Multiplayer Game

301CR – Assignment 1

Contents

[Introduction 3](#_Toc532521703)

[Conceptualisation 3](#_Toc532521704)

[Mood board 3](#_Toc532521705)

[Sketches 4](#_Toc532521706)

[Creating the animation 4](#_Toc532521707)

[Real world footage 4](#_Toc532521708)

[Camera Tracking 5](#_Toc532521709)

[Creating the container 6](#_Toc532521710)

[Water components 7](#_Toc532521711)

[Inflow 8](#_Toc532521712)

[Container 8](#_Toc532521713)

[Liquid (Domain) 9](#_Toc532521714)

[Materials 10](#_Toc532521715)

[Liquid 10](#_Toc532521716)

[Pipe 11](#_Toc532521717)

[Wood material 11](#_Toc532521718)

[Copper material 11](#_Toc532521719)

[Animating 12](#_Toc532521720)

[Rendering 12](#_Toc532521721)

[Renderer 12](#_Toc532521722)

[Lighting 12](#_Toc532521723)

[Render settings 13](#_Toc532521724)

[Post production 13](#_Toc532521725)

[Reflection 13](#_Toc532521726)

[Bibliography 14](#_Toc532521727)

[Texture bibliography 14](#_Toc532521728)

[Reference Images Bibliography 14](#_Toc532521729)

# Introduction

This assignment is separated in two tasks. One is creating playable game and implementing different subsystems to it and the other one is adding multiplayer functionality to it.

## Playable Game

For the playable game I must implement the following subsystems: Graphics, Physics, Resource Management, Human Interface, Initialise/Shutdown, Networking and Profiling.

## Networking Multiplayer Functionality

The playable game should be extended to support multiplayer functionality, using a middleware library (SFML)

# Middleware Used

For this project I’ve used SFML as middleware for all my subsystems.

# Game Systems

For this project I managed to implement the following systems: Windowing, Graphics, Networking, Audio, User Input and File Loading. Below are all the systems explained.

## Windowing

The task of this subsystem is to create the window in which other subsystem will execute (Figure 1.1). This subsystem uses observer pattern, which notifies it when to close the window (Figure 1.2). The window makes use of the event system and waits for a close command. The window also uses the singleton pattern as the clients will never need more than one window running.

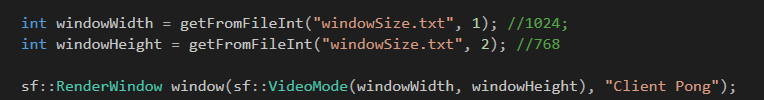


Figure .1 Window creation

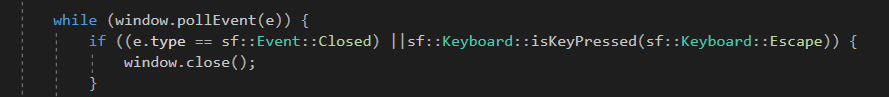


Figure 1.2 Observer Pattern

The windowing subsystem also checks if the user has selected the current window and only then proceeds to take user input. This is an extra protection to guarantee that each client can control only its own bat. The reason I take this step is SFML doesn’t separate user input from the active window.

## Networking

The networking subsystem is tasked with sending and receiving data between the server and all of the clients. For the making of this subsystem I used packets and SFML/Networking. The architecture of this system is the following:

1. The server starts and wait for connections.
2. Client tries to connect to the server.
3. Server checks if the client is still connected. If it is, it proceeds to the next step. Otherwise goes back to one
4. The server receives the movement state from the client.
5. The server does all of the calculations for every movement in the scene.
6. The server sends the updated locations of the players and the ball to the clients.
7. Clients update their own positions and draw.

Both the server and the clients use fixed time step for sending packets and frame limit. This guarantees that the game will run with the same speed on every client and every machine. The data send between the clients and the server makes use of the SFML types, which guarantees that the types sent take the same amount of bytes on both 64 bit and 32 bit systems. Dead reckoning is not implemented which results in latency issues in the application

## Audio Subsystem

For the audio subsystem I use SFML/Audio. This subsystem is applied as header to the client class and it has a couple different functions – loadAudioFile, setAudioState and runAudio. The sounds are declared in the client, outside of the game loop. The application looks for a background music with a given name, loads it and runs it before the game loops start.

