FRASER BARKER

1600196

1600196@uad.ac.uk

ASSESSMENT Report

CMP301

Contents

[Overview 1](#_Toc532114144)

[Objects 2](#_Toc532114145)

[Brief Response 2](#_Toc532114146)

[UI & Controls 2](#_Toc532114147)

[Standard Controls 2](#_Toc532114148)

[ImGUI Controls 3](#_Toc532114149)

[Algorithms and Data Structures 6](#_Toc532114150)

[Vertex Manipulation 6](#_Toc532114151)

[Post Processing 7](#_Toc532114152)

[Lighting & Shadows 8](#_Toc532114153)

[Tessellation 9](#_Toc532114154)

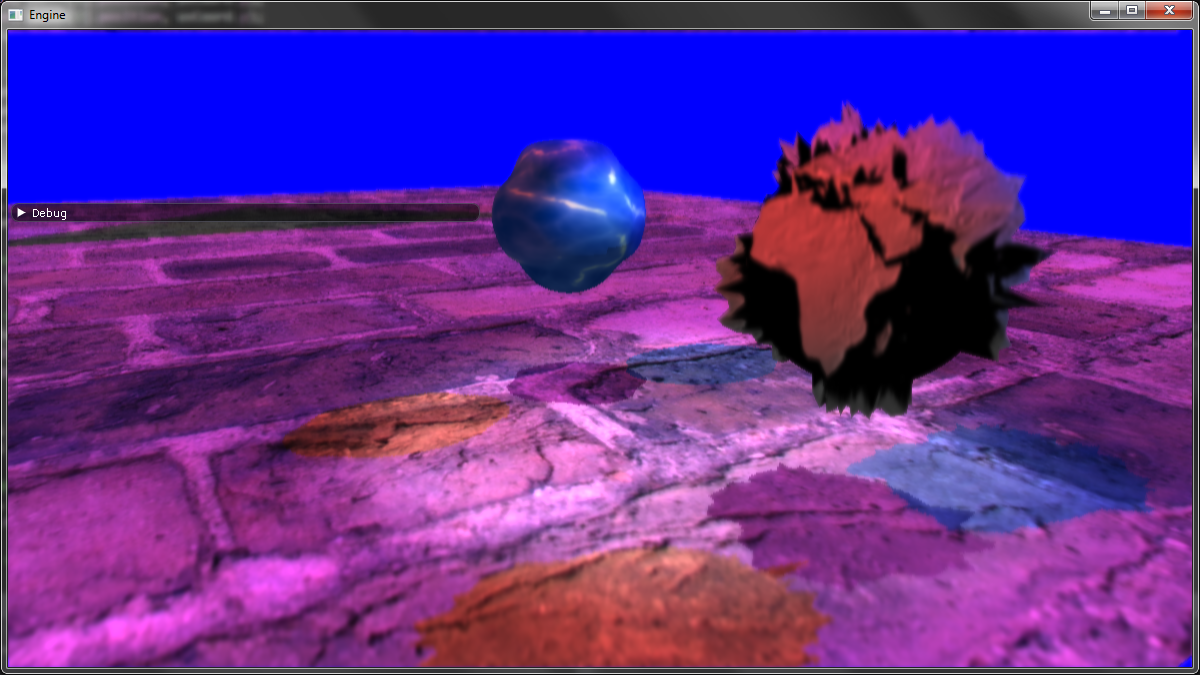
[Geometry Shader Stage 9](#_Toc532114155)

[Critical Reflection 11](#_Toc532114156)

[References 11](#_Toc532114157)

# Overview

## Objects



There are three objects within the scene. Two are spheres and the third is a plane. The two spheres both showcase tessellation and vertex manipulation. Both spheres have their depth calculated from each light and the camera. This allows the spheres to have their shadows cast dynamically onto the plane below them. The plane is more basic and has a plain texture applied to it. It also has its depth calculated from each light and the camera, but its main use is to have the spheres shadows cast onto it.

