Low Poly Game Environment

A flat shaded experience

Giang Chau Nguyenová

Abstract

This report summarizes my final project of the Virtual Environments course (T-723-VIEN). The main idea was to make an easy jumping puzzle game in a third-person camera perspective. The emphasis was put on the appearance of the environment – the user should be able to experience a consistent and dynamic world and not just static objects placed in the scene. The environment is made in a low poly style, which gives the overall cartoon feeling to the game. Furthermore, this approach works well with the uncomplicated level design.

1 Motivation

There is no free standard tool to make a dynamic low poly terrain in Unity (at the time of writing this report), therefore it was not possible to get the effects such as wind zones or water with the basic Unity Editor. These terrain effects are quite crucial to create a compelling and ever changing scene setting which emit the feeling of a functioning virtual environment. Thus I have decided to construct my low poly game with some satisfying wind and water effects.

There are some paid low poly terrain engines in Unity Asset Store, for example Polaris[1]. Currently a second version[2] is in the store.

[1] https://assetstore.unity.com/packages/tools/terrain/polaris-low-poly-terrain-engine-123717

[2] https://assetstore.unity.com/packages/tools/terrain/polaris-v2-ultimate-low-poly-terrain-engine-144645

2 Related Work

The mechanics of my game were inspired by an indie game called Omno [Manke 2020], which was funded by a Kickstarter[3] and is now still in development. The game has various obstacles as the player walks through it. To overcome these jumping puzzles, the player is made to solve them, either with the help of his magical staff or by clever jumping techniques. The game is also set in a third-person perspective and a lot of attention is put into the animation of the controlled character, which makes the game graphically pleasing and artistic. In my project, I would also like to let the player experience the environment through actions and not just letting the player walk around without a purpose. Moreover, I don't think the environment I have in mind would be sufficient to create a captivating walking simulator.

The environment itself was inspired by a game called Equilinox [ThinMatrix 2018]. The foundation of this game is the simulation of a whole ecosystem. It is a dynamic event-driven world, where the player sees the environment changing in time according to his previous actions. The mimetic constraints are defined so the player feels like he is in control of everything – from grass to animals, which is necessary for this nature-themed sandbox game. For me, the game further stands out in animations, whether it's grass waving in the wind or a hopping bunny. These details form the "dynamicity" that I want to bring into my project.

[3] https://www.kickstarter.com/

3 Approach

The following section describes my approach to achieve a good-looking low poly game environment. As the project stems from the first programming assignment, I will also shortly write about the base of my jumping puzzle game.

3.1 Basic Mechanics

The goal of the game is to finish the game by getting into the black tower and proceed to the next level (that has not yet been designed). The player is in control of a squared cat figure, which can change its color of fur when it interacts with a floating sphere. The cat can only walk on platforms that have the same color as its fur, so to get through the environment, the player has to change the color according to the platforms ahead. The only exception are the grey platforms because they are active all the time (meaning that the player can use them regardless of the color of the cat).

There are collectibles (fish) scattered around the world so the player can somewhat track which places he has been to. These collectibles are important to advance with the story and get into the black tower.

This sums up my first programming assignment on which I am building the rest of my environment.

3.2 The Story

I have reused my cat model to create characters that the player can interact with to get a better idea of the virtual environment. There are in total of three characters with different kind of dialog situations. The dialog system is a class with predefined sentences. These sentences are inserted into a queue right after the user decides to talk to a character. The sentence is typed out into a dialog box one letter at a time using the functions of coroutines in Unity.

During the story-telling dialog, the player is forced to make some decisions. These decisions can result in a game over screen and the player has to replay the game. To make the game story progress steadily I have placed fade out/in screens to make a smooth transition between scenes (e.g. teleporting the player to a different part of the environment).

The dialog interface can be seen in figure 1. The same dialog system is used for the end part of the game, where the player enters the black tower. The dialog changes when the player meets all the requirements to go inside. During other times the dialog will not let the player get in the tower.

