SS23 Proximity

Firebreath Forest

Alpha Release

Johannes Madest

Mingyang Li

Bassant Elnaggar

Task Progress

Excitement Minigame:

The excitement minigame scene as well as the accompanying game logic have been finished. In Addition, the UI has been upgraded to incorporate our assets. The only missing part is the Dragon chasing the player, which will be done during our next milestone. Furthermore, another aspect that will be focused on improving the adaptive difficulty to offer an adequate challenge.

Anticipation Minigame:

The Scene for the Anticipation minigame has been finished. In addition, the game logic is in a complete state with the implementation of the dragon's behavior as well as a time limit represented by the sunrise.

In general, the main focus regarding the minigames moving forward will be revolving around iterative testing and fine tuning.

Main Scene:

The terrain design and fundamental functionalities of the main scene have been finished. Moving forward, our next focus is to fine-tune the terrain details

to complement the roller coaster tracks. Additionally, optimizing the frame rate will be a crucial aspect of our development process.

If time permits, we also plan to enhance the realism of the rollercoaster experience by implementing physical calculations for speed adjustments during uphill and downhill sections.

Implementation

Excitement Minigame:

The Cart the player rides in follows a path created with a Spline Animation. Yet, this type of component is not particularly designed for situations like ours, in which the speed of the animated object is dynamic. The Spline Animator works by defining either a constant speed or a completion time to calculate the object's traversal. Altering either one of those mid-animation leads to unwanted behaviors, as the altered value will be treated as if they have been set from the start, causing large jumps in position.
To circumvent this issue, the progression of this animation is controlled manually by converting the cart's varying speed values to increases in the animation's progression.

The UI has been upgraded to incorporate our assets. A dictionary has been used to access the needed asset for the respective requested button inputs. In addition, an indicator panel has been added to highlight the currently active button prompt.

Anticipation Minigame:

The main focus during this milestone in regards to implementation is the behavior of the dragon. In its current state, the dragon will cycle through multiple states: Deep, Light and disturbed sleep as well as awake. If the player keeps their distance, the dragon will transition from deep to light to awake to deep again on a fixed timer. The dragon’s state can be influenced by the player making noise. Each action the player takes has a noise level attached to them. Depending on the player’s noise as well as the current distance to the dragon, a noise value is calculated. If this value exceeds a certain threshold during the dragon’s normal sleep phase, it will transition

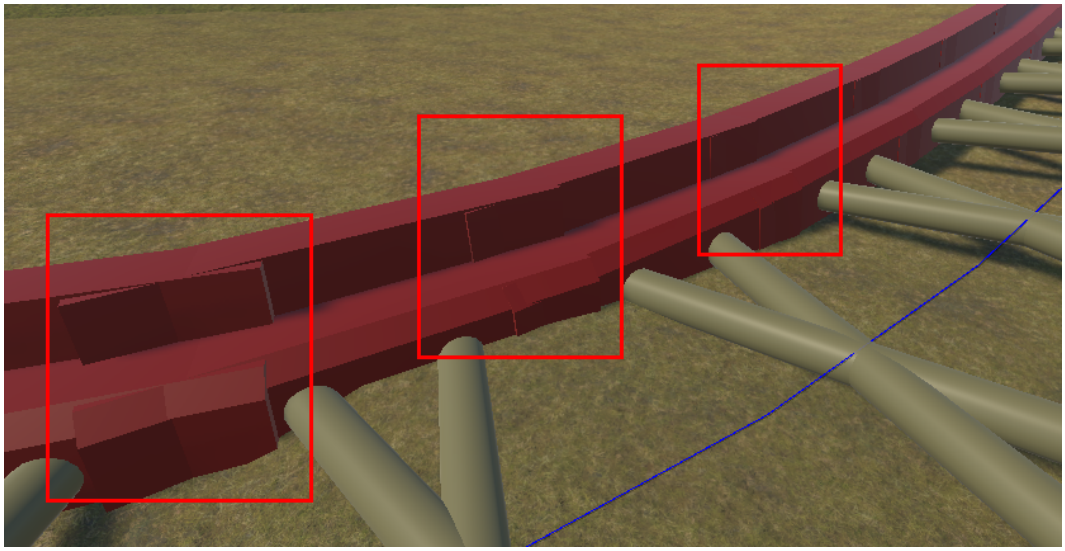
to the disturbed sleep state, after which the dragon will awake shortly after. Once the dragon is awake, it will look around and check whether it can see the player. If the player remains in the dragon's vision for too long, they will be considered detected and the game will end.

