**Mini Project – 3D Worlds – Budget Spiderman**

Christian Petersen – Student Nr. 20206244

**Project Repository:**

<https://github.com/Kermyeet/Swing-and-grapple-game>

**Idea and Project Parts:**

The game works around the idea of having to swing on multiple building pieces and get away from a tsunami that is following behind the player, but also making sure not to fall into the fog below. The game depends heavily on the movement of the player and overall physics to get the player moving forward.

Scripts:

* **Camera Move:** One line that makes sure the camera’s position stays in the same position as the player’s head.
* **Game Manager:** Contains the overall system that allows the game to be an endless runner type. It makes sure that the game only keeps a limit of 14 prefabs spawned at a time, so in theory it could go on forever and not take too much processing power. It also contains the method used to spawn in other prefabs later in the game.
* **Grappling:** Much of this script came from Dani’s Tutorials video but works on making a spring joint being made between the player and the grapple location. The spring joint is important as it allows for the distance between the player and their grapple location to change, thus feeling like a more natural swing, instead of being stopped when grappling straightforward and just sticking to that distance.
* **Movement Script:** Much of this script also came from Dani’s tutorials video on movement, as its way of controlling the movement is very smooth and worked well with the swinging compared to other janky movement scripts I could make or find. This script also contains the controls for the UI and how much score the player currently has.
* **Rotate Grapple:** This script only controls the direction that the grappling gun is facing, so that it rotates to look at where the grappling point is.
* **Random Height:** This script is put on the prefabs for the level pieces and randomly scales the buildings on the left and right to different y scales.
* **Trigger Exit:** This script is used to add onto the score whenever the player goes through, but also lets the game know to spawn a new random prefab level piece ahead.
* **Wave Move:** This script allows for the big wave to move forward having a constant acceleration so it gets faster, but also limiting how fast the wave can move.

Materials / Objects / Levels:

* **“Player” Object:** This object contains all the necessary objects and positions needed for the player to move, rotate, grapple, etc. There is a skybox game object that is just a sphere that also renders the normal on the inside of the sphere. The sphere is necessary to get the *foggy effect* on the camera, as the camera’s fog effect did not render on top of the normal skybox.
* **“Wave” Object:** This object is what is following behind the player and contains different particle systems that are supposed to portray a bubbling of water and has a mesh collider to get more accurate collision detection.
* **“UI Screens” Object:** Contains all the UIs used for the game. The actual canvases for the different situations are children to the UI Screens object. All the text on the UIs uses a font from the unity asset store to make the text look pixelated.
* **Level Prefabs:** All the prefabs that are spawned in to continue the game. Each level piece contains an object that can be grappled with tall buildings to each side limiting where the player can move between. There is also a cube at the bottom that is supposed to be a form of fog and another “death” area that the player can hit.
* **“Clouds Mat” Material:** Material used for the particle systems rendering that makes up the clouds above the area that the player can swing on.
* **“Dynamic Vertical Fog” Material:** This material was found on the unity asset store and has its own folder containing the shader that it uses.
* **“Water” and “Sky” Materials:** Each material is a very simple base color, except *water* is transparent and put on the wave, while *sky* is opaque and put on the *Skybox* sphere.

**Utilization:**

The movement script and grapple scripts make use of the normal physics engine from the unity to get the main feature of grappling to work. The levels were made with unities pro builder and put into prefabs that can be spawned. The randomized prefabs work by having 6 prefabs spawn from the start (each 100 units apart as they are 100 units in length), but then when spawning new prefabs for later, they are spawned 600 units ahead to make sure the levels are spaced properly. After the first 6 level pieces, there is a “Spacing” prefab put in between all level pieces by checking if the random chosen level piece is named “Spacing”. Then when it has spawned once, it sets a Boolean to true, which is then set to false after a prefab not named “Spacing” is spawned. All the orange pieces on the level pieces have a layer mask named “Grapple” to make sure the grappling gun can only be activated when looking at the appropriate object. Each level piece contains a flat object in the middle of it that has a collision trigger to check whenever the object labeled “Player” has gone through it.

Time Schedule:

|  |  |  |
| --- | --- | --- |
| **Process** | **Time Spent** | **References / Sources** |
| Player + Movement | ~3 Hours  Much of the time was spent trying out different variations of moving the player, the physics of the player and such, until finally ending at using a tutorial to get something that worked like I wanted. | [Movement](https://www.youtube.com/watch?v=XAC8U9-dTZU):  <https://www.youtube.com/watch?v=Xgh4v1w5DxU>  [Grappling](https://www.youtube.com/watch?v=Xgh4v1w5DxU):  <https://www.youtube.com/watch?v=Xgh4v1w5DxU>  [Original Script](https://github.com/DaniDevy/FPS_Movement_Rigidbody):  <https://github.com/DaniDevy/FPS_Movement_Rigidbody> |
| Level Prefabs | ~4-6 Hours  Most of the time was spent trying to find out how to get the level pieces to be spawned at correct distances and figuring out how the overall system should work and fit together. The rest was put into making the simple level designs. |  |
| Wave | ~1-4 Hours  The wave in the final version was made quick, but a lot of time was spent trying to make a wavy material that looked like a water surface but gave up on as nothing useful came out of it. |  |
| UI | ~2 Hours  Simple UI was made first and then added more UIs nearing the final version. | [Font](https://assetstore.unity.com/packages/2d/fonts/free-pixel-font-thaleah-140059):  <https://assetstore.unity.com/packages/2d/fonts/free-pixel-font-thaleah-140059> |
| Particle Systems | ~2 Hours  Spent a lot of it messing around with settings trying to get the wanted effect with the particles on the wave. The clouds were made from a tutorial. | [Clouds](https://www.youtube.com/watch?v=H0jUEuPENKI):  <https://www.youtube.com/watch?v=H0jUEuPENKI> |
| Fog | ~3-5 hours  A lot of time was spent figuring out a way to limit camera visibility, until landing on making custom skybox for fog. (fog doesn’t render on unity skybox) | [Fog Solution](https://discussions.unity.com/t/how-to-see-a-sphere-from-inside-it/173174):  <https://discussions.unity.com/t/how-to-see-a-sphere-from-inside-it/173174>  [Fog Asset](https://assetstore.unity.com/packages/vfx/shaders/dynamic-vertical-fog-189939):  <https://assetstore.unity.com/packages/vfx/shaders/dynamic-vertical-fog-189939> |
| Game Manager | ~1 Hour  Had to implement a game manager to make spawning prefab level pieces easier. |  |