* Water sphere demonstrates vertex manipulation, post processing, lighting and shadows, tessellation and the geometry shader stage.
* Earth sphere demonstrates vertex manipulation, post processing, lighting and shadows, tessellation and the geometry shader stage.
* Plane demonstrates lighting and shadows.

## Brief Response

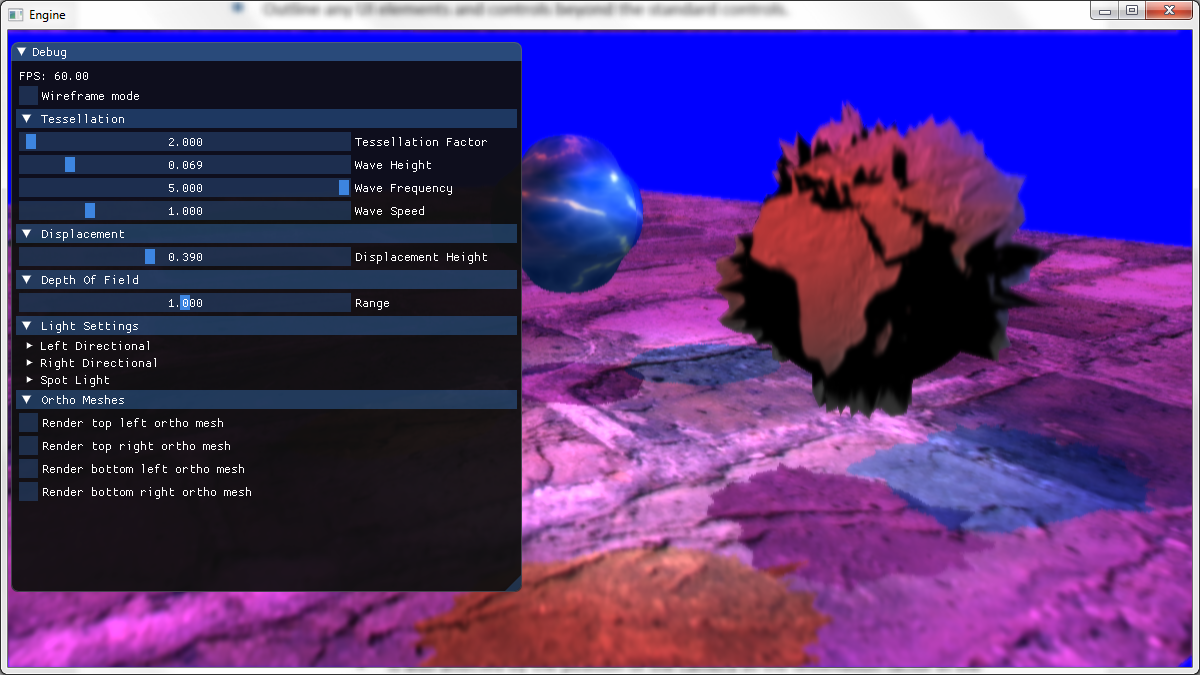
Initially, a cohesive desert island scene was planned but changed over due to being able to demonstrate the same process with much less hassle. Even straying from the proposal submitted in week seven although the project came close. Two spheres demonstrating most of the shader stages. Just missing the third (and arguably more desired, personally) sphere which would have showcased algorithmic morphing and reflection.

## UI & Controls

### Standard Controls

* Move the camera with W, A, S, D.
* Alter the camera’s height with E, Q.
* Alter the camera’s rotation with either the Up, Down, Left and Right arrow keys, or hit Spacebar and use the mouse to alter rotation.