To limit the total amount of time the player has to collect gold, we implemented a sunrise. In its current state, it functions by locking the dragon's state to awake after a certain time has passed, which is indicated by a constantly increasing light intensity.

Roller Coaster and Railroad Tracks (Main Scene)

Since our last milestone, we have made the decision to utilize the Spline package from Unity to create the rollercoaster's route and enable it to follow along. As we progress to the next phase, our focus is now on placing the tracks along this route. After numerous attempts, we have discovered that utilizing the Spline Instantiate component provides us with the most convenient method. However, we have encountered a challenge: the track pieces generated by this method do not fit seamlessly along the path.

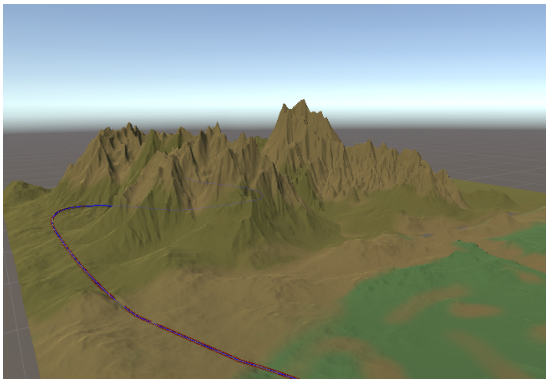
One of the main reasons for this issue is that the Spline Instantiate component does not generate a mesh that adheres precisely to the path. Instead, it positions prefab objects along the spline. This becomes particularly problematic when we enlarge the track pieces, as they fail to combine smoothly, especially in curved sections(see figure below).

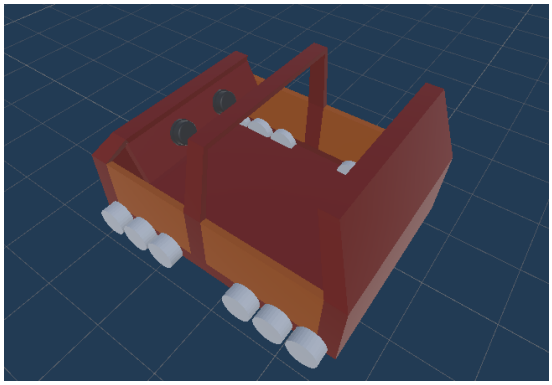


When we reduce the size of the track pieces to create a smoother appearance, we encounter an issue: it increases the rendering cost, subsequently leading to lower frame rates. This can create performance problems that need to be addressed.

The reason behind the higher rendering cost is that reducing the track piece size results in more individual pieces being rendered, increasing the overall complexity of the scene. Each track piece requires resources and processing power to render, and when there are a large number of them, it can strain the system and lead to lower frame rates.

To tackle this problem, we need to find a balance between achieving smooth track transitions and maintaining optimal performance. One potential solution is to implement a Level of Detail (LOD) system for the track pieces., which we will talk about in the future plan section.





