GRA Coursework: Football Stadium

# Introduction

For my scene I initially had the idea to create a football pitch containing players, as I am a big fan of the sport and would enjoy creating the scene. This would allow for some complex animation and a realistic looking scene in OpenGL, making it a good idea from a technical standpoint. From the coursework examples given in lectures, I knew that I wanted to create a built up area containing buildings and cars, as they produced good visuals for the end product and made the scene more life-like. I found the most fitting way to do this was to create a stadium around the football pitch with a surrounding car park and roads. This would involve complex hierarchical modelling and give many lighting and texture opportunities to help me achieve an impressive scene. This led me to the final scene I have created, containing a football stadium, football pitch complete with players and an external car park with roads.

# Modelling

The majority of my models are created hierarchically, with push and pop matrixes found all throughout my code, as this is the most efficient and easy-to-read method of creating complex models. This in accompaniment with using different functions for each model section allowed code to easily be reused and the model placed anywhere in the scene using translations and rotations.

## Stadium

The stadium is the largest structure in my scene, and it composed of two different building blocks – stadium sides and stadium corners (see lines 133 and 164 in the stadium class). These are made up of four different sections – the base, the wall, the roof, and the seating. The stadium sides are composed of several cubes from a custom Box function (see line 322 in the stadium class), where the cubes are composed of six GL\_QUADS, which are then scaled in whichever direction to create the desired shape. These scaled cubes are translated and rotated to create each section, of which one side of the stadium is made up of several sections to allow for the textures to be scaled correctly. I decided to angle the roof upward slightly to mimic a real stadium, which would be used to allow rain to run off. The seating is angled downward to meet the base to give the impression of layered seating, which in combination with the appropriate texture would give a life-like look.

Each stadium corner contains one stadium side in the middle with one triangular version on either side to create the overall sixteen sided stadium, giving the large scale look I desired. The corner pieces are very similar to the sides apart from the roof and seating, which are comprised of custom GL\_TRIANGLES to account for the inner perimeter being smaller than the outer perimeter of the stadium.

Finally, to finish the stadium structure I added a player’s tunnel coming out of the left side of the stands. This was made from one scaled cube and one black rectangle to signify the entrance on the right side. This broke up the symmetry of the stadium well and was the ideal place for the red floor lights to be placed. These comprise of four small red spheres, which are semi-transparent so that they mimic light bulbs.

