Contents

Task 1

Introduction

The aim of this task was to create an animation of a marble run. I decided to create a simple course for the marble to follow including different shapes and slopes as well as drops where the marble falls. I decided to use Blender to finish this task, as I am familiar with the program and it is capable of doing the same tasks as other software such as 3DS Max.

Conceptualisation

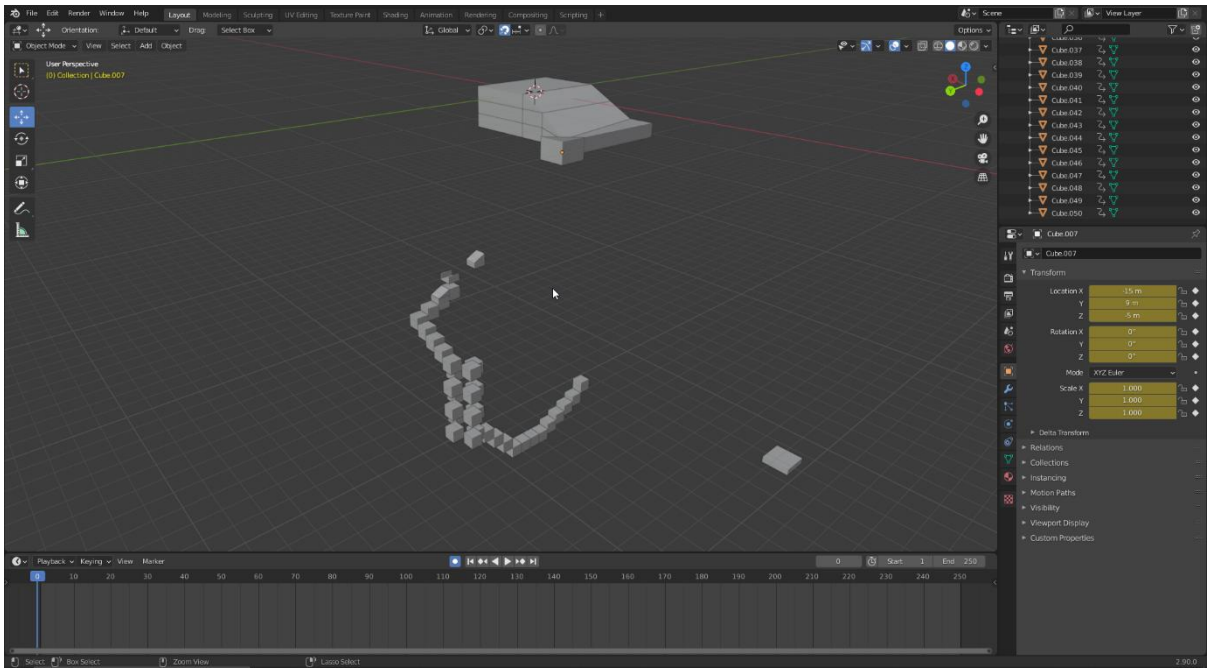
I did not do much planning on the layout of the run, as I wanted to simply experiment with what sort of scene I can build and improve on it from there. My main inspiration for the run was from the NVIDIA Marbles demo, which shows a marble run on a table full of objects and wooden blocks. I did not go into the same level of detail with the environment as the demo, however I did use some simple wooden blocks to make up the main part of my scene. Rather than doing sketches, I first set out the entire layout of the course using simple cubes.

Figure 1. NVIDIA Marbles at Night

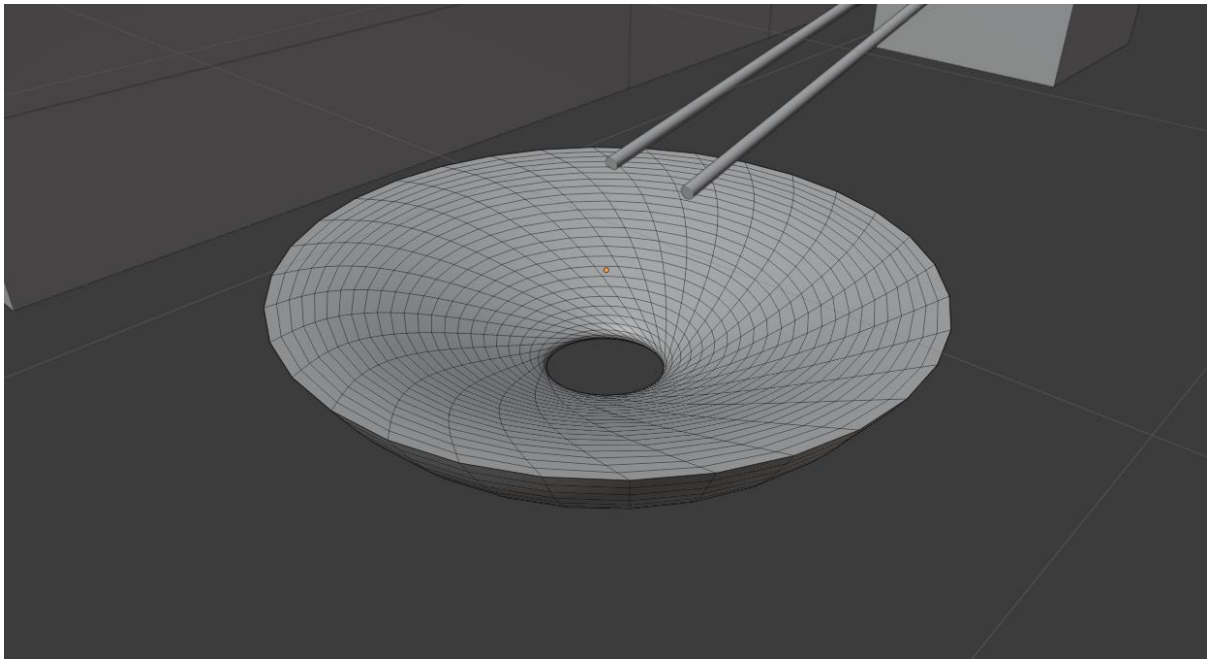
Method

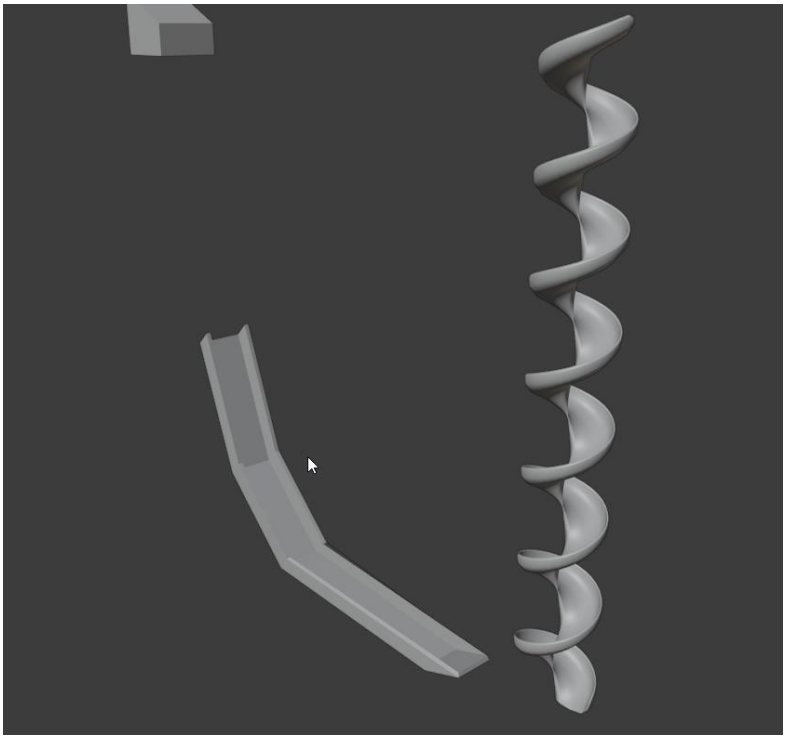
Scene

First, I set about creating the layout of the level. To do this, I simply duplicated and scaled the default cube into a basic path, whilst thinking of how the marble will move through it. I left space for the more detailed object such as the funnel and focused only on tracing a pathway which ended up staying the same all the way to the final product. After I was happy with the layout, I also modelled some of the cubes into ramps, thinking about how they would need to be angled for the marble to follow along the path.



Next, I created the two most complex objects in the scene, which are the funnel and spiral. The funnel is simply a Bezier curve with the subdivision and curve modifiers applied. I then grabbed some of the vertices and transformed them until the shape looked how I wanted. Meanwhile, the spiral is a plane, which once again has a screw modifier applied with several screws and a bigger angle value to get the desired look. To fill in the plane and get the cup shape, I extruded the edges and used the solidify modifier to give it more volume. I also added a final ramp which is a very simple extruded plane, once again using the solidify modifier.

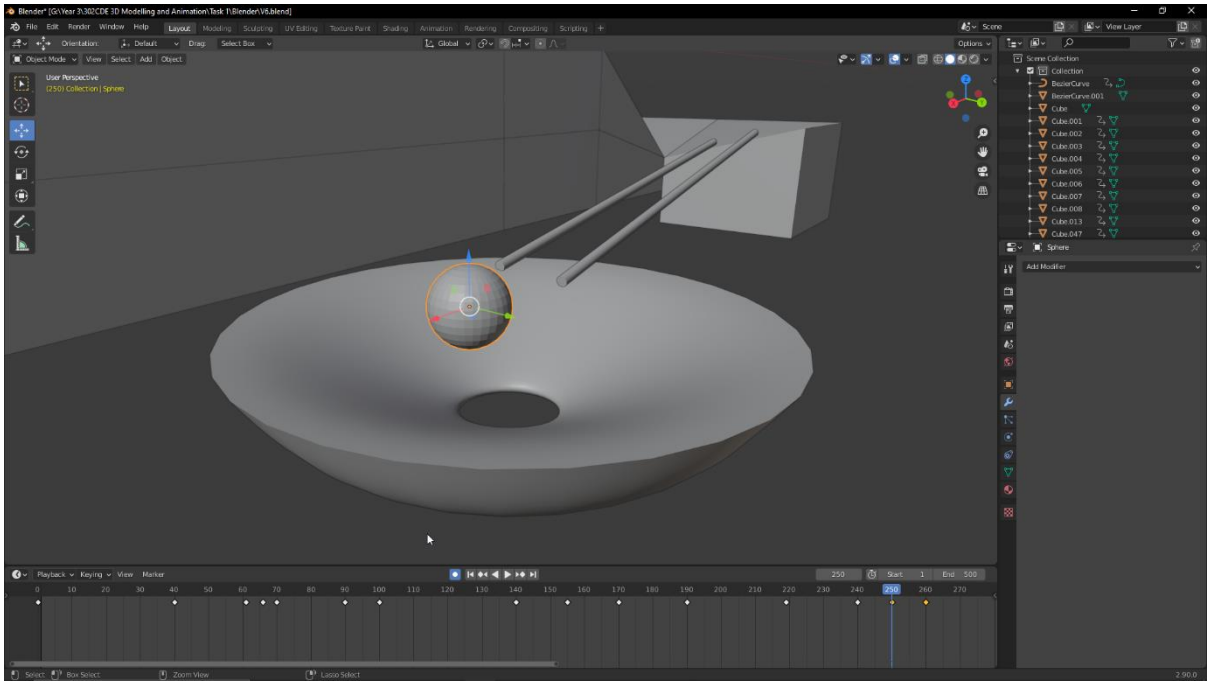




With these objects, the scene was almost complete. The only difference from this point to the final product was changing the position of some of the objects to fit the animation and pacing better. Therefore, I begun working on animating the marble.

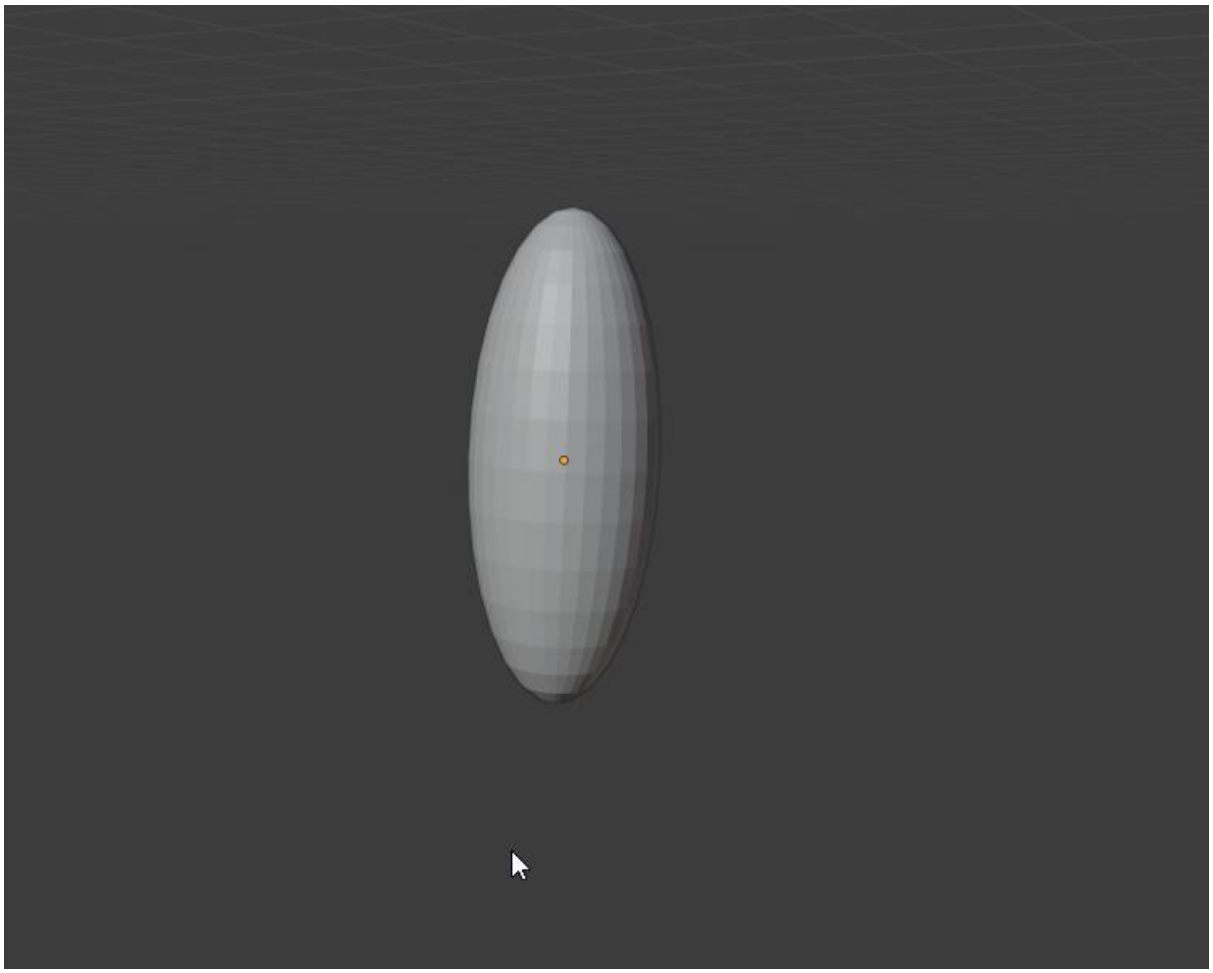
Animation

I spent a lot of time considering how I will animate the marble. At first, I experimented with using curves and animating the sphere on the curve. This seems to be the most common way of animating a moving object, as it allows for quickly setting the path that the object will take and being able to refine it step by step. However, I was not happy with the effect I got from this method, as it didn't seem convincing enough to be a real marble that is following Newtonian physics. Therefore, I made the decision to simply keyframe the entire animation manually. Therefore, I set about carefully positioning the sphere every few frames, adjusting the distance between each frame to change its speed trying my best to make it appear as fluid and realistic as possible.

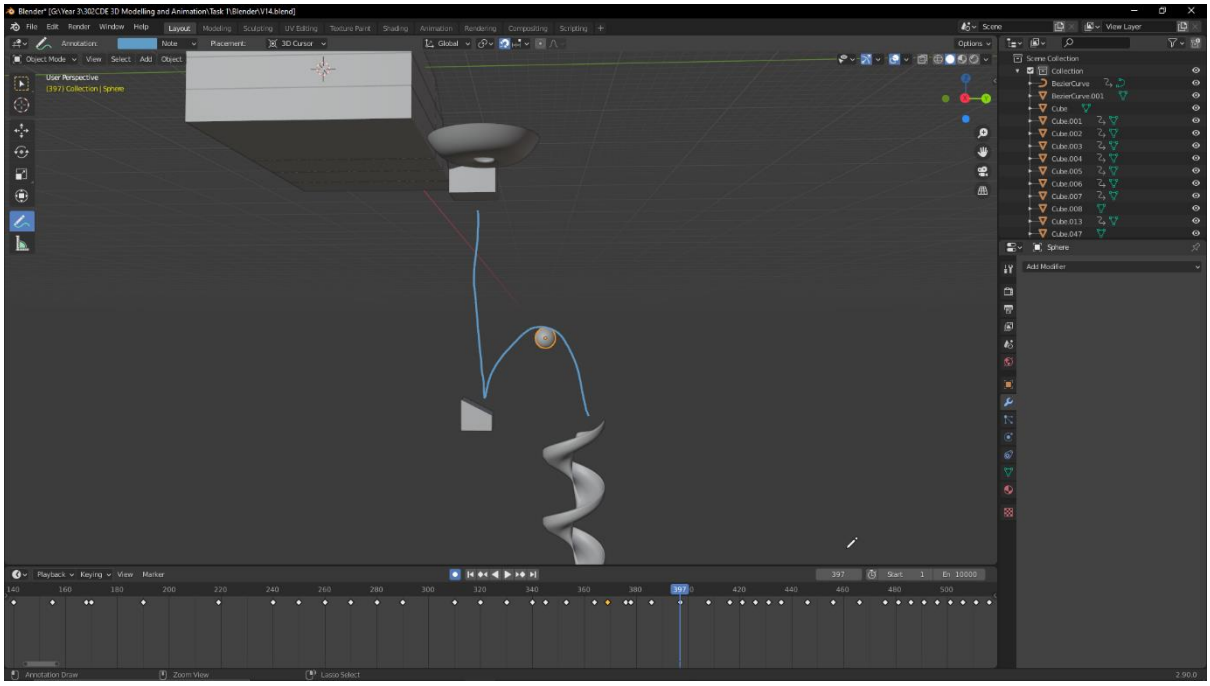


This process took a lot of time, and I had to go back several times and redo some of the frames to really make it look how I wanted. My strategy was to animate the sphere and stop when there is a big change in direction or speed, then scroll through the frames and see how it looks compared to what I'd expect a marble to do in real life. This method really did give me a lot of control over how the sphere moved and reacted to its environment, but it relied a lot on me being able to manually keyframe it well enough to look convincing. The three hardest parts where definitely the funnel, the bounce and the spiral. For the funnel, I used a lot of very closely spaced keyframes, with the idea that every time the sphere loops around it will rise, slow down, fall, speed up and repeat until it is all the way at the bottom. Originally, I ended up making it loop too many times, so to reduce the length of the animation and make it more exciting I moved some of the keyframes back to get rid of the loops. This is one case where I believe the curve method would have been harder to use.

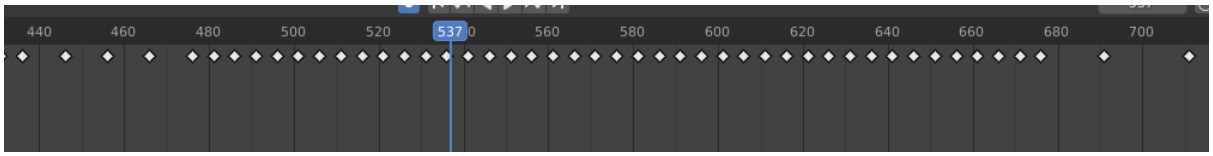
I thought a lot about the bounce segment and whether I should even do it. I wanted the animation to look like it abides the laws of physics, but I also wanted to have a personal and animated flare which reveals that the animation was actually manually keyframed and not simulated. At first, I considered using the squash and stretch principle from the 12 principles of animation. I was going to make the ball stretch downwards during its fall, then squash completely on contact with the slope, before forming back and bouncing towards the spiral. This was going to be also manually keyframed in the same way as the movement of the sphere.