### ImGUI Controls

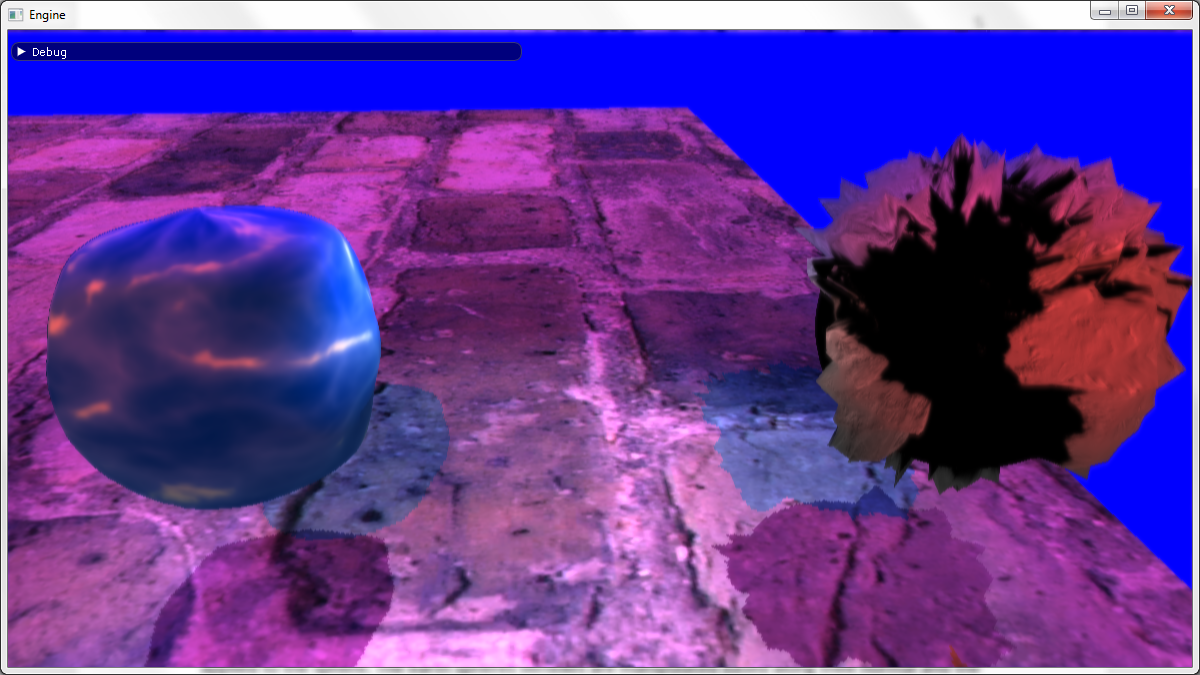
Separated into specific sections to alter the shape that displays a specific graphical technique.