## Graphics Subsystem

The graphics subsystem uses SFML/Graphics. It’s used both in the client and in the server. Although the server never draws anything, I use graphics types (e.g. RectShape) to make the ball Calculations. This is by far the most important subsystem as without it, nothing would be displayed on screen. The Graphics subsystem also makes use of Object Oriented Programming – every client has an object for the ball and for the bat and on received packets it updates their positions.

## User Input

User input uses the observer pattern. The subsystem is waiting to detect a pressed key. When this happens, it changes the state of that client. As long as the state goes through a few conditions, it gets sent to the server in a packet.

## References

## Storyboard

# Creating the animation

## Characters

The first thing I decided to do was the two characters. Both were made with a cube, which is subdivided into smaller faces and modelled by the reference image. The heads and limps were extruded from the body.. (Figure 4).

Figure 4 Characters

The hair of butters is made by multiple small planes which are modelled accordingly to his head (Figure 5) and the hat of Cartman is created by duplicating half of his head, scaling and applying solidify. (Figure 6)

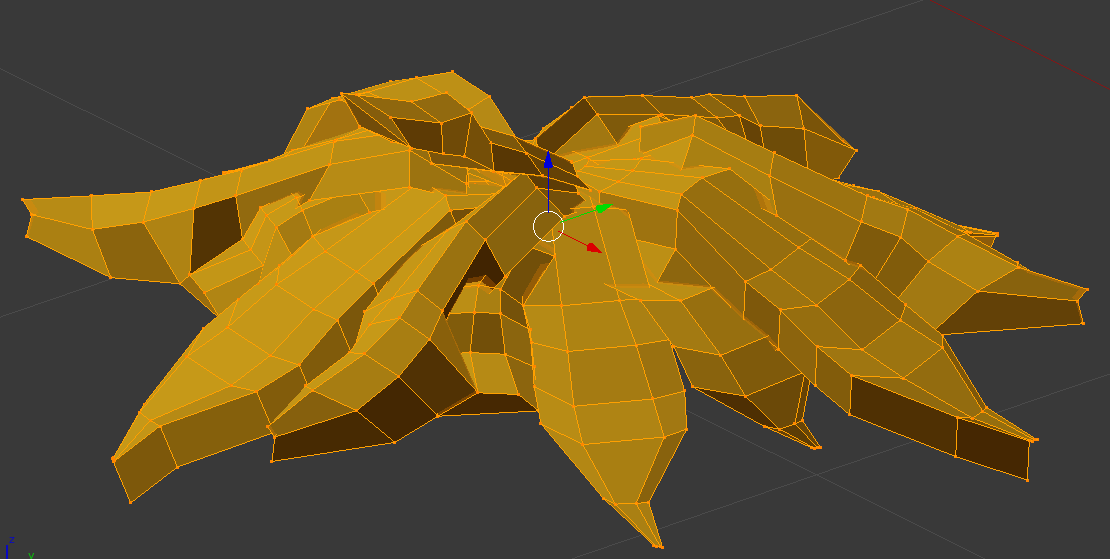


Figure 5 Butters Hair

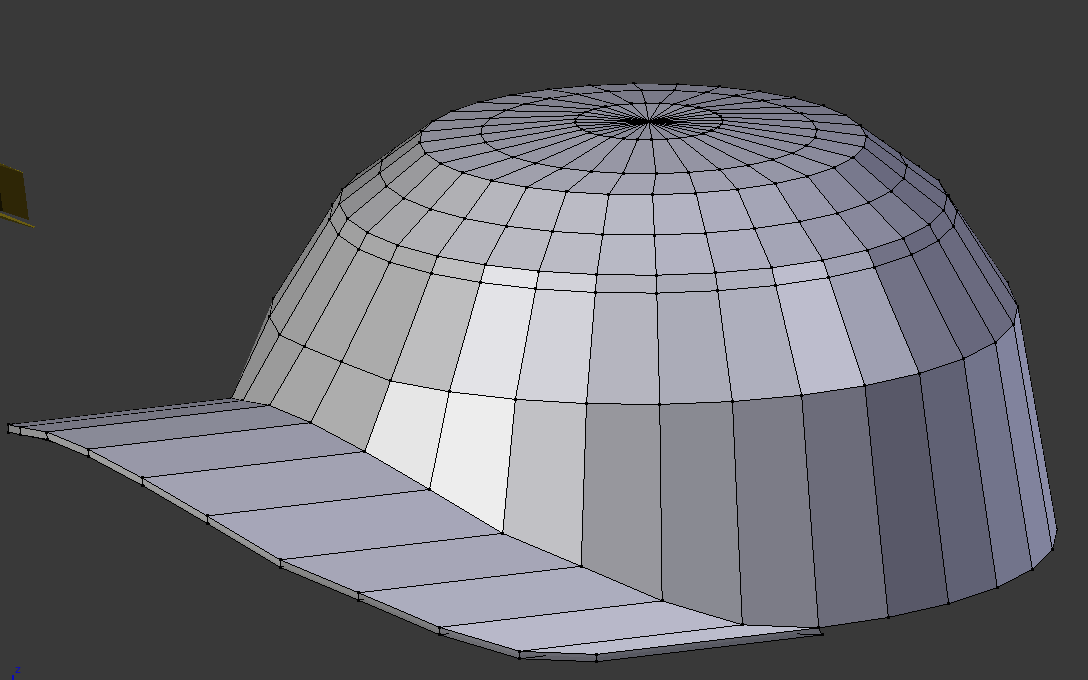


Figure 6 Cartman Hat

The eyes of the characters were created by creating two cylinders (Fig. 7), which were cut into the model by using the Boolean modifier (Fig. 8)

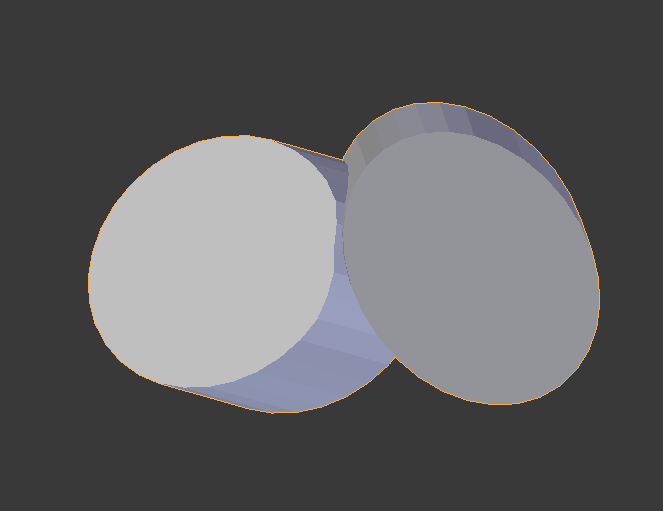


Figure 7 eyes

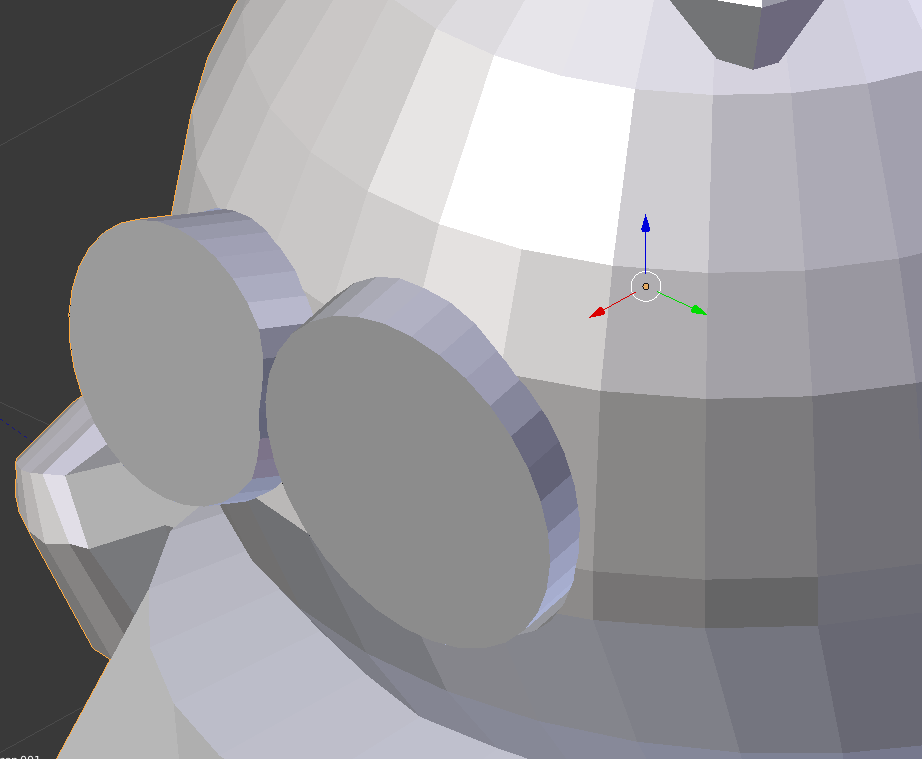


Figure 8 Eyes in the model

## Props

When I was happy with the characters I still had to create the environment. The models I created are garden plant, bush, house and snow piles