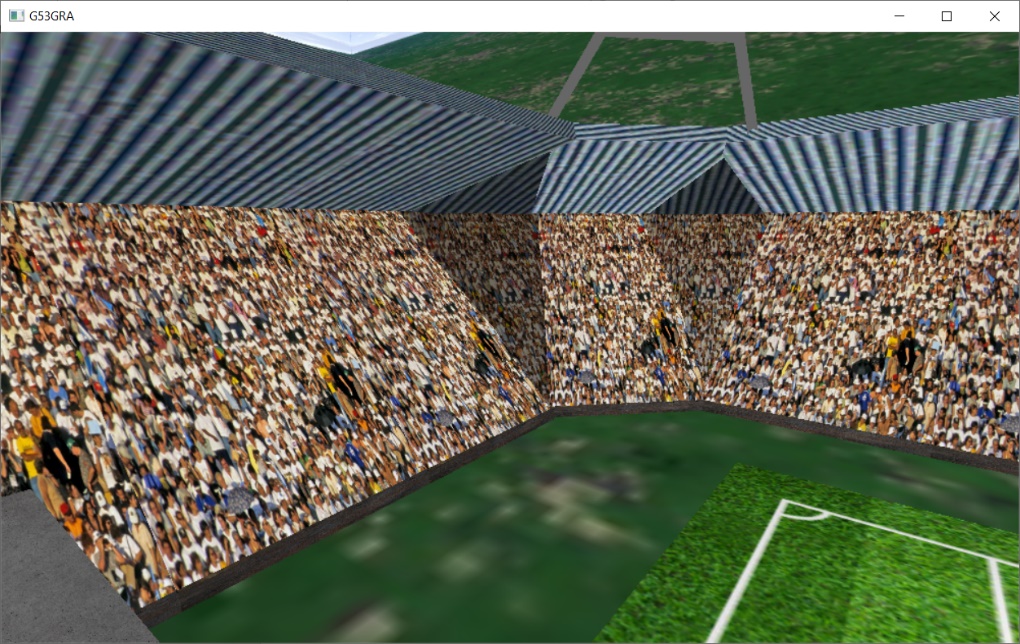
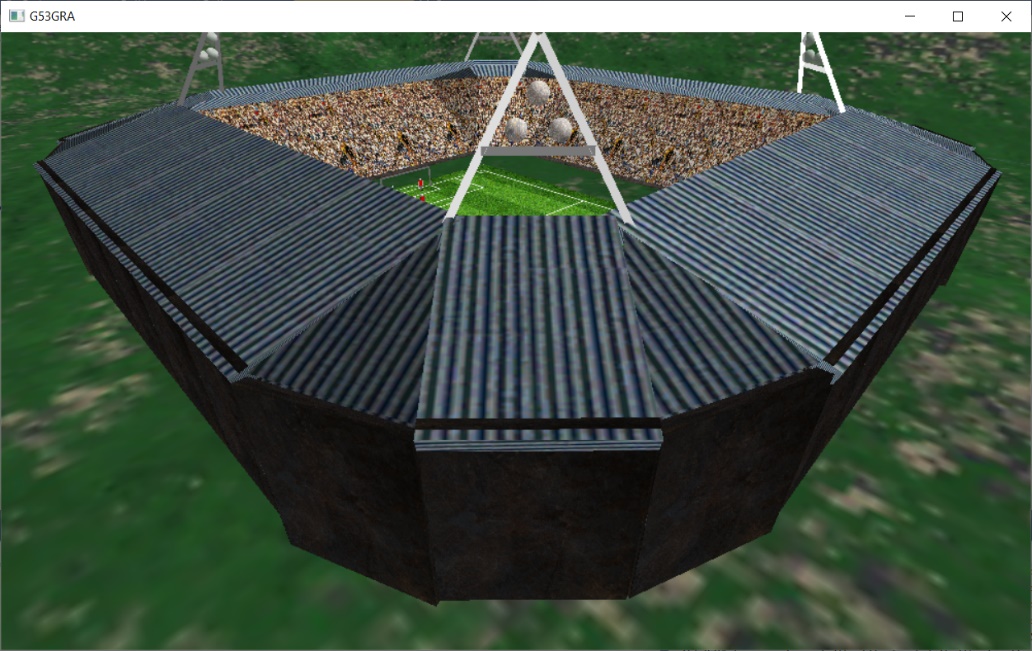
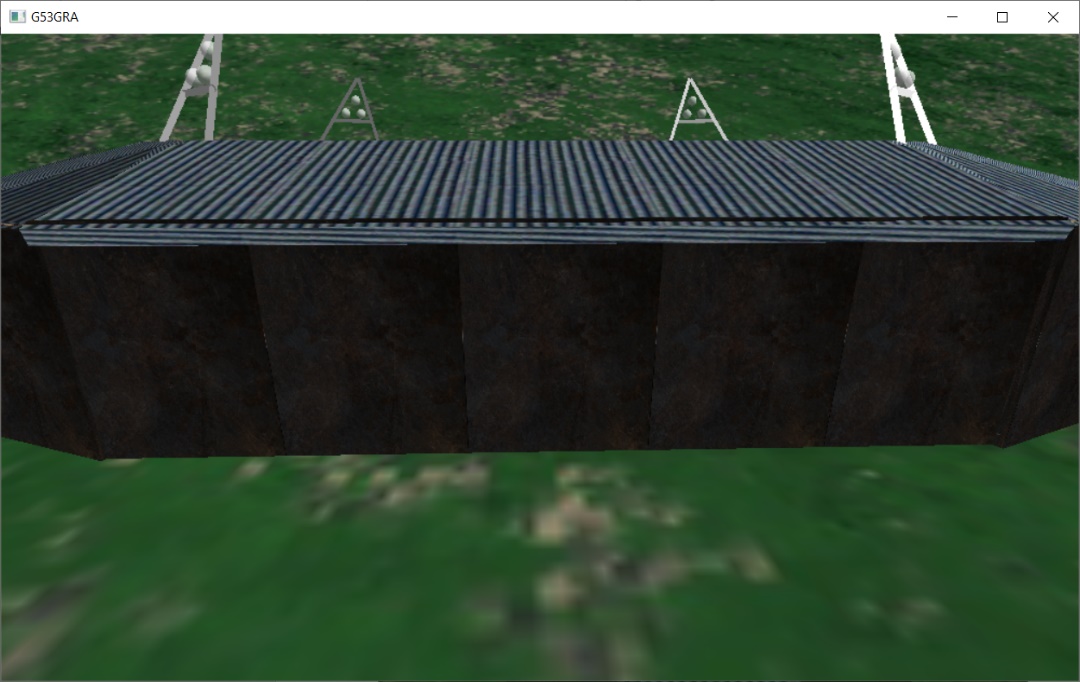
