Final project report:

Advanced Topics in Computer Graphics

Rover Vos

Student Number: 122358345

Final project report

As a final project, I wanted to generate trees in Babylon.js. I encountered many different problems along the way and went through multiple different design iterations and multiple goals. In the end, nothing seemed to work for me and after 5 days of coding, I decided that it was time to stop being a programmer and start being a scientist. Thus I went back over everything I did and took a more explicit look at what went wrong and what went well.

I tried creating 3 different VR experiences. A forest which you can take a look at. A box with a maze inside which you can grab and move a ball inside of. And a Perlin noise field where you fly over. Below I talk about what I did and what went wrong.

Tree's

To generate trees I first tried to change the centre of a cylinder in such a way that it's at the bottom instead of at the centre of the cylinder. My goal with this being that I could create a vector tree and span a cylinder between the branch points by picking the lower vector and then rotating the cylinder to be in line with the other vector. However, I found that with my math this only worked in small angles. Angles that are smaller then about 10%. Anything larger and the cylinder moved far away from the intended point. I'm not sure why the math wasn't correct as I fully worked it out on paper and everything functioned the way it should when I put numbers in.

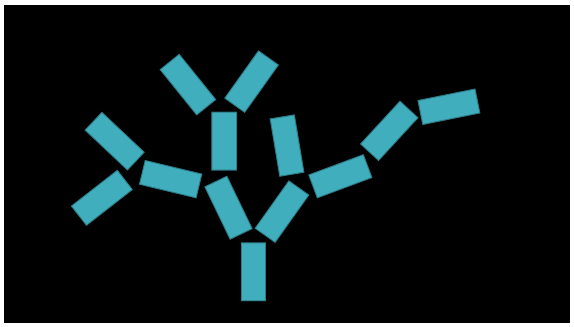

Figure 1 2D representation of the Tree generation idea

After playing around with cylinders I tried to work with lines instead(*Figure 2*). As I found that lines are a lot nicer to work with if you are using a vector system. Lines want any number of vectors and create lines between them. I could create trees with this, however, I quickly found out that lines have the problem of not being physics objects. They do not interact with light and thus when you have many lines in a scene there is no depth/shadow. I also briefly tried playing around with creating something of a cocoon around the camera, this had the same problem as the light didn't have any effect on the lines and it almost just looked like being inside a sphere.

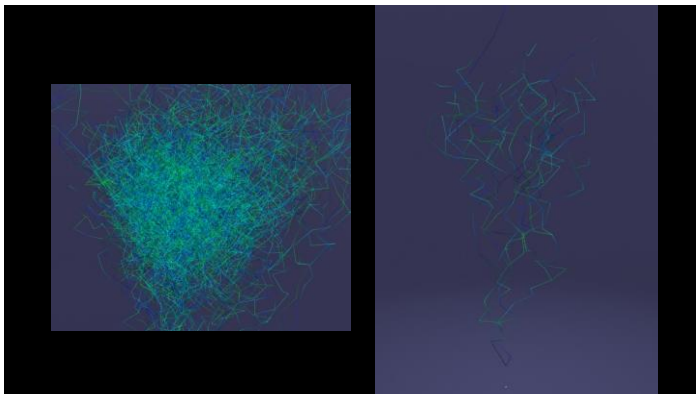

Figure 2 Result of playing around with lines

Maze Box

The maze box idea sounded very straightforward. Create a box, put a maze in by copying a maze I made before and put a ball in. Now add some physics, add controls and done! The first couple of steps went without issue. However, I decided to skip the maze part at the start(*Figure 3*). The problems arose when I started looking into how the controls should work. First I just wanted to be able of moving to the box around and thus I implemented a gizmo with which you can move the box with 3 degrees of freedom. This already introduced a new problem, physics. Most of the time it worked fine, but sometimes the ball randomly fell through the box. Also if you move the box fast enough the ball got shot out of the box. Which was easily fixed by placing a reversed plane on top of the box. Implementing VR controls was where the real problem occurred. As I did figure out how to do it and did implement it, I however had no way to test it. I did not have access to a VR headset with controllers at the time. As this is a very limiting aspect of the work I decide to continue and make a stationary VR experience instead.