### Bushes

The bushes (Figure 9) are in fact simple cubes, which I have subdivided and then added the “Subsurf” modifier. This modifier makes them rounder, while keeping the original amount of faces. I also made a layer of duplicated faces on top of the bush, which I will use later for texturing.

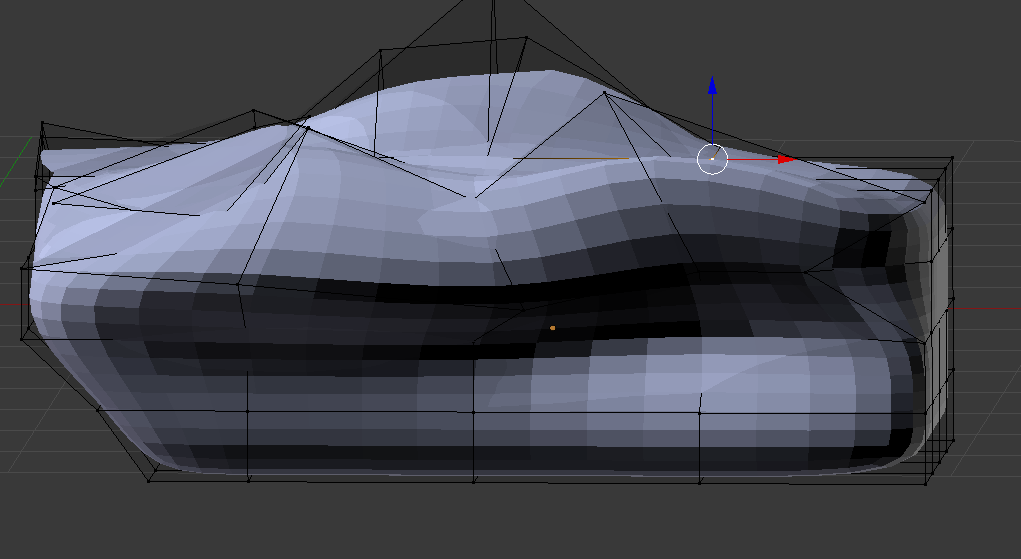


Figure 9 Bush

### Garden Plant

The garden plant (Figure 10) was made out of the box. The desired shape was achieved with scaling the array divisions around the side until it reaches the required volum

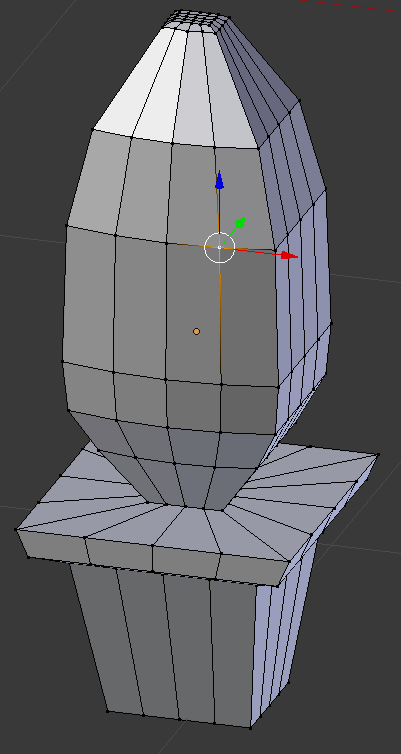


Figure 6 Garden plant

I tried to take the solve error down below 1, but for the sake of doing this I had to reshoot another video, so I decided to just go with it.

### Snow Piles

The snow piles are made of one big, thin cube (Figure 7). I deleted random faces from the middle of the cube using the brush tool (“B” hotkey). After that I filled in all the missing faces and applied the subsurf modifier (Figure 8).