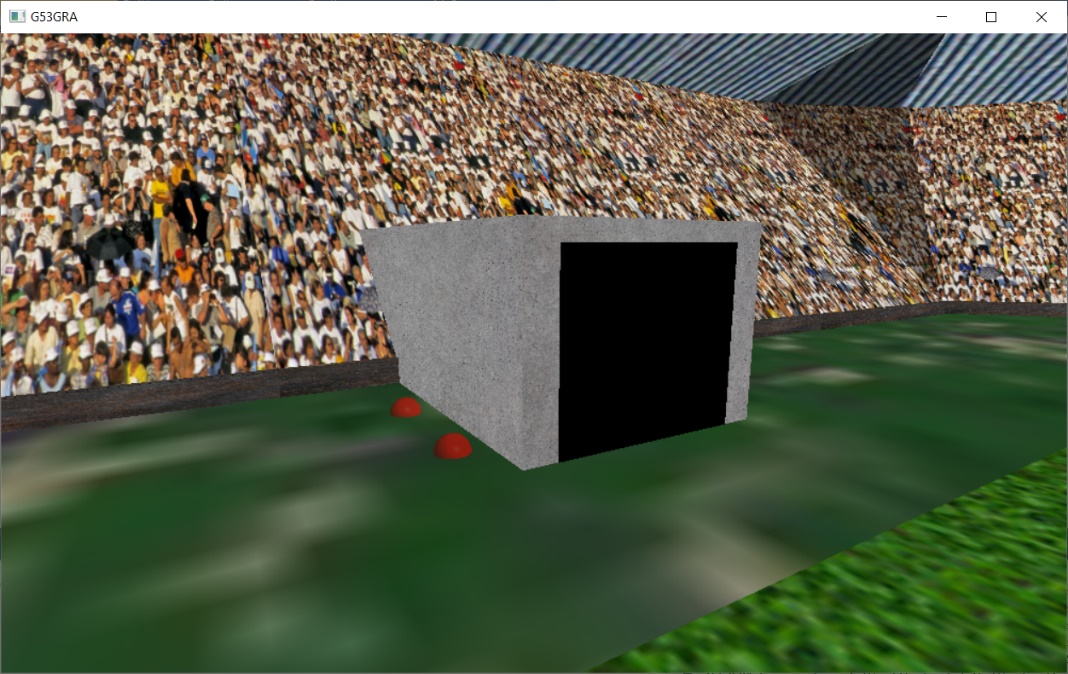
Fig. 1 A front side view of the stadium sides and a stadium corner

