COMP1685 Feedback Sheet

The following is to help assess your work and give feedback for the individual components. However, the grading criteria override it – so a report which is missing a section will not be awarded more than 49%, even if the marks from the marking scheme indicate a higher mark.

Name: Nicholas Allen **Overall Mark:**

Marking scheme	**achieved well**	**partially achieved**	**not achieved**
Report & deliverables (approx 40%)			
deliverables uploaded as required, report laid out correctly and appendices as requested			
an account of how the scene was designed			
a clear description of the user interaction			
an informative critical analysis			
a scene-graph of the code (approx 15%)			
Visual design of the scene (approx 15%)			
good use of shape			
good use of lighting			
good use of materials			
Interaction design of the scene (approx 15%)			
effective interaction via the keyboard			
effective interaction via the mouse (including the GUI)			

Coding (approx 10%)			
well designed & efficient code (e.g. loops, methods)			
clear commenting			
Imaginative use of 3D graphics techniques (approx 20%)			
via interaction			
via other features			

Feedback:

3D Computer Graphics – Report

Account of how the scene was designed

When designing the scene, the first stage was to come up with an idea for the scene and to sketch out some rough drawings on paper. The idea was formed around a future where humans are long gone and due to invasion from an alien species, only 2 robots remain. In order to regain control the 2 robots must find each other and unite to drive the alien invaders away.

The idea came to have a robot whose position could be controlled by the user, who would have to direct the main robot to the secondary robot. Once the two robots were within a set proximity to one another, they would start to rotate in unity, which would cause the alien invaders to leave the scene.

To start the process, several sketches were made to illustrate the layout of the scene, positioning of objects within the scene, and to work out what form the robot would take. After these initial sketches were completed, a rough scene graph was created to help work out the hierarchy of objects in the scene, and to help with establishing the parent-child relationships between the objects.

For each 3D object, groups, geometries and materials were decided and documented. This was a very useful process as it allowed a higher-level visualisation of how the scene was formed. Before starting work developing the 3D objects, a camera and directional light was added to the scene, as well as a plane that would be used as the flooring.

At this point, it was time to start developing the robot. The previously documented groups were added to the scene, and child groups were added to their parent group. A new THREE.cubeGeometry was added, and also a grey coloured THREE.MeshLambertMaterial that would be as the material for the base shapes of the robot. The cube geometry was set to with values x=1, y=1, z=1, allowing for the same geometry to be used several times, which was scaled to create the correct size mesh for the various parts of the 3D object.

A mesh was created for each part of the robot, and these meshes were added to the correct groups, which were then positioned to form the shape of the robot. Some objects, such as the arms and legs, were generated in loop; this was done to avoid repeating code over and over to fulfil the same task. When creating multiple 3D shapes in a loop, the objects were added to a group, which was then rotated on the correct axis and positioned, essentially giving a mirror effect.

Once the robot was created, it was added to the scene, and positioned in the correct place on the x, y, z-axis. At this point, the code used to generate the robot was added

to a new function called makeRobot(). This was done so the code for this particular task was separated from the rest of the source code, improving the overall design and readability of the code. To make the robot, this function was then called before the render method.

It should be noted that several geometries and materials were used for the construction of the robot, not just those previously stated. This includes using sphere and cylinder geometries, which were used for forming the joints on the arms/legs and smaller features, such as the eyes and mouth. In some cases, new geometries were created where existing geometries could have been used and scaled instead. This was purposely done to show competence with both techniques.

After creating the first robot, a second function was added to create the second robot, the code used for this is similar to the first makeRobot() function, however this robot has a different material, colour etc.

Having created the robots, work was undertaken to create the ufo's that would rotate and revolve around the scene. Groups, geometries and materials were created for the various shapes required and the 3D objects were created using similar techniques to those listed above. In addition, a function was created to rotate and animate the movement of the ufo's. This function was amply named rotateUfos(), and was added to the render function.

The next stage was generating and adding the roads and houses to the scene. This was mostly carried out using simple cube geometries, which were reused for the various 3D shapes. The road is a simple grey coloured LambertMaterial, and a loop was used to add lines to the road. To achieve the spacing between the lines on the road, a spacer variable was created, which was added to the z-position when adding the line to the scene.