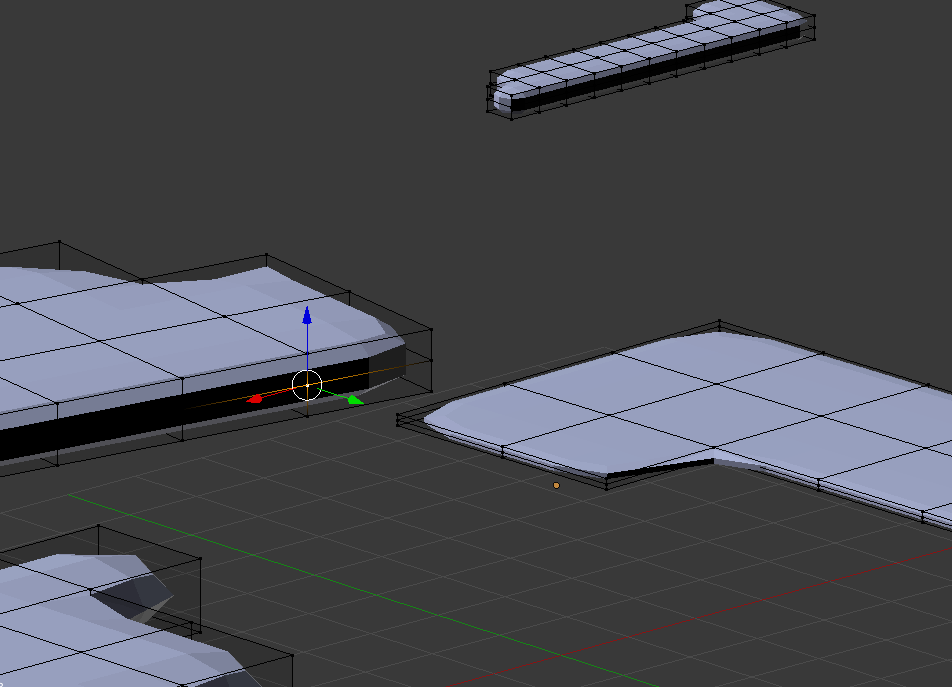
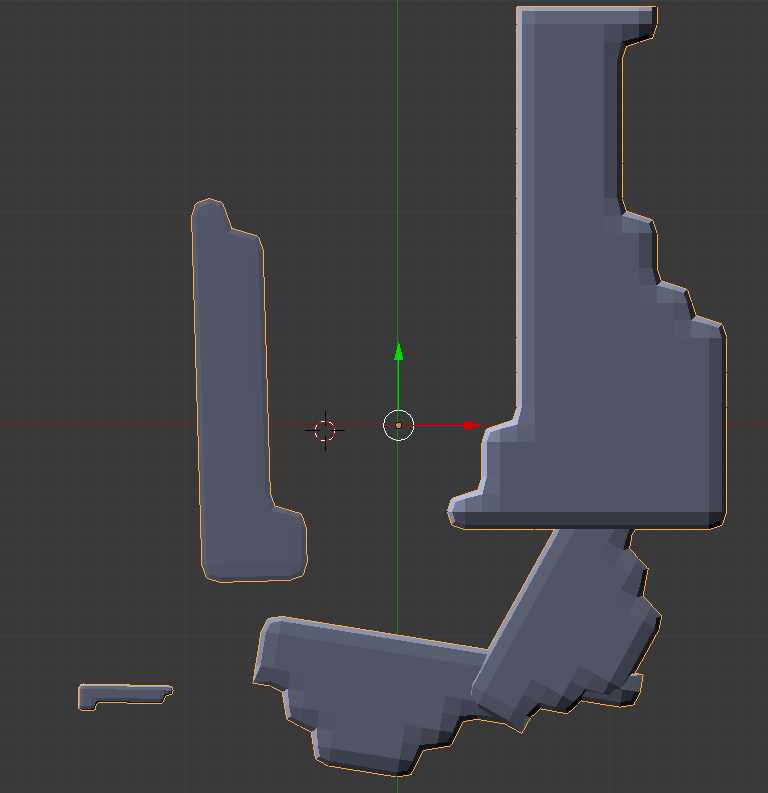


Figure 7 Snow piles, top down view Figure 8 Snow piles Subsurf

### House

The house (Figure 9) was the simplest model. Since it is visible only from the front, this is the only side I modelled. I extruded faces for one window on the left and repeatedly subdivided the faces where I had to UV map the door