Fig. 2 A top side view of the stadium sides and stadium corner roof



Fig. 3 A back side view of a stadium side illustrating the multiple sections it is made from

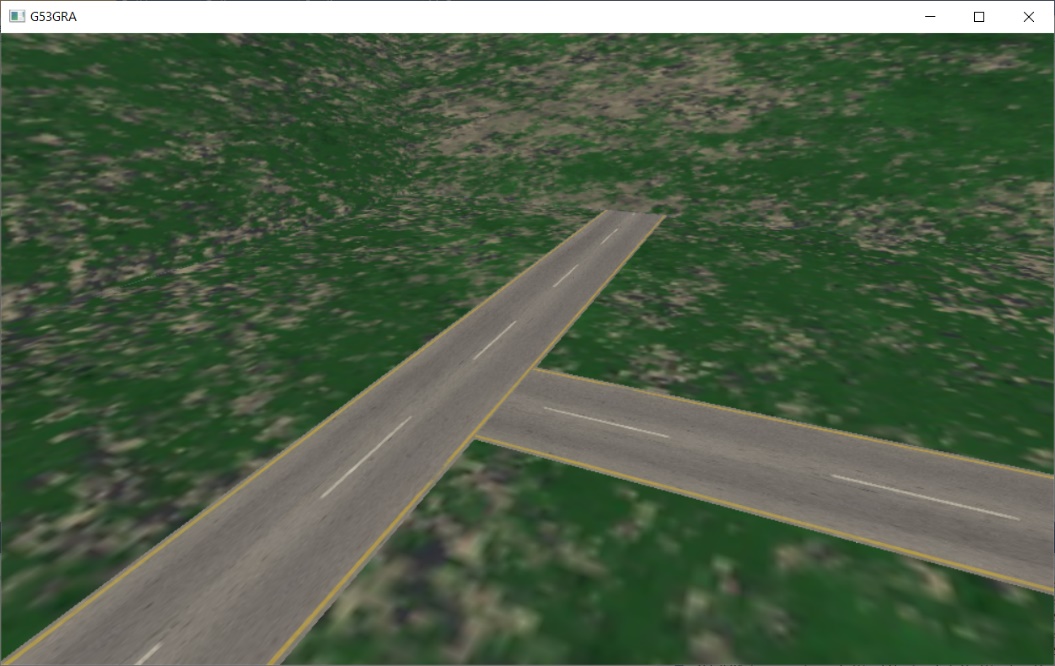
Fig. 4 A side view of the player’s tunnel with red floor lights

## Car Park

The car park forms the entirety of the rest of the scene outside the stadium. There is one main road stretching from the left side of the skybox to the right and another smaller road connecting it to the car park. These roads are cubes scaled so that they are very thin the y-direction so that they appear flat. They are also, like the stadium, made up of several smaller sections as to not stretch the texturing and make sure they look visually pleasing.

The cars are a fairly complex structure with many different basic shapes being used to create them. The cars are constructed from the bottom up, starting with the wheels and finishing with the roof. Each wheel is made up of one cylinder rotated around the x-axis and two disks on either side – one black one used to cap the cylinder to form the wheel and a smaller grey one for the hubcap. The main body of the car is modelled simply by one scaled cube, but given how a car roof is not the same size as its body, the rest of the car is modelled differently. This is done by using GL\_QUADS for each side of the car connecting the roof to the body, which was necessary due to the sides being a trapezium shape. For the same reason, the side windows were also made this way, however the front and back windscreen were rectangular and so could be made using thin scaled cubes. The car function has one Boolean parameter to change the colour of the car from red to blue, to add some diversity to the scene. I have seven stationary cars parked in my scene and one which drives along the road in an animation.

The last model used in the car park are the lamp posts which are placed throughout. These are made from two thin cylinders placed in a “T” formation meaning there is one light on either side. A scaled cube is placed on either end with a sphere inside, but which is only slightly showing on the bottom side of the cube to represent the light bulb. These are at regular intervals within the car park, as they would be in real life, spacing out the cars appropriately.

