Hochschule für angewandte Wissenschaften Europa
Campus Hamburg
Game Design Semester 4

Performance Requirement
Game Feature

Game Programming 3 SoSe 2021

Lecturer: Slawa Deisling

Contents

Introduction

Over the course of this semester we had a deep dive into different Unity Packages and got an introduction into working with these. The task for this class was to create a game feature for a potential game, using two of the many packages we talked about. In the following we will explain our potential game a bit more, take a closer look at the features we implemented and at the end we will describe who worked on which part of the project.

The potential game

The potential game we developed is a space exploration game called *Pocketspace*. The concept of the game is inspired by the typical Walking Simulator Genre. The player should simply relax and enjoy the vastness and calmness of space, with pretty visuals to discover along the way.

We thought that this simulator-like game would work well with the packages we chose and could lead to a nice little atmosphere in the game scene, on which we put our main focus. Other features might include discovering alien life forms (Plants, animals, etc.) or visiting the different planets and exploring the landscape.

The main target platform would be the PC. Therefore we decided on using the mouse and the keyboard for controlling the player and moving through the atmosphere. We could imagine implementing controls via the gamepad because of the new Input System, but because of the scope we just settled for the keyboard and mouse input.

Implementation

We started by setting up the repository and unity project for the task before diving into our assigned topics.

After a brief discussion, we decided that using the following Unity packages would be the most beneficial for our game:

- Shader Graph
- Pro Builder
- New Input System

These packages will be implemented into the space exploration game we already talked about, where the pro builder objects function as blockouts and provide a feel for the game and the shaders give the planets and stars a celestial feeling.

Pro Builder

Tornquist started by creating a space station from the pro builder objects which could be placed in the scene. This worked rather well as pro builder has an easy and accessible interface that allows you to create basic shapes very quickly.
A variety of planets and asteroids were created as well to randomly fill the scene.
The tools were simple to use, especially with the Autodesk Maya knowledge we have from our first semester. This allowed us to understand and move the faces, vertices and edges of the objects in a similar manner to how it would be done in Maya.
Unfortunately, when pushing these objects to the repo, we encountered a merge conflict that led to all the progress being deleted. Therefore, Tornquist had to create all the objects a second time. However with the learned routine, this thankfully did not take as long as the first time

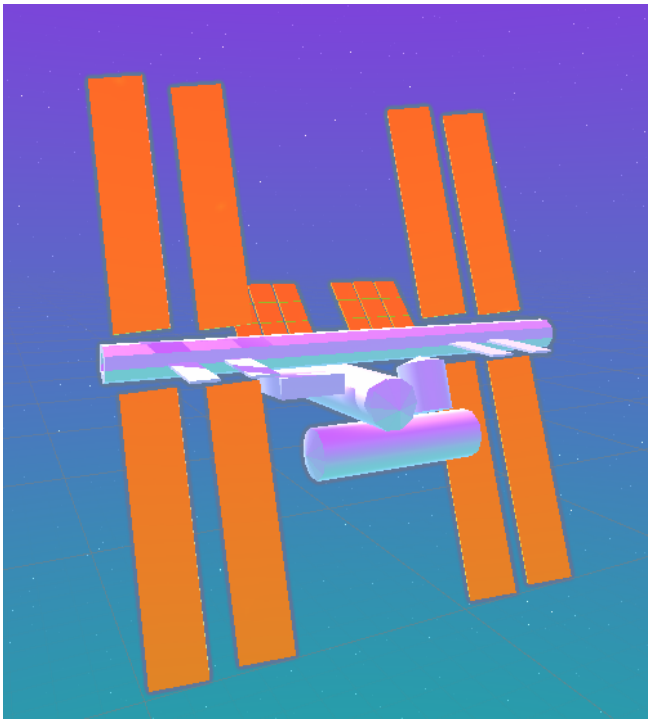


Figure 1: New space station with materials

Everything then got added to the scene, planets scattered throughout the small constraints of the map and 3 asteroid fields that use 3 different asteroids which were turned into different positions to create the illusion of variety.
As Alessa finished her first shaders, the scene finally took shape.

