DH2323 dgi22 VT22-1 Computer Graphics and Interaction
Andreas Wrife, David Åstrand, Léo Noharet
awrife@kth.se, dastrand@kth.se, lnoharet@kth.se

DH2323 LAB 2

1. Setup of the Lab Environment

To begin with, each group member installed Unity on their own computer, and set up the three different scenes; Tank, Fireflame and Shell. Each group member got used to the environment of Unity throughout the lab by following the setup steps preceding the tasks for the respective scenes. Compared to the previous lab using SDL, no problems arouse during the installation and setup phase. Only issue was downloading the correct version of Unity required for the lab.

2.3 Tank Tasks

After inspecting the Tank environment, including the tank hierarchy, starting the scene, playing around with the movement of the tank, the camera controls and the associated script using Visual Studio Code we began with the tasks.

2.3.1 Rotate Wheels when Tank Moves

When trying to get the tank wheels to move, one of the first things that was done was moving the camera to a position where the wheels could be easily observed. An idea that cropped up was to configure the Tank Hierarchy to where the camera control was a child node to the tank, see Figure 1. This meant that the camera would follow the tank instead of being idle floating somewhere in the scene. After that, the camera was adjusted to look at the wheels in a profile perspective, see Figure 2. Important note: to observe rotation of the wheels, the left and right most wheels had shapes that gave a good indication if they were rotating.

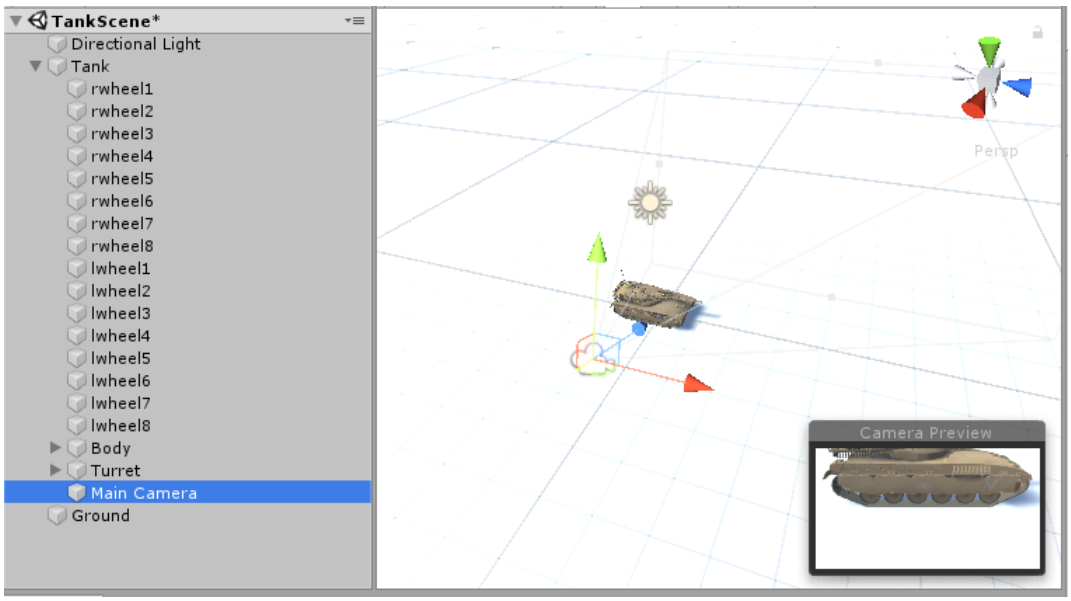

Figure 1: Showcase of the Tank hierarchy, where the Main Camera is located inside the Tank object.

Figure 2: Camera view of the wheels in a profile perspective of the tank.

A quite funny part of this task was trying to figure out which axis the wheel was rotating on. After the implementation of the RotateWheels() function was implemented using transform.localRotation and the Quaternion system, described in the lab assignment, picking the wrong axis gave some hilarious results. The wheels would then of course rotate out of its socket around the wrong axis.

