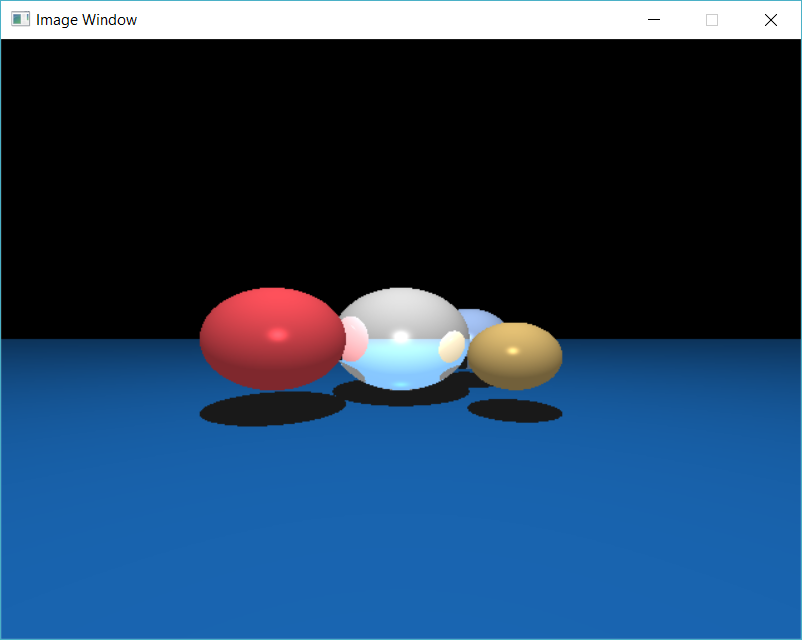
After completing hard shadows, I set out to accomplish Soft Shadows, the effect of soft shadows can be seen in Figure 3. This shadow effect looks more realistic that hard shadows due to the shadow fading from full ambient to whatever colour the object that the shadow is being imprinted onto.



Figure

As shown in Figure 4, the hard shadow effect is created solely by a point light source shining onto an object and creating the shadow. Whereas the soft shadow effect is created by an area light source (the sun) and smoothens the shadow. As you increase the sample size of the area light, you will also increase the smoothness of the transition from base Ambient shading (Umbra) to the object colour which improves the Penumbra effect.

## Reflection



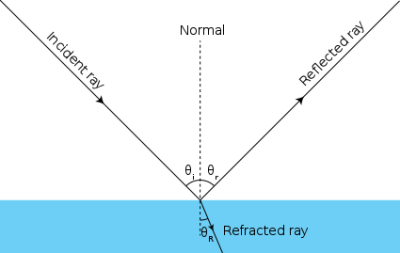
Figure

After I had completed Phong’s shading and hard shadows I decided I wanted to try and challenge myself to something completely new, this challenge came in the form of reflection. From the lectures that I attended I knew the most efficient way to implement this was recursively.

From what I previously knew, the angle of incident is equal to the angle of reflection. This means that if a ray hits an object, the reflection ray will “bounce” off that object with the same angle that it hit with the object with, this can be seen in Figure 4. The point of intersection then becomes the new ray origin, these are then passed into the recursive function along with a +1 to depth, this then continues until the ray hits an object that isn’t reflective, or the maximum depth has been reached. The original point of intersect colour then becomes the current point of intersect colour. This then emulates reflection within code.

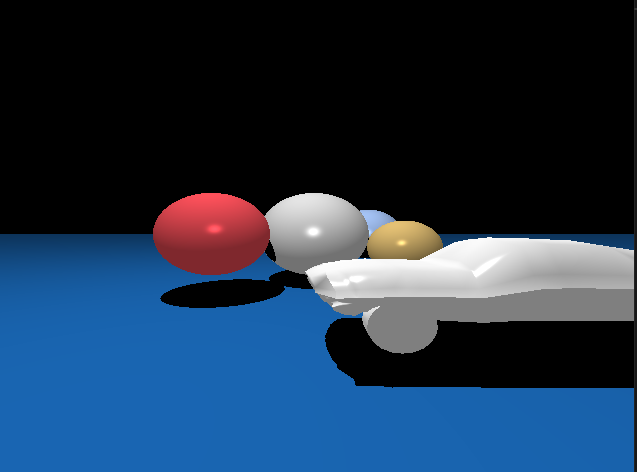
A maximum depth is used to prevent an infinite mirror as this will cause the code to crash as the rays are continuously travelling which would take up too much memory space.