The houses were again constructed using simple geometries, added to a house group. Once one house was built, a loop was used to build the houses on both sides of the road, again making use of a variable to space the shapes correctly.

Once the basic scene was completed, controls were added, and functions were written to animate, and allow user interaction with the scene. These are described in the next section.

Description of the user interaction

Robot Interaction

The most significant part of the user interaction is the ability to move the robot using the wasd keys, which allows the user to move the robot around the scene. If

the user moves the robot onto to path on either side of the street, the robot y-position increases to give the impression the robot has stepped up onto the path. Also while moving the robot around the scene, if the user moves the robot too far on the x-axis (using a,d keys), the robot will hit the boundary. This will cause the robot to float up into the air, and then be placed at the original position.

When the user successfully moves the robot to the other end of the road, and meets the other robot, both robots start to rotate. This marks the unity of the robots, and drives the ufo's out of the scene. Once free of ufo's, the user can continue to move around the scene or refresh the page to start again.

Finally the user is able to change the speed the robot moves by adjusting the slider on the Gui on the scene. This slider is titled "changeRobotSpeed", and will adjust how far the robot moves per when the wasd keys are pressed.

Ufo Interaction

The user is able to adjust the speed that the ufo's rotate and revolve using the slider on the Gui. The slider is titled "changeUfoRotation".

Camera Interaction

The user is able to interact with the scene camera in a number of different ways. The most basic is by using the arrow keys to move the camera forward, backwards, or rotate the camera left and right. In the Gui on the top right of the scene, the user can also switch the camera into first person mode or set the camera to view the scene from either end of the road (Behind main robot, in front of main robot).

Lighting Interaction

The user is able to interact with the lighting of the scene via the Gui. There are 2 options, one is a colour picker for the lighting of the scene and the other is a slider that adjusts the intensity of the light.

Analysis of difficulties encountered

Ufo beam opacity: I encountered issues with setting the opacity of the beam coming from the bottom of the ufo's. After looking over the lecture notes, I noticed that the transparency attribute of the material needed to be set to true to allow the opacity to be adjusted, viewed.

Robot foot: Each part of the robot leg was added to a group and then the legs were generated in a loop, and rotated accordingly. This led to one of the feet facing in the wrong direction. This was solved by checking the value of the loop counter and for the second leg, adjusting the positioning of the foot.

The final difficulty, which remains unsolved, is interaction between the ufo's and the main robot. Initially, when the robot got too close to a ufo, the robot was supposed to be abducted and returned to the starting position. However, it was discovered that as the ufo's are not added to the scene and instead are added to a group, added to another group, the x, z position corresponds to the x, z position from the group the ufo is added to.

It was realised that to get the true x, z position of a ufo, calculation of the x, z position of each group would need to be calculated with relative to the containing group. These results would then be used to work out the position of a ufo, in relation to the scene. It was decided to skip doing this in order to focus more time on the Final Year Project.