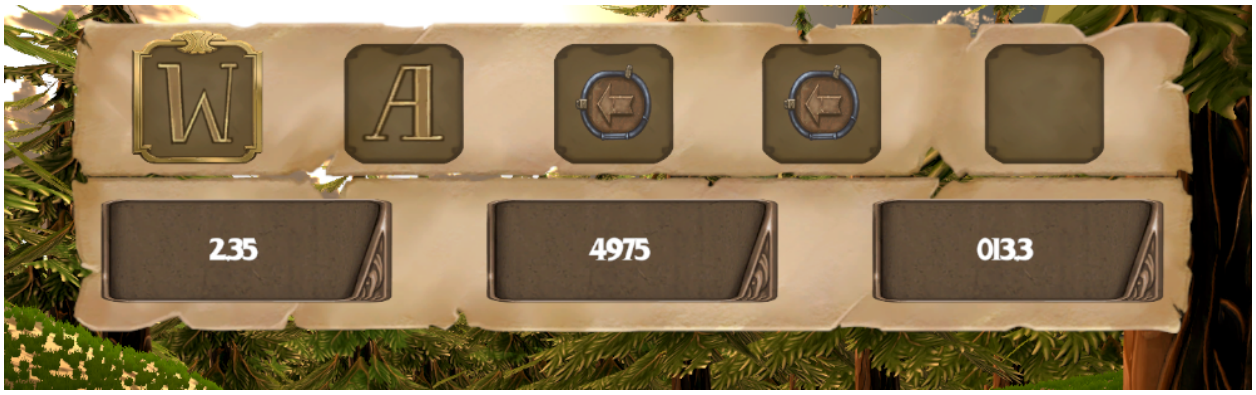
Regarding the open forest area, we have employed Unity's Terrain tools to create a realistic terrain with mountains. The roller coaster track has been designed to traverse through these mountains in the first-person view. Following the mountainous region, the rollercoaster enters a vast forest area.

Within the roller coaster car, we have included two buttons that players can interact with. When a player presses one of these buttons, the cart accelerates, propelling them toward the corresponding mini-game.

Designs and UI

Excitement mini-game:

Instead of placing the tracks manually, which was an inefficient way to consider, we used a method built-in to the spline package to place the tracks automatically into the terrain. We added more scenery to this mini-game and integrated the UI along the level design with the functionalities. The player is followed by the flying dragon all along, and the shadow of the dragon from above is visible to the player. The only remaining thing to be added to the scene is to add an end destination for the player, so they know that the dragon is no longer chasing them, along with audio to the scene as well.



Anticipation mini-game:

After trying different layouts for the anticipation mini-game, we agreed on having a centered open field in the forest for the player to go around and

allocate the gold piles. The dragon is placed in the middle of this open field, sleeping initially then wakes up according to the sleeping pattern to check the field and if the player is allocated then they will die and start the scene again if not then the game goes on and the player keeps collecting the gold and the dragon will be back to sleep. This mini-game starts with a gloomy dark lighting and them is keeps lighting slowly, it will be harder for the player to hide and easier for the dragon to locate the player. We will be using post-processing in the next milestone, to make the gold piles shine in the forest to be identified by the player faster, also the scene will have more fine-tuning and audio added. Finally, we integrated the UI along with the level design with the functionalities for the game.

UI:

For the UI screens, we used the same assets that we introduced in the previous milestones. We added a start scene for the user to either choose to start the game immediately or to go to the options section. For the options scene, the user either see the instructions for each mini-games, set the audio settings (will be done later in the next milestone), or to quit the game. The UIs used are again following the adventure theme that we picked starting of our game implementation.

Future Plans

Frame Rate Optimization:

Although we have experimented with techniques like incorporating grass and flower textures instead of placing individual objects, we are still encountering low frame rates when traversing through the mountains and forests with the cart. One of the contributing factors is the presence of a large number of track pieces, reaching tens of thousands. In our next phase, we are determined to explore potential solutions to enhance the frame rate.

One approach we are considering is consolidating the tracks into a single object, thereby creating a mesh that encompasses the entire track system. This consolidation would reduce the overall number of individual elements that need to be rendered, potentially leading to improved performance and higher frame rates. By treating the track pieces as a unified entity, we can streamline the rendering process.

Additionally, we are also investigating ways to optimize the Level of Detail (LOD) system. By carefully calibrating the transition distances and creating appropriate LOD versions for the track mesh, we aim to dynamically adjust the level of detail based on the viewer's proximity. This technique ensures that only the necessary level of detail is rendered, further optimizing performance without sacrificing visual quality.

By implementing these measures, we anticipate significant improvements in the frame rate during cart travel through the mountains and forests. Our team

remains dedicated to finding the most effective strategies to enhance the overall performance and deliver a seamless and enjoyable experience for our users.

Focusing on the emotions:

As the underlying components of our game near their completion, we will revisit our main objective: Invoking the emotions tied to rollercoasters with our minigames. In that regard, we will heavily invest into testing, fine-tuning and extending our minigames with the goal to further deepen the connection to their intended emotion.