* Tessellation
  + Tessellation Factor
    - Ranges from 1 – 64. Alters the tessellation value that is applied to the insides and edges of each tessellated shape.
    - Is also affected by the position of the camera as the tessellation factor in the shader stages is divided by the distance the camera is away from the mesh.
  + Wave Height
    - Ranges from 0 – 5 as the shape is manipulated along its normal. Alters the height of the wave.
  + Wave Frequency
    - Ranges from 0 – 5. Alters the frequency of the waves appearing on the mesh.
  + Wave Speed
    - Ranges from 0 – 5. Alters how fast the waves moves across the mesh.
* Displacement
  + Displacement Height
    - Ranges from 0 – 1. Alters the vertexes position based on the height map applied to the mesh.
* Depth of Field
  + Range
    - Ranges from 0 – 2. Alters range at which objects around the centre of the depth texture are perceived to be in focus.
* Light Settings
  + Left Directional
    - Ambient
      * Drag on the 4 bars to alter each colour of the ambient accordingly.
    - Diffuse
      * Alter each bar on click on the small square to be presented with a colour picker to set what the diffuse colour will be.
    - Direction
      * Drag on the three bars to alter the direction of the light. There seems to be an issue with the X direction being zero so I tried to implement a rudimentary if statement check to avoid this error. Assert for the X direction being zero being hit.
    - Position
      * Drag on the three bars to alter the position of the light.
  + Right Directional
    - Ambient
      * Drag on the 4 bars to alter each colour of the ambient accordingly.
    - Diffuse
      * Alter each bar on click on the small square to be presented with a colour picker to set what the diffuse colour will be.
    - Direction
      * Drag on the three bars to alter the direction of the light. There seems to be an issue with the X direction being zero so I tried to implement a rudimentary if statement check to avoid this error. Assert for the X direction being zero being hit.
    - Position
      * Drag on the three bars to alter the position of the light.
  + Spot Light
    - Ambient
      * Drag on the 4 bars to alter each colour of the ambient accordingly.
    - Diffuse
      * Alter each bar on click on the small square to be presented with a colour picker to set what the diffuse colour will be.
    - Direction
      * Drag on the three bars to alter the direction of the light. There seems to be an issue with the X direction being zero so I tried to implement a rudimentary if statement check to avoid this error. Assert for the X direction being zero being hit.
    - Position
      * Drag on the three bars to alter the position of the light.
    - Advanced
      * Angle
        + Ranges from 0 – 90. Alters the angle that the spot light is calculated with.
      * Constant Factor
        + Ranges from 0.01 – 1. Alters the constant factor that the spot light is calculated with.
      * Linear Factor
        + Ranges from 0.01 – 1. Alters the linear factor that the spot light is calculated with.
      * Quadratic Factor
        + Ranges from 0 – 0.5. Alters the quadratic factor that the spot light is calculated with.
* Ortho Meshes
  + Render top left ortho mesh
    - Boolean to enable/disable rendering this mesh.
  + Render top right ortho mesh
    - Boolean to enable/disable rendering this mesh.
  + Render bottom left ortho mesh
    - Boolean to enable/disable rendering this mesh.
  + Render bottom right ortho mesh
    - Boolean to enable/disable rendering this mesh.

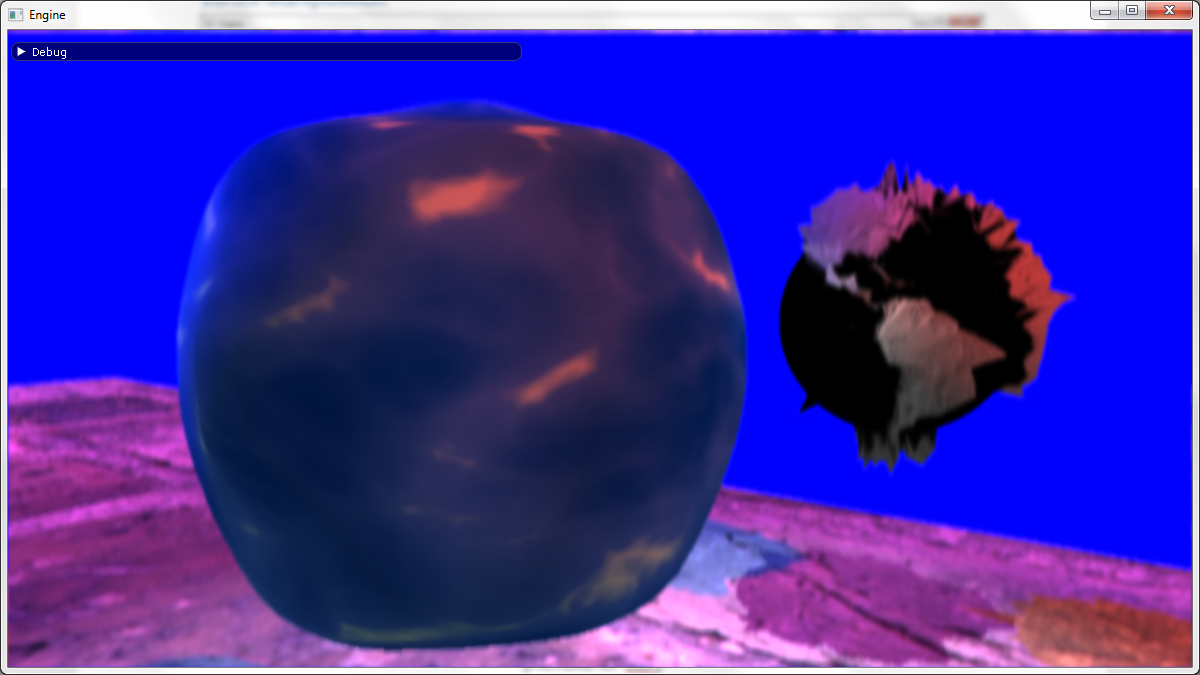
# Algorithms and Data Structures

* Important calculations used, data passed and shader stages.
* This should focus on the hlsl/shaders written.
* Providing diagrams, code snippets and supporting screenshots as required.