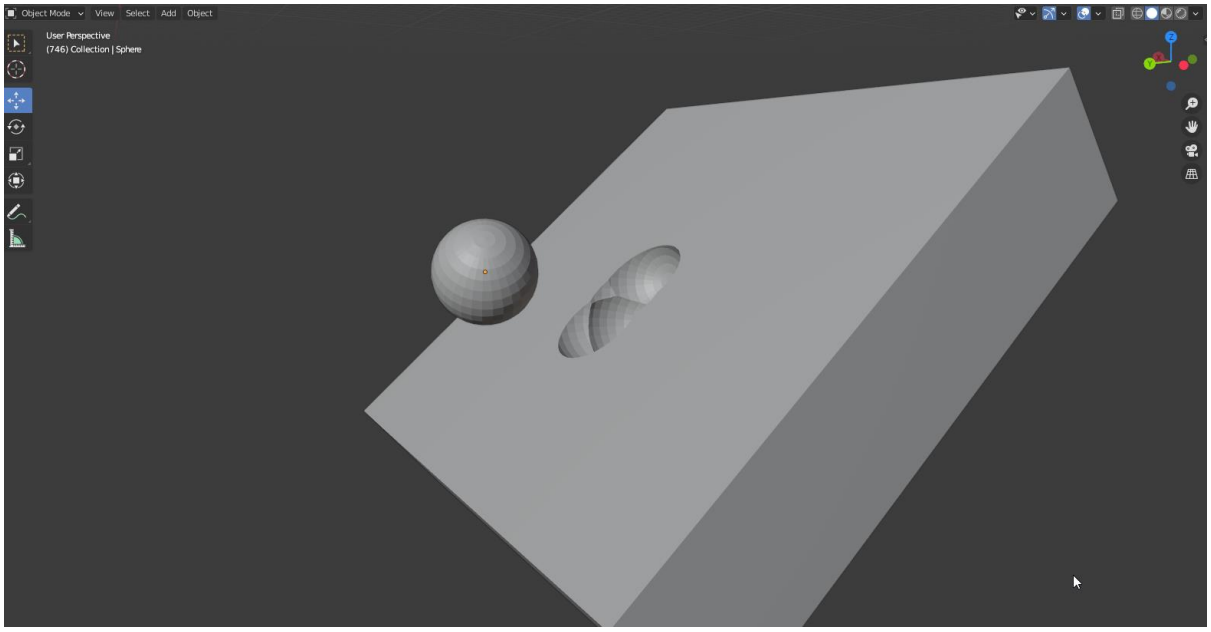
In the end, I steered away from this idea and decided the marble will keep its shape the entire time to make the animation appear more realistic. A marble that stretches and bounces around will seem like it's made out of rubber, which wasn't my intention. Instead, I decided to use exaggeration, which is a more subtle principle that still works well for showing stylization in an animation. In this instance, I made the sphere hang for a long time in the air at the peak of its fall, which breaks away slightly from the realistic nature of the animation without significantly breaking the idea of following Newtonian physics.



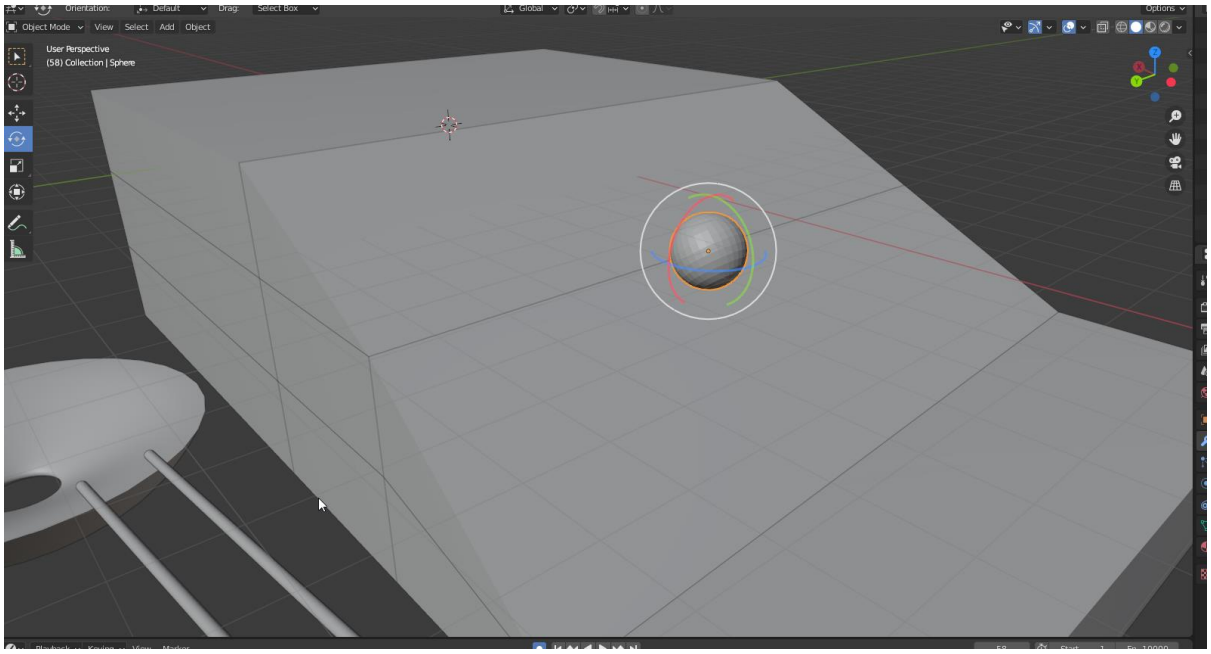
The final notable part of the animation is the spiral. This part wasn't as difficult as it was tedious, as there is no sort of tool for repeating the same movement as there would be with a curve. My process was to keyframe every 45 degree angle the sphere takes down the spiral, adjusting the position to be on the outside of the curve. This segment is easy to spot in the timeline, as it goes from having lots of carefully timed keyframes to having a keyframe every 5 seconds for each turn.



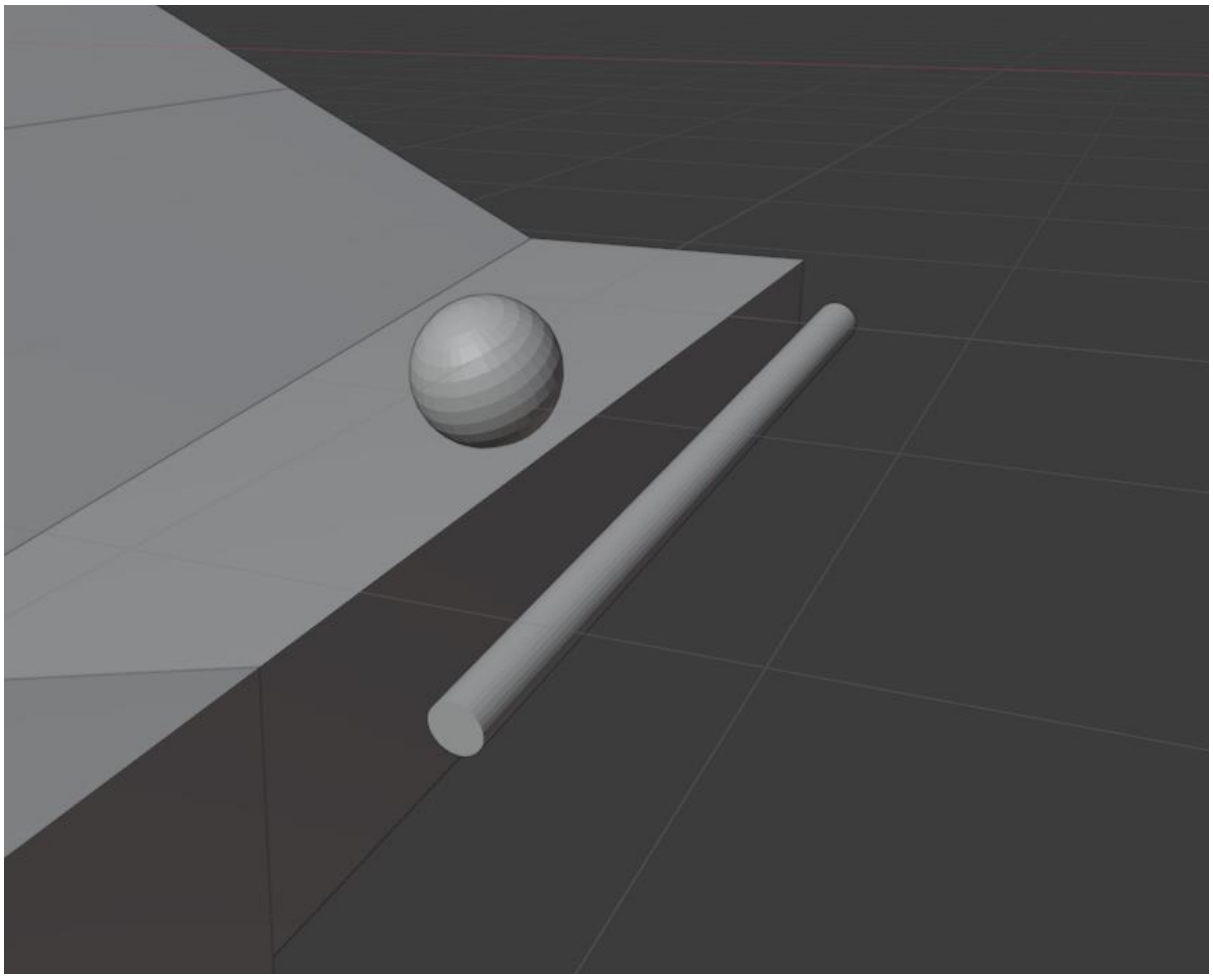
I repeated the same process for the second jump, carefully considering how the longer the sphere falls the faster it should be going. Whilst looking at its speed, I had an idea that when it lands it should either smash through the ground or leave an indent, so I created a cube and left an impression with the Boolean modifier. However, I never figured out how I could animate this, so the indent is there before sphere actually hits the cube.



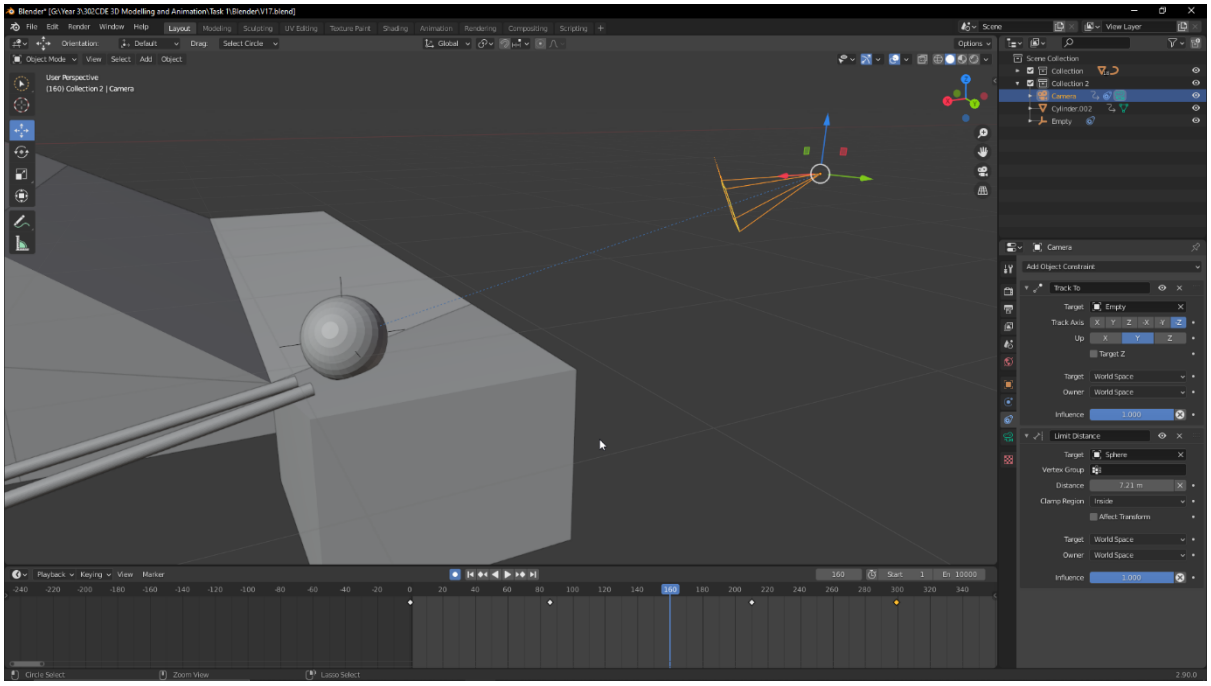
Up until this point, I only edited the transform of the sphere, so it appeared to float along the ground rather than rolling across it. With a curve, I may have been able to use a driver to automatically calculate the rotation based on its position in the curve, however with my method I simply decided to keyframe it the same way as I did the movement. I went across every keyframe that I already had and changed the rotation to match the speed and direction of the animation. Unfortunately, there are some instances where the rotation goes all the way back around instead of just rolling straight, however in motion it is not as noticeable and so I do not believe it completely ruins the effect. This is one of the mains things I would like to have gone back and redone.



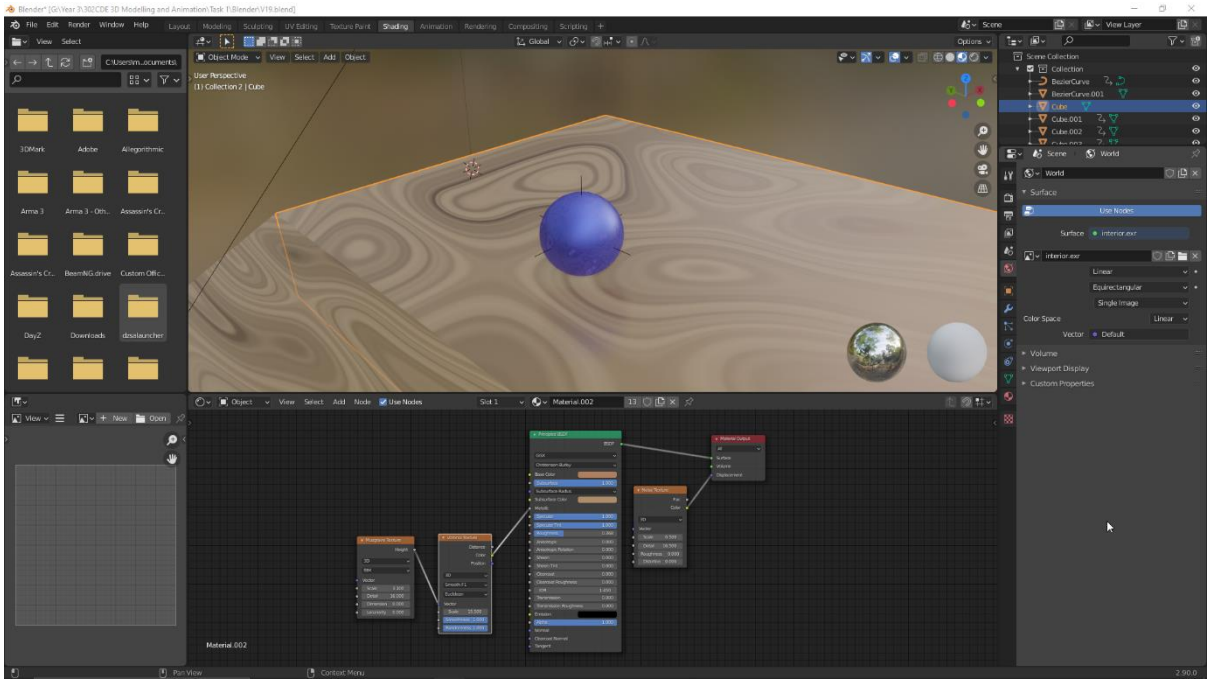
With this, the animation was almost complete. The only thing left was to add a second object that the sphere bumps into and off the scene, which is something I had planned from the start and was inspired by the physical objects in the NVIDIA scene.



With the animating done, I decided to add the camera to the scene. Unlike the sphere, small changes in rotation and position can be very noticeable, so I wanted the camera to follow the sphere by itself and for me to animate its position in relation to the sphere. To do this, I created an empty axis, which copies the location of the sphere. Then, the camera has a track constraint to that empty, causing it to always be facing that empty axis. This is good because whilst the sphere is constantly rolling, the orientation of the empty stays the same and therefore the camera does the same. Finally, I added a limit distance constraint to the camera with the sphere as the target, causing it to always stay within 7 m of the sphere no matter its position or rotation. Finally, I keyframed the transform of the camera whenever I wanted to view the scene from a different angle, which let me put focus on the marble and always keep it in view.



As a final touch, I added a very simple wood material to all of the blocks to make it look a bit like the inspiration material. I made it very quickly using a Musgrave and Voronoi texture to give it a wood grain look and tweaked the values and colour slightly to give the sphere a marble-like appearance. This was a very small and quick change, and I definitely could have made it behave a lot more like wood such as not being so reflective but I believe it makes the final render much more visually appealing and I wanted to focus more on the animation for this task rather than final polishes.



Reflection

Overall, I am quite pleased with my final render. I think it looks quite convincing in terms of the physics of the animation, while still having an animated flair when it comes to the falls and bounces. I also believe I did well in having a variety of different obstacles to make the animation as exciting as possible and to show the movement of the marble in different situations. However, there are several things that I wish I had done differently or improved on.

Firstly, there are a number of times when the marble visibly clips into the terrain. This is something that I could have solved through carefully redoing each frame, however due to time constraints I decided that the clipping was minor enough to leave in the final render.

Secondly, I would have liked to animate the final part of the animation, where the marble falls into and indents the cube. This is something that I may have been able to fake using keyframes and replacing objects, but I decided I should focus on more important parts of the animation such as the actual sphere movements.

Finally, if I had to do the task again, I would definitely try using a curve track instead of manually animating it, as I believe it would have saved me a lot of time and with careful enough edits I could have made it look just as if not better than what I achieved.

Task 2

Introduction

This task involved creating a detailed 3D Model with a rig and animation controller. I decided that I will work on a model of a land vehicle, as it is something that I had never tried to do before. Once again, I used Blender for the entire creation process of this model.

Conceptualisation

There were two models that I was considering making; either a tank, or a modern car. I thought a tank would be more interesting, but in the end I chose the car because I thought a tank has too many sharp edges and is too cuboid, whilst I wanted to attempt to actually make a smooth and highly detailed model. I thought about what kind of car I could base it on and decided to use a BMW E90 as my reference. This is simply because I believe it is a car that is instantly recognizable whilst still giving me enough room to make it more generic and my own. I also used blueprint images of the car as a reference guide for the shape and size of my model.