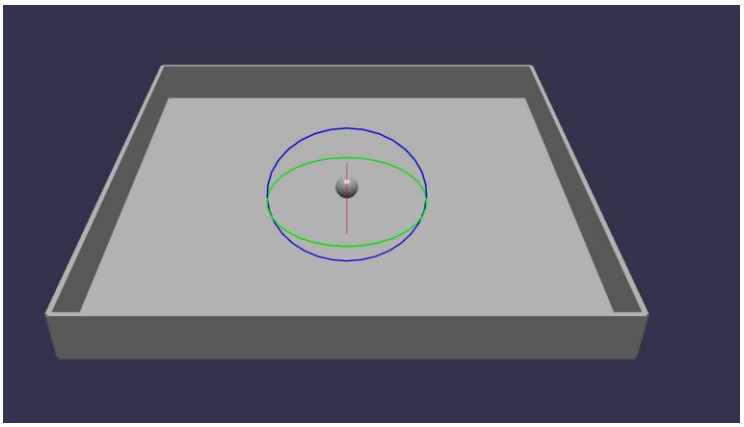

Figure 3 Ball in a box

Perlin noise

Perlin noise field(*Figure 4*) has always been an interesting thing to me. The infinite amount of variations you can create with only a couple of simple lines of code. Then you can also modify it in many different ways to create different types of effects. For this field, I kept it very simple and implement a very basic Perlin noise which is not very flowy, but created a more crystalline structure. I created the field by mapping each point on a grid and giving it a height(y position) based on their x and z positions and the Perlin noise. Then fill in the between the point with custom meshes.

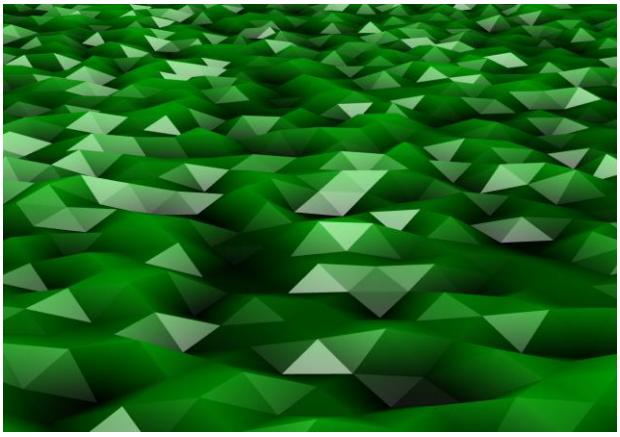

Figure 4 Perlin noise field

Using custom meshes is where the largest problem arose. For each point on the grid, I created one mesh minus width and height. This already lowered the frame rate quite a bit while not being animated. But non the less I animated it over time and who would have thought the frame rate went below 10 fps. I optimized a couple of things and reduced the size of it and managed to reach about 90fps(*Figure 5*), but the field was way too small to create any meaningful full experience. Being at a loss for what to do I scoured the internet on how other people did this in Babylon.js and found out that other people used two different things. Firstly they used someone's Perlin noise from a GitHub library and secondly they used ribbons instead of custom meshes. I also found a couple of fully functioning examples however not animated.

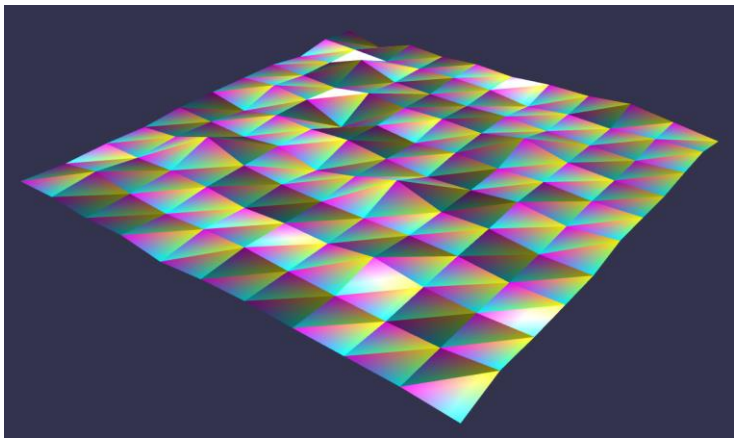

Figure 5 Animated Perlin noise field(Not here)

Takeaways

So what did I learn during this process of coding in VR? Babylon.js is weird, but also really cool. It allows for quick prototyping and easy VR implementations. However, as it uses webGL it's not very powerful and I don't think I would use it again. It's also not a very broadly used GUI as it was more difficult than it should have been to find an answer to the thing I needed. It also doesn't have a lot built into the engine it's very barebones(at least for what I wanted to do). It is however javascript based and thus a lot of code snippets are available on the internet.

Think a bit more about what I want to do before doing it. I could have saved a lot of time if I did a bit of research before starting to code. For example, reading about lines and that they don't interact with light before experiencing it myself would have been a good thing to do. Also knowing that ribbons are a less computationally hungry mesh is something I could have found out by reading a bit more into it.

Working in 2D is so much easier than in 3D. Adding a third dimension increases the complexity of a project significantly and for future projects, I think it would be smart while working in 3D to see if you can create it in 2D first. I don't mean to make it in a 2D engine. I want to see if I can make a 2D representation in the 3D engine and then with a couple of extra lines of code convert it into 3D. Like making a maze. Make a 2D maze and just places walls between the two points now it's a 3D maze from a 2D maze.

Don't attempt to work on a project if you don't have the capability of testing it. Not having easy access to a VR headset made this project a whole lot harder than it had to be. I did not have a lot of opportunities to test anything and just had to hope it was going to work in the end.