Fig. 5 A top down view of the roads in the scene

## Fig. 6 A side view of a stationary car with a lamp post in the background

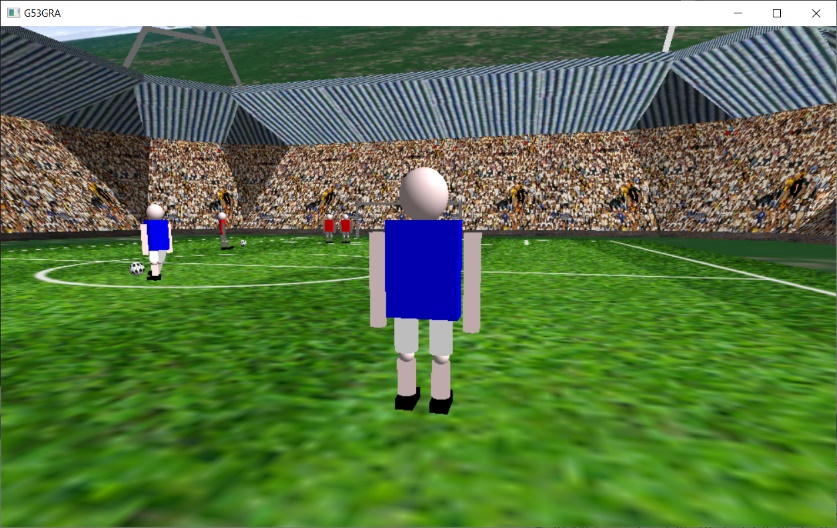


Fig. 7 A front view of a car showing the front windscreen as well as the side



## Players

Opposite to the cars, the players are built from the top down, starting with the head using a scaled sphere and a scaled cube for the body. Each of the arms is one cylinder as I felt this would give the best look and an arm joint was not necessary for the animations I had in mind. However, the legs were going to have some complex animation, so I made sure to model a knee joint in the leg, made from two small cylinders at the top and bottom of a small sphere. Finally, the feet are made from two small scaled cubes. This is by no means an exact model of human, but it is complex enough to facilitate the animations and fits in with the graphic style of the scene. Similar to the car function, a parameter is passed to change the colour, but only for the player’s shirt, with everything else being a fixed colour to keep in fitting with the scene. The red and blue players are split up into two separate classes for the purposes of animation (see interaction section for more details).

Fig. 8 A stationary player from the blue team

## Miscellaneous

For some models it did not make sense for them to be a part of the larger classes, so I decided to split them up for ease of use - these are the flood lights, goal posts and the ball classes. The flood lights are made from three cylinders to form an “A” shape, something which I copied from many real life stadiums. Three large spheres are placed in the middle as the actual lights, again similar to the red lights, these are semi-transparent to mimic a light bulb. One flood light is placed on each corner of the stadium to create an aesthetically pleasing design. The goal posts are a very simple model, made form three cylinders, with the top one being longer than the two vertical ones to create a goal which is correct in size in relation to the players. The ball classes are simply a sphere the size of a football in relation to the players, but for animation purposes it made the most sense to create two separate ball classes. This allowed me to have two different update functions, allowing for greater control over each animation.

