COMP3011 Computer Graphics

Spring 2025

Assessment 2

Report Sheet

This report will help you prepare for your demo. **Submit this report to Moodle**.

**Student Name: Toby Surtees**

**Student ID: 20477669**

**Username: psyts20**

**I agree for my code & report to be published, with my name, to future students as an example (yes/no): no**

|  |  |  |
| --- | --- | --- |
| **Introduction** | | |
| *Please explain why you implemented this scene* | *Describe your inspirations* | *Provide a general description of the scene.* | |
| I implemented this scene to act as a unique way to demonstrate several simple to advanced computer graphics techniques using several different types of objects. This scene uses a low lighting setting to better demonstrate shiny surfaces in simulated moon light, while encompassing different glowing objects to showcase interactions between light sources and effect on transparency, which interactive elements to remove said light sources for a different effect. | *The inspiration for this scene were mainly from Egypt and the pyramids, as well as folklore regarding how the pyramids were created, specifically the theory that aliens built them, to put a more interesting spin on a typical desert scene. This also helped to showcase more lighting elements with an interactive UFO.* | *The scene shows procedurally generated desert dunes in a starry night sky. On the dunes there is a 3D pyramid surrounded by a transparent cone of light leading up to a large, spinning UFO. The UFO is producing a glowing green colour, while projecting 4 red spinning spotlights on the ground around the pyramid. Circling the UFO is a jet plane. In the background there are some Egyptian ruins showcasing different transparent layers, aimed to represent stained glass, and sitting on those ruins is a large ancient vase. Looking up at the sky we see procedurally generated, animated shooting stars.* | |
| **TR 2 – 3D Modelling** | | |
| Object 1 - procedurally generated | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of object* | |
|  | *cylinder.h (all)*  *Assessment2.cpp (845 – 899, 1427 - 1437)*  *plane.h (all)*  *Assessment2.cpp (955 - 964, 1270 - 1280)*  *tangent.h (all)* | *The first object is a Cone / Cylinder shape, representing a beam emitting from the UFO. This is generated in cylinder.h using two radius values to determine the size of each end of a cylinder, forming a cone shape. The shape is made of two circles at each end, surrounded by a specified number of strips which make up the sides of the shape. Also implemented is a gradient colour, from green to blue, to represent the travel distance of the beam, as well as a gradient of transparency which decreases from top to bottom.*  *The second object is a plane of desert dunes, generated using a grid of triangles which form together to create a flat plane. I then manipulate this plane using Sine waves to change the height of each triangle, creating the dune effect. I also implemented roughness to make the waves appear more interesting and random.* | |
| Object 2 - OBJ parser | | |
| *Please give a screenshot* | *Provide the URL for the OBJ files you submitted* | *Description of object.* | |
|  | *Jet Plane:* [*https://www.cgtrader.com/free-3d-models/aircraft/military-aircraft/dassault-rafale-b-bdc1590e-5936-4912-ba44-79f5c2e09f07*](https://www.cgtrader.com/free-3d-models/aircraft/military-aircraft/dassault-rafale-b-bdc1590e-5936-4912-ba44-79f5c2e09f07)  *UFO:*  [*https://www.cgtrader.com/free-3d-models/space/spaceship/free-flying-saucer*](https://www.cgtrader.com/free-3d-models/space/spaceship/free-flying-saucer)  *Ruins:*  [*https://sketchfab.com/3d-models/asset-pack-egypt-low-poly-128dbc9ec9d24b58bdd1d9e4ed49fc17*](https://sketchfab.com/3d-models/asset-pack-egypt-low-poly-128dbc9ec9d24b58bdd1d9e4ed49fc17)  *Vase:*  [*https://www.cgtrader.com/free-3d-models/architectural/decoration/flowervase*](https://www.cgtrader.com/free-3d-models/architectural/decoration/flowervase)  *Stone Texture:*  [*https://freepbr.com/product/stone-block-wall/*](https://freepbr.com/product/stone-block-wall/)  *Sand Texture:*  [*https://3djungle.net/textures/sand/1779/*](https://3djungle.net/textures/sand/1779/)  *Skybox:*  [*https://tools.wwwtyro.net/space-3d/*](https://tools.wwwtyro.net/space-3d/index.html#animationSpeed=1&fov=80&nebulae=true&pointStars=true&resolution=1024&seed=6c6p72hxz5a8&stars=true&sun=true) | *The first object shown is a Jet Plane rendered using an .OBJ file and .mtl for texture mapping and colours. The parser from Assessment 1 has been extended to support tangent space calculations, textures and colours as well as automatic translations into vectors of floats rather than triangle objects for use with OpenGL buffers.