2.3.2 Rotate Turret Following Mouse

The turret rotation task was probably the most difficult task given in this lab assignment. There were simple solutions, such as one liners of code using defined functions such as LookAt(). However, wanting to do a more mathematical implementation meant understanding how the positions of the mouse and turret could affect the current rotation of the turret. With the help of sketches on paint and trigonometry the solution was found. For a showcase of the turret following the mouse, see Figure 3 and Figure 4

Figure 3: Showcase of the Tank Turret following the mouse, looking left.

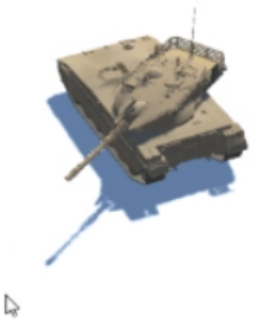

Figure 4: Showcase of the Tank Turret following the mouse, looking down.

3.3 Shooter Tasks

Before the Shell scene was attempted the group played around with the FireFlame scene. Following the step by step guide from the lab assignment as well as just playing around with it a bit, the group members got used to the particle system and some of its features, See Figure 5.

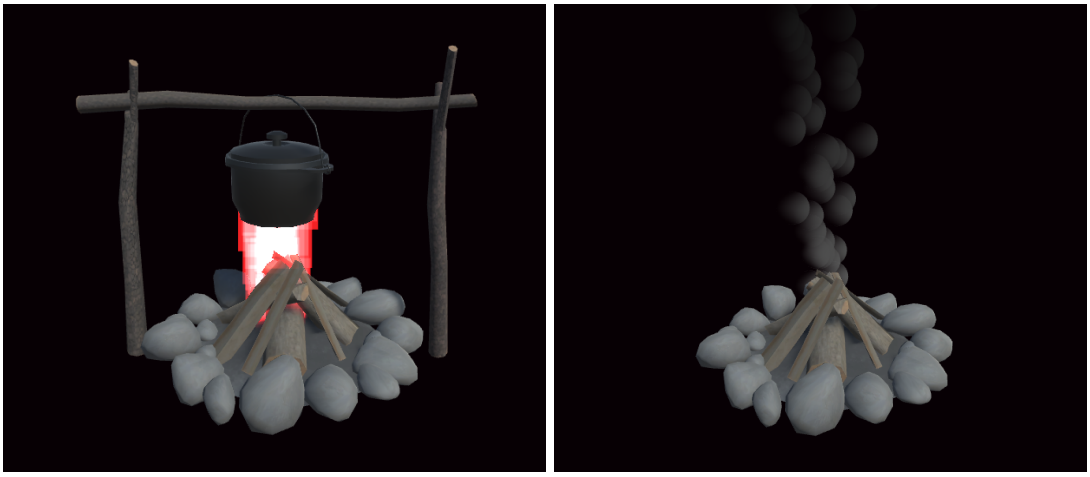

Figure 5: Images showing FireFlame scene, with kazanok and fire on the left and with ash particles on the right.

3.3.1 Fire the Target!

After creating a particle system for the explosion and making it as well as the shell into prefabs tinkering with the explosion effect could commence. To begin with the shape of the explosion was made into a sphere since the object would explode into all directions. Second the material was set to the FireMaterial used in the previous scene since it seemed to be fitting. Furthermore, the colour was changed to be of a gradient over the particles lifetime. Starting with a bright yellow and turning red seemed to look pretty well. The last thing was to play around with the $Lifetime$, $Duration$ and $Startspeed$ parameters since they either made the explosion too short or made it so that the particles drifted off into space for an uncomfortable amount of time. The group settled on a $Duration = 0.1$, $Lifetime = 2.5$ and $StartSpeed = 2$. See Figure 6 for snapshots of the explosion.