Figure 2. BMW E90 from Wikipedia.org

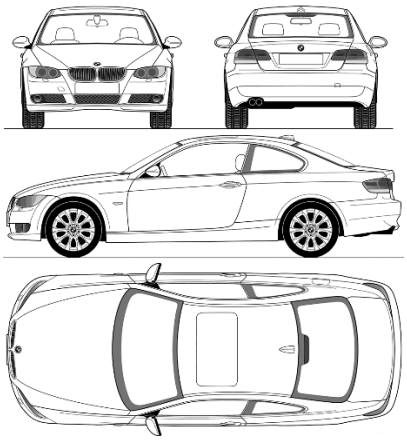
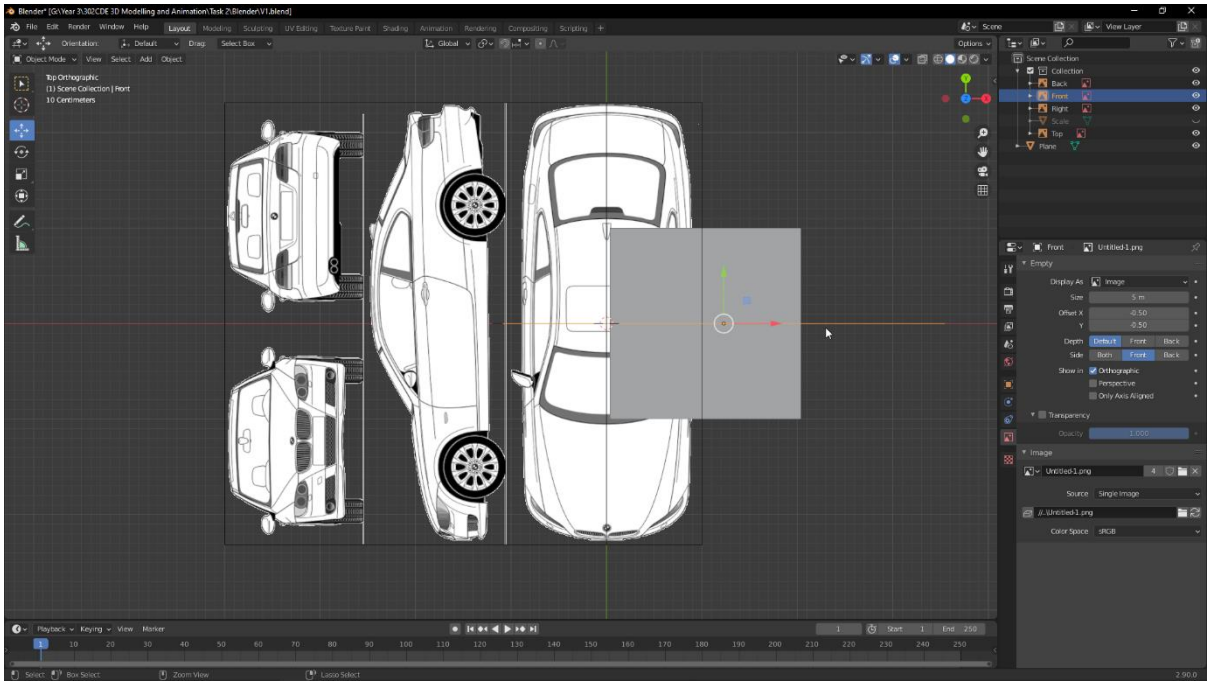


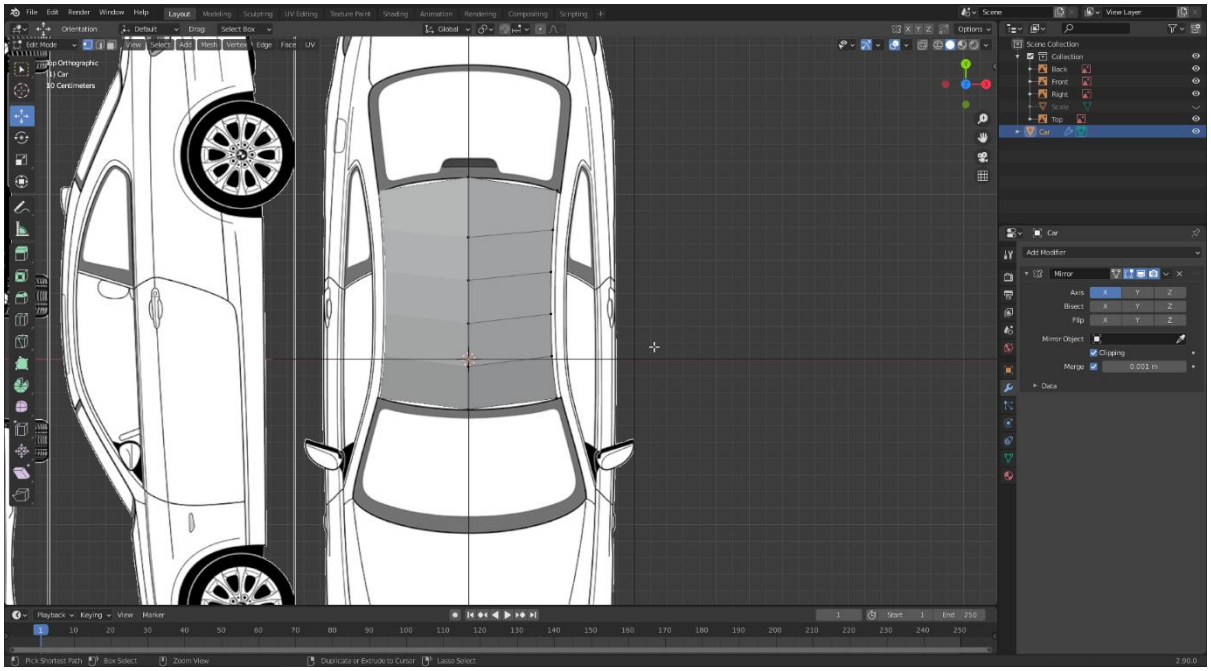
Figure 3. Blueprints

Method

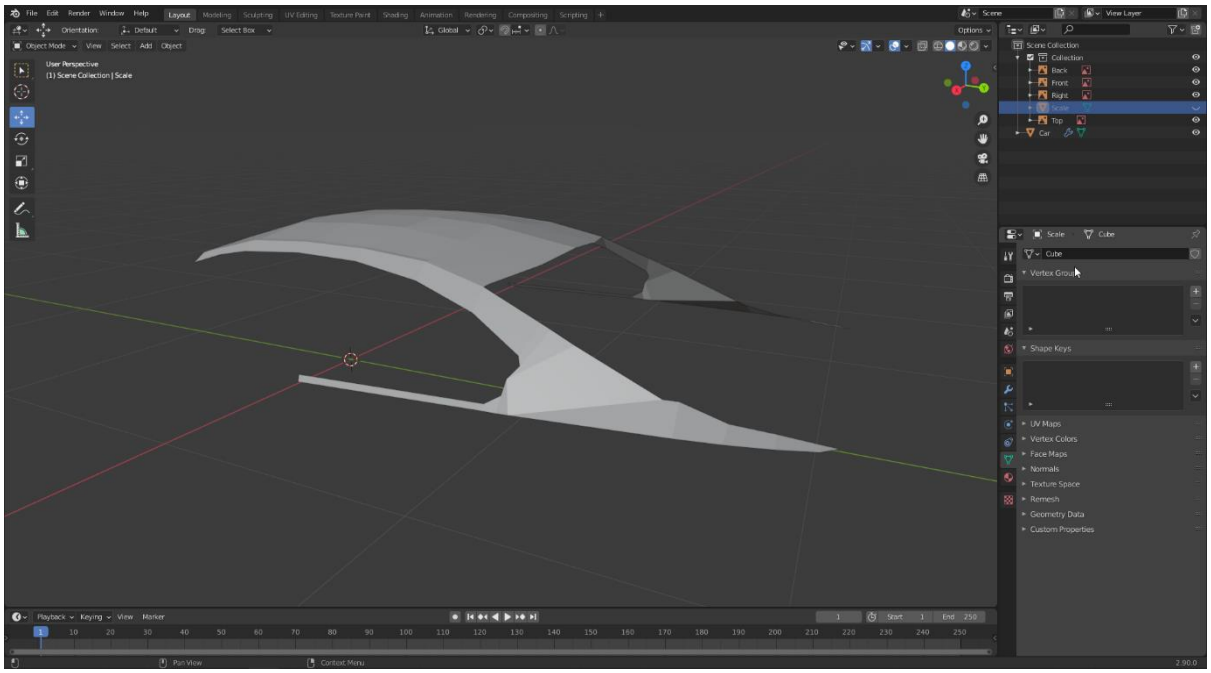
First, I put down reference images that I can follow to create the model. There is a total of 4 images, for the front, back, right and top. These images are only visible in their specific orthographic view thanks to the “Show in” option in the reference image, which makes it very easy to see how the model is looking without the images getting in the way. I aligned them based on a base cube to make sure that they match up in all directions, and thankfully due to the nature of the blueprints they were scaled correctly in all directions.

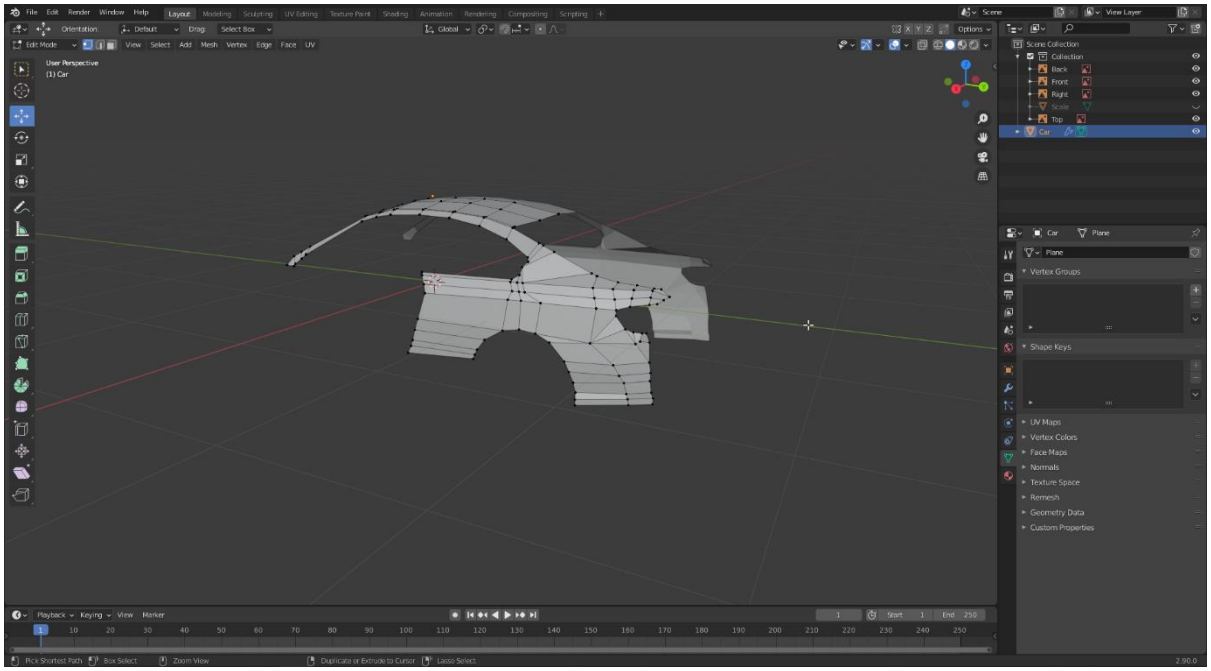


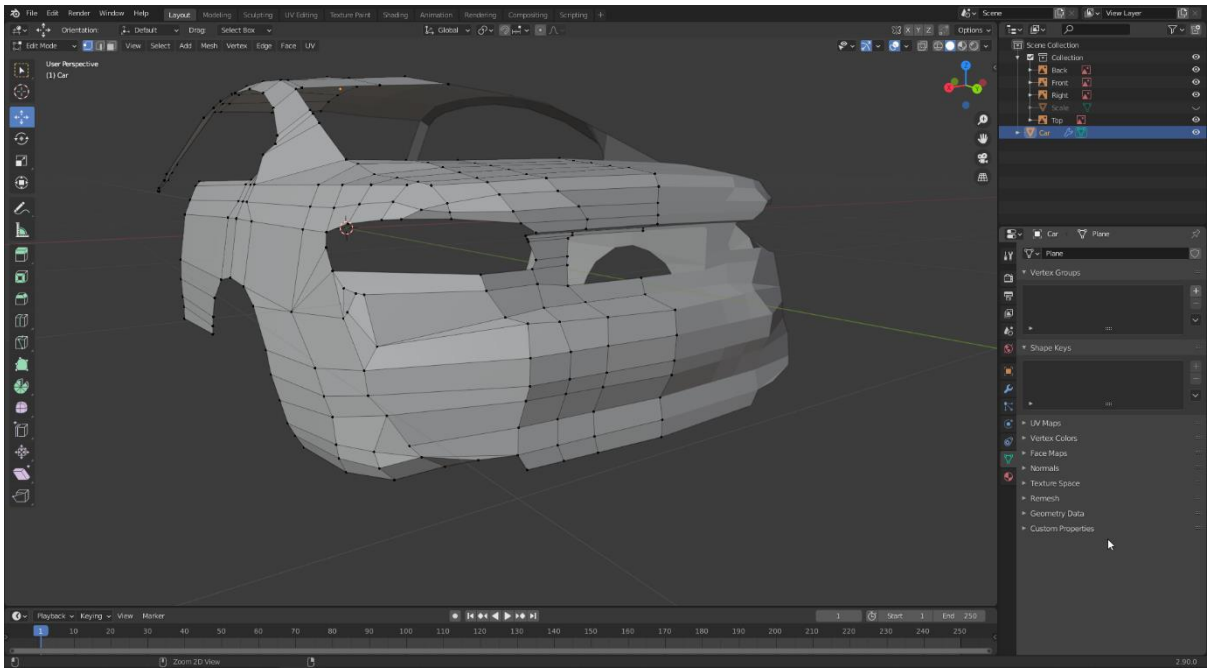
Next, I created a plane as the first part of my model. I gave it the mirror modifier with clipping enabled, which allows me to only work on one side of the model and for the other half to be created automatically alongside it. Any differences in the sides can then simply be made as separate objects. I used loop cuts to give the plane more vertices for me to work with and began shaping the model by tracing over the reference image. My workflow was to work on aligning it from one view first, and then correcting it from all the other views in order. With this method, positioning it from the top and right side would usually mean it is already correctly aligned from the front.



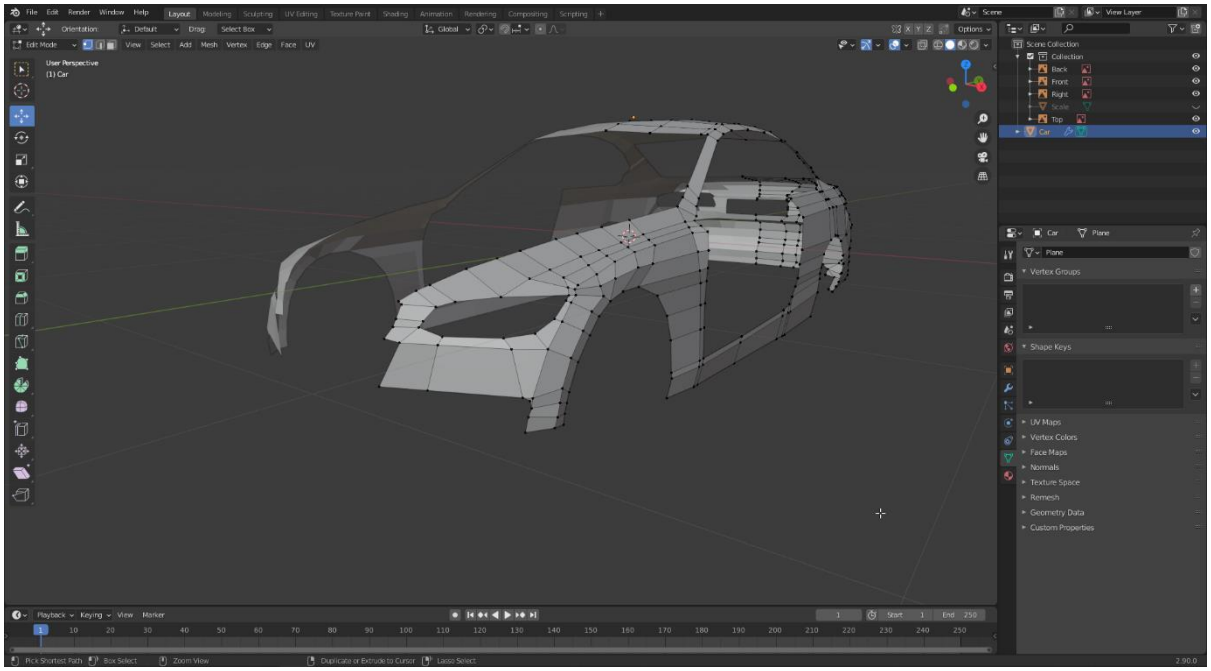
From here, it was simply a careful and long process of extruding the vertices out, loop cutting them and aligning them from each viewpoint to make sure they look correct. One thing I considered heavily is to make sure I'm working with quads by extruding out and not being afraid to use several loop cuts in one long shape. There are however several times where I used triangles due to the curved nature of the car.





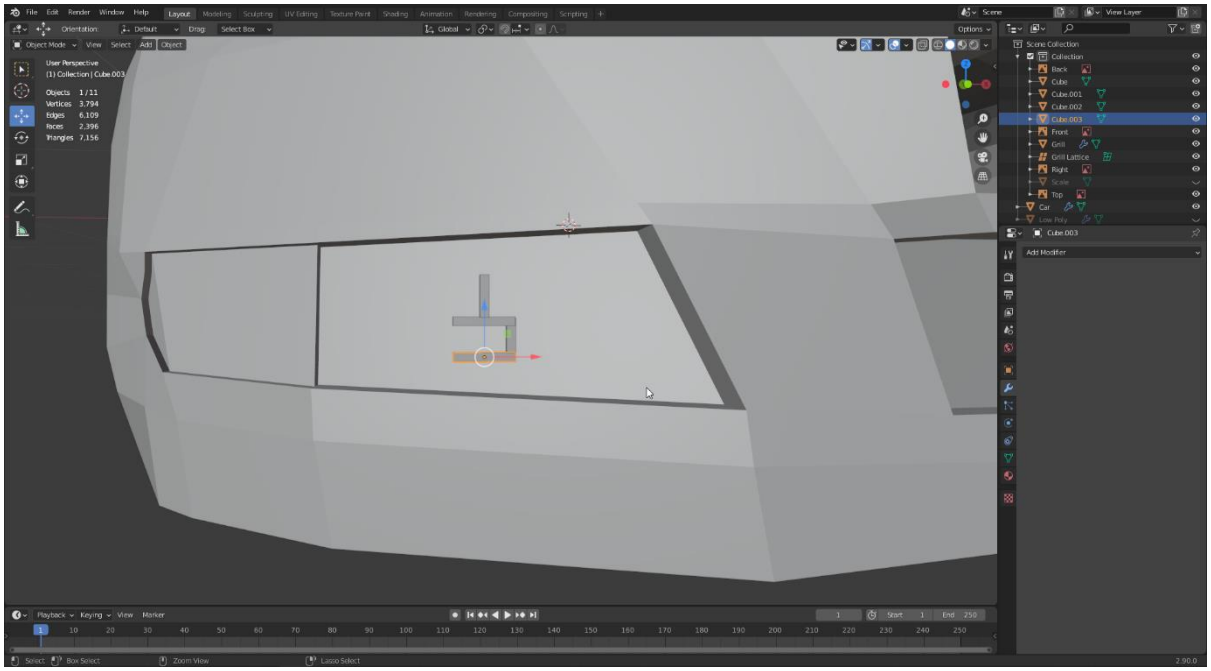


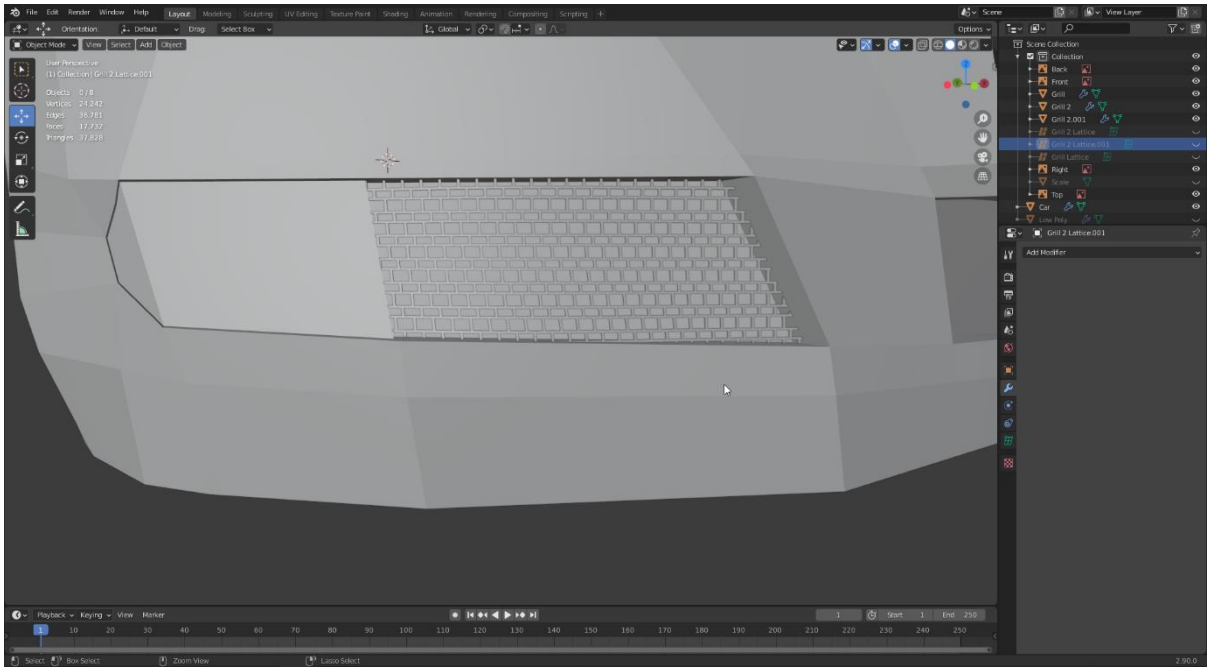
Due to my inexperience, I struggled a bit on the back of the car, and it ended up not looking as good as I would have liked, as well as using many triangles. However, I used this as a learning experience, and the front ended up looking a lot better.