Appendix A – Screenshots of Scene

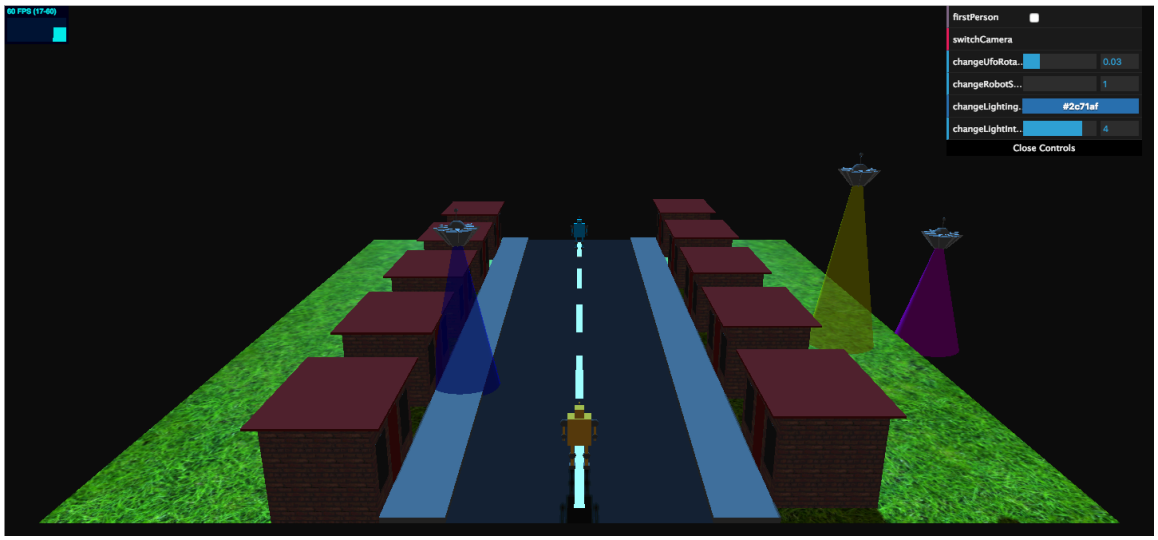

Fig 1.1: Starting view of scene

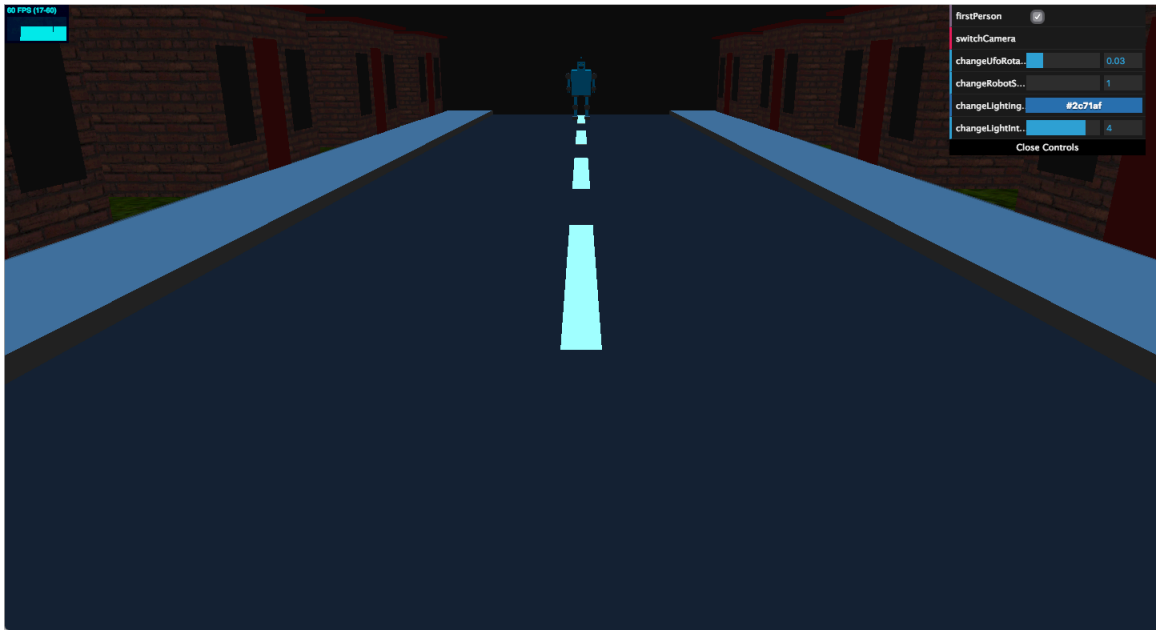