*  *Loaded objects are rendered with Shadow and react to lighting, as seen by the green reflections from the positional light in the image with the plane. Objects also support transparency as shown in the image with the UFO. The Ruins object is used to showcase my PBR shader as a matte colour object, while the vase is used to showcase a shiny PBR object.* | |
| **TR 3 – 3D Transformations** | | |
| Object 1 - procedurally generated | | |
| *Please give a screenshot of transformed object* | *reference specific code (filename and line)* | *Description of transformations* | |
|  | *Assessment2.cpp (845 – 899)* | *Here we can see the cone (beam) portrayed as leaving the base of the UFO. It has been scaled, translated and rotated in order to properly fit under the UFO and cover the pyramid. When the UFO is clicked, the cone is then transformed in an animation to reduce in scale first in the Y and then X and Z to represent shrinking.* | |
| Object 2 - OBJ parser | | |
| *Please give a screenshot of transformed object* | *reference specific code (filename and line)* | *Description of transformations* | |
|  | *Assessment2.cpp (1058 – 1074)* | *The vase has been scaled down to fit on the pedestal in the ruins object, rotated to face straight on the pedestal and translated to the correct position.* | |
| **TR 3 – Animation** | | |
| *Please give a screenshot of animated object* | *reference specific code (filename and line)* | *Description of animation* | |
|  | *Assessment2.cpp (975 - 1003)* | *The plane object has been animated to circle the UFO in a consistent radius by translating it over time. The plane is automatically rotated to bank towards the UFO to simulate a real plane in flight.*  *The UFO also rotates in place, as do its projected spot lights.* | |
| **TR 4 – Camera** | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of camera* | |
|  | *camera.h (all)*  *Assessment2.cpp (28-52, 240-326, 351-375, 391-414, 747 – 778, 1598-1618)* | *There are a total of 7 cameras implemented in this scene. The first is a simple Model-Viewer camera that follows a central point in the scene and can be manipulated using the arrow keys and R / F for zoom. The second (Pictured) is a first-person fly-through camera that can be controlled using the mouse to change view direction, and WSAD for movement. Next is an animated rotating camera that circles the scene, its speed can be changed with the square bracket keys. Finally, are 4 fixed cameras that are set at each corner of the scene.* | |
| **TR 5 – Texture** | | |
| Object 1 - procedurally generated | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of texture* | |
|  | *texture.h (all)*  *lighting\_vertex.vert (all)*  *lighting\_fragment.frag (82-117, 380-390)*  *Assessment2.cpp (1005 – 1039, 1133-1450, 1509-1560)*  *Skybox.frag (all)*  *Skybox.vert (all)* | *The first image shows the sand dunes with a sand texture. This texture uses an array of mipmaps from 1x1 to 1024x1024 mapped to texture coordinates.*  *The second image shows a skybox using cube mapped textures to create a realistic and seamless sky effect. This is achieved by binding 6 separate textures to a cube object and applying depth properties to ensure the cube stays in position in the camera, so it always looks far away.*  *The third image shows a texture applied to the UFO object, which uses a normal map for raised details, a glow map for enhanced glowing areas such as the green lights, a specular map for shiny areas and a bump map, again for smaller details. This creates the detailed effect you can see in the image.* | |
| **TR 6 – Lighting** | | |
| Object 1 - procedurally generated | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of lights on object* | |