The outcome of adding reflection to my project can be seen in Figure 3



Figure

## Object Loading

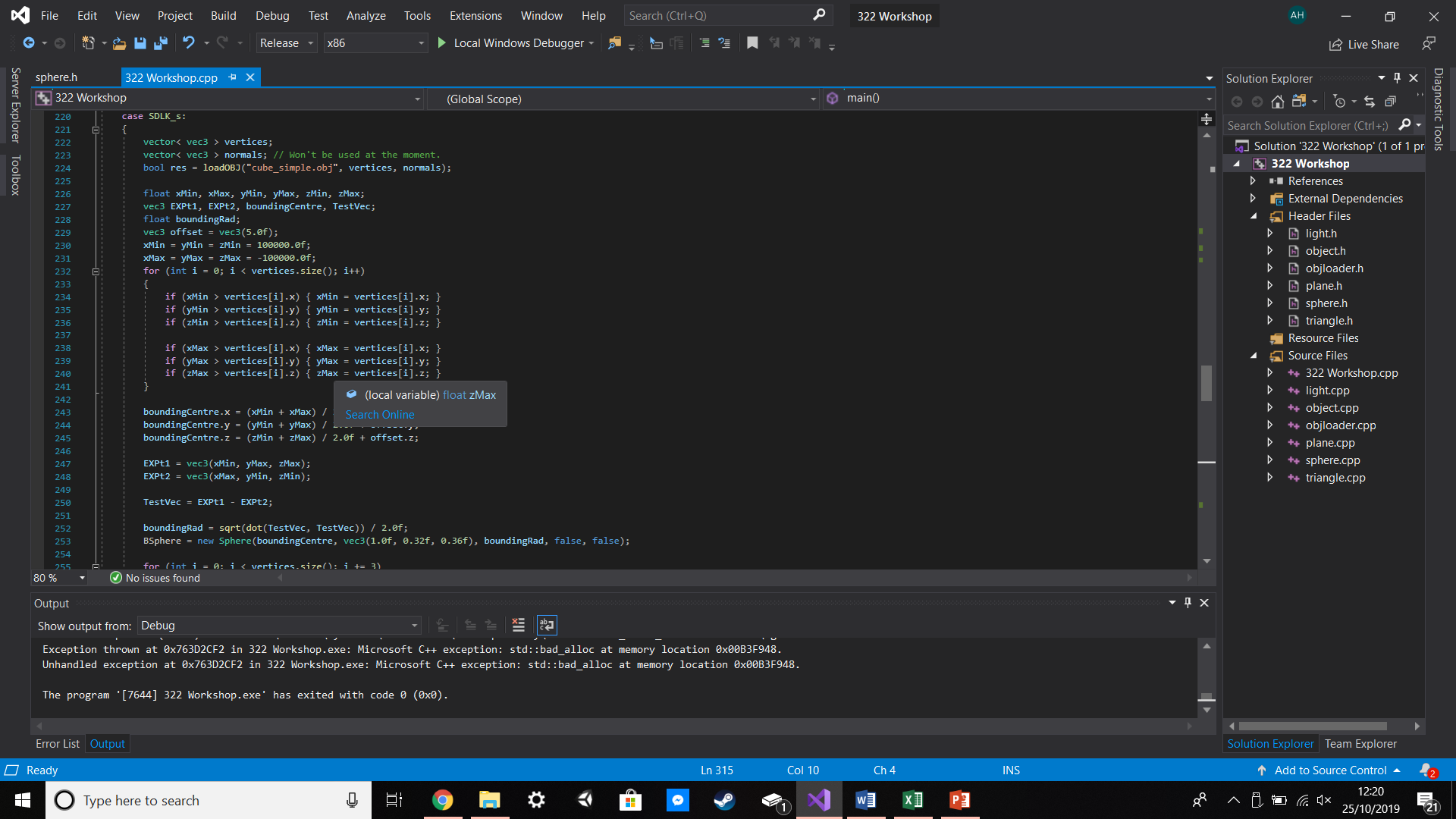


Figure

After completing all the steps that I outlined above, I began working on importing objects into my scene, since I had already made code to import .OBJ files from any 3D modelling software, this was fairly easy for me to implement, my main issue was choosing what model I could import that I felt could show off some of my skills, so I opted for the vehicle model that was part of my 302CDE assignment one.