Fig. 9 A front view of one of the flood lights on the stadium roof

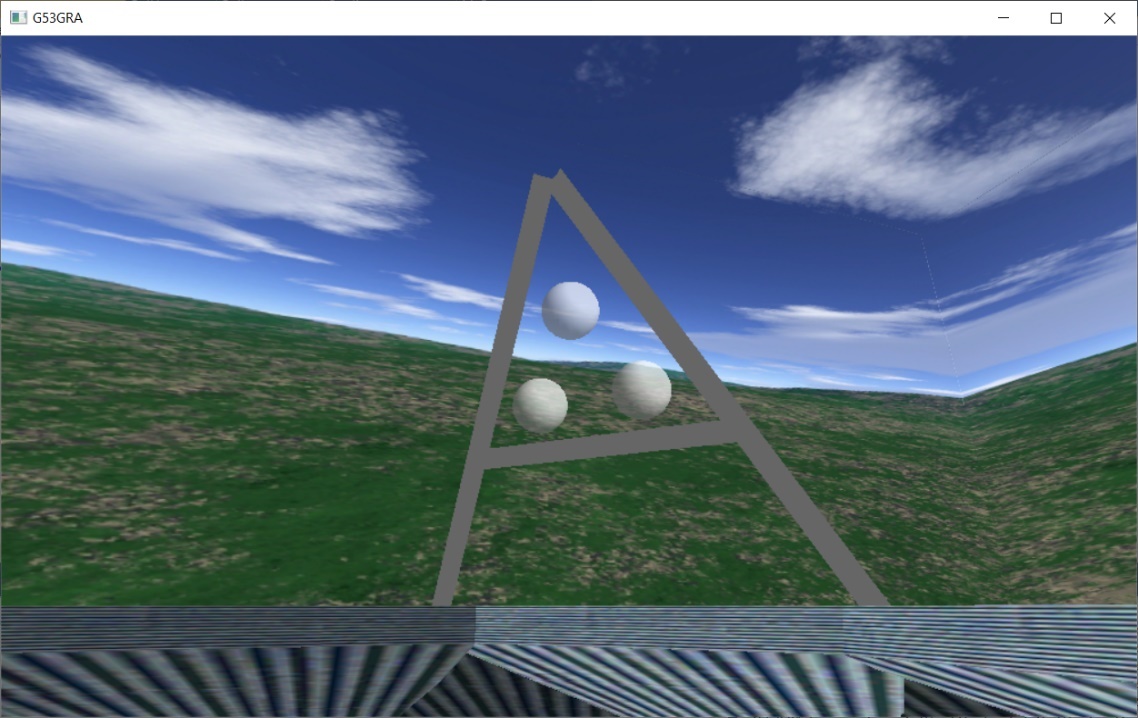


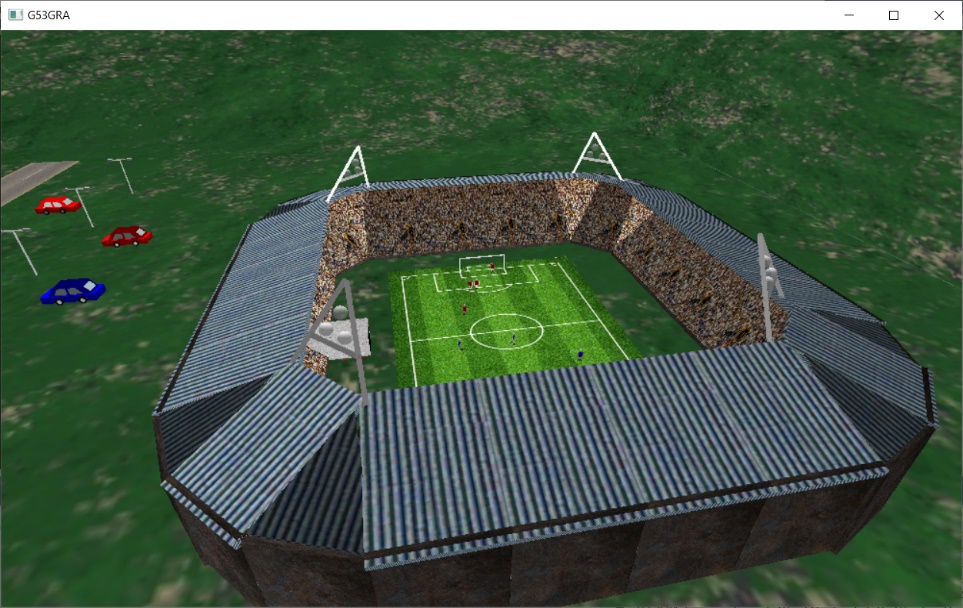
Fig. 10 A front view of the blue team goal posts

# Lighting

The lighting in my scene has two options – daytime and sunset. I chose these to give some variance to the scene, with daytime showing more detail in the modelling and sunset creating a more thematic environment (my preferred lighting for the scene). The daytime lighting exclusively uses GL\_LIGHT0 which is declared in the Engine class. I chose this lighting to be directional as I wanted it to appear to be coming from the sun in the top left corner of the skybox. Therefore, the direction is set to -1 in the x-direction and so that the light matches what the sun would do if it were real. Its position is also set above the scene to make sure all the models receive the correct lighting.

In the sunset lighting the daytime light is disabled and replaced with GL\_LIGHT2, which is of the same height and direction as the daytime light but, much weaker in the diffuse and specular strength, to emulate the reduced lighting of the sun. In the scene the reds are the home team, so I added a red light (GL\_LIGHT1) to the stadium next to the tunnel to give some atmospheric lighting to the players and surrounding stadium. To fit this purpose the light had to be positional, centred in the tunnel, with the correct attenuation values to make sure the light was not overpowering. Since shadowing is not implemented and the light would not realistically escape through the stadium walls, I disabled this light for the car park so that those models would not appear red.

Shading for the scene is defined in the MyScene class (see line 18 in the initialise function) and is the same for all the modelled objects. I found this gave the best results for the look I wanted – with low specular and shininess values preventing too much glare from the lighting.