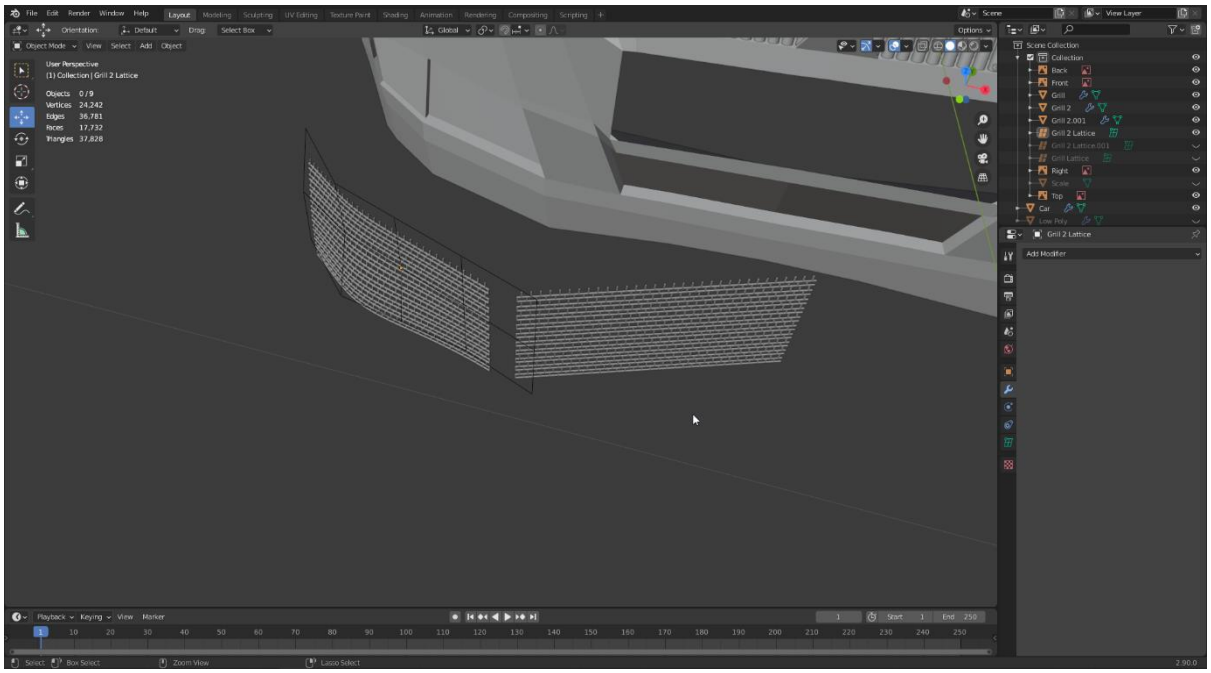
With the base model completed, I used the subdivision modifier to make the model a lot smoother and give it a more realistic appearance. I did 3 levels, which increased the vertices from 996 to 55,920, however made the car look much better.

Next, I worked on the grills at the front of the car. These are made up of straight cylinders for the top part, and a pattern for the bottom. To make the pattern, I made a simple shape out of cubes before applying the array modifier to copy them across, and again to copy them downwards. This gave it the appearance of a grid.

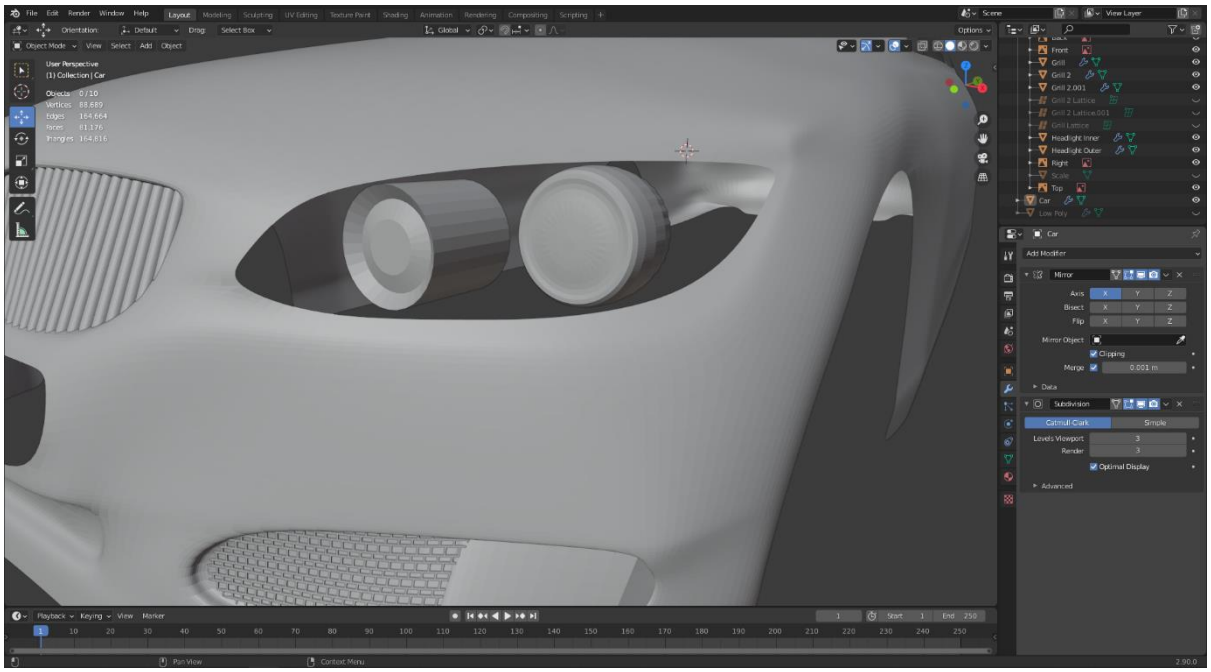




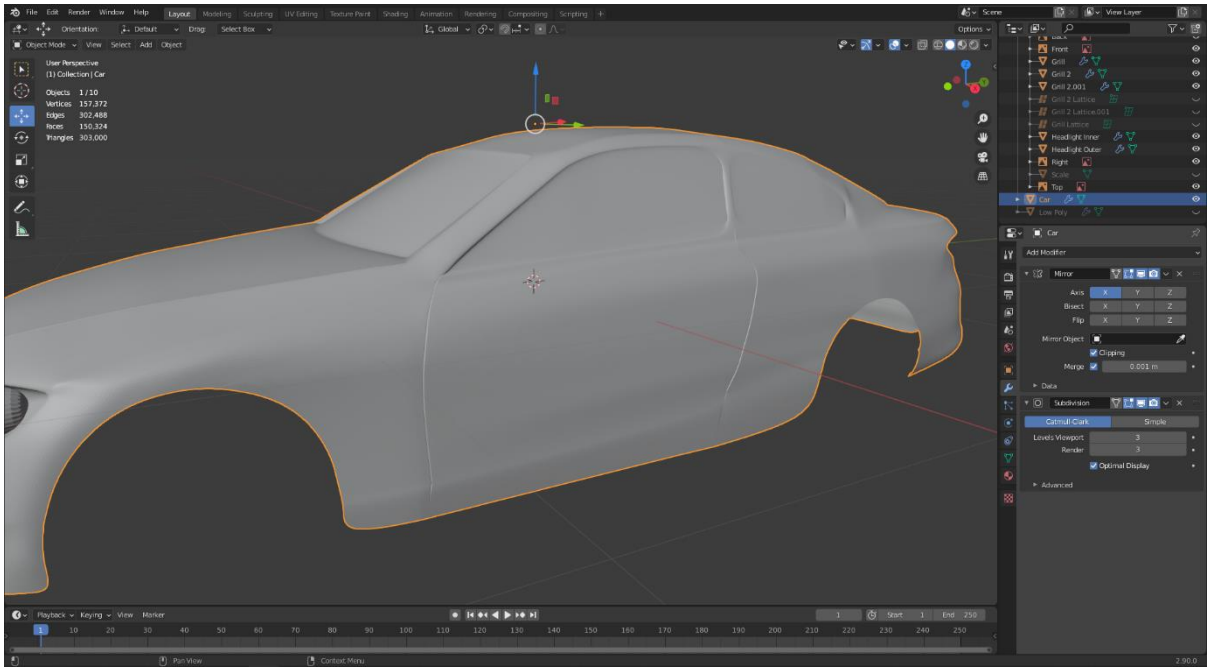
For the top parts, I did the same, except only using a cylinder instead of the cubes. However, the grills were straight whereas the model was curved, so in order to morph them into shape, I crated a lattice, which is a grid that allows you to set its resolution. I gave it a resolution of 5, 1, 3 and then applied a lattice modifier to the grill. This made the grill take the shape of the lattice, meaning I simply had to transform the vertices of the lattice in order to make the grill fit correctly.

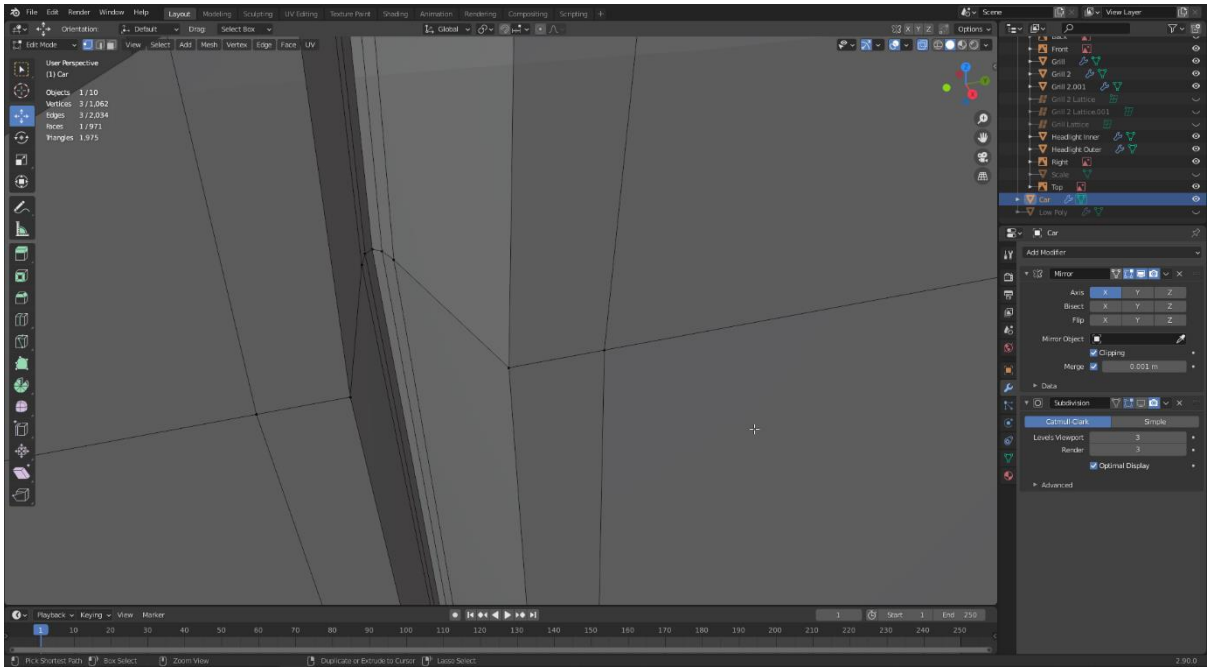


For the lights, I simply extruded and scaled a cylinder to give a similar appearance to the headlights of a car based on my reference image, and then applied subdivision to make it look more detailed. I then mirrored this to the other side.

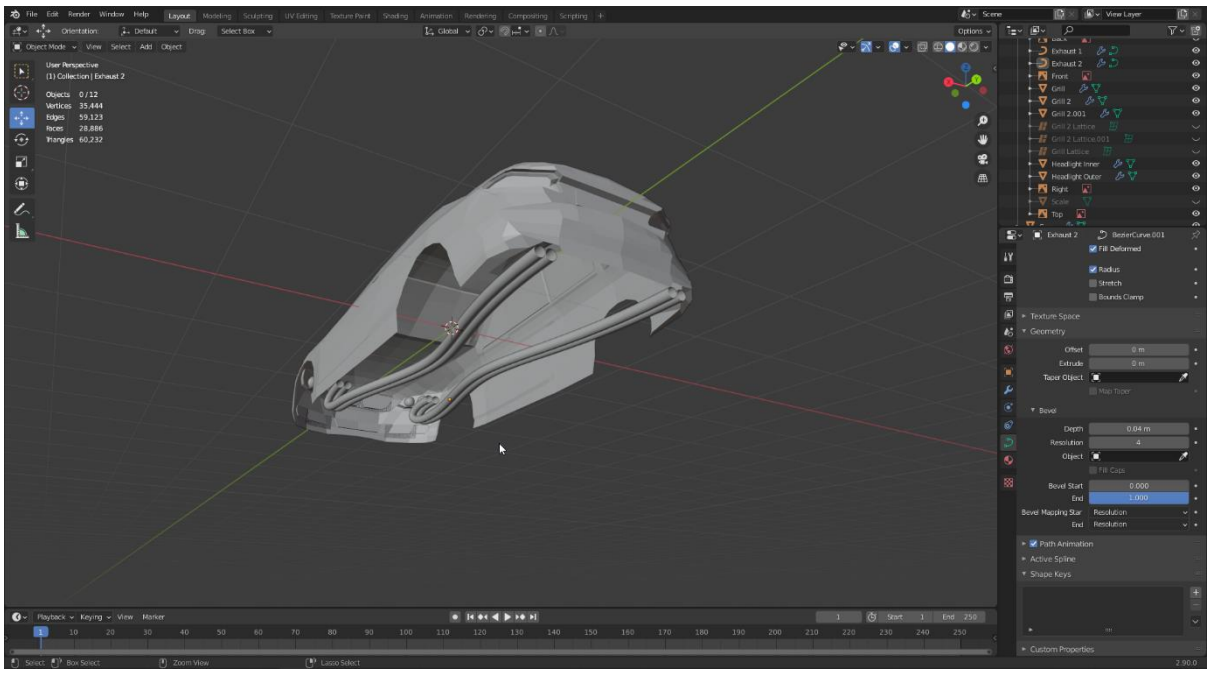


Next, I filled in the windows, which was a simple case of extruding the trim of the windows inwards, and then creating faces between all the vertices on either side. This allowed me to curve the window inwards how I wanted them. I also added spacing between the car hood and doors to make them appear separate to the car. I did this with a series of very small loop cuts, which I then dragged inwards and bevelled. This makes very smooth lines between the door and the body, especially when subdivided.

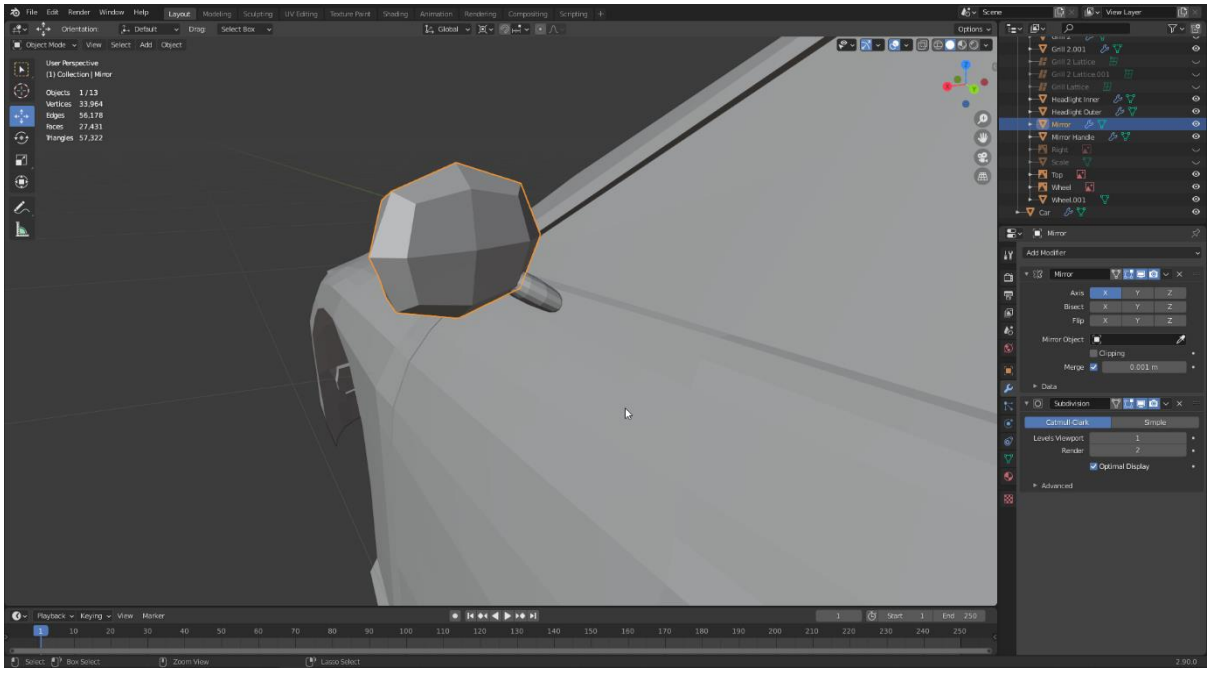




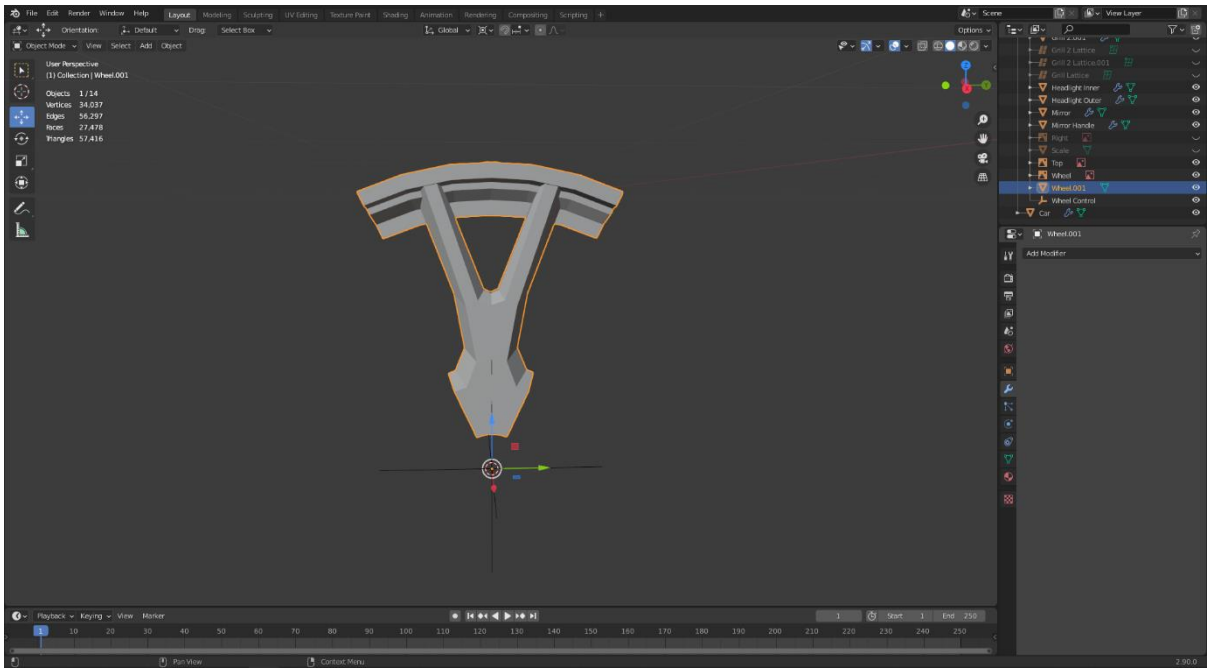
For the exhaust, I used a Bezier curve which I extruded to resemble the exhaust pipe of a car going to the engine. Then, to make it appear, I simply added a bevel which created a perfect pipe alongside my curve. I used a small resolution of 4, as I didn't want to have too many polygons on such a small part of the car which is typically hidden underneath.



