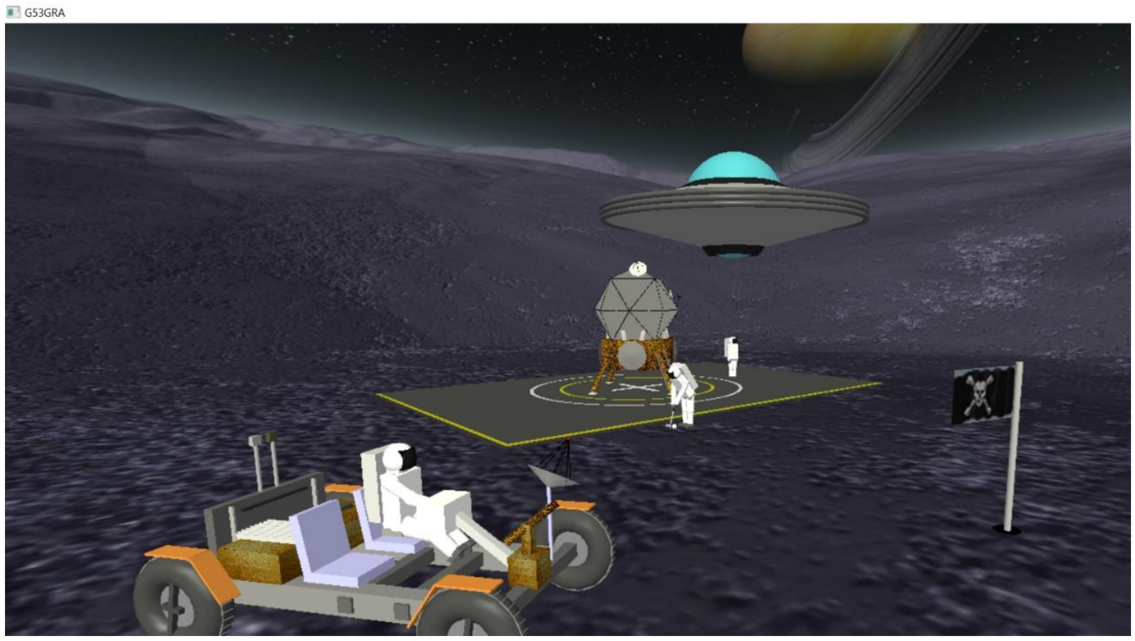
George Castle psygwc 4287671

G53GRA – Final coursework report

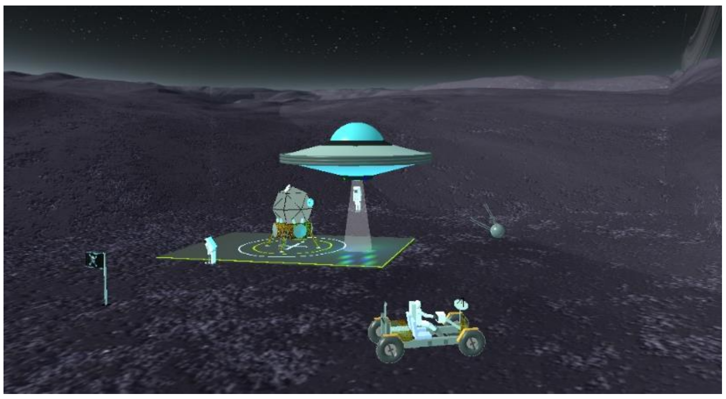
For my final coursework, I created an extra-terrestrial scene featuring significant events in space exploration. The scene includes; Sputnik 1 the first artificial Earth satellite, the lunar rover from Apollo 15, the lunar module from Apollo 11, the space X landing pad and the infamous golf swing from Apollo 14. I also decided to add an extra event, the abduction of an astronaut by an unidentified flying object, this addition was mainly to utilise some of the OpenGL lighting techniques.