Fig. 11 A top view of the stadium in the daytime, showing the left side being brighter than the right as a result of the directional lighting

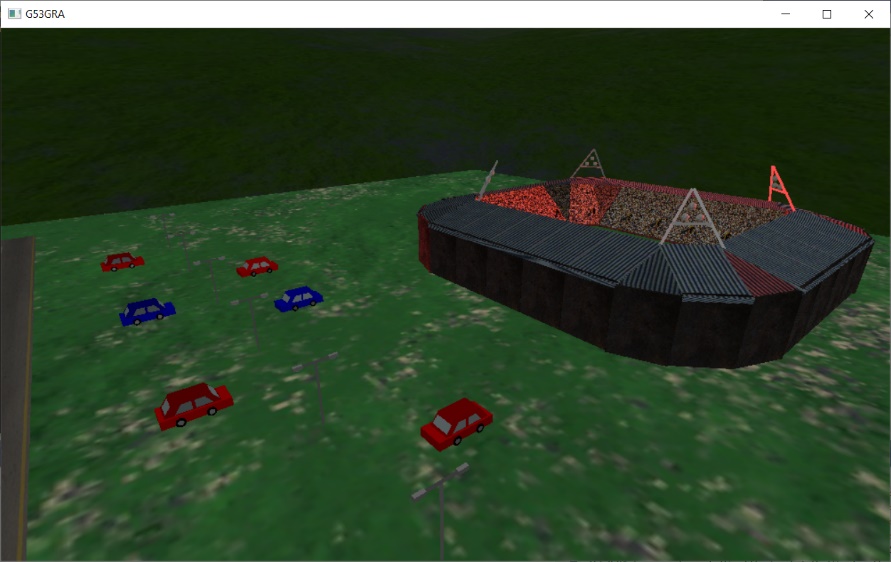
Fig. 12 A view of the car park during sunset showing the models unaffected by the red lighting in the stadium

Fig. 13 An inside view of the stadium with the attenuated red lighting

# Texture

My scene consists of many different textures most of which are used within the stadium models. As mentioned in the modelling section, the majority of the models are scaled cubes which allowed me to texture each cube within this function. See line 322 in the stadium class for the Box function, showing how the texture coordinates are mapped with vertex coordinates of the cube. Also note the scale parameters which are used to manipulate the shape of the cube before it is drawn.

I briefly touched upon the fact that the larger models were broken down into smaller pieces in the modelling section. The reasons for this become apparent in this function, where you can see that the cube is scaled before it is textured, meaning that the textures would be otherwise stretched. This method of texturing the cube, however, was not possible for the stadium corners where triangles were used. This led me to use a different method of creating texture coordinates for these triangles. The metal texture I was using for the roof had lines going from left to right, meaning I had to create a rotated triangle within the texture to achieve the correct texturing. See line 201 again from the stadium class, where this method is used for texturing each triangle. The same method was used to texture the corner seating in the stadium.

The skybox is created and textured in a very similar way to the modelling cube, apart from multiple GL\_QUADS statements are used instead of just one. This is done so that each quad can be textured differently to achieve a correct skybox. Each texture is loaded beforehand and bound as a 2D texture before each quad is drawn; in the case of changing the skybox for sunset an if statement is used at the start of the draw function to change the textures’ source (see line 20 in the skybox class).

The ball classes are textured as traditional black and white footballs but require more complex texturing due to their shape. This is done by applying a 2D texture to the gluSpheres the footballs are made from. The draw style is set to fill and the normal is set to true to give me the desired football texture.

Although the outside of the stadium was already textured, I felt that it lacked life, as many real football stadiums often have animated banners or TV screens. Therefore, I decided to implement an animated board showing which teams were playing in the stadium. I picked two club logos (Liverpool and Chelsea) which would change from one to the other every five seconds in a ten second animation loop. This all occurs in the Update function (line 120 in the stadium class), where a variable “currentFlag” stores the texture that is currently being displayed.

# Animation

There three different animations in my scene of varying complexity. The most basic is a red car travelling across the scene from left to right on repeat. This is done simply by translating the car in the x-direction by a constant speed and resetting it after it exits the skybox (see line 88 in the Update function of the CarPark class).