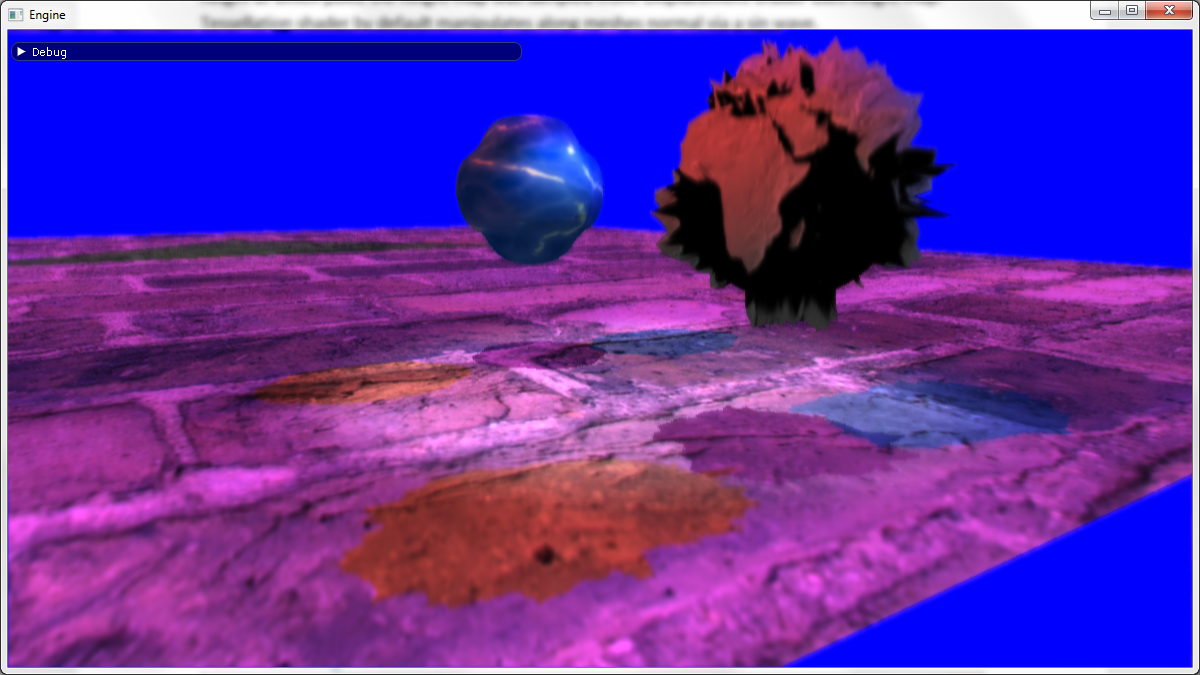
## Vertex Manipulation

**Tessellation and Displacement Shaders.** Both spheres within the scene showcase a different example of vertex manipulation. The water sphere has its vertexes manipulated along their normal via a sin wave. The earth sphere showcases vertex manipulation via a height map which has been applied to the sphere. The earth spheres vertexes are manipulated based along their normal and the height at which point the height map was sampled from. Displacement shader uses height map. Tessellation shader by default manipulates along meshes normal via a sin wave.