Figure 1: *A dialog box which displays what the characters say with decision buttons*

3.3 Controlled character

The character has a script connected which determines how it's controlled (and also has a character controller component attached to it). The player manipulates with the standard WASD keys and the spacebar to move the cat character.

The player can see the cat that he is controlling in third-person perspective. I have chosen this camera perspective so the player can observe the color of the fur directly. It was a bit tricky to script the camera to follow the cat (player) properly. As a result the camera can also be rotated horizontally and vertically and the player can zoom in and out via a mouse.

Because the player can see the cat, it was important to have the cat animated. I have used the Blender armature system and rigged my squared cat model. I have left the weight painting in default state. In the next step I have created the key poses and let the tool interpolate between the poses. I also made sure the animation would loop to create a plausible idle animation.

The last detail I wanted to add was the cat's footprints to make it more believable that the player is in contact with the environment. I have attached a particle system to the controlled character which renders exactly one billboard of paws right under where the character is standing. The particles slowly disappear after 0.7 second or when the player jumps. The result can be observed in figure 2.

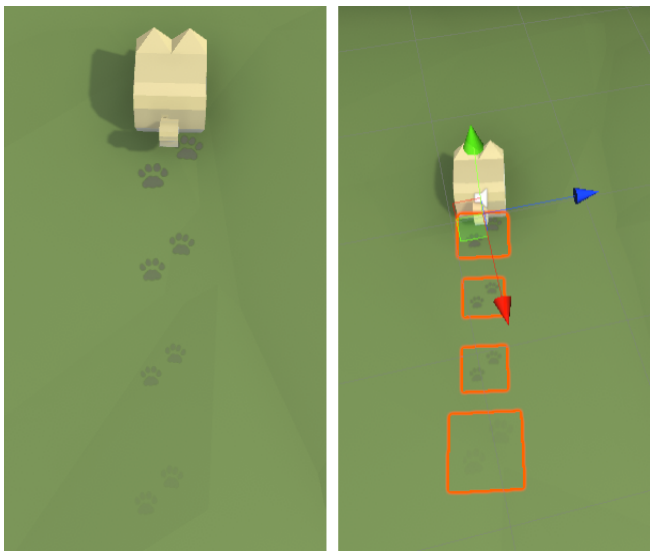

Figure 2: *Footprints of the cat character*

3.4 Animating vegetation

The animation of the wind is the core of this project. Here I am focusing on getting the visual simulation of wind through shaders in Unity. The main reason for this approach was to be able to parametrize the wind "simulation" (similar to editing the wind zone parameters in the Unity Editor).

3.4.1 Swaying trees

As all the vegetation in the environment is downloaded from the Unity Asset Store, it was rather limiting in terms of materials and textures attached to these models. In Unity, each material has one shader which defines the rendering pipeline. In my situation, the trunk of the tree was the same material as the leaves. I did not want to spend time on making my own assets for the time being, so I have decided to transform every vertex of a tree the same regardless of the part of the tree.

The TreeWind shader displaces each vertex of the model each frame. It changes the x and z position of a vertex according to parameters that change the cosine value in time `_Time.w` which

is a built-in shader time variable in Unity multiplied by 3. The variables I am using to parametrize the wind are *Wind direction, Turbulence, Speed* and *Vertex influence* – the vertex influence parameter determines how much the vertex is affected by the wind (a float value between 0.0 and 1.0 – for more details see figure 3). I have used Renan Bomtempo's Polygon Wind shader [Bomtempo 2017] to create a plausible wind parametrization.