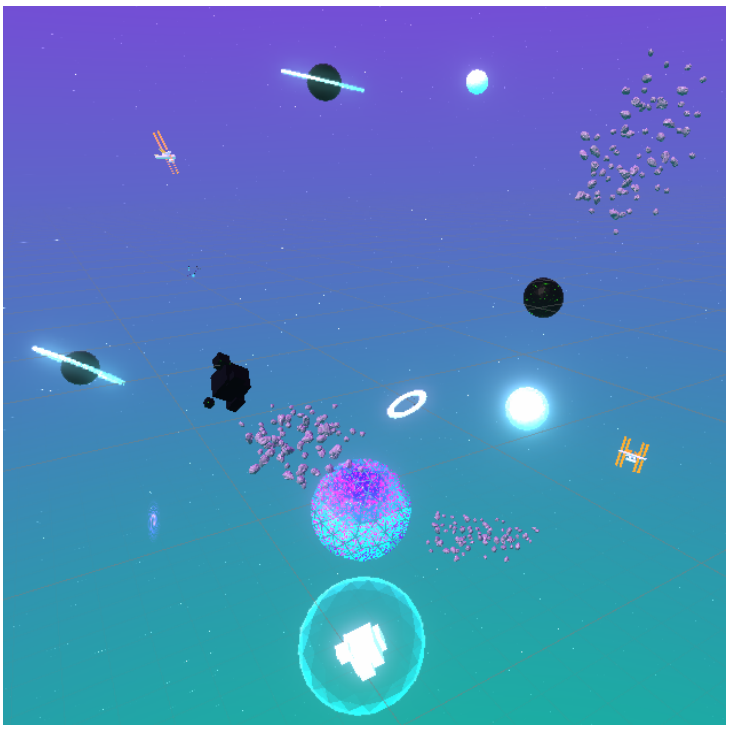


Figure 2: Final Scene with all pro builder elements and shaders

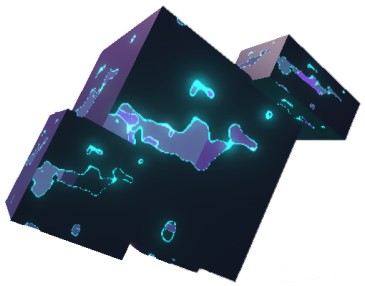
Overall, we have gathered a lot of experience with this package and will definitely use it again in the future to create simple and quick block outs for our scenes.

Shader Graph

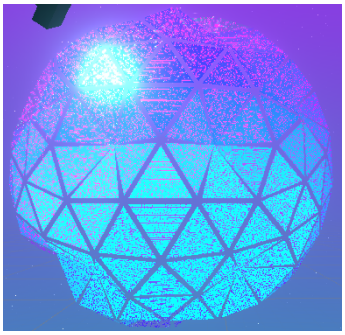
Shader for planets

Since the lessons for the Shader Graph took place some time ago I decided to watch a tutorial first and follow along with it in order to refresh my learnings from the class. After that I tried to find some other tutorials on top of that, in order to create specific shaders that might fit our galaxy setting. As a result I created the first shader for our game which was the *Glow* Shader. This shader lights up from time to time and changes the color back to normal. But when I tried to install the URP Unity crashed and did not boot up anymore. Luckily I already had the same problem when working on another project of mine, so I was able to fix it quickly by renaming and moving the folder of the repo. After following the tutorial along I noticed that the shader is working in theory but doesn't really glow. After a bit of research I knew that I had to install Unitys Post Processing Package and add a Global Volume with a Bloom profile. Doing this

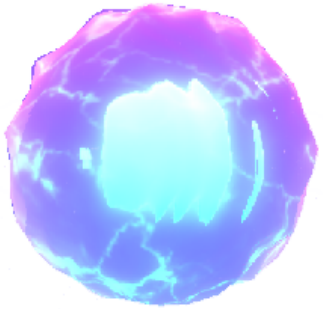
solved the problem and also added the glowing effects to the other materials made of the Shader Graph.