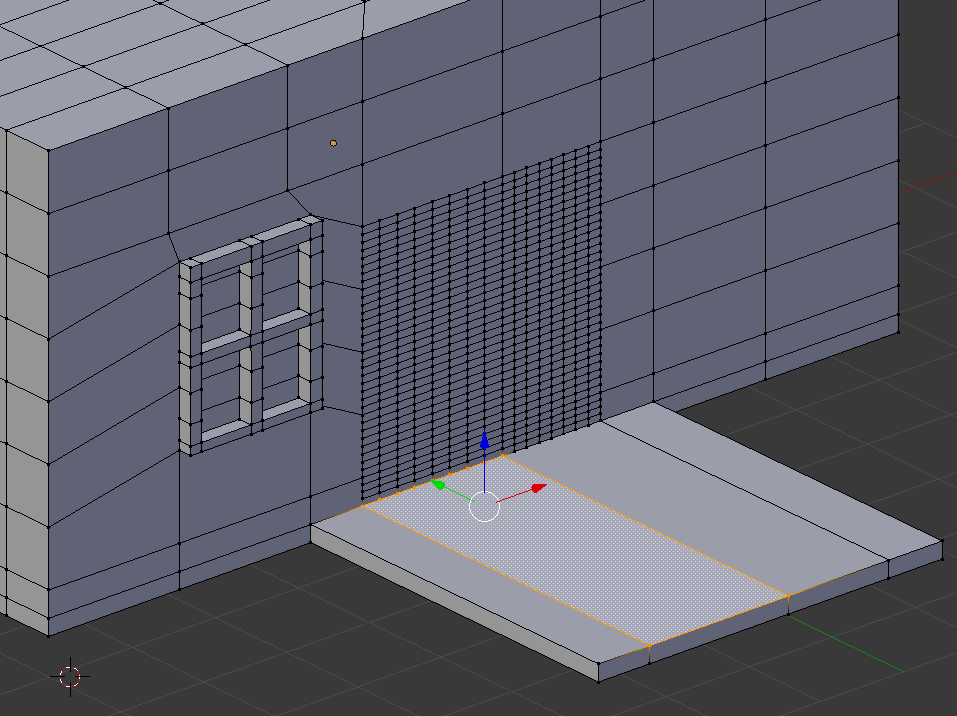


Figure 9 House

# UV Mapping

UV mapping was extremely tricky as I didn’t have any textures to work with. For this reason, I made a screenshot at the specific scene I am using (which you can see in the Reference Images chapter) and I use that screenshot as the texture for UV mapping wherever possible (Figure 10). When it wasn’t possible, I used the screenshot as a colour palette. (Figure 11)

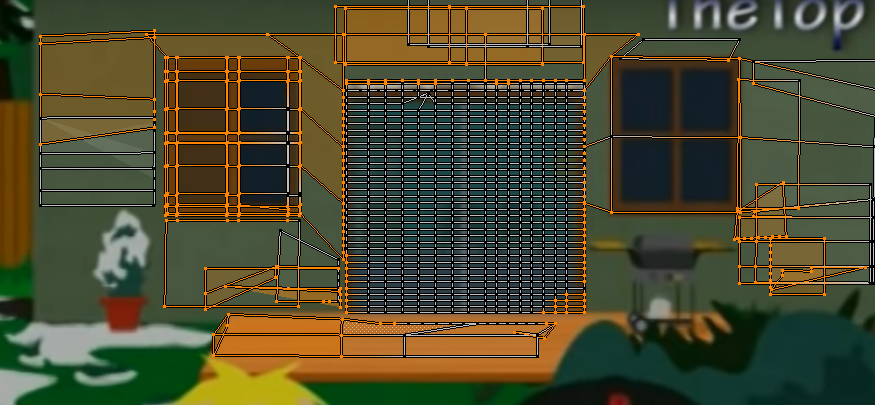


Figure 10 UV Mapped door and Window



Figure 11 Colour Palette

## Animated Textures

I also tried to do animated textures for the mouths and eyes. The way this was supposed to happen was creating a separate plane, which uses bone system to move the UV map. The UV map itself is a grid holding different expressions (Figure 12). The reason I couldn’t do that is because I had one big UV map for the whole character, and I should’ve had a separate one for the mouth and the eye. For me to make this work I had to do full retexturing, for which I didn’t have enough time.