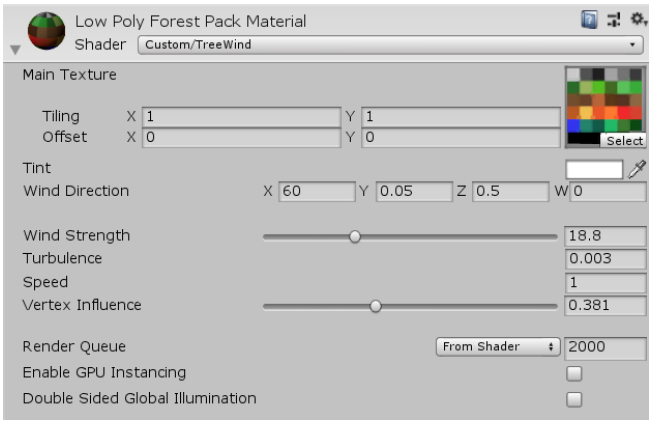

Figure 3: *Wind parameters for the trees*

3.4.2 Waving grass

For this shader I followed a tutorial written by Linden Reid [Reid 2018]. The grass waving animation is achieved by sampling a gradient texture (see figure 4) and moving the x and z positions of the vertices according to this sampled value. The wind ripple effect is done by scrolling the sample position based on time, meaning that the position shifts in x and z direction of the wind. Eventually, the sampled position will get out of bound, so it is necessary to clip the value between 0 and 1. Finally the sine (for the x position) and cosine (for the z position) values of this position are used together with other wind parameters (*Wind speed* and *Wind amplitude*) to move the vertices of grass.

Figure 4: *Gradient texture used for sampling*

To stop the grass from moving as a whole object (as the base of the grass shouldn't move because it's deep in the ground) and making the movement a bit realistic, the animation is modified based on the height of the processed vertex. The grass bends more, the further away it is from its root. A new variable is introduced in order to make the animation exponentially larger with the increasing y-value of the vertex position.

The final editable parameters for the wind to create the grass waves are displayed in figure 5.

3.5 Other details

This section is dedicated to other little features that I have added into my game.

Starting with the post-processing stack that I have used to make the unpolished low-poly environment look visually appealing: the

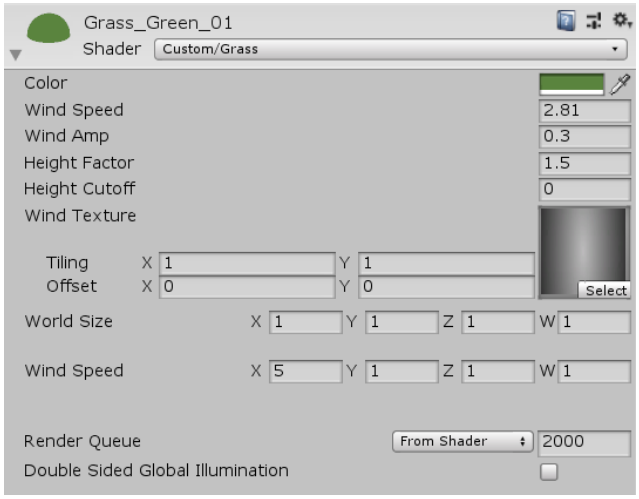

Figure 5: *Wind parameters for the grass*

effects I have used were *ambient occlusion, depth of field, bloom* and *color grading*. The scene is set in late afternoon so I have tried to produce this with a warmer color temperature and a yellow directional light. Besides the depth of field, the viewing is further limited with a fog of the same color as the lower part of the skybox. The difference between polished and unpolished environment can be observed in figure 6.

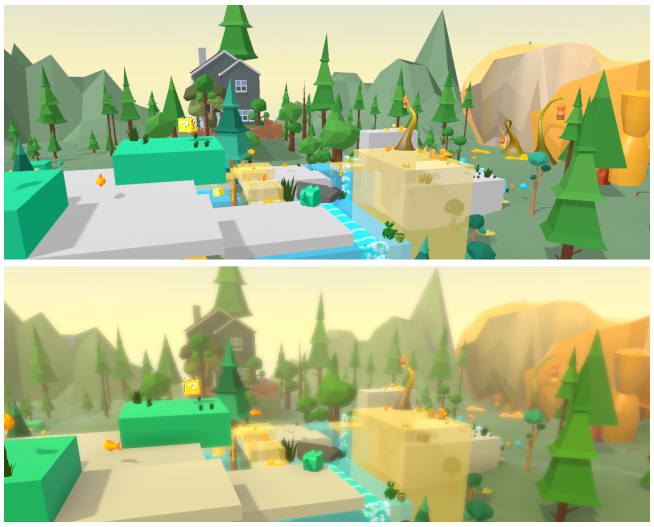

Figure 6: *Post-processing results*

Another fine detail was changing the material of the platforms from opaque to transparent when they were inactive (meaning that the player can't walk on them because he doesn't have the corresponding color). During the testing of the game, this feature was surprisingly successful among the testers because they could finally see the cat character even though they were walking through or inside the platform objects.