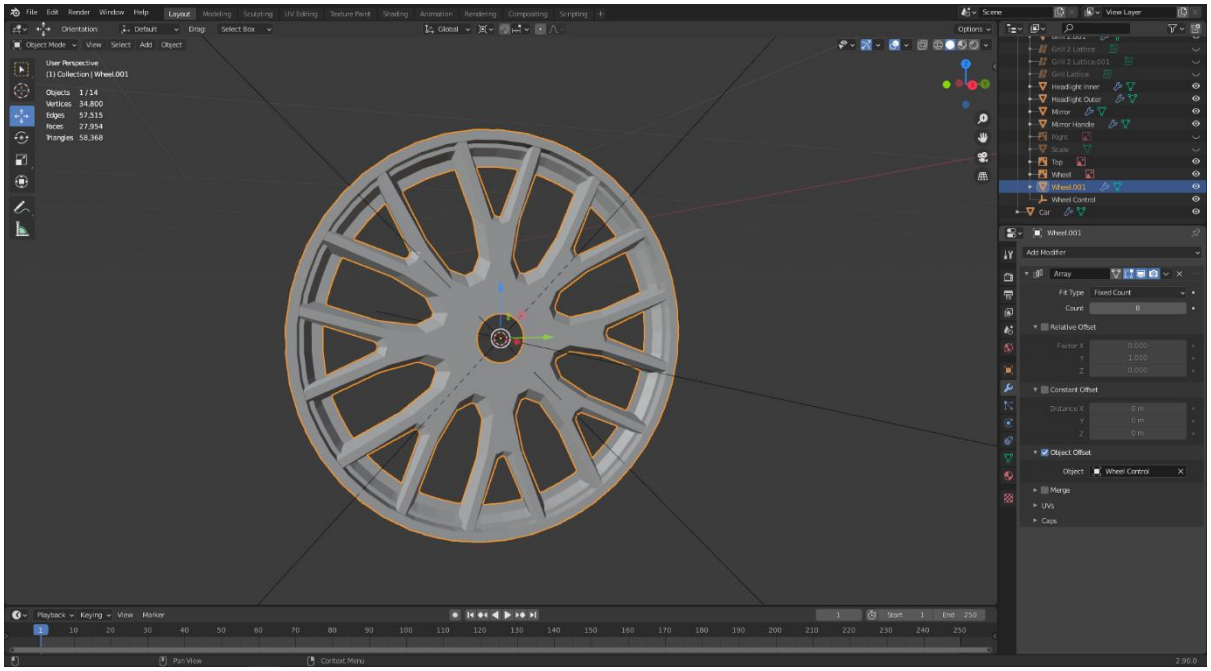
I also modelled the mirror using two subdivided cubes. This gives them an almost round appearance, which I morphed into a mirror shape.



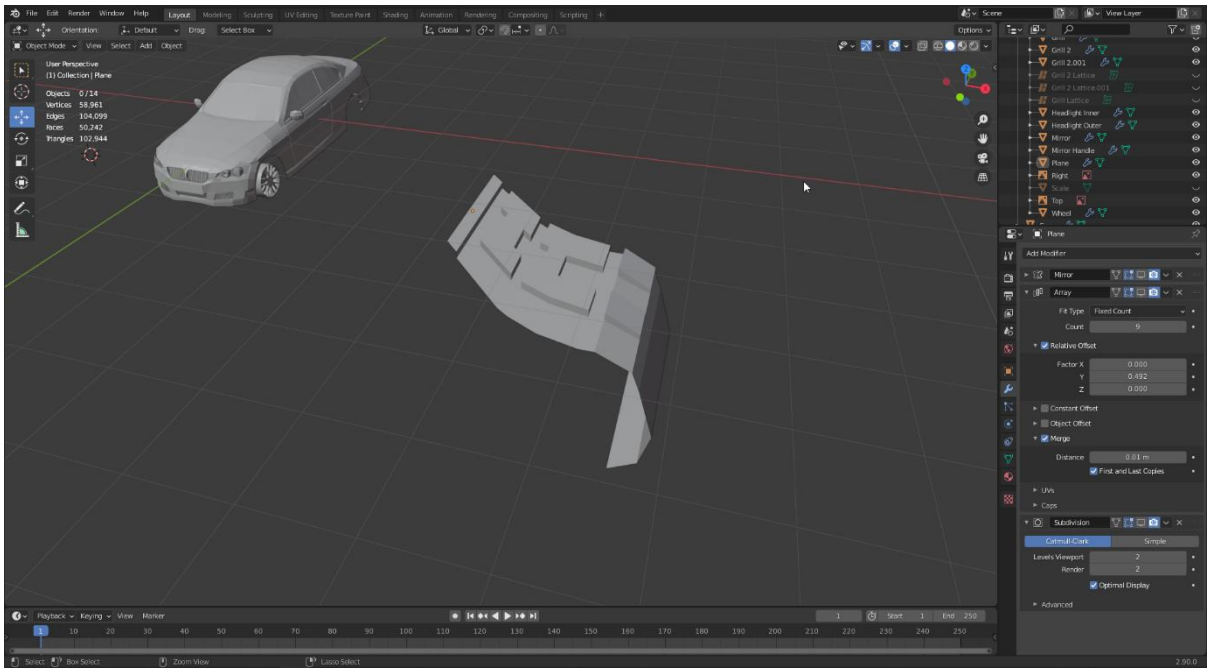
Creating the wheels was a bit tricky but using modifiers I was able to get a nice result without much manual work. First, I created one part of the rim.



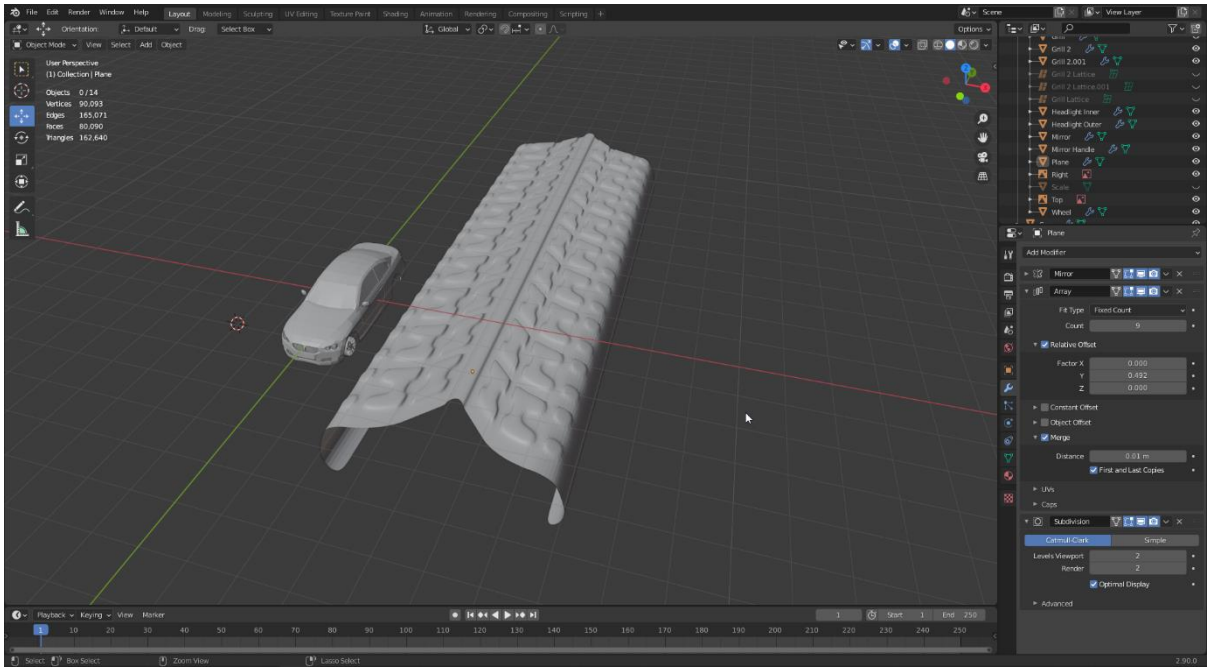
I put an empty axis on the wheel, and added an array modifier to the rim. Then, I set the object offset as the empty, as well as set the iterations to 8. Spinning the empty then moved the array objects, allowing me to position them into a perfect wheel.



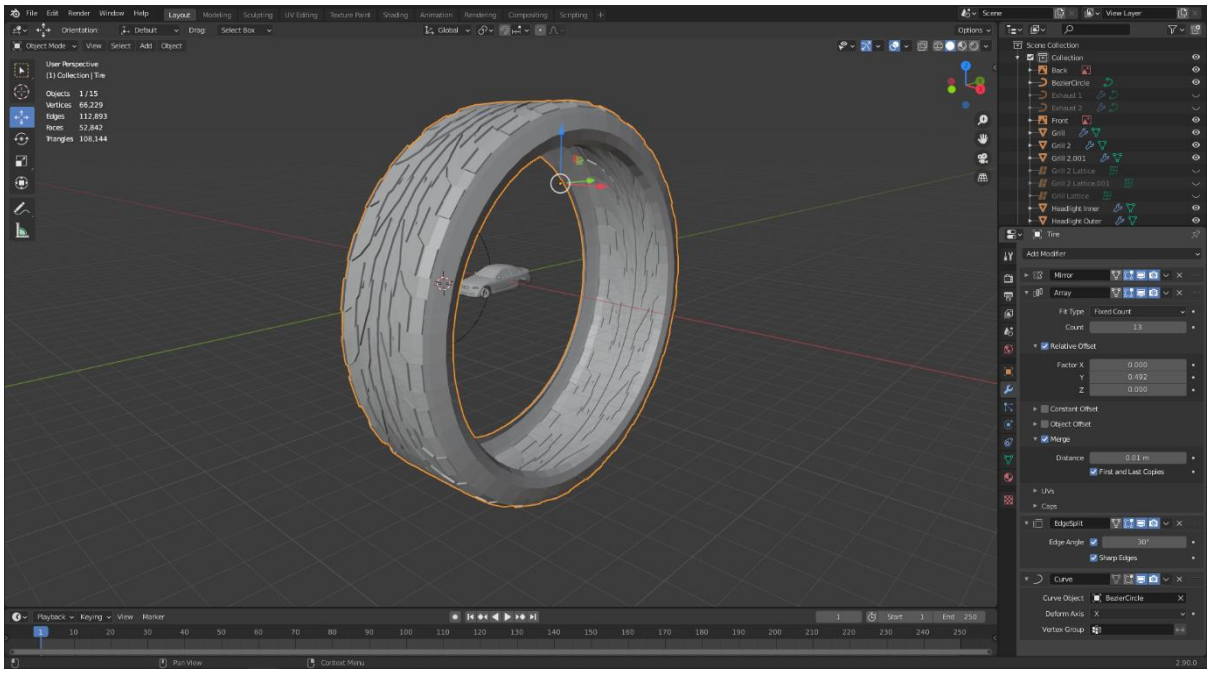
For the tire, I started out with a slice of the pattern that the tire would have.



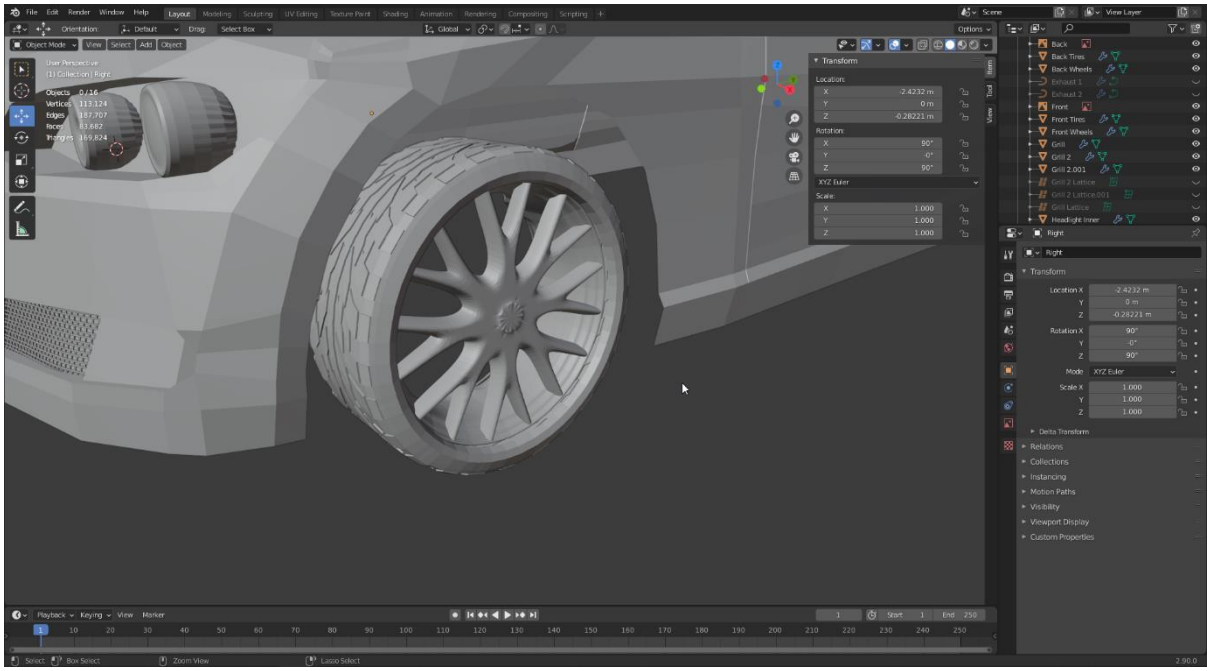
I added a mirror modifier to make the other half, an array to make it longer, and a subdivision to make it look more detailed.



Next, I created a Bezier circle, and used it as the curve object of a curve modifier on the tire. This shaped the tire into the shape of the circle.



Finally, I placed the tire onto the wheel, and positioned it on the car.

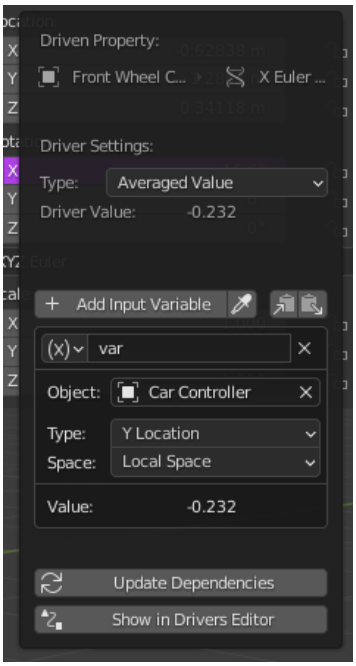


Finally, to finish off the model, I extruded the bottom vertices of the car inwards, and then created faces between them. I also filled in the wheels and grills.

Finally, I set about making the rig and animation controllers for the car. For animating the position of the car, I used an empty called Car Controller. The body of the car is parented to this empty, meaning if the empty moves the entire model will move without having to mess with the model itself. Next, I created an axis for both back wheels, and separate axes for the front wheels. The wheels are parented to their respective axes, and the axes are parented to the car controller. Now, if the car controller is moved, both the car and the wheels will follow despite being independent.

The reason the rotation x value is purple is because I added a driver to the rotation. These allow the value to be changed by a function or equation. The driver type is averaged value, the type is Y as that

is the direction of the car, and the space is local space. This takes the average value of the car controller' location and wheel controller's Y rotation and applies it to the wheel controller in real-time. This means when the car controller position is changed, the wheels of the car rotate as if the car is driving across the ground.



Next, I created an empty cone at the front of the car. Once again, this is parented to the car controller so that it follows it along. Then, I gave the front wheel controllers a copy rotation constraint, making them take the target's rotation and apply it to themselves. I set the target as the directional controller (empty cone), the axis as the Z axis, and the mid as Add. The effect of this is that the empty can be rotated to make the wheels rotate accordingly, allowing them to be animated without the model ever being touched. Finally, the directional controller has a limit rotation constraint with a min and max value configured to stop the wheels from rotating all the way around like a real car would.

Reflection

Overall, I am satisfied with my final model. I believe that it is a visually appealing model, with a lot of detail in the body mesh and wheels. I also think its easy to tell what the car is based off, without out right being a full copy of the real version. However, there are several issues that the model has, as well as a number of improvements I would like to have added.

Firstly, the vertices count of the model is very high, at 1,175,087. This is not necessarily a problem, as most of it is caused by the subdivision modifier, especially on the smaller objects like the lights and mirrors. This could be easily solved by simply reducing the value of the subdivision modifier.

Secondly, the back part of the car does not have as much detail as the front part. This is mainly because I was not experienced with creating such a complex model and thus made a lot of mistakes. If I went back and redid the entire back section, I believe I could have put in a lot more detailed and made the back section appear much better. However, despite being worse than the front, I'm still quite pleased with how the result looks.

Finally, the model could be greatly improved by adding an interior and underside model. This would take a lot of work and effort, and I could not achieve it in my time frame, however if I could have achieved it, it would be a massive improvement to my model.

Task 3

Introduction

Task 3 is to create a scene based on a reference image using custom made models and materials. Out of the three images, I decided to create the outside house scene, as I like the lighting in the scene as well as the inclusion of glass and water. I used Substance Designer to create the materials, and Blender for the models and render.

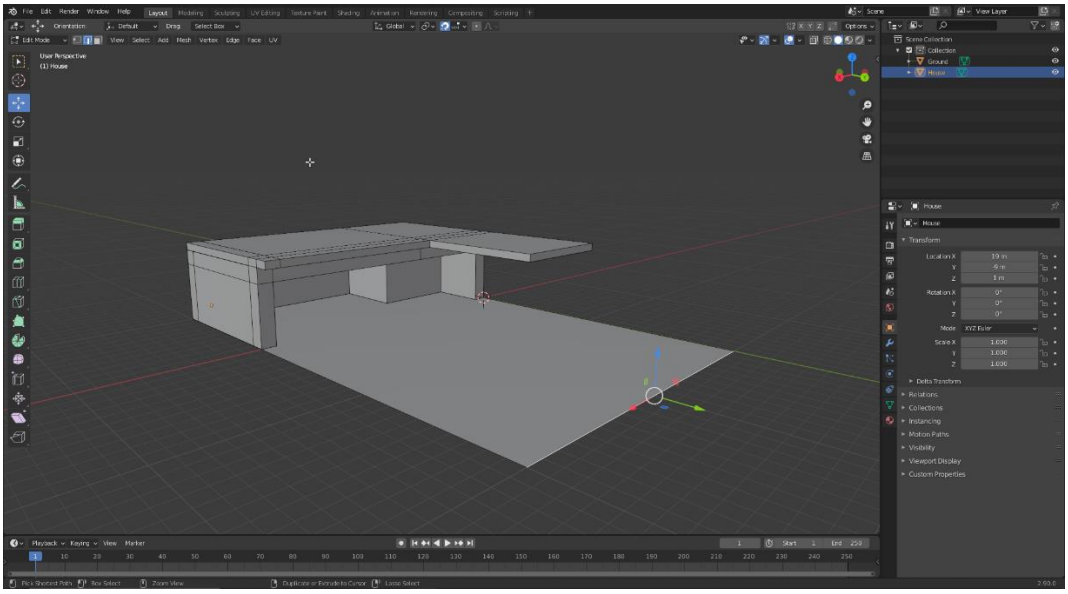
Conceptualisation

This task did not require much planning nor research, as I was working off the reference image provided in the resources. I based both the model and materials all on this image alone.