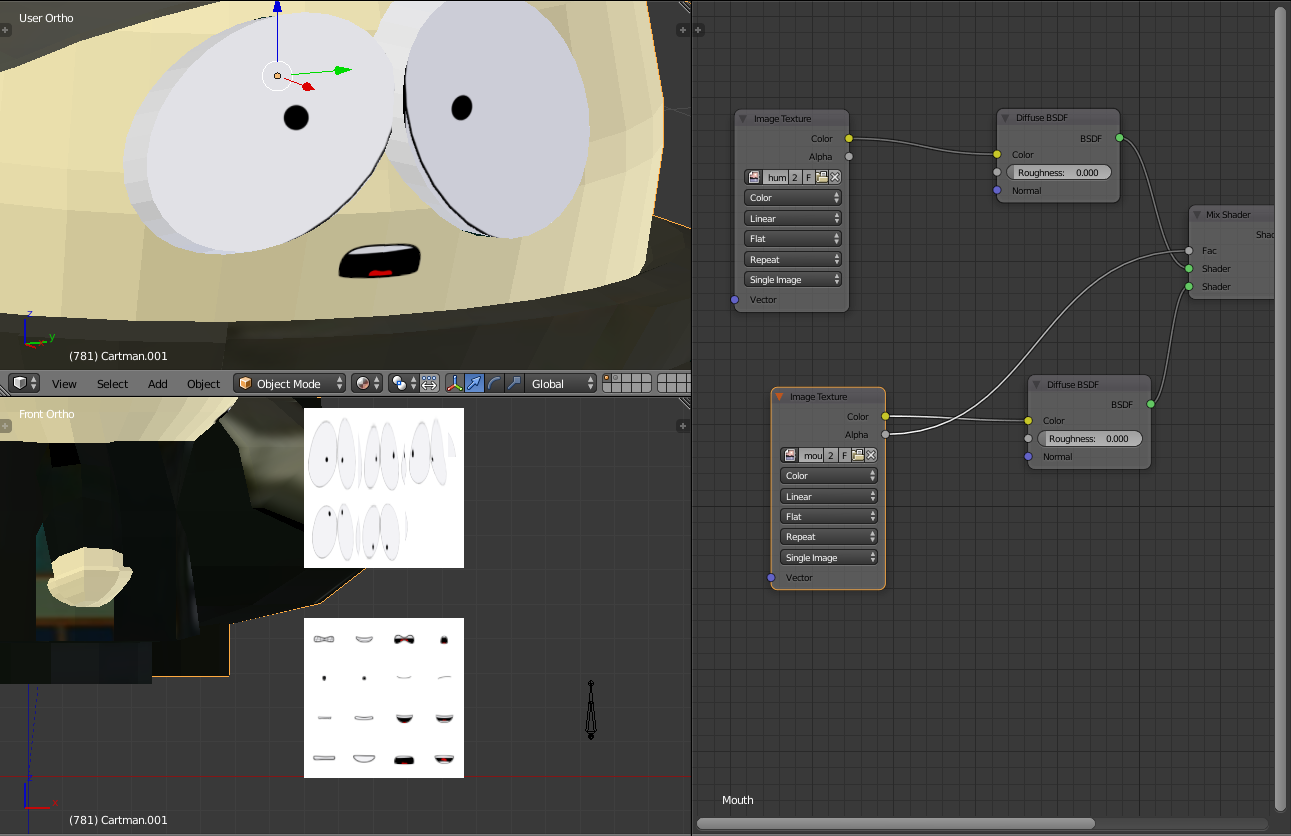


Figure 12 Animated Textures

# Materials

This animation has considerably high amount of materials, but most of them are as simple as diffuse shader and texture image. The only more complicated material is the snow one.

## Snow

For this assignment, I found a website that provides an amount of free full pack textured. Those include not only the image, but a bunch of bitmaps, reflections, etc.

The Snow material (Figure 17) uses a principle shader, which provides the functionality of multiple smaller shaders into one big. You can see that I am using also 4 textures, of which 1 is an image, holding a color data, and 3 are images that hold extra data for the shader such as reflection, roughness and displacement.

