Report on the Application for Virtual Reality Carla Sophie Beranek 3048170

Originally, the idea of the application included a less realistic scenario. However, early on, I realised that moving an avatar would be easier through a relatively realistic terrain. Thus, I created a grassy terrain and used different shaders and elements to add to its appearance. Specifically, I designed edges for the terrain in the form of a mountain range. Additionally, I added a broad open space, in which I could later place objects, and I added a certain area to the terrain, which I covered with trees. To create a more surreal atmosphere, I added floating objects, specifically floating mountains. To create the flying mountains, I added them to my assets from the asset store and placed them on levels above the ground plate. The terrain also includes different assets, such as mountains or grassy hills that were created either through shaders, or through adding them from the asset store and placing them underneath the terrain, so that their peaks would just pierce through the upper layer, creating the feeling that they grew out of the plane. As for trees, I created broad woodlands for the scenery, through adding pine-trees, using the mass-placing method.

Later, I created different objects using Blender and added them to the terrain. Some of these objects were fitted with scripts determining certain movements for them, some are just being used as stationary additions to the terrain. These stationary objects are all abstract shapes, which is reflected in their unconventional naming. The first object, *wobbly_blob_1*, was created in Blender through adding a sphere, colouring it, and giving it bumps on its surface. It is located in the forest of the virtual environment and its script allows it to change its speed into the z-direction at random, with the speed resting within a range of -0.2f to 0.2 f. The second object was named *wobbly_blob_2*. Its location places it on the right-hand side of the terrain, floating high above the ground. The script attached to it permitted it to rapidly change its colour, choosing colours from a random pool of options through using the Object Renderer, setting the new colour within a random range of colours, of which the object renderer chooses random colours at random times. The object called *cube_round_edges* is located at the right-hand side of the terrain, floating over the ground. Its script allows the object to change its position and speed at random. The speed ranges from -5f to 5f. Lastly, the *cube_and_ball* object was moved through spinning it. The object is placed near the player-characters starting point. The script enables the object to spin around its own axis through using the following function: *this.transform.Rotate(rotation*1*Time.deltaTime)*.

The next step included the movement for the player character. I used the free basic motions dummy model provided by the Unity Asset store. Specifically, I chose the yellow-skinned *BasicMotionsDummy* provided through Kevin Iglesias. As for the Character Controller, I added the one attached to the model and used the pre-set numbers. Additionally, to allow the model to move, I added the animator for the Idle walk, which allows animated walking movements forwards, backwards, and sideways. Finally, I added a box collider to the dummy. The box collider permits the Dummy to manage physical collisions, so that the character will stop walking or bounce back from objects in the terrain, instead of walking through them or getting itself stuck in them. To make the model walk, I added a C#-script, which I called *Move.cs*. The script includes determinators for speed (4f), jump Speed (7f), and for the impact of gravity (9f). To direct the movement of the dummy, the script determines the arrow keys of the keyboard as the steering elements for the dummy, using the > -key to

direct movement to the right, and the < -key to direct it to the left. To break the characters movements, once the player does not engage the specified keys anymore, the script contains a section which determines the next actions of the dummy, using an if-else statement:

```
// controlling whether movement exists
    bool isWalking = move != 0 || strafe != 0;
    animator.SetBool("isWalking", isWalking);
    //Vector3 moveDirection = transform.forward * move * speed * Time.deltaTime;
    charCont.Move(moveDirection);
    if(move != 0)
    {
        animator.SetBool("isWalking", true);
    }
    else
    {
       animator.SetBool("isWalking", false);
    }
```

To make the character jump, the player can use the space-key. To prevent the jumps from lasting too long, I set the jump height to a lower level and increased the gravitational influx.

Throughout this project, I have been confronted by a multitude of different problems and mistakes. The most difficult part when it came to coding were issues with error messages. Because the software I had to use due to the model of my Laptop did not display any error messages in the coding environment, but only in Unity once the program broke, correcting them grew quite tedious. Other problems were more specific: In the beginning, getting the player character to move presented as a difficulty. Firstly, the program did work, but the key input had no impact on the movement of the dummy that could be seen on the screen. After some investigation, I found out that I had imported a second dummy model, to which I had attached the script, but which I had not used or even placed in the game. To solve this error, I attached the script to the correct model and deleted the obsolete one. After this, the program started, but the dummy did not move. Additionally, at some point, it returned the error message *Parameter isWalking does not exist*; and it took me some time to recognise that I had made a spelling mistake in one of the lines in the middle of the code, in which I used the parameter *isWalking,* but had spelled it wrong. Once I had fixed this mistake, the model could be moved freely through using the key inputs. When trying to make some last adjustments, I made the mistake of moving the camera, without noticing it. Because I thought that the program worked, I exported it. When opening the exported program, I only found the image of the sky in the frame. I could not determine whether the model worked or not, and I feared that I had somehow broke the entire application. Because I could not find an explanation myself, I showed the application to a friend of mine, who quickly found the mistake: the camera

had been facing and filming the sky, instead of the dummy and its surroundings. After I had fixed this error, the program worked.

Overall, I think the application turned out quite well. The one thing I need to change in future projects is the approach to it. In the future these projects need to be managed with a bit more structure and a concise image in mind, as well as an improvement in time management. When it comes to this specific project, in retrospect, I should have asked for help sooner, to avoid some of the time pressure. But all in all, I am satisfied with the result.