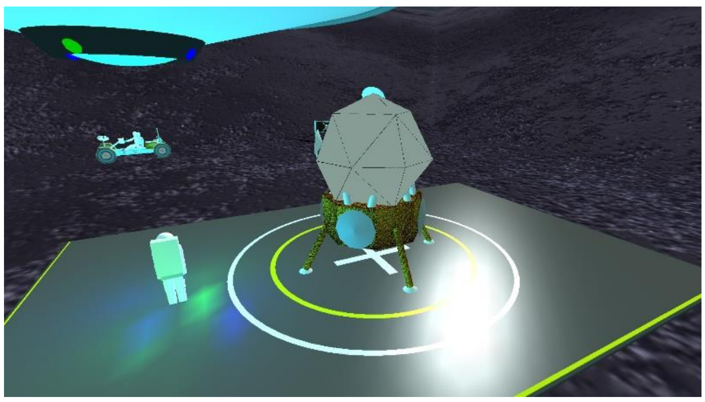


User instructions for operating scene:

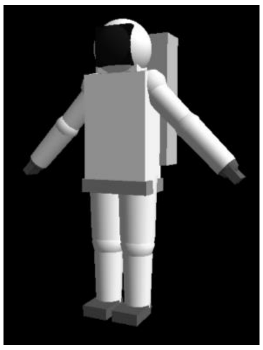
- Up arrow – to make the rover accelerate
- Down arrow – to make the rover decelerate
- ‘p’ – to make the UFO appear
- ‘o’ – to abduct the astronaut (when UFO is in place)
- ‘2’ – to toggle blue lights
- ‘3’ – to toggle green lights
- ‘0’ – to toggle all positional lights
- ‘v’ – to switch to first person (space bar to reset)
- ‘c’ – to switch to car view (space bar to reset)
- ‘i’ – to hit a hole in one

I began by creating the skybox for my scene, I did this by creating a large box around the camera, with only the inside walls visible. I then textured this box using a space texture pack I found on the internet. I lined up the images, using the texture coordinates to make the scene look less like a box, creating a 360-degree horizon. I then created the models.



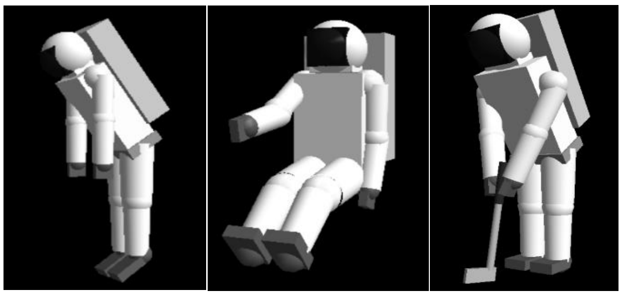


Hierarchical Modelling



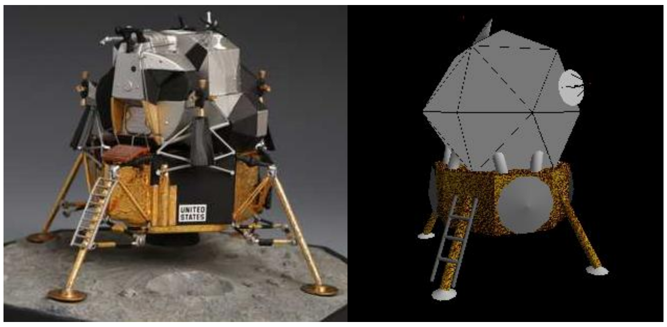
I started by creating a model for the abducted astronaut using hierarchical modelling. First, I created the legs, this was so that I could rotate the rest of the body independently to make the abduction look more realistic. Each leg is made of two cylinders, two spheres and a box for the feet. Using multiple shapes allows for a more realistic model. I used hierarchical modelling as it allowed me to add different animation points which could affect different limbs in a modular way.

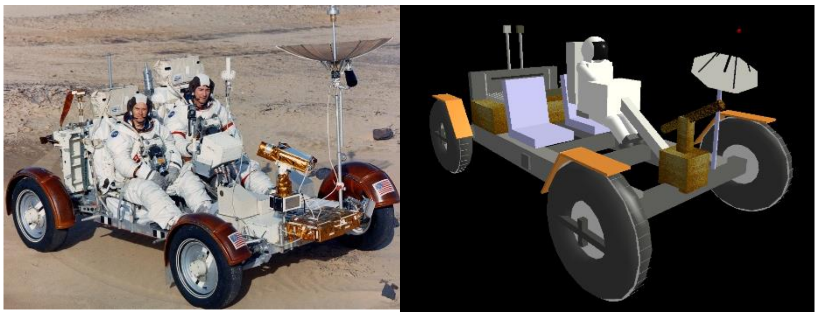
I used the same base astronaut model for the golfer and lunar rover driver as I could change the animation points depending on what motion I desired. Below are the three different positions I have placed each astronaut in.