Fig 1.2: First person camera view

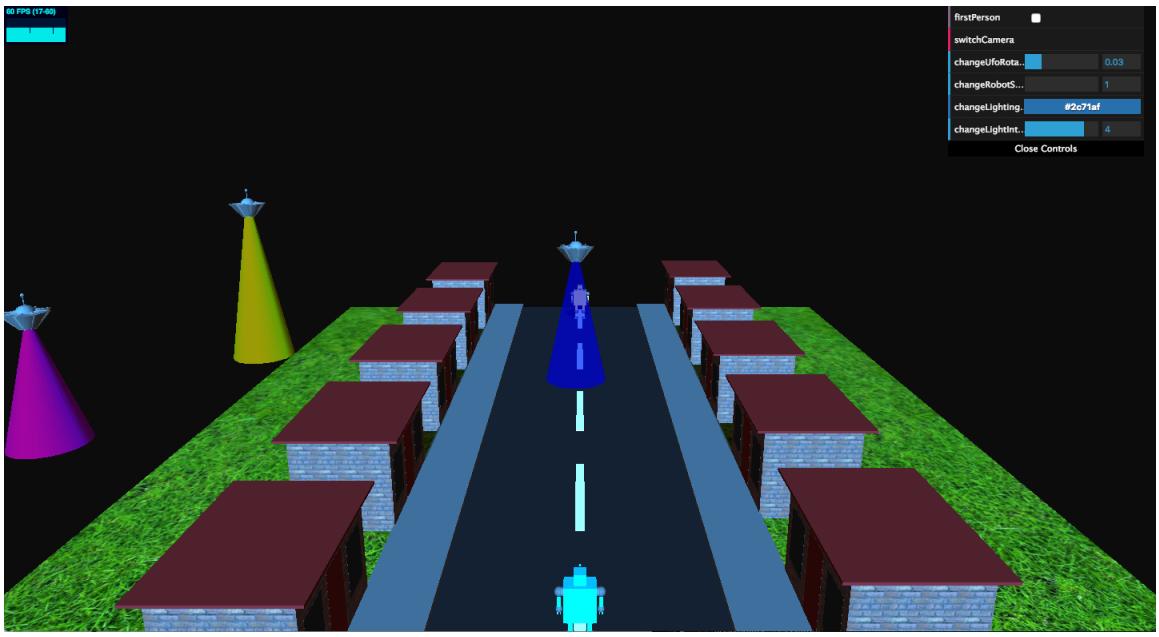

Fig 1.3: Switch camera (camera from front)