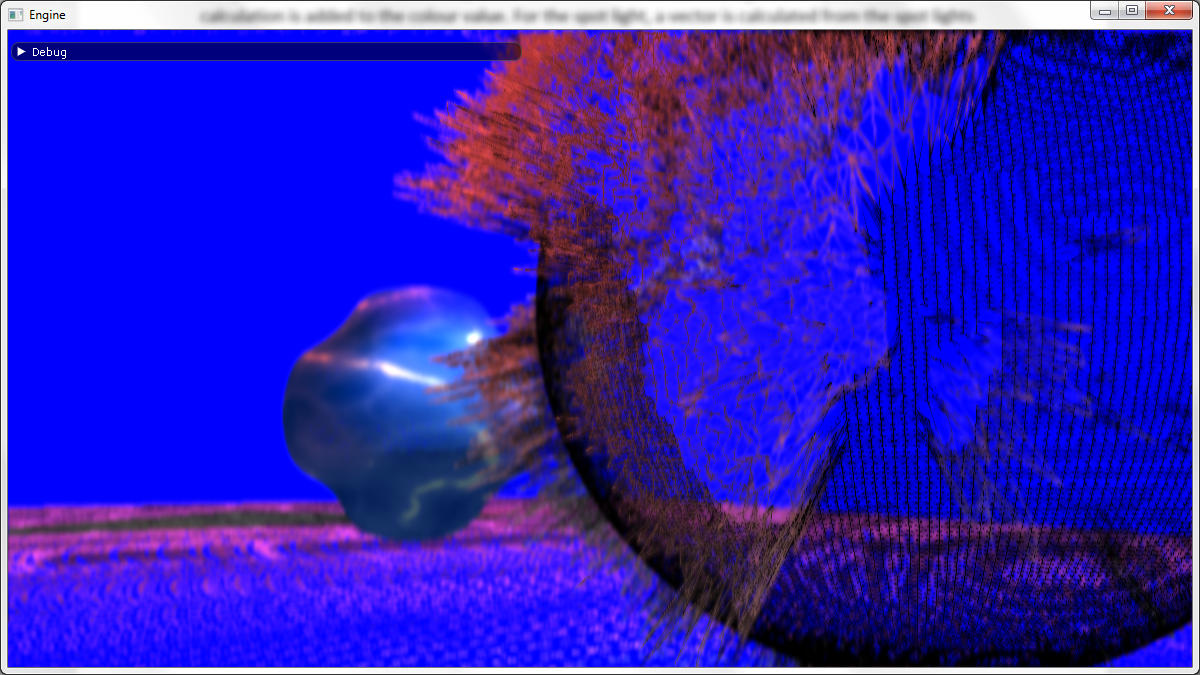
## Post Processing

**Vertical Blur, Horizontal Blur, Depth of Field and Texture Shaders.** The scene utilises the vertical and horizontal Gaussian blur with down/up sampling to apply a final depth of field post processing effect. The depth of field shader takes in the normal scene texture, the combined blur texture after down sampling and the depth texture created from the cameras depth pass. The depth of field shader also takes in three floats, one of which is the range which is user controlled via ImGUI, the other two which are the SCREEN\_NEAR and SCREEN\_DEPTH. These floats plus any padding required are passed into the pixel shader where they are used to calculate the blur factor. The blur factor is used to linearly interpolate between the two textures (normal and blurred), based on what the camera is looking at. The three textures passed in are sampled based on the texture coordinates of the ortho mesh which this shader is applied to. The depth texture is sampled twice, once using the given texture coordinates and another based on the centre of the depth texture. These values are then flipped to equal 1 minus their value to put them in the 0 – 1 range. They are then both multiplied by the result of SCREEN\_DEPTH – SCREEN\_NEAR and passed in to the calculation of the blur factor. **TALK ABOUT REQUIREMENTS.** The result blurs the scene based on what the camera is looking at.