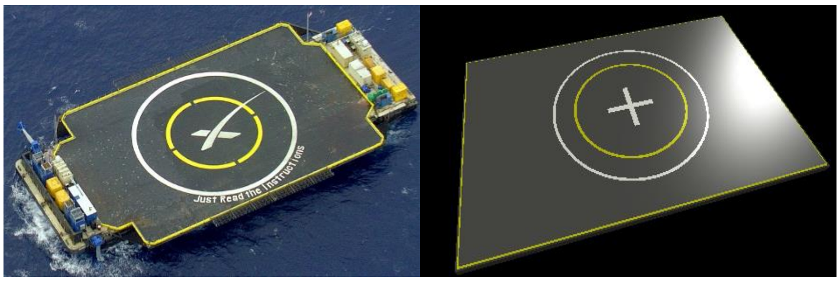


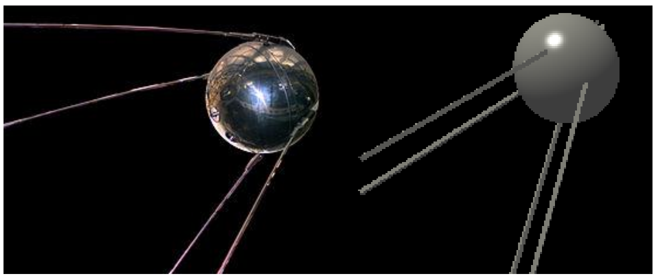
I also used hierarchical modelling for the other models in the scene. For example, on the lunar rover I used hierarchical modelling to create the rotating wheels, satellite and telescope. I did this by creating a function that draws the wheels and one that draws the satellite dish, I then call these functions from the body of the main draw function after adding a rotation point, allowing the whole wheel to be rotated at the same time, instead of having to make each component rotate on its own.

For the modelling of the 'MoonLander' I used surface subdivision whilst only calling the recursive function once, this was to create a structured sphere shape for the top of the lunar module, similar to the top of the real module used in the Apollo 11 mission. Below I have compared some of my hierarchical models to the real structures which inspired them.









Animation

Almost all the objects in my scene contain some form of animation. Most of this animation is triggered by user input, for example when the user presses 'p' a flying saucer appears from the left of the scene and flies across the sky until it is hovering above one of the astronauts. I did this by calculating how far the UFO had to fly in coordinates per frame of animation and increased its 'x' and 'y' coordinates by an appropriate amount each frame to create a smooth flight across the screen until stopping above the astronaut. To make the entry flight smoother I split it into three parts and changed the rate at which the coordinates were increased to make the UFO look as if it were slowing to a stop. I did this using two simple conditional statements, with slower speeds in each one. For the exit flight I used a different animation technique. I mapped out each coordinate position of each frame, this made the flight less smooth but allowed for more control over the UFO's position per frame, this enabled me to experiment with different animation techniques.

The animation of each of the astronauts was also done by mapping out each angle and coordinate position per frame of animation, allowing for complete control of where they are positioned per frame. For objects that had easier movement I used simple calculations to compute the position of each object relevant to the runtime of the scene, for example the positional lights of the UFO were calculated using simple trigonometry to update their position to rotate around the bottom of the flying saucer. The lunar rover moves by simply increasing its rotation angle around a midpoint using the current velocity, increased by user input.

To animate sputnik falling from the sky and crashing to the lunar surface I used keyframe animation. I split the beginning of the animation in to three keyframes where the rotation of sputnik is changed to give more life to the satellite to look as if it was malfunctioning before falling from the sky. I then used interpolation to generate the frames between each angle of the rotation to allow for smooth movement of the satellite. To calculate the falling motion of the satellite I used SUVAT equations for both the 'x' and 'y' translations of the satellite, with a higher initial velocity in the 'x' direction. For the deceleration value in the 'y' direction I used the gravity of the moon '1.62m/s', however for the 'x' direction there was no deceleration as there is no air resistance on the moon, this made the motion of the satellite more realistic to the setting of the scene.

I used the same technique for the golf ball animation. Originally, I mapped each coordinate to ensure the ball landed in the hole, but this created a choppy animation which did not look realistic. To fix this I used SUVAT equations using the moons gravity to work out a realistic trajectory for the golf ball ending in the hole. Again, the SUVAT equation for the X direction does not have an acceleration value as air resistance on the moon is negligible, which is why the X translation is constant. The accurate animation of the golf ball and Sputnik are parts of my coursework which I believe to be exceptional. The flag is animated by rotating slightly around the pole by changing the angle each animation frame, as well as having a sprite animation, which is explained in the next section.