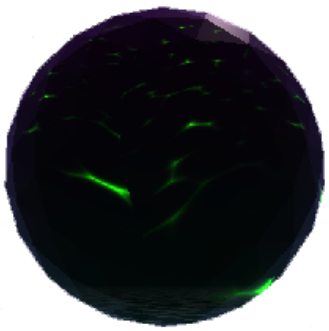
The *Dissolve* Shader was also created for some of the different planets. First it lets the planets dissolve with some glowing outlines and after that it lets them appear again. I didn't really have any major problems while working on this and it turned out the way I attended it to be.



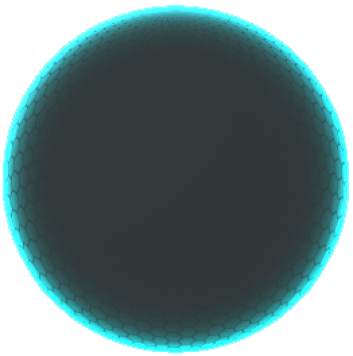
The R*ift Sphere* Shader was originally meant to have some kind of bubbles / waves going in and out of the planet (see in Shader Scene). After transferring the shader from the shader test scene to the Game Scenes it kind of changed its behaviour. Probably because of the size and the different polygon count. In the end we decided to let it be like that, because we also liked this effect, even though it looks a bit different than originally intended.



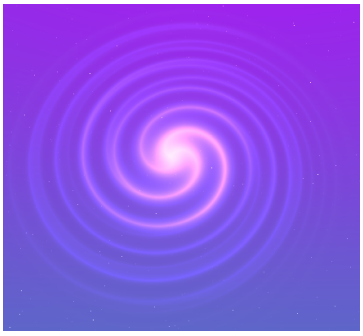
This *Plasma Orb* consists of two different shaders on two different Spheres. The one in the middle is simply made of a light glowing material and the outer one is made of a transparent shader with an electricity map as texture.



The *Water/Lava* Shader was originally meant to be for the ground in the second level. But it turned out to be a bit too much with all of the other materials and assets, which were already implemented. So in the end we just reused it as a material for more planets. With this shader I had kind of a similar problem as with the Rift Sphere Shader, because it turned out to look different in another level but after adjusting the variables of it a bit it got better.



As well as the Water/Lava Shader this *Force Field* Shader wasn't meant to be for the planets, but rather for the limitation of the levels. In the end we did not use it for the level border because we preferred having it transparent in the middle, which made it barely visible for the player. For this Shader I had quite some trouble finding a matching texture, because I did not know how strong it should be and if the Sprite itself needs to be transparent or not, etc.



Portal Shader:

The *Portal Shader* was inspired by the shader you showed us in class and worked out pretty well for our needs and without any further problems. We used it to port the player to another level when colliding and placed it in both of our scenes.

Skybox Shader:

The *Skybox Shader* was an idea Tornquist had and asked me to do. Also this Skybox Shader basically added the whole color scheme to our game. The only problem with this shader is that the skydome is unwrapped in a way that creates an aberration in the form of a stretched-out line rotating around the z-axis, which is why some of the stars are distorted. I tried to make it less noticeable by increasing the tiling. On top of that I was planning on adding some sparkling to the stars, but I didn't do so because on the one hand I thought that it might be too much with all of the moving planets and on the other hand because it didn't work out the way I wanted it to.