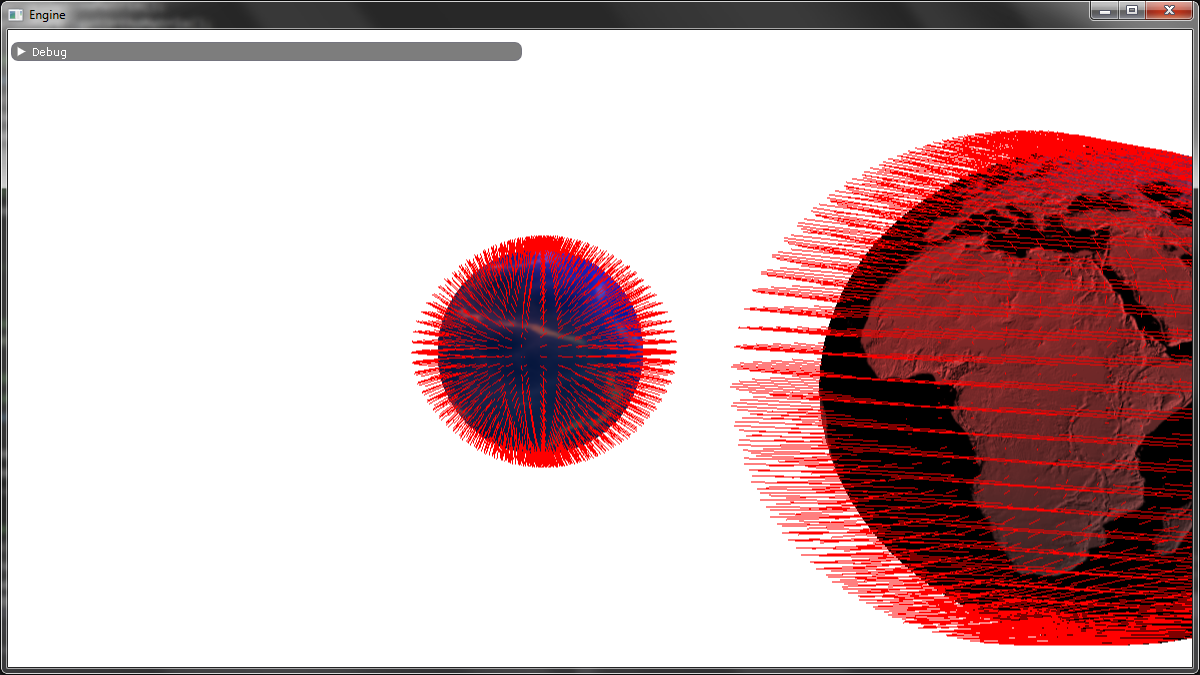
## Lighting & Shadows

**Depth, Displacement Depth and Tessellation Depth Shaders. Also Shadow, Tessellation and Displacement Shaders.** There are three lights within the scene, two are directional and the third is a spot light. Each light has its own shadow map generated via the three depth shaders based on what object it is. Theses shaders are functionally similar to their non-depth counterparts with the main difference being that only the position and depth value are passed to their pixel shaders. These shadow maps are then used within the shadow, tessellation and displacement shaders. The displacement shader uses the same pixel shader as the tessellation shader. The pixel shader is more or less the same between the shadow shader and tessellation shader. Looping for each light, the projected texture coordinates are calculated by dividing the light view position by its W component and then multiplying and adding by 0.5 to get them in the 0 – 1 range. If these values are outside of the 0 -1 range then lighting is not calculated for the specific mesh. The depth value is sampled based on the depth texture and the lights depth value is calculated by is Z component divided by its W component minus the shadow map bias. For the first two directional lights a simple diffuse calculation is added to the colour value. For the spot light, a vector is calculated from the spot lights position and the meshes world position. The length of the spot lights vector is then used to calculate the attenuation factor of the spot light using the values passed in via a buffer. The spot lights vector is then normalised and is used for a basic diffuse calculation which is multiplied by the attenuation calculated earlier. This final spot light colour is multiplied by the intensity which is calculated based on the direction of the spot light, its normalised vector and its angle. The intensity of the spot light is the resultant of the smooth transition between the minimum cos of the angle given in radians, maximum cos of the angle given in radians and the dot product of the spot lights direction and the negative spot light vector. This results in a nice cone being calculated for the spot light. There is an issue with the spot light where it seems to think a part of the plane should be shadowed. From debugging it seems to be either an issue with the spot lights light view position or how lighting/shadowing is calculated in the pixel shader.