## Optimisation

To optimise my code, I opted for bounding spheres. I chose these as when it comes to loading complex objects, they optimise the code to run smoother.



Figure

As Figure 8 shows, to create my bounding spheres I first iterate through the entire complex object and work out which point is at the absolute minimum and which point is at the absolute maximum, I then minus the minimum from the maximum to get what will become the centre point of the bounding sphere, if this bounding sphere exists then the program will switch to the object loader which in turn will load the object that is chosen.



Figure

As can be seen in Figure 9, optimising my code with bounding spheres for the complex objects did improve my run speeds when the objects were loaded. If I were to improve on this I would include bounding cubes for all my primitive objects to improve the run speed without the more complex shapes. I did not include the mustang object that can be seen in Figure 7 in these calculations due to the fact that adding in Soft Shadows and reflections made the run time way too long (excess of a minute) because of the amount of vertices involved in creating the model. I included the screenshot as proof that I did once have it working in the code.

# Reflective Review

When starting the project, I originally got the primitives to load into a simple .PPM file. This was done so that I could constantly check that the primitives were being detected by the ray tracing algorithm. Once I had made sure that all the primitives worked alongside the correct colouring, I began working on the program so that it would display directly to the screen, to accomplish this I used the graphics middleware SDL. I chose this middleware over OpenGL so that when it came to code the shaders later it would become easier as I wouldn’t have to code the entire fragment shader or vertex shader as OpenGL would require me to.

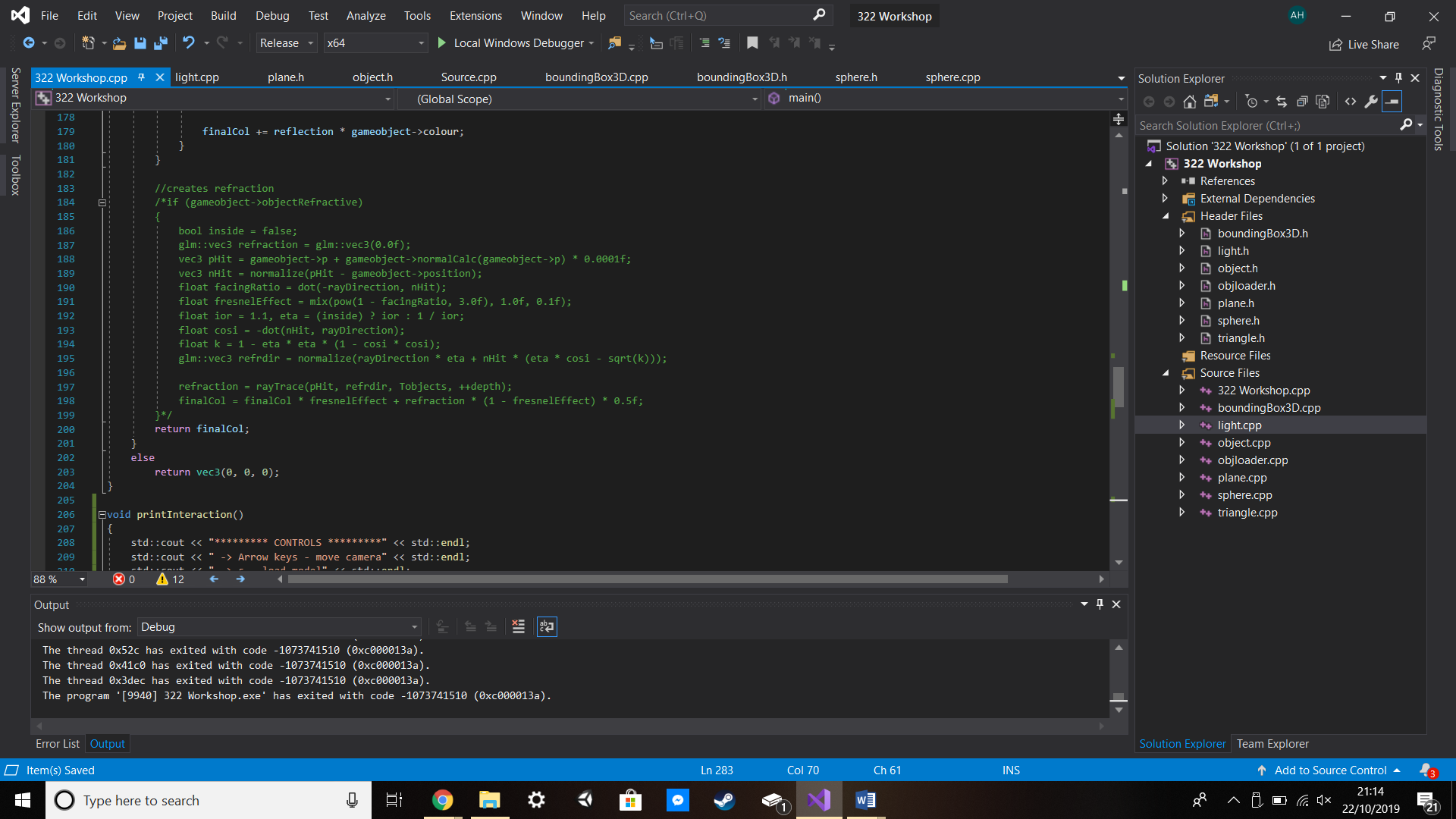
Whilst working on this project I found the initial change from outputting to .PPM to outputting via SDL fairly easy since I have been working with SDL and OpenGL for a few months now and have come to learn the basics of both along with a bit of advanced programming. On the flipside however, I found that when it came to programming phong’s shading I did initially struggle since I wasn’t used to coding light without coding the shader files, this did provide a challenge that I eventually overcame through lots of experimenting.