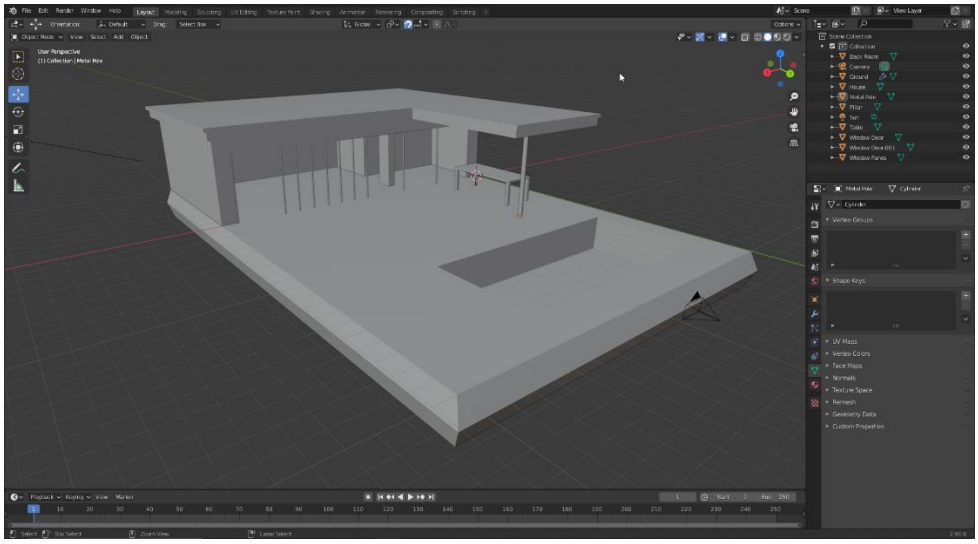
Method

Model

First, I worked on creating a simple scene in Blender based on the reference image. I started with a plane as the ground and a cube which I extruded out into a basic shape.

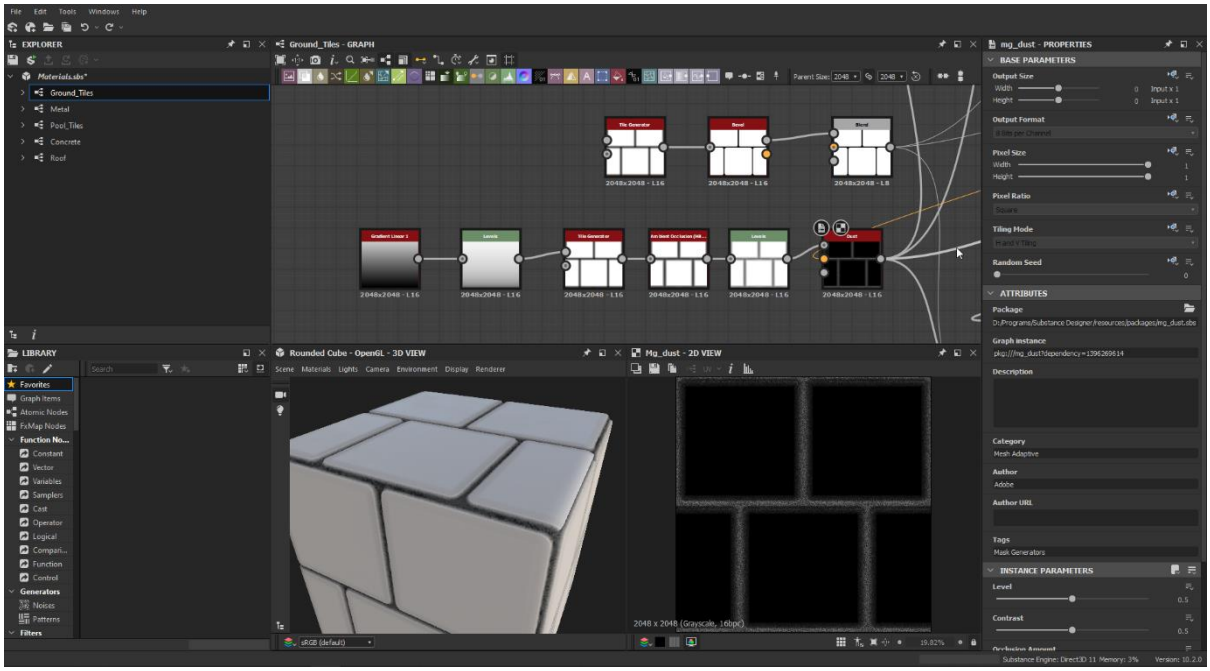


I refined this model by adding different objects that specifically would have different materials. For example, the back walls would be a different colour, the top of the roof would be a metal material, and the pool would have its own tile material.

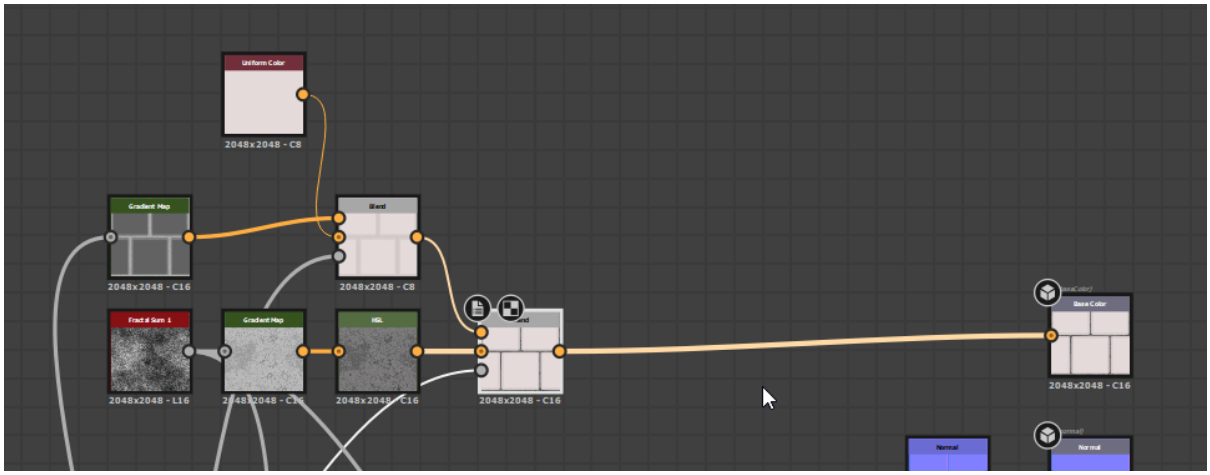


Materials

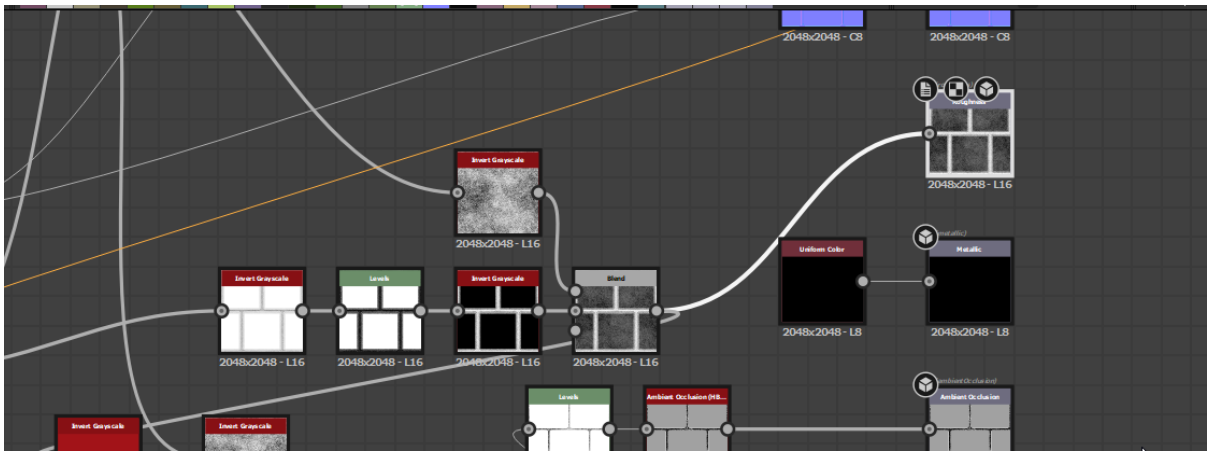
First, I created the ground tiles material. I wanted the top to be almost perfectly smooth, with a few surface imperfections, whilst the spaces in between being filled with rough cement. To achieve this, I used a tile generator to create the main shape of the tiles, offsetting it by half in order to get the pattern. I gave it a bevel to round out the shape and used this for several of the other inputs. Similarly, for the 'dust' or cement part in between the tiles, I used the same tile generator, but with a linear gradient brought through a levels node, then slotted into a 'mg_dust' map to get the grainy effect.



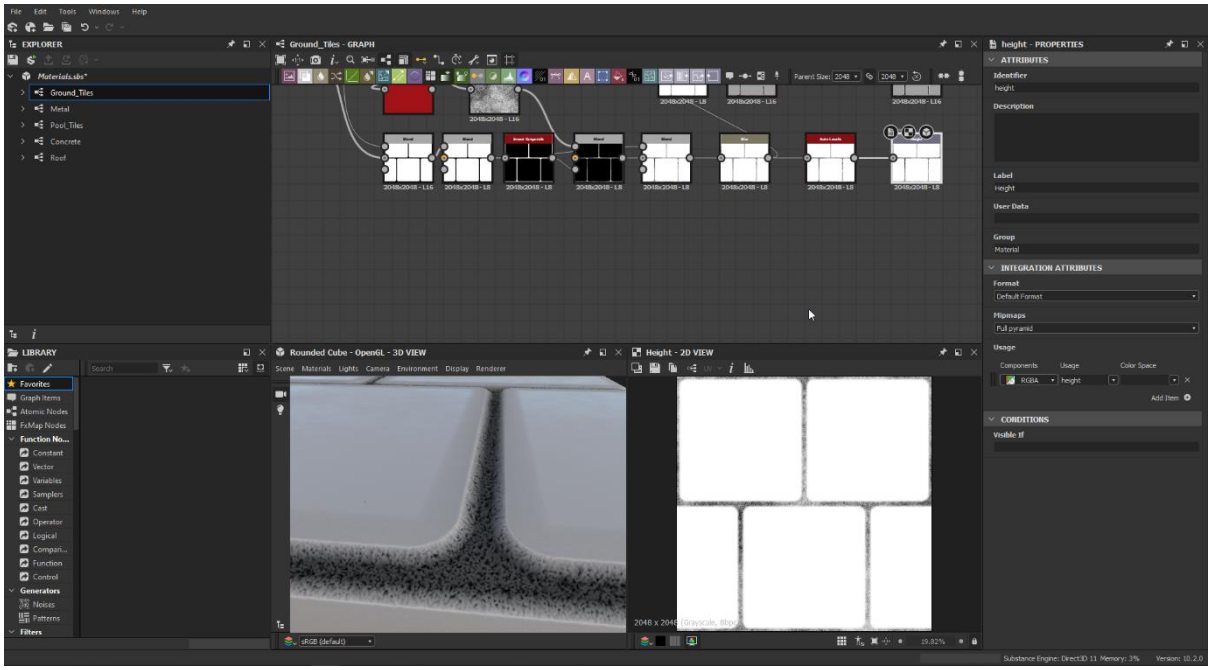
For the base colour, I brought it through a grey gradient map, blended with a whiter grey. At the same time, I used a fractal sum to get more noise, through a HSL node to adjust the lightness levels, and finally blended back to combine with the whiter panels. This is used as the base colour of the material.



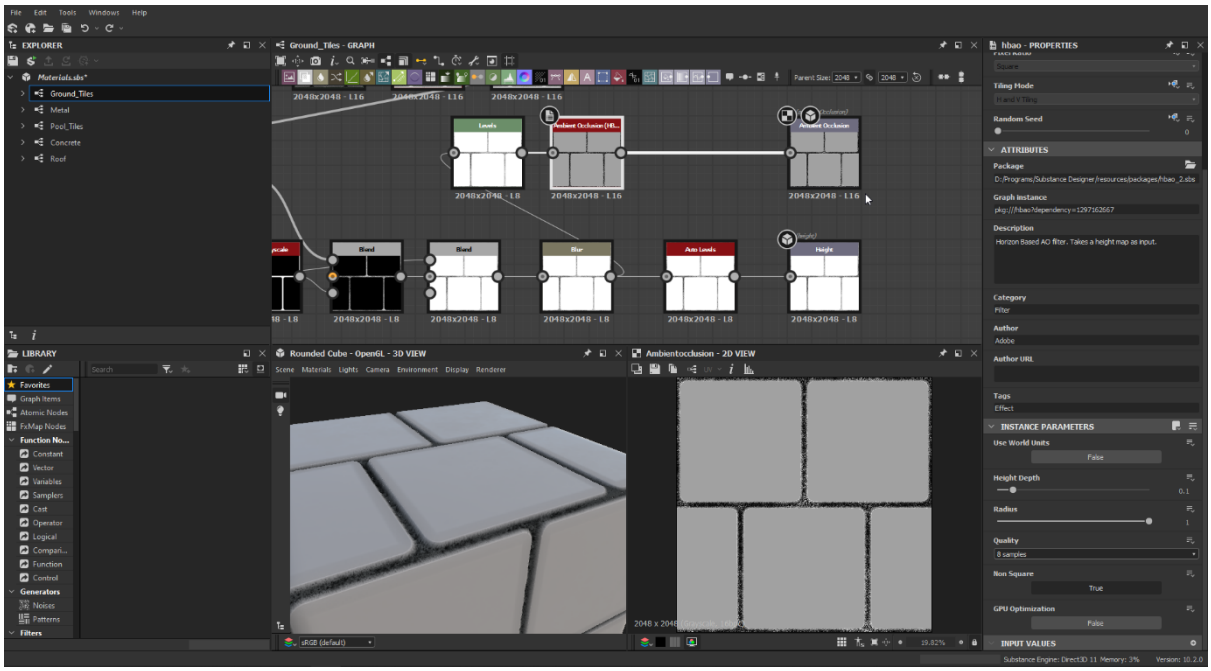
For the normal, I only needed to use the output of the tile generator, and the metallic is a simple black value. The roughness map uses the dust output, which I invert and blend together with the gradient map that is used in the base colour. The normal is used to show which areas of the material are lighter or darker to give it artificial depth. Meanwhile, roughness shows how glossy the surface is.



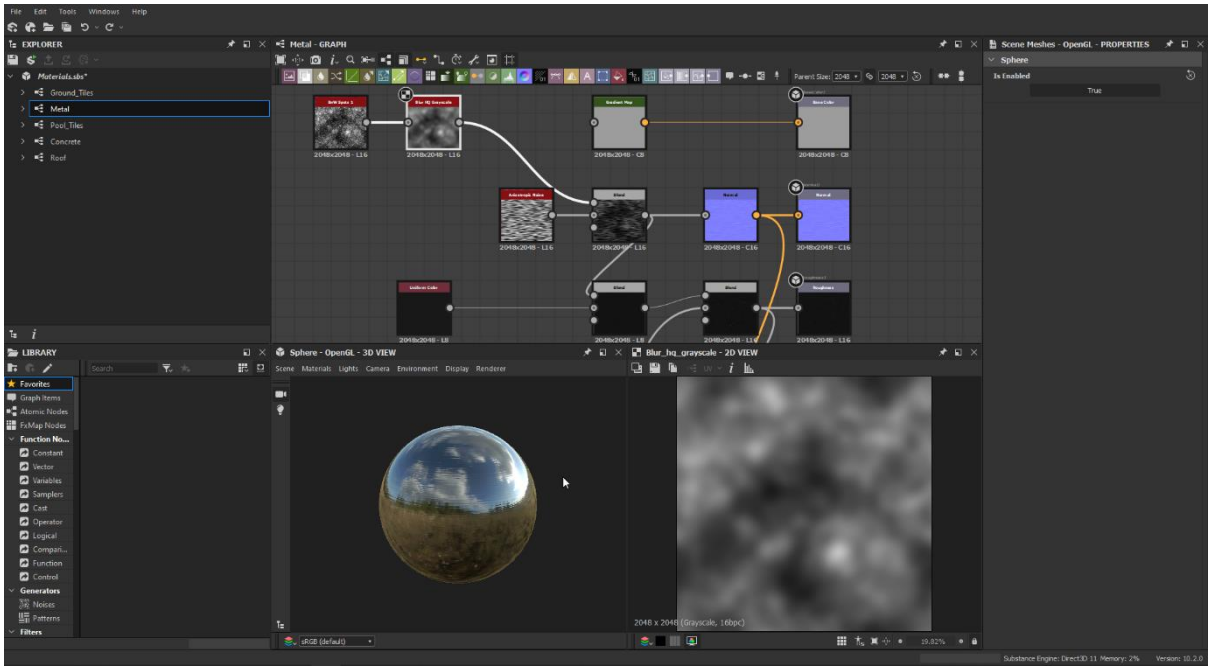
For the height map, I needed to keep the panels white, but make the in between areas grey and noisy to give them varying bumps. I did this by inverting the tiles, blending them with the fractal sum to overlay it on top, then placing it back onto the white tiles. This makes the tiles straight and the dust areas bumpy.



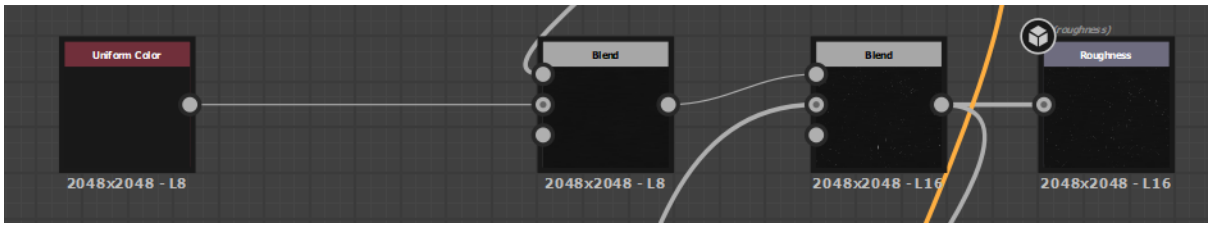
Finally, for the ambient occlusion, I took the height map and converted it in an ambient occlusion node. Ambient occlusion is simply a value that controls how much of each part is exposed to the ambient light in the scene.

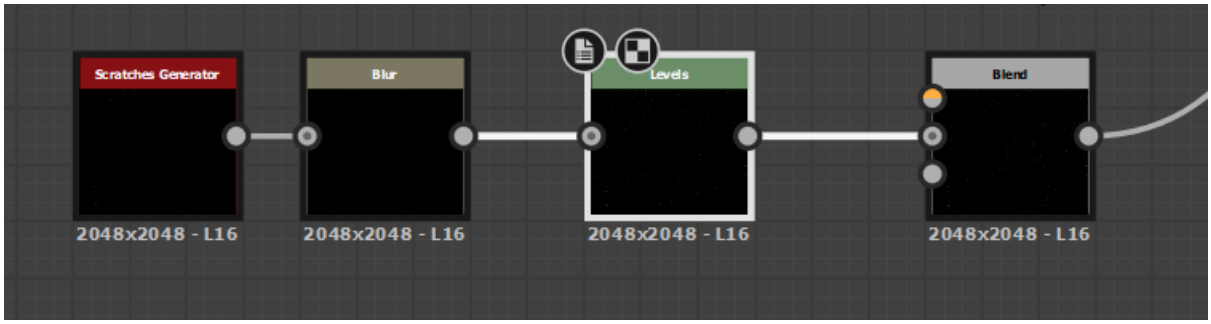


For the metal pole material, I used a grey gradient for the base colour, and wanted to give it a brush finish. For that, I used “bnw spots” and anisotropic noise, blended after blurring the spots. This firstly goes into the normal map to make the material appear slightly bumpy.