## Tessellation

Both spheres within the scene are dynamically tessellated. The tessellation factor which affects the inside and edges of both spheres can be altered via ImGUI. Both spheres are subject to distance based tessellation also which divides the tessellation factor passed into the domain shader by the distance each mesh is away from the camera.

## Geometry Shader Stage

**Render Displacement Normals and Render Tessellated Normals Shaders.** A rather primitive use of the geometry shader stage but it is useful none-the-less. Functionally similar to their Displacement and Tessellation shader counterparts. Main difference being their geometry shaders output a line stream and have a max vertex count of 2 compared to outputting a triangle stream with a max vertex count of 4. Looping for each input point for the triangle two points are appended before restarting the line strip. The first point of the line is at the vertex position, the end of the line is based on the vertex position plus the normal position multiplied by the length of the normal. This then creates lines around the sphere showcasing all of its normals. There is a weird issue though where to lines seem to rotate with the camera. The main suspect would have to be the calculation used.

# Critical Reflection

* Discussion of what you learned, any shortfalls, areas for improvements, how you might extend the application, what you would do differently if doing it over again.
* Offer possible solutions to the challenges or how it could be taken forward.
* This should refer to the coursework plan done during week 7/8.
* How would you take it forward? Errors/success, where are the flaws?

In short, a lot was learned. During this module many new concepts have been introduced and a few took previous knowledge further. Vertex manipulation, tessellation and the geometry shader are all new concepts but after learning about them they are some of the most important. Furthering previous knowledge from OpenGL with the lighting and post processing effects that we can now implement in greater detail using DirectX or re-implementing via glsl in OpenGL again. Areas for improvement include starting work on the project sooner and managing time better as a base line. There is an issue with the spot lights shadow calculation would could be resolved if there was more time. Reducing the branching within the shaders would be the next step. Increasing the number of neighbours used in the blur shaders and possibly introducing a dynamic weight calculation for each neighbour. Use of texture arrays within shaders that have multiple textures being sent to them. Furthering the current implementation from pure diffuse calculations a blinn-phong or cook-Torrance model could have been implemented to showcase specular highlights on the water sphere. The dynamic tessellation could be improved via ensuring that the vertices don’t tear as a result of the distance-based tessellation. Also, sampling the points around a specific patch to check if tessellation would make much of a difference. Doing more complex processes within the geometry shader would be a good start. Such as bill boarding or doing a water droplet effect. Extensions for the application would include the implementation of the shader blob put forth in the proposal, this would showcase reflection as a post processing effect, but refraction could also be applied to the water sphere. The introductions of more spot lights or a point light to further push the boundaries within the depth calculations and pixel shaders. If the project was to be redone, the major points would be to break down and plan as far as possible each shader that was being sought after. Gather information from multiple sources (books are a thing, ha-ha). Utilise available time better.

# References

Digitalerr0r. 2009. XNA Shader Programming. [ONLINE] Available at: <https://digitalerr0r.wordpress.com/2009/05/16/xna-shader-programming-tutorial-20-depth-of-field/> [Accessed 09/12/2018]

Joao Paulo. 2017. Water 001. [ONLINE] Available at: <https://3dtextures.me/2017/12/28/water-001/> [Accessed 09/12/2018]

Aster. 2004. GDEM V1 Black & White Map. [ONLINE] Available at: <https://asterweb.jpl.nasa.gov/gdem.asp> [Accessed 09/12/2018]

Ocornut. 2018. ImGUI. [ONLINE] Available at: <https://github.com/ocornut/imgui> [Accessed 09/12/2018]

Gabriel Lacey – Gave some pointers about implementing the depth of field shader.

Paul Robertson – Texture, Shadow, Vertical and Horizontal Blur shaders which have either been extended or used outright.