|  | *lighting\_vertex.vert (all)*  *lighting\_fragment.frag (252-376, 642-658)*  *Assessment2.cpp (1133-1198)* | *Three lights can be seen on this object. The first and most obvious is the green positional light, which is positioned directly above the pyramid in the image, casting a green glow. The second are the red spotlights that circle the pyramid, which cast a red spot on the ground, calculated using a cutoff angle for a sharper shape and a faded colour as distance increases from the source. The red glow can also be seen cast down the side of the pyramid, reflecting off the parallax mapping. Finally, is a directional light positioned at a distance from the scene which casts a yellow shine over the entire scene, contributing to reflections and overall colour depending on where the light is positioned. Each of these lights use Blinn-Phong lighting.* | |
| Object 2 - OBJ parser | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of lights on object* | |
|  | *lighting\_vertex.vert (all)*  *lighting\_fragment.frag (393-576, 613-641)*  *Assessment2.cpp (1339-1371)* | *This object is implemented using Physically Based Rendering (PBR) which more accurately simulates the properties of different materials. The directional light can be seen producing a nice shine on the side of the vase, while the positional light contributes to this shine by adding a green tinge. The lights can be seen reflecting off small imperfections on the object, as well as realistically wrapping round the handles. In this screenshot, we can also see a matte PBR object reacting differently to the light. Normal mapping is also used here to improve depth perception.* | |
| **TR 7 - Shadow** | | |
| Object 1 - procedurally generated | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of shadow on object* | |
|  | *lighting\_vertex.vert (all)*  *lighting\_fragment.frag (213-250)*  *shadow.frag (all)*  *shadow.vert (all)*  *Assessment2.cpp (1105 – 1450)* | *By moving the directional light to a low-down position, we can see the shadows cast by the procedurally generated dunes. This is achieved using shadow mapping to determine whether an object is in shadow. This effect is enhanced by using Percentage Closer Filtering (PCF) to soften the edges of the shadows by sampling neighbouring texels and getting an average colour result.* | |
| Object 2 - OBJ parser | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of shadow on object* | |
|  | *lighting\_vertex.vert (all)*  *lighting\_fragment.frag (213-250)*  *shadow.h (all)*  *shadow.frag (all)*  *shadow.vert (all)*  *Assessment2.cpp (1105 – 1450)* | *This image shows the shadows projected from the UFO and Plane objects, produced by the directional light. These shadows are projected onto the ground and stretched by the unlevel surface, producing a more realistic effect.* | |
| **TR 8 - Interactive object** | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of interactive object* | |
|  | *Assessment2.cpp (327-349, 854-900, 901 – 953, 1177-1186)*  *interactivity.h (all)* | *The interactive object shown is the UFO, light beam (cone) and associated lighting elements (positional and spotlights). By simulating a sphere around a central point, set at the middle of the UFO, I can then check if the user’s mouse is within that radius when a mouse click is registered.*  *If this is detected, the lighting elements mentioned are removed from the scene by changing the associated uniforms. As well as this, a shrinking animation is played on the light beam, making it disappear into a point. The UFO is then translated away into the distance, while reducing its size to simulate a further distance. Size is then reduced to 0 to remove it from the scene.*  *We can see the difference without the extra lighting in the scene and observe differences in colour contribution at each object.*  *The scene can be reset by pressing E, returning the ship, beam and lighting to the scene.* | |
| **TR 9 – Curves** | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of curves* | |
| GIF | *Assessment2.cpp (779-803)*  *casteljau.h (all)* | *This implementation of Bezier curves is used to simulate shooting stars moving across the sky. Each shooting star's path is generated procedurally using four randomized control points, forming a smooth, curved trajectory. The curve is evaluated into points and rendered as a GL line strip to create streaks in the sky. This is animated by progressively drawing segments of the line, creating the illusion of motion. Once the animation completes, a new path is generated, making each shooting star appear unique and dynamic.* | |