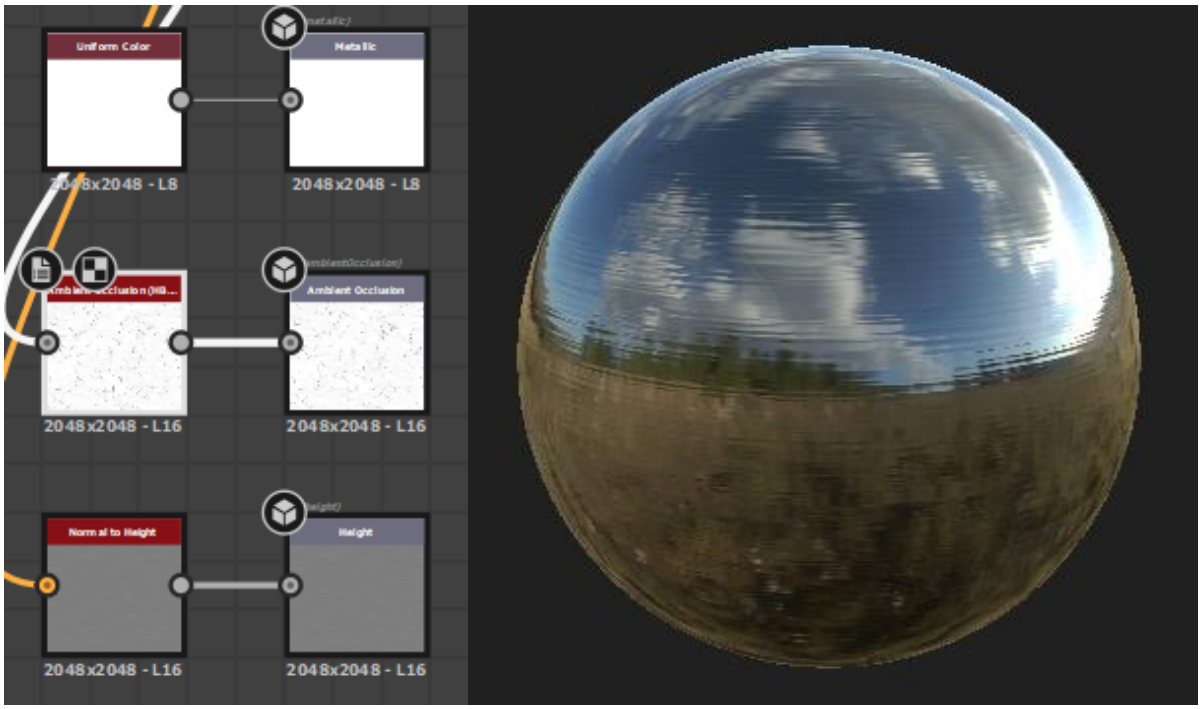


For the roughness map, I simply had to take the previous nodes and blend them with a black uniform colour. This is then combined with scratches, which I created using the scratches generator with a blur node to make them rounder and levels to make them more defined.

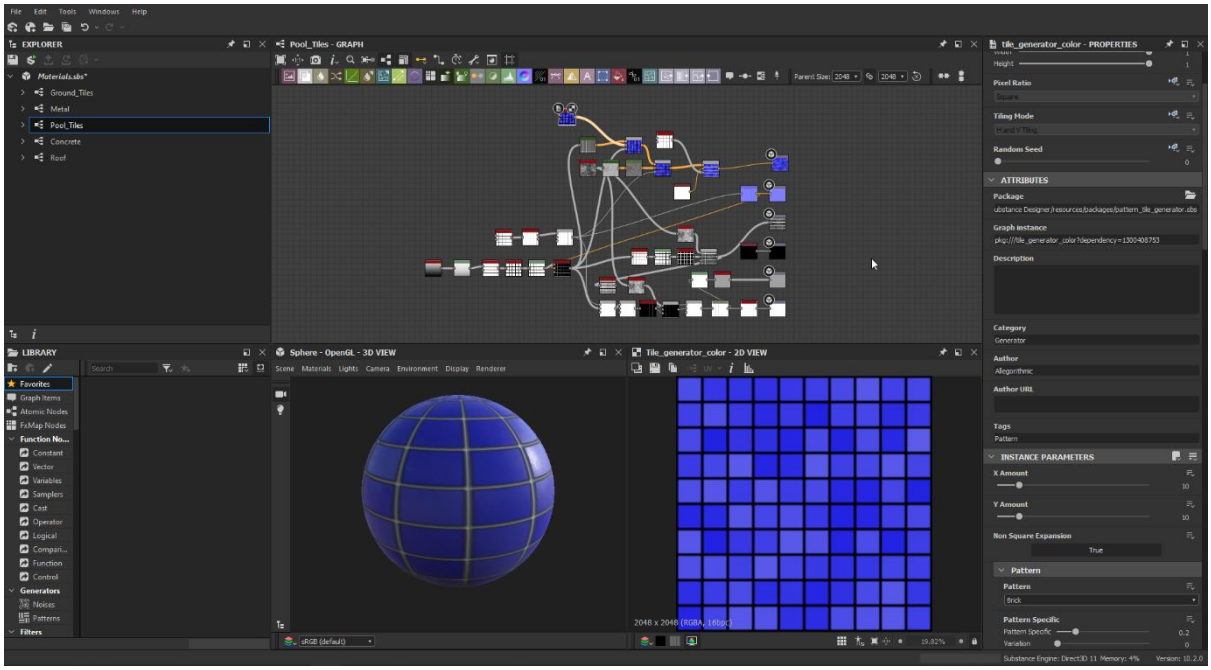




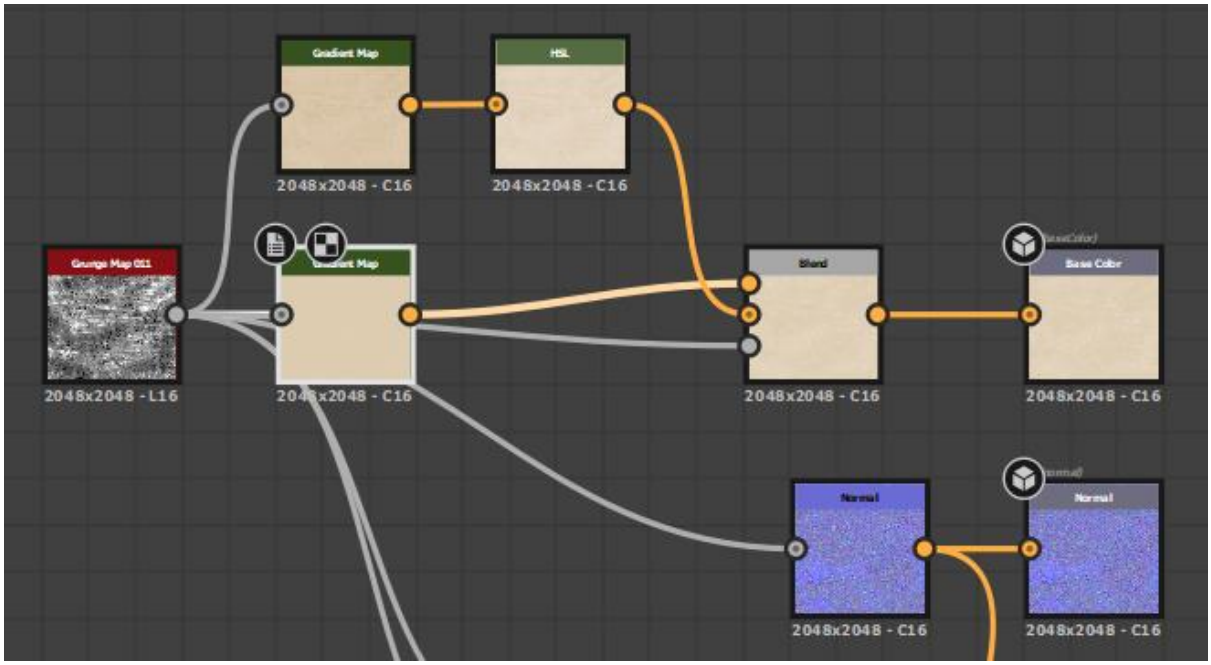
As the entire material is metal, the metallic map is just a white colour. The ambient occlusion is the roughness map converted into an ambient occlusion map, and the height map is the normal map converted to a height map. This gives a very reflective material with a lot of imperfections in the form of scratches.



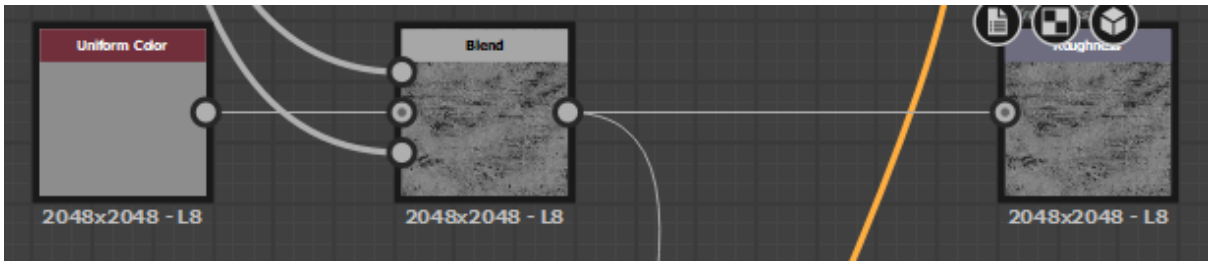
The pool tiles material is almost the same as the ground tiles, but I used a tile generator colour node instead of the uniform colour and added 10 instances of tiles instead of 2. This gave a much different look to the tiles despite the very small changes.



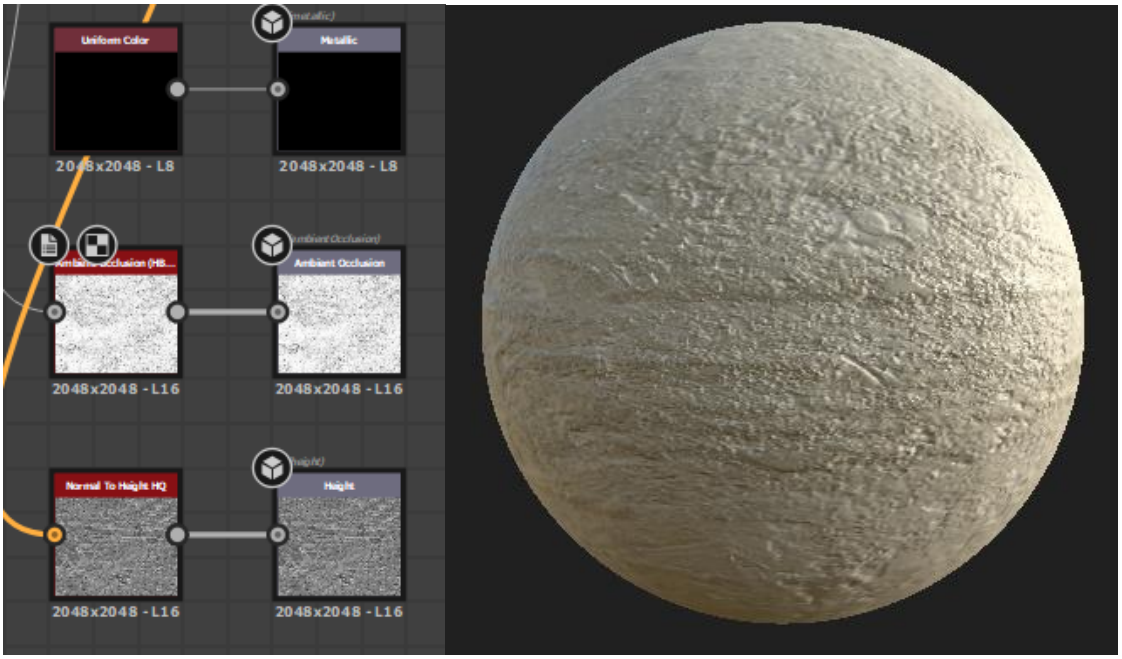
The concrete material is a very simple but effective material, as PBR materials work very well with a lot of micro details. The entire material is covered in spots thanks to a grunge map, which I blended with two cream gradient maps to get the bottom and top colour layers. The background is darker and the foreground very slightly lighter to give the effect of different bumps. The grunge map is also directly used in the normals to give it even more depth.



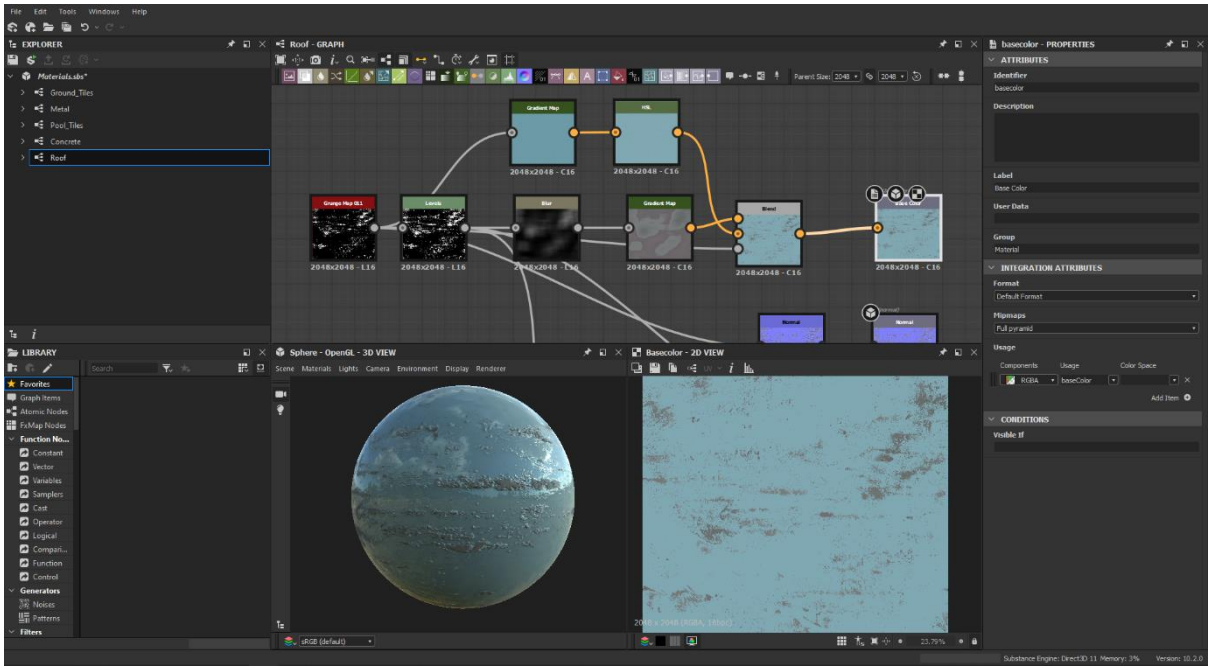
The roughness map is once again the grunge map but blended with a grey colour to give the deeper parts less gloss. In a roughness map, the black parts are less glossy, and the lighter they get the glossier they are.



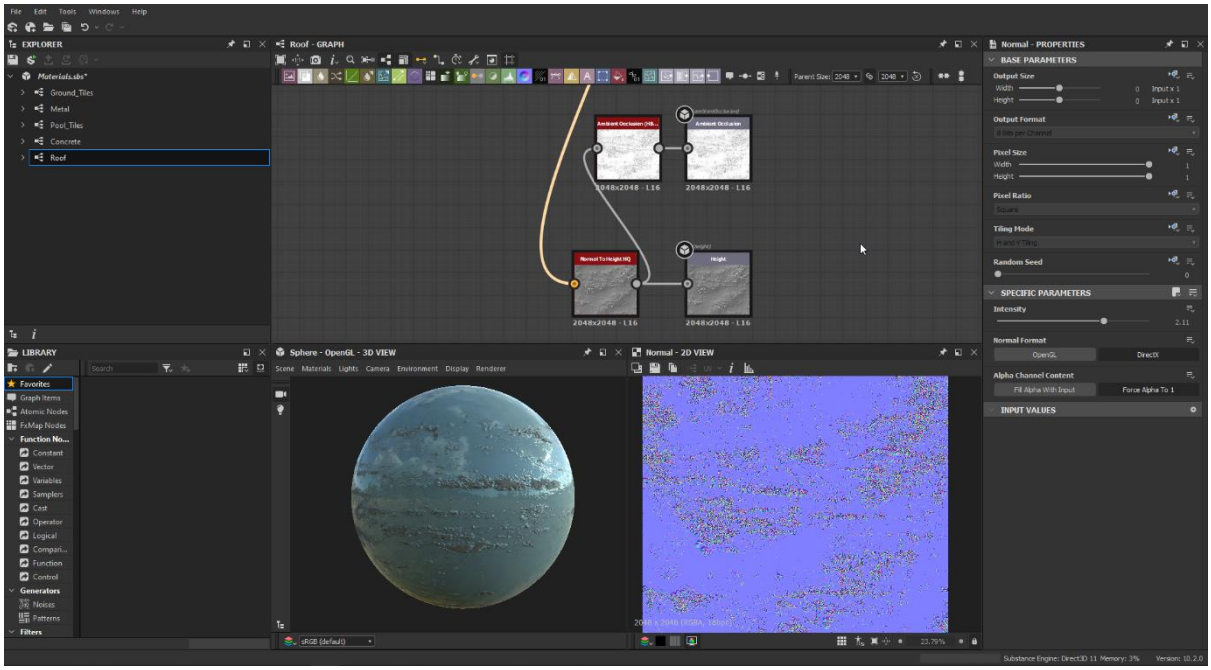
As the material is not metallic at all, the metallic map is completely black. This is because white signifies a completely metallic surface whilst black signifies the opposite. The ambient occlusion is then the roughness map in an ambient occlusion node, and the height map is the normal map through a normal to height map. The result is a very bumpy, cream coloured material with very noisy roughness and ambient occlusion maps.



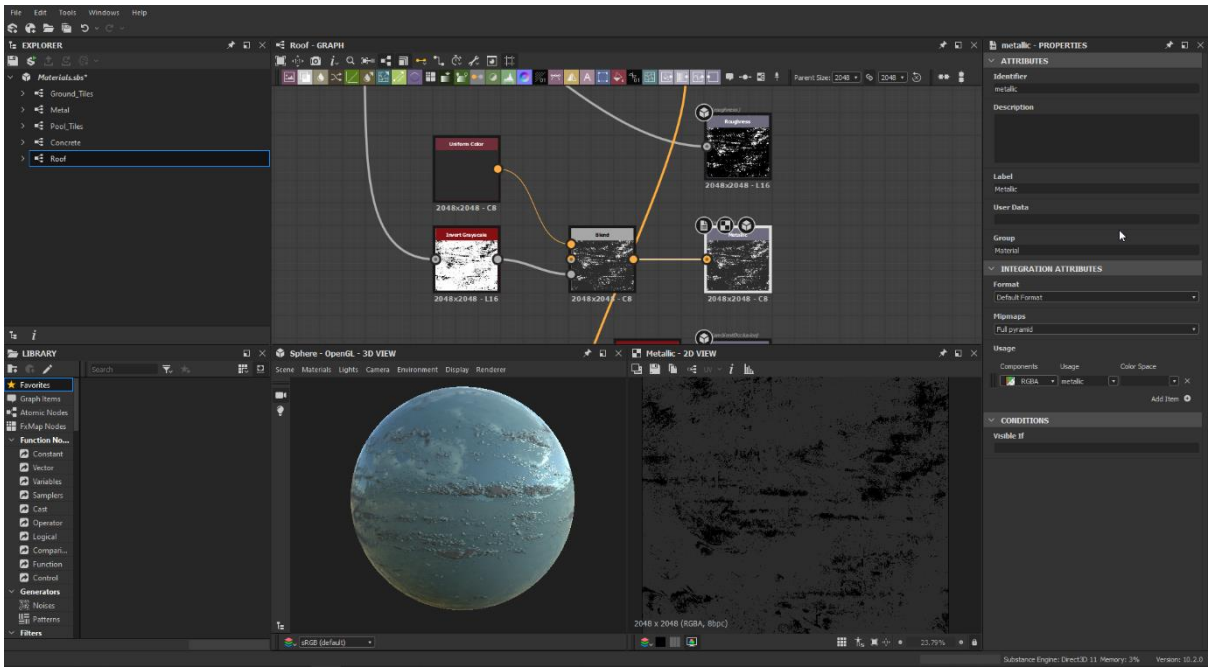
Finally, the roof material is a slightly dirty or rusty metal, where the metal bits are quite reflective compared to the matte dirt. This one also uses the same grunge map, brought through a levels and blur node to reduce the amount of dirt on the material, and blended with a light teal gradient map for the metallic parts. This is used as the base colour, with a large teal surface area and small amount of brown dirt scattered throughout.