There are hints and interactive objects scattered around the world to further tell the story and set the mood. The hints are shaped in Mario-game-like boxes that rotate and display a hint when the player is close enough. The interactive objects emit soft "glitter" to grab the player's attention. Once activated, the dialog box will appear, showing the cat's (or players?) thought about the object.

Lastly, I created an intro scene where the player can either choose to start or exit the game (screenshot is shown in figure 7). It is a completely new scene that loads the main game environment. The scene consists of a birch tree moving in the wind with falling leaves (made with the particle system – this particular effect is also used in some places in the game). This scene was created for presentation purposes (for more details see the Results section).

Figure 7: *Screenshot of the intro scene*

4 Results

The finished project is a low poly game with the first level completely designed and implemented. The mechanics work as expected (despite the clunky controls of the character and the third-person camera at the beginning of development) and there is a purpose to the game due to the added story (although short and unfinished in the first level).

The animation of the vegetation turned out quite pleasing (at least with the right setting of the parameters). There is a slight problem with the swaying trees, as mentioned before (there was only one complete material for the whole tree), so the trunk is visibly moving with the leaves making it look like a floating jellyfish. However, it is noticeable only if the player completely focuses on the tree model, therefore I have discarded this issue in the project.

In the original plan I also wanted to write my own water shader, however I spent too much time debugging the wind shaders and therefore didn't have enough time to make another one. I ended up using a well-implemented low poly water asset[4] and some tricks with the particles to make good-looking ponds and waterfalls.

The intro scene was supposed to have a button for credits, which would display the resources I have used to make this game, but again, my time management was poor and therefore the intro scene feels quite useless.

5 Future Work

In the future, I would like to start and finish the planned water shader (at least for the pond). Had I known about the complication of debugging shaders in Unity I would have created a project which could be compatible with the visual shader editing tool Shader Graph. However, this feature was only available in Unity projects with the High-Definition Render Pipeline (HDRP) or the Lightweight Render Pipeline (LWRP) [Unity Technologies 2019]. After some research I found out that there was no effortless way to convert my current standard 3D project into a project with the scriptable render pipeline [Katstodian 2018]. I have never used the Shader Graph in Unity so it could be a valuable learning experience.

Another task is to make the intro scene more useful. I would like to add the mentioned credits and also make the intro scene available during the playtime as a pause menu. This will most likely have to be re-implemented differently because as it is now, the loading of another scene completely resets the current state and therefore cannot be used as a pause screen.

[4] https://assetstore.unity.com/packages/tools/particles-effects/lowpoly-water-107563/

References

BOMTEMPO, R., 2017. Shader - open source: Polygon wind - for low poly tree assets. https://forum.unity.com/threads/shader-open-source-polygon-wind-for-low-poly-tree-assets.499240/. Accessed: 2019-11-15.

KATSTODIAN, 2018. Installing lwrp into an existing project. https://github.com/Unity-Technologies/ScriptableRenderPipeline/wiki/Installing-LWRP-into-an-existing-Project. Accessed: 2019-11-15.

MANKE, J., 2020. Omno. https://www.playomno.com/. Accessed: 2019-11-12.

REID, L., 2018. Waving grass shader in unity. https://lindenreid.wordpress.com/2018/01/07/waving-grass-shader-in-unity/. Accessed: 2019-11-15.

THINMATRIX, 2018. Equilinox. https://store.steampowered.com/app/853550/Equilinox/. Accessed: 2019-11-12.

UNITY TECHNOLOGIES, 2019. Getting started with shader graph. https://docs.unity3d.com/Packages/com.unity.shadergraph@6.9/manual/Getting-Started.html. Accessed: 2019-11-15.