After completing the project, I am really proud that I managed to get my 3D model, that I used in my 302CDE coursework, to load into this coursework and then have that model lit via Phong’s lighting and then cast shadows onto anything that would be in the models shadow depending on where the light source is. If I could follow it up further, I would like to try texturing the model in the scene so that it would hopefully become more aesthetically pleasing to those who view my work and in the long run would add even more to my e-portfolio.

The lectures that I have attended in the forerunning to handing in this project have helped me to understand how to ray-cast in almost real-time along with how to manipulate the rays to “bounce” off objects so that I can reflect and refract if the object is a body of water or a mirror.

If I were to redo this project I would attempt to have one object have multiple different properties, by this I mean that if I were to load a car, as an example, the windows would refract the light to represent a glassy texture and the body of the car would reflect the light to represent a metallic colour.

When it came time to add user interaction to my project, I opted to allow the user to edit where the camera was focusing so that they get an essence of being able to view the entire scene. I then also enabled key presses that would allow the user to decide whether they wanted soft shadows or hard shadows, pressing another key would allow the user to import a complex model, then pressing a third key would toggle whether the objects become reflective or not. If I were to reattempt this project, I would ideally like to include controls for the light source to be moved by the user to imitate a day/night cycle. I would also opt for multi-threading to optimise my code further but at the time of this project I do not understand the concept fully so it would be something that required a lot of research on my part.



Figure

Even though it was never included, Figure 10 serves as proof that I attempted to add refraction into my program and even though it doesn’t work I left what I had in the project so that if I were to ever redo the project I would have my notes and edits so that in theory I would be able to implement it effectively.

# References

Figure 2 -> <https://en.wikipedia.org/wiki/Phong_shading>

Figure 4 -> <https://www.google.com/url?sa=i&source=images&cd=&ved=2ahUKEwjg4MTHnbflAhX5A2MBHYrOCHsQjRx6BAgBEAQ&url=http%3A%2F%2Fwww.polygonpi.com%2F%3Fp%3D318&psig=AOvVaw0x8q0Mkem_6zlOwz3ZxC0M&ust=1572086723422288>

Figure 6 -> <https://study.com/academy/lesson/what-is-the-law-of-reflection-of-light-definition-lesson-quiz.html>