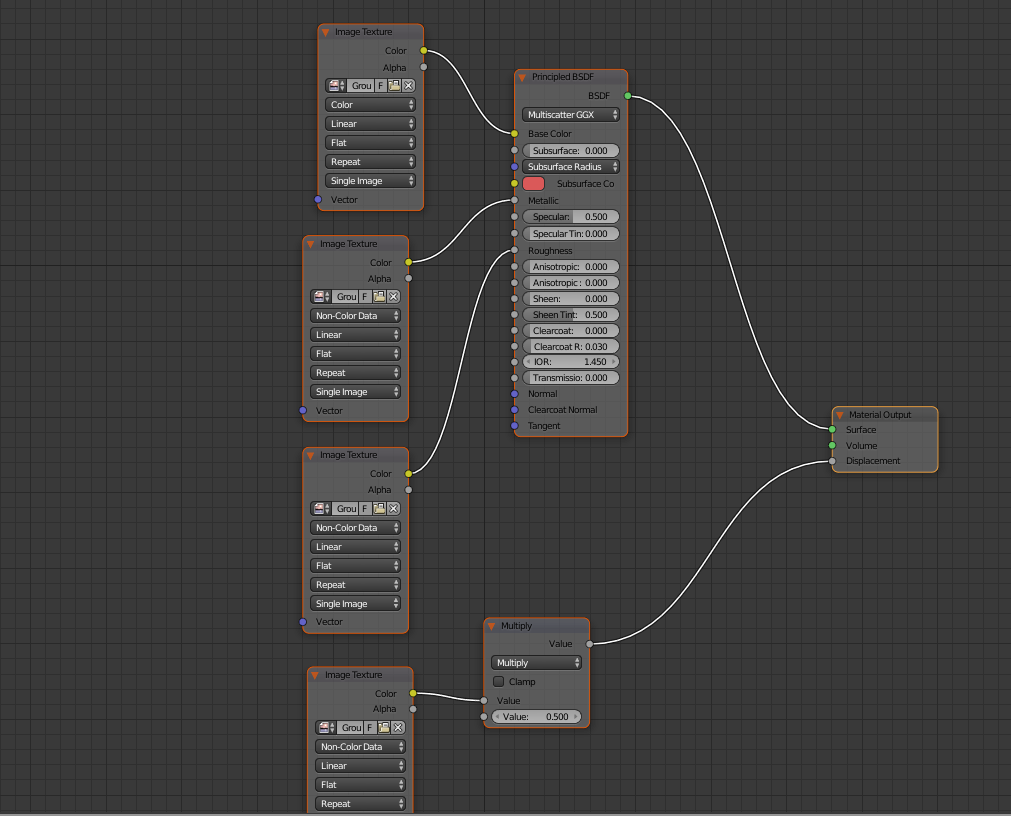


Figure 12 Snow Material

# Animating

I created two separate animations for each scene. Both are consisted of line animation and keyframing. The line animation is created using constraints to the characters and is used thorough the whole animation to get them from point A to point B. The keyframe animation is used to move the characters on the Y axis while they walk and to give the characters a bit of wobble when they talk, since there is no other way of identifying which character is doing the talking.

# Rendering

## Renderer

I rendered my scene using the Cycles renderer provided by blender. The main reason for my choice was because the convenience of using Node materials. I find them to be very easy to work with and since I was already familiar with them because of our previous assignment, it was straight forward choice.

## Lighting

For lighting the scene, I added 2 single point light and positioned them close to the characters. Since the skybox has a night image, the scene is quite dark, so I needed the lights close to the characters



Figure 13 Lighting the scene.

## Render settings

The animation was rendered at 24 frames per second. I figured out since the scene is low poly, slightly more choppy movement will look better than the smoothness of 60 fps.. I also reduced the number of samples in the render settings, which lowers the noise in the final version. I chose to go with 4 samples, because I was advised to do so. When I tried different sample values, I stopped seeing a difference after the 4sample steps, so I stuck with that amount.

# Post production

I used Adobe Premiere Pro for my post production. I added intro captions, background music and did a small colour correction.

# Reflection

I am extremely happy with the quality of my 3D models. I had a clear vision of what I want to achieve, and I think I managed to do so. However, this assignment was also the most frustrating one because of the animation. I was determined to have animated textures, but since I manage to break the whole scene while trying to fix them I decided to rollback and leave the textures as they were. For my next project I will be aware of what are the steps to execute a technique before I start the project to avoid similar mistakes

# Bibliography

//TODO:

# Reference Images Bibliography

The vintage aviator, 2014. *The Vickers Machine Gun.* [online]   
available at: <http://thevintageaviator.co.nz/projects/reproduction-guns/vickers-machine-gun>  
[accessed 01.11.2018.].

International Military Antiques, *Original British WWI Fluted Vickers Display Machine Gun with Tripod.* [online], available at: <https://www.ima-usa.com/products/original-british-wwi-fluted-vickers-display-machine-gun-with-tripod?variant=26169729669>  
[accessed 02.11.2018.].