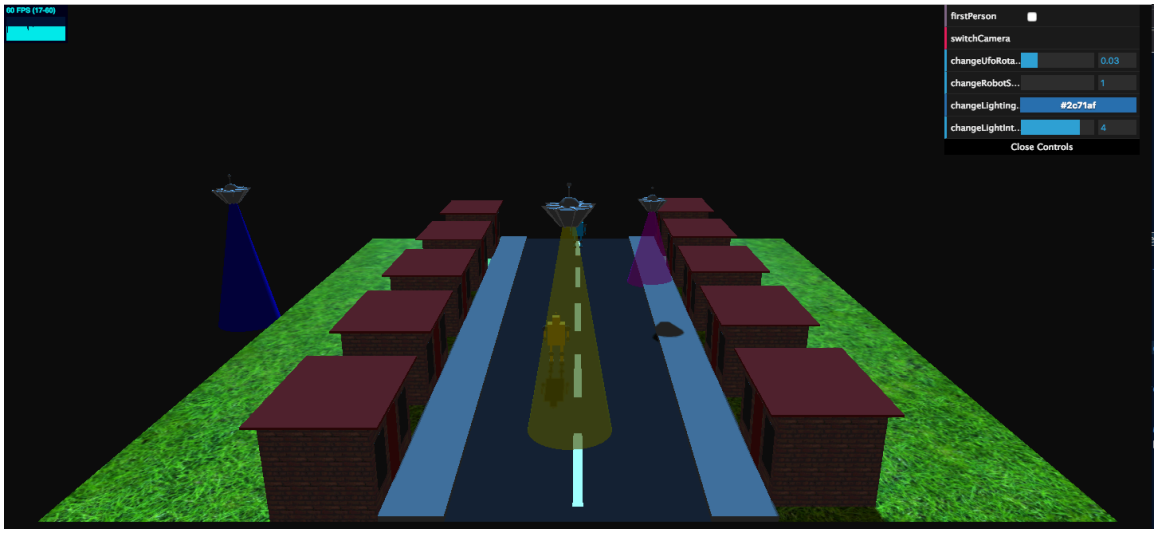

Fig 1.4: Moving robot using wasd keys

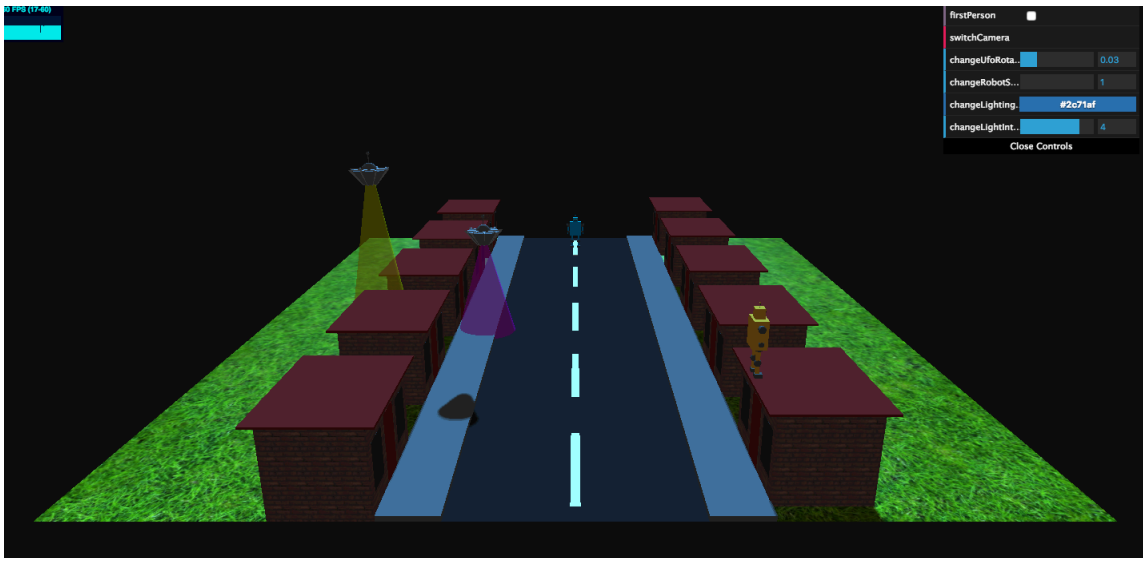

Fig 1.5: Out of bounds

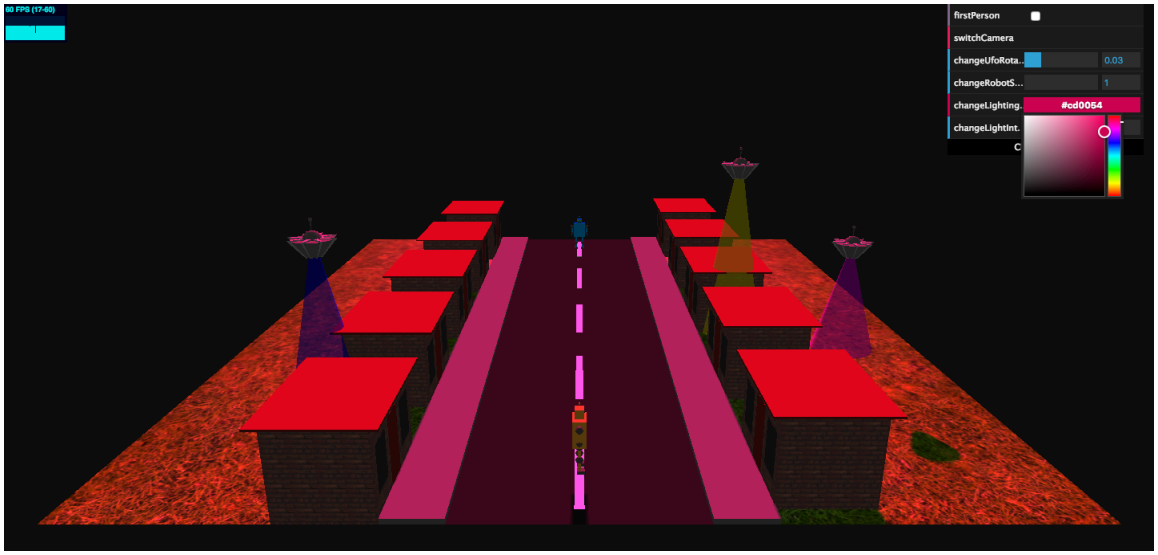

Fig 1.6: Changing lighting colour