| **TR 10 – Transparency** | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of transparency* | |
|  | *Assessment2.cpp (205 – 227, 845-945, 1075-1104, 1374-1449)* | *The first example of transparency is the three panes in the ruins. Each pane has a different colour, and by looking through the first pane we can see the colours combining as each pane contributes to the final colour. This was achieved by defining three simple squares with alpha values, and disabling depth masking while rendering, so that other objects appear correctly, as well as rendering opaque objects first, then rendering transparent object back-to-front to enable multiple transparent objects.*  *The second example is the light beam (cone) which uses a gradient of alpha values to change its transparency from top to bottom, simulating strength declining over distance.*  *Finally, the UFO uses transparent textures so you can see into the cockpit and into the ship through the side windows.* | |
| **R&D**  Please provide details of any research and development you conducted, as additional techniques not in the lecture notes. | | |
| *Please give a screenshot* | *reference specific code (filename and line)* | *Description of Research including websites, articles, references, etc.* | |
| PCF:    PBR:    POM:    Blinn-Phong: | ***Percentage Closer Filtering (PCF):***  *lighting\_fragment.frag (212-249)*  ***Physically Based Rendering (PBR):***  *lighting\_fragment.frag (393-576)*  ***Parallax Occlusion Mapping (POM):***  *lighting\_fragment.frag (120-207)*  ***Blinn-Phong Lighting:***  *lighting\_fragment.frag (253-376)* | ***Percentage Closer Filtering:***  *Sources:* [*https://developer.nvidia.com/gpugems/gpugems/part-ii-lighting-and-shadows/chapter-11-shadow-map-antialiasing*](https://developer.nvidia.com/gpugems/gpugems/part-ii-lighting-and-shadows/chapter-11-shadow-map-antialiasing)  [*https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping*](https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping)  *PCF improves shadow mapping by sampling nearby texels and forming an average value for softer, more realistic looking edges.*  ***Physically Based Rendering:***  *Sources:*  [*https://learnopengl.com/PBR/Theory*](https://learnopengl.com/PBR/Theory)  [*https://learnopengl.com/PBR/Lighting*](https://learnopengl.com/PBR/Lighting)  *PBR aims to replicate how light interacts with real-world surfaces by using physical properties like albedo, metallics, roughness, and ambient occlusion.*  *I implemented a Cook-Torrance BRDF model to simulate surface reflectance, simulating roughness, shadowing and specular highlights.*  *Material appearance is influenced by metallicity, roughness, and ambient occlusion, allowing for realistic variation between shiny metals and rough materials. I also used HDR tone mapping and gamma correction for more control over the scene.*  ***Parallax Occlusion Mapping:***  *Sources:*  [*https://learnopengl.com/Advanced-Lighting/Parallax-Mapping*](https://learnopengl.com/Advanced-Lighting/Parallax-Mapping)  [*https://rdmilligan.wordpress.com/2018/03/09/parallax-mapping-with-opengl-and-c/*](https://rdmilligan.wordpress.com/2018/03/09/parallax-mapping-with-opengl-and-c/)  *POM is used to simulate depth and surface detail by offsetting texture coordinates based on a height map and the viewer's perspective, adding visual complexity without increasing mesh detail.*  *I used ray marching and a height map in tangent space to find where the view ray intersects the virtual surface layers. Once the intersection is found, I perform linear interpolation between the last two steps for smoother results. These texture coordinates are then used to sample a normal map, which is transformed back into world space for realistic lighting interaction.*  ***Blinn-Phong Lighting:***  *Source:*  [*https://learnopengl.com/Advanced-Lighting/Advanced-Lighting*](https://learnopengl.com/Advanced-Lighting/Advanced-Lighting)  *Blinn-Phong improves on Phong lighting by computing specular highlights using a halfway vector between the light and view directions. This results in more realistic looking lighting with better specular highlights and shininess.* | |