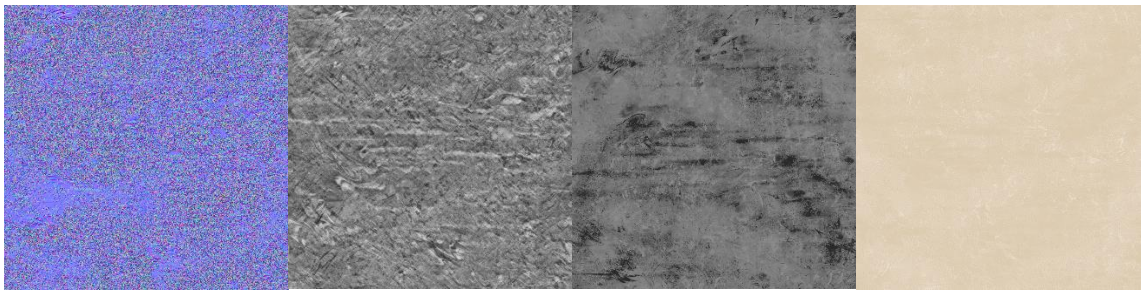
Like before, the normal is the grunge map, the height map is a normal to height map node, and the ambient occlusion is the result of that height map node.



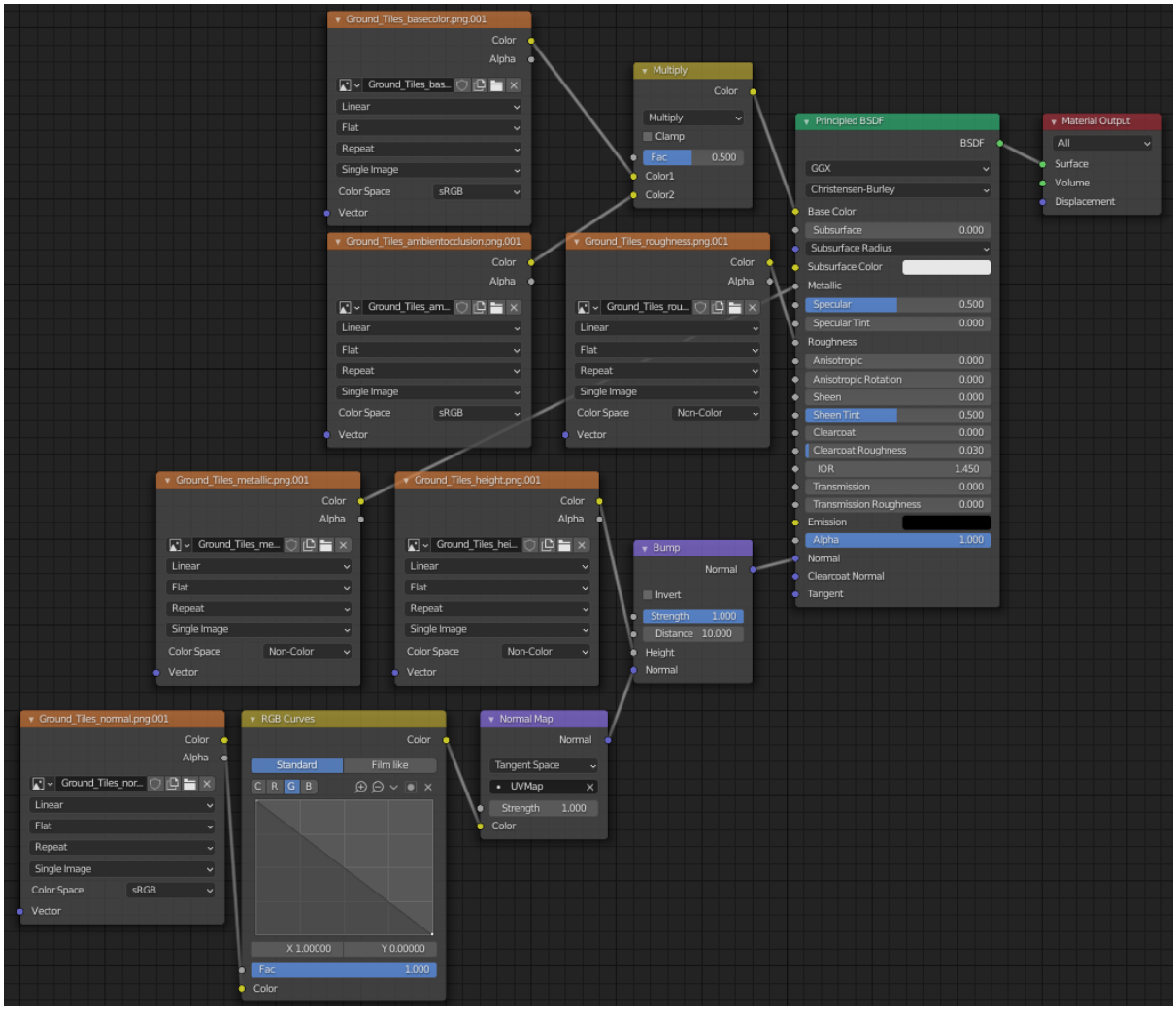
The roughness map uses the grunge map to make certain parts in between the dirt glossy. Meanwhile, the metallic map makes the dirt bits non-metallic by inverting the grunge map and blending with a grey colour to make the non-dirt parts metallic.



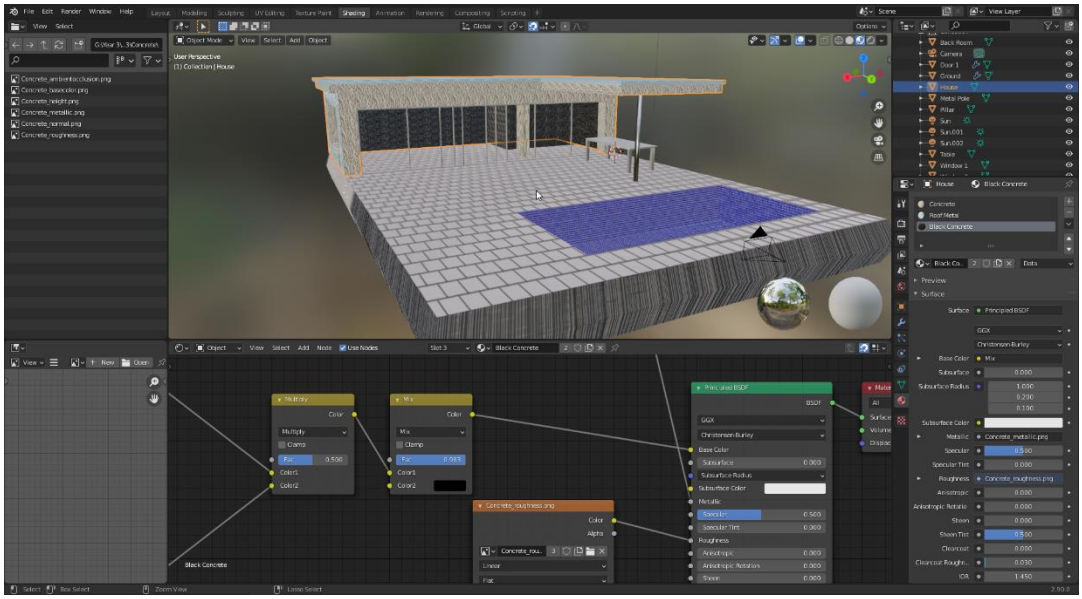
To import the materials into Blender, I first exported the material into separate maps.



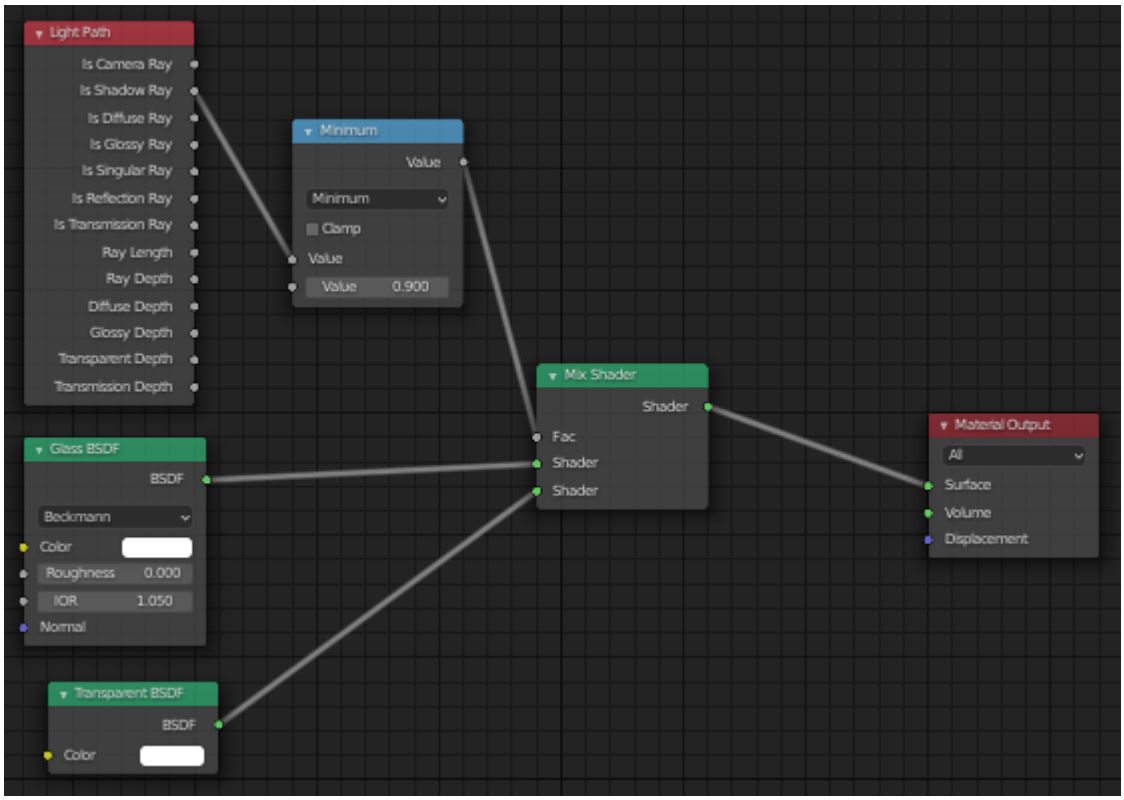
Next, I create a new material in Blender, and simply slot the materials into the ‘principled BSDF’ node. The ambient occlusion and base colour nodes go through a multiply node to control the effect of light on the material. The normal map goes into an RGB Curves node to make it readable by Blender, and then into a bump map which takes both the normal and height map as input to be slotted into the normal map. Finally, the ambient occlusion, base colour and normal maps use the sRGB colour space whilst the rest are non-colour. These settings work across all of the materials by simply slotting the correct maps into them.



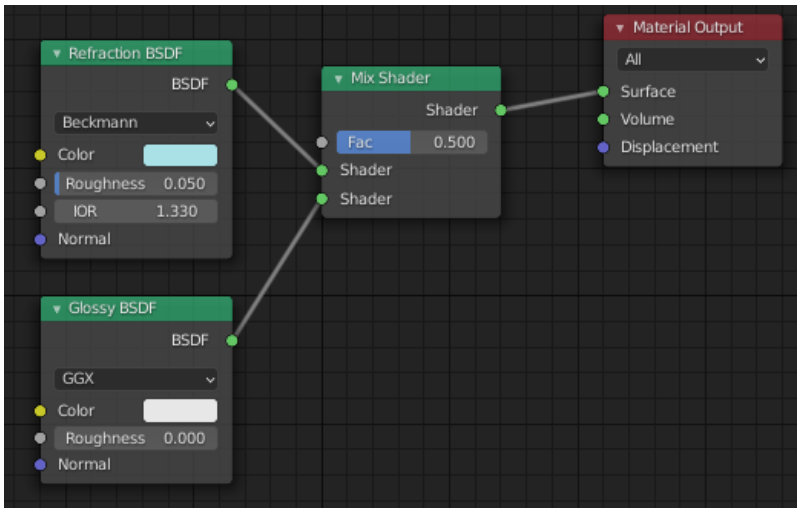
I went across all of the faces that can be seen in the camera and applied the materials to them individually. For the interior, I used the concrete texture with a black colour node mixed in to make it darker.

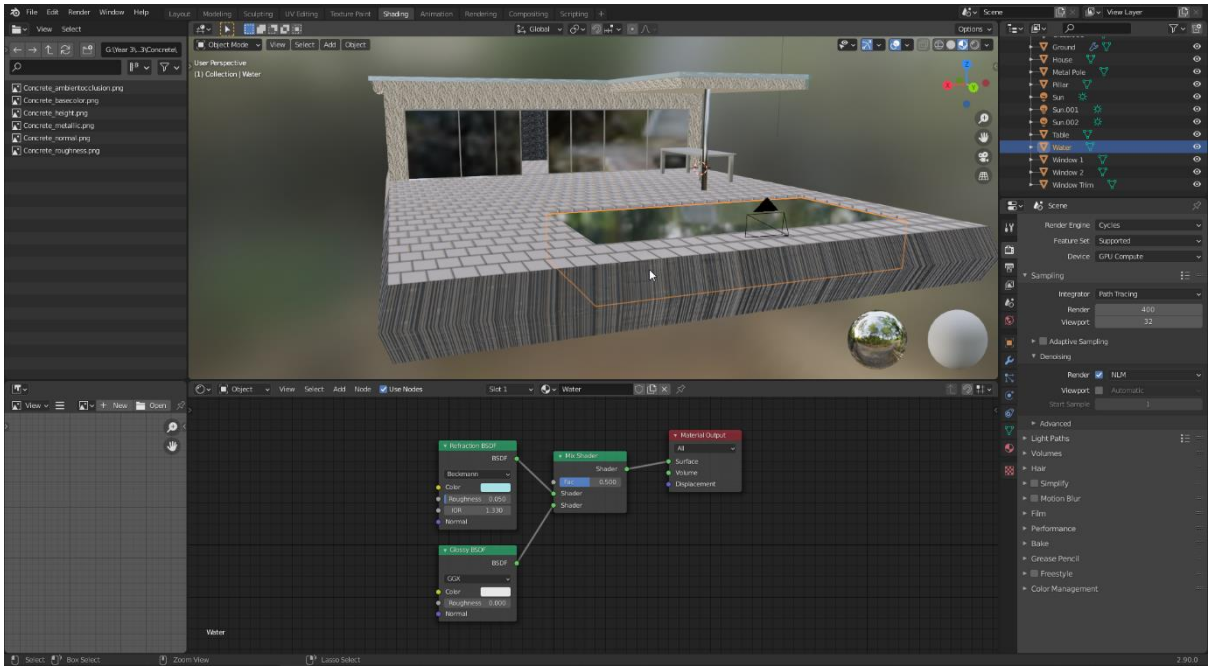


The glass and water materials are simple materials made in Blender and are not PBR materials. They are applied over a cube for the water and a plane for the glass. The IOR value is the index of refraction, which affects how much the light bends through the material. The transparent shader mixed with the glass shader makes the glass less reflective.

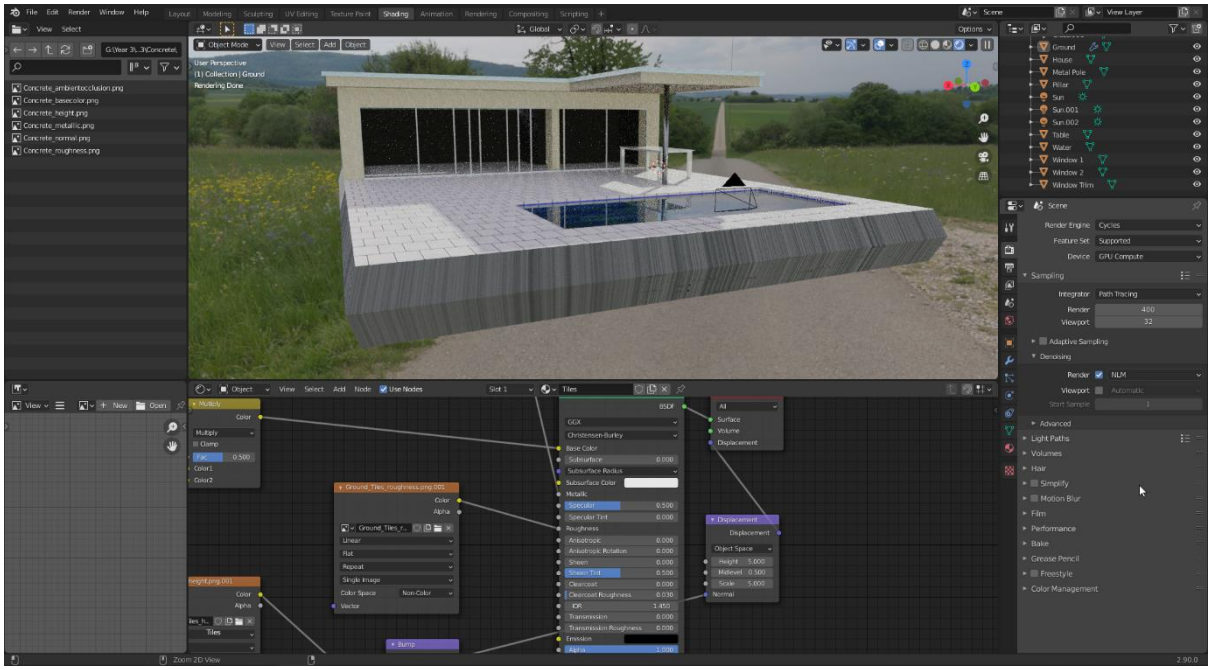


The water material is very similar, using a refraction shader and a glossy shader with a slight blue tint.





This does not look very impressive in the Eevee render engine, however in Cycles, which is Blender's physically based renderer, the light rays are affected by the glass, water, and all the maps of the materials. For instance, the ambient occlusion map affects how much light different areas of the material receive, such as lower parts receiving less indirect light from other objects.



Reflection

I am pretty happy with the final outcome of this task, as the image looks quite realistic, especially on the concrete material. However, there are several things I could improve on.

Firstly, there is a very tiny amount of noise towards the back of the house. This is barely noticeable due to the denoising feature of Blender, however with a higher number of samples, this noise could have possibly been completely removed.

Secondly, I could not get the normal and height map to have a very noticeable effect on the materials. There was a certainly a difference, however from the Substance Editor preview I expected a much larger imprint in the tiles. This could possibly be solved using displacement on the geometry.

Finally, I could have attempted to make the water and glass inside of Substance Editor, such as to give the glass imperfections like dirt. However, I feel like this might not be very noticeable, especially with the positioning of the camera.

References

Figure 1. NVIDIA Marbles at Night 1
https://www.youtube.com/watch?v=NgcYLIvlp_k

Figure 2. BMW E90 from Wikipedia.org 11
https://en.wikipedia.org/wiki/BMW_3_Series_(E90)#/media/File:2006-2010_BMW_335i_(E92)_coupe_(2011-07-17)_01.jpg

Figure 3. Blueprints 11
https://www.the-blueprints.com/blueprints/cars/bmwcars/27529/viewsingle/bmw_3-series_e90/