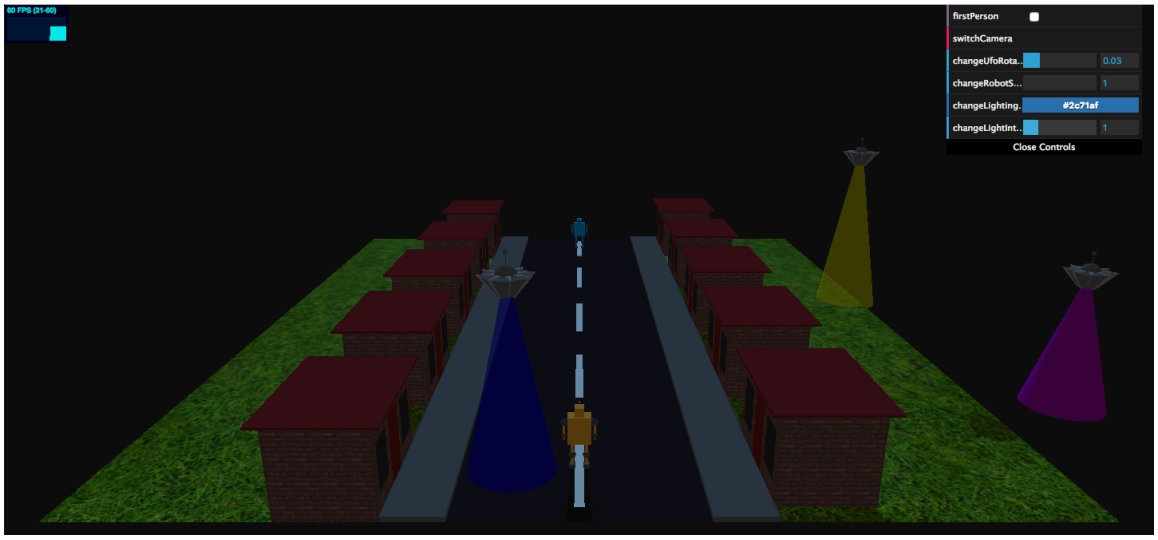

Fig 1.7: Changing lighting intensity

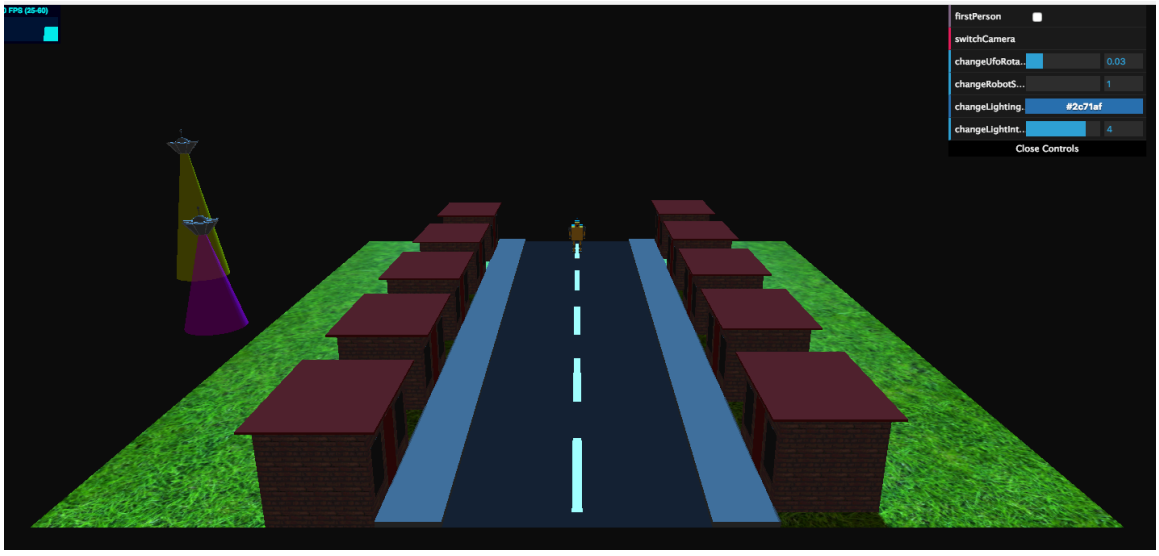

Fig 1.8: Robots meeting, ufo's leaving scene

Appendix B – Scene-Graph