Lighting and Texturing

For the lighting of the scene I created a 'SunLight' file to create a more realistic light coming from the position of the sun that was in my skybox textures, this illuminates the scene from the back-left corner of the skybox. You can see the sunlight reflected in certain objects, for example I used Gouraud Shading for each of the astronaut's visors, the main sphere of the UFO and for the body of Sputnik, this creates a metallic look making the objects more realistic.

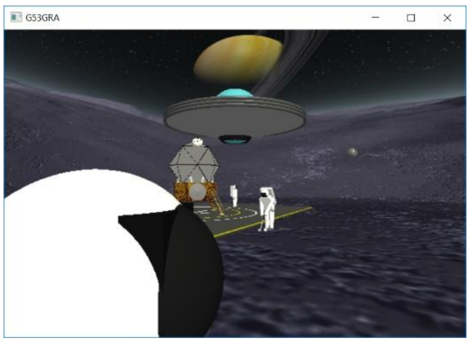
I also added 4 positional lights to the UFO for when the astronaut is abducted, two green and two blue, to create a turquoise lighting which rotate around the bottom of the UFO and are reflected in the ‘LaunchPad’. To create reflection in the launch pad I had to tesselate its top surface, I did this using two nested ‘for’ loops which created a grid of 400 small squares, each with their own diffuse, ambient and specular light values added to create a Phong reflection model across the launch pad.

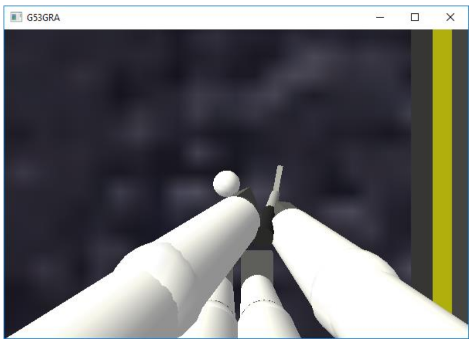
The real Apollo 11 lunar module has a base covered in gold to reflect infrared radiation to protect it from the sun’s heat, to create this effect on my model I textured the base using a gold texture. I created the hexagonal base by combing eight textured gold boxes with each side mapped to my gold texture, I then textured the legs of the module by binding a quadratic object with a thinner gold texture and wrapping the cylinders with the objects, I also used the same technique on the lunar rover.

An iconic element of the first lunar landing which caused a lot of controversy was the waving flag, to create this, I used a sprite texture of 9 images which are mapped to each corner of the flag box. I then used an array to cycle through the 9 sprite images to create the illusion of movement in the flag. To make both sides of the flag visible I had to inversely map the texture coordinates to the back of the flag so that the flag could still be facing the right direction on both sides. Unfortunately, the only flag sprite I could find on the internet was a pirate flag, so I had to use this instead of the American flag, transforming my astronauts in to space pirates. There is also a small outline on the flag, which is on the sprite images. I tried to make the background of the sprite transparent but when I did this it made the texture render as translucent red, because of this I kept the outline. The waving flag is part of my coursework I believe to be exceptional.

Viewing and Projection

For the viewing and projection of my scene I have been using the framework to allow camera movement with ‘w’, ‘a’, ‘s’, ‘d’ and mouse movement. However, I did also add some extra functionality, I changed the initial start camera position to be further back to allow a better initial view of the scene. I added the ability to move in the ‘y’ plane allowing up down movement using the mouse and I created two new functions, one which allows the user to view the scene from the first-person perspective of the ‘AstroGolfer’, the user can switch to this view by pressing ‘v’. Whilst in this view the user is unable to use ‘wasd’ to move, they can reset this by pressing the space bar. The other function switches to a view from the rover, which follows the rover’s movement. To calculate this camera position, I used trigonometry with the radius of the rover’s track to find the z and y coordinates. I also changed the view direction to always be looking at the origin, to get a good view of the scene. This is another part of my coursework which I believe to be exceptional.





The projection of the scene uses a maximum distance of 10000, a FOV of 60 and calculates the aspect ratio by dividing the width of the window by the height, so that each object can be rendered with the correct prospective projection. I used prospective projection instead of orthographic as it made the scene look more like real life as the rays converge at the centre of projection.