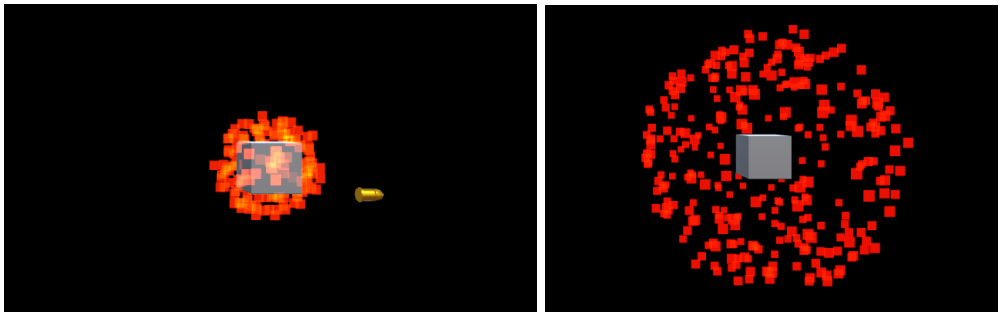

Figure 6: Image showing (orange) explosion particles following the Shell hitting the Target and (red) explosion particles following a second later.

3.3.2 **Create Your Scene**

Figure 7: Image showing off the costume scene with the 3D imported Mjölnir hammer before and after texturing.

For the custom-made scene, a Thor's hammer was imported from the Free3D website [1]. Using this object a scene was created featuring Mjölnir with lightning coming out from it in a circular manner. See Figure 8. This was accomplished by adding a noise map at high frequency but low strength as well as adding the trail property to the default particle material. This particle system was copied and made smaller but with higher emission to create even more lightning particles closer to the hammer. This can be seen more clearly in the right image in Figure 8. For the longer particles the *SimulatedSpeed* was turned up to 3 to sell the lightning even further. Since the smaller lightning bolts had a greater emission it looked better to keep the normal speed ($SimulatedSpeed = 1$). Otherwise it seemed almost too fast and cluttered. To see the lightning bolts more clearly, the colour value of the Directional Light of the scene was tuned down, creating more contrast between lightning and background.

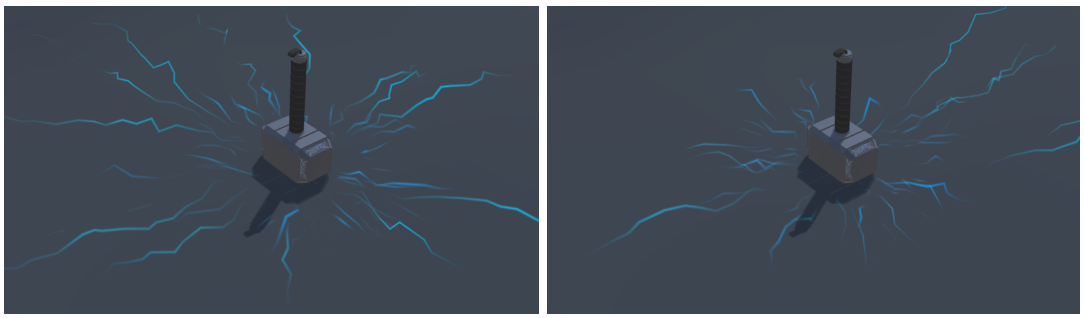

Figure 8: Image showing off the costume scene with the 3D imported Mjölnir hammer together with a particle system of lightning.

Workload Distribution

Compared to the previous lab, this one was done individually on each group members' personal computer since it required Unity and we believed that everyone should get hands on experience operating Unity on their own system. Two of the three group members had some previous knowledge of Unity from playing around with the features, however no one had done anything major with the engine. One idea for the upcoming project was to simulate a more complex particle system where Unity could perhaps come in handy. Throughout the lab, we discussed the tasks and then tried different approaches individually. When everyone felt ready we compared and discussed our implementations, sharing screen and pointing out pros and cons with each application.

References

[1] dualityy, "Thor hammer (mjolnir) 3d-modell," Apr 2017. [Online]. Available: https://free3d.com/3d-model/thor-hammer--62945.html