While working with the Shader Graph I slowly got a hang of it I would say and I started to understand which nodes have which effect and what I have to do in order to change specific studd. Although it looked very confusing at first sight when I started working with it. To me it was a lot of fun working with it, especially because you can see such a great effect immediately when adding shaders or editing them and to be honest I kind of regret that I did not work with them before, because it feels like they just add so much to the game feel.

Input System

Setting up the new input system was simple as we simply installed it from the package manager and replaced the old event system with the new one. This was all that was necessary to have a working main menu navigation with the new input.
For the movement, we ran into a few issues as we did not quite know how to add a Y rotation to our player to allow up and down movement in a 3 dimensional space. In the end we chose to only have free movement in the second scene and a pre-set animated track for the first level.

For our movement, we created a new input for the player via the Input Actions from the new input system package. This way, we can get our buttons and axes instead of setting them via script as we used to do with the old input system.
Here, we set an Action for Horizontal movement, which contains the WASD buttons to move the player forwards, backwards, left and right.
We also created an action for Jump, which is the space bar, MouseX and Y, which get the input from the mouse on both the X and Y axes.
To handle the input and movement, we created 3 scripts; Input Manager, Movement and Mouse Look.
The Input manager translates the movement from the Movement script to the new input system so that we can use the buttons and axes we set in the Input Actions.
The movement and Mouse Look scripts handle the actual code for moving the player and mouse to look around.
Finally, we attached all scripts to the player and had a working character controller, with which the player is able to walk along the planet.

Distribution of work

We decided to split the task so that Alessa would create the shaders and Tornquist the pro builder objects and implement them as well as the shaders in the scenes. Both of us would work on the input system and code for movement. During our working process we were constantly in a Discord call to ensure that both of us would be able to talk to the other one when problems or questions occur.

Bibliography

- Brackeys. (2018, 27. May). *Basics of Shader Graph - Unity Tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=Ar9eIn4z6XE
- Brackeys. (2019, 28. April). *FORCE FIELD in Unity - SHADER GRAPH* [Video]. YouTube. https://www.youtube.com/watch?v=NiOGWZXBg4Y
- Brackeys. (2018, 10. June). *DISSOLVE using Unity Shader Graph* [Video]. YouTube. https://www.youtube.com/watch?v=taMp1g1pBeE
- Brackeys (2019, 19. May). *SIMPLE CARTOON WATER in Unity* [Video]. YouTube. https://www.youtube.com/watch?v=Vg0L9aCRWPE
- Gabriel Aguiar Prod. (2018, 04. December). *Unity Shader Graph - Portal Shader Tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=w0znZIuvQ2I
- Licke. (2020, 13. June). *Unity Rift Sphere Shader Tutorial: How to in Shader Graph* [Video]. YouTube. https://www.youtube.com/watch?v=kr_tdKCT2Xs
- Binary Lunar. (2020, 30. August). *Magical Plasma Orb Part 1 Shader Graph Unity Tutorial.* [Video]. YouTube. https://www.youtube.com/watch?v=DpXPhGeCqus
- The Game Guy (2021, 01. January). *How to make a Gradient Skybox with Stars in Unity | Shader Graph Unity* [Video]. YouTube. https://www.youtube.com/watch?v=yw2J9NWRdow&t=11s
- Iconian Fonts. *Raider Crusader.* 1001 Free Fonts. https://www.1001freefonts.com/raider-crusader.font
- HummingNature.(2021, 19. März). *Space Ambient - No Copyright [FREE]* [Video]. YouTube. https://www.youtube.com/watch?v=a39Mu8hp0rI
- *Vektorgrafiken Hexagonal grid*. depositphotos. https://de.depositphotos.com/vector-images/hexagonal-grid.html
- *Electric Power Lightning Texture Seamless*. Textures4Photoshop. http://www.textures4photoshop.com/tex/bokeh-and-light/electric-power-lightning-texture-seamless.aspx
- Packages Used: Input System, Shader Graph, ProBuilder, Polybursh, Post Processing, TextMeshPro, Universal RP