Next, I wanted to bring the players on the pitch to life – for which I created an animation where two players on the blue team are passing a ball between themselves. This is done on a four second loop, starting with the player’s right leg being moved backwards and rotated about the z-axis in the negative direction. This takes half a second and another half a second is used to rotate the leg positively and move it forward, creating the action of the player kicking the ball. Once this action completes the ball moves either positively or negatively in the x-direction depending on which player kicks the ball. A real ball rolls across the ground, which is emulated in this animation by rotating around the z-axis, once again, positively or negatively depending on which player is passing the ball.

The third animation in my scene occurs on the red side of the pitch where the players are set up to practice a free-kick. This animation is also on a four second timer but only happens once the user presses the “3” key. The same leg kicking animation is used as with the blue player animation, but with the red player stood back from the ball and his whole model being translated forward as he kicks the ball. This imitates a real player taking a free kick where a run up is often used to add power to the shot.

The ball trajectory has three separate parts – the first where it rises in the y-direction and moves backwards in the z-direction, while also moving slightly to the right. Once the ball is above the wall of players it then begins to dip in the y-direction, straightens in the x-direction and slightly decreases in speed in the z-direction. In the third and final section the ball dips more in the y-direction and continues moving backwards. Throughout these three sections the ball is being rotated in both the x and y direction continuously, as often players taking a free kick will put spin on the ball to manipulate its direction.

Furthermore, in this third section the goalkeeper dives in an attempt to stop the ball from going into the goal. This player animation is done by rotating it around the z-axis positively and rotating his arms negatively about the x-axis. Since the rotation of the player is based at the feet, it needs to be raised in the y-direction to prevent it from going through the floor. It is also moved left towards the ball in the attempt of the save, giving the overall dive a smooth animation similar to that of a real goalkeeper.

In the first 0.1 second of each animation the positions of the models are reset, making sure that the animations are consistent each time they loop.

Fig. 14 A screenshot of the blue players passing the ball

Fig. 15 A screenshot of the free kick animation from the taker’s perspectives

# Interaction

The user is able to move about the scene using the “wasd” keys, using the standard code from the framework, while also using the basic mouse movement to look around the scene. However, for a scene like mine it can be beneficial to view from different heights, so I added the use of the “q” and “e” keys to move up and down in the y-direction, respectively. This is implemented in the camera class. Both the update and the handle key function were modified to allow for this functionality, with the update function calling the add and sub functions using the up vector as opposed to the right or forward ones (see lines 68 and 71 in the camera class).

In the skybox class I added the ability for the user to change the lighting settings using the “1” key for sunset mode and “2” for daytime mode. The lights are set up in the MyScene and Engine class, but due to GL\_LIGHTs being global variables they can be accessed from any class. This allows the skybox class to implement the HandleKey function and enable/disable the correct lights depending which key is pressed (see line 92 in the skybox class). When the lighting changes, the skybox also needs to change to keep in setting with the theme. This is done with a Boolean variable “isNight” which changes the skybox textures in drawSkybox function (line 20 in the skybox class).

Lastly, I felt as though some interaction with the animations was needed within the scene. Therefore, I added a feature to allow the user to control when the red team player takes the free kick. This is done by pressing the “3” key and can be paused at any time by pressing the “4” key. If the “3” key is pressed after the animation is paused, the animation timer is reset, and it will play from the beginning again. The blue player animation runs continuously in the scene whereas the red player animation only needs to play once the “3” key is pressed - this is the main reason why I decided to split the red and blue players/balls into separate classes, as they would need separate animation timers.

# Conclusion

Overall, I am very happy with the way my scene turned out and it encapsulates what I had envisioned from the start. I would say its strengths lie in the modelling of the stadium in which its shape and texturing closely resembles a real stadium. I am also pleased with the player animations and their interactions with the ball, which match some older football games I used to player when I was younger. In terms of weaknesses, the flood lights serve as a good structural addition but I had hoped that they would also be used as spotlights for the players. I attempted to use the OpenGL spotlights for this but to no avail and struggled to achieve the desired effect, which I am sure could be done, but is out of my coding skill in this domain. The addition of shadows would have also been beneficial for this but was not taught on the course, and without any additional help I decided not to attempt this. The stadium roof texture produces some aliasing at distance, which detracts from the realistic look of my scene. Once again, I attempted to fix this but it was beyond my skill and it remains in the scene.