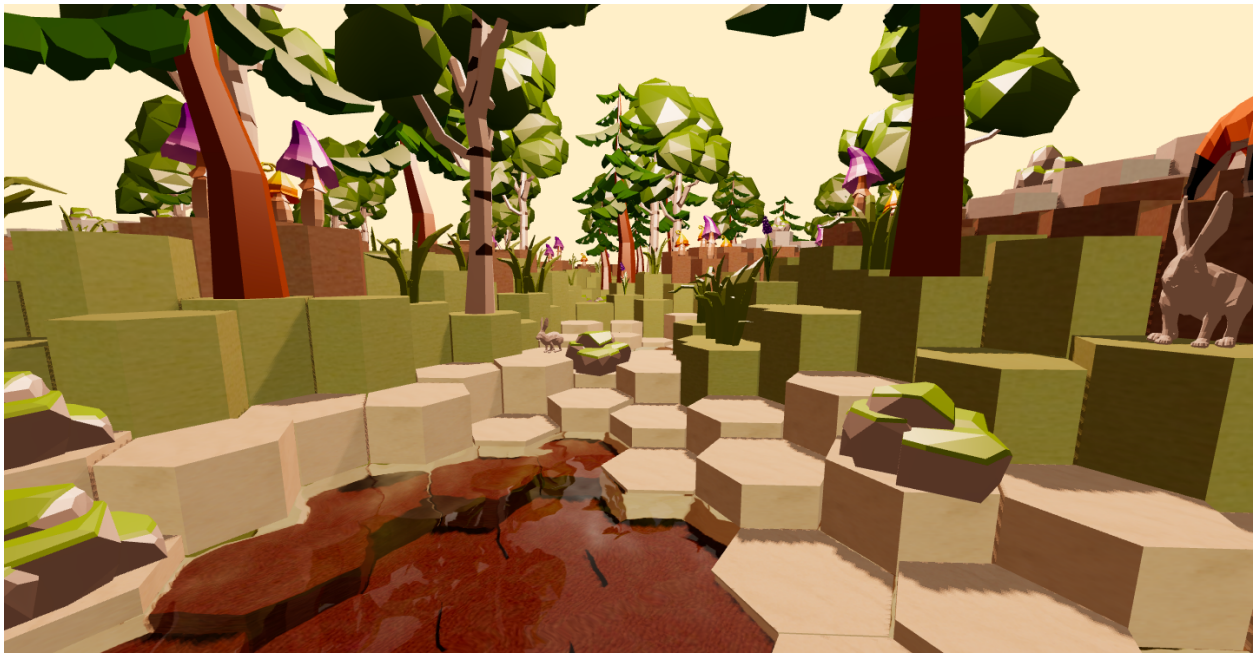
Ryan Xia, Sunny Yoo
COS426: Rabbit Rescue Final Report



Abstract

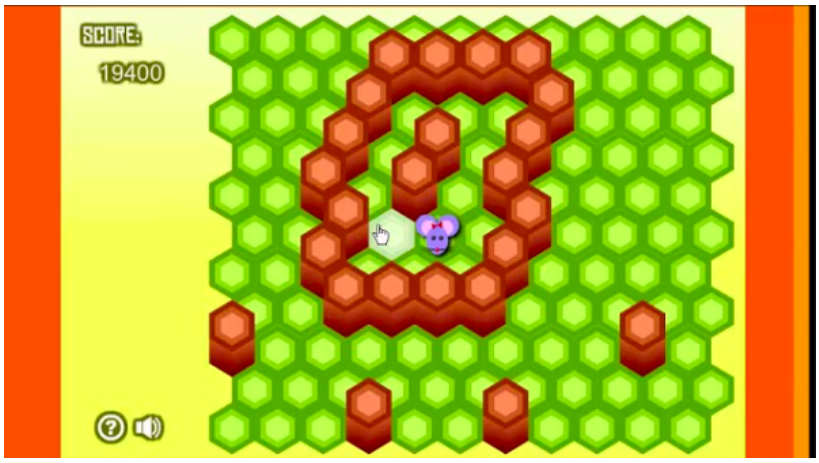
We developed an app called *Rabbit Rescue*, a survival game where the user controls a rabbit and jumps between tiles hoping to rescue his baby rabbits before the game ends. The rabbit must avoid bear traps on the loose, as well as foxes that jump from tile to tile in pursuit of the rabbit and its babies. When the user completes the challenge and receives a score as a function of the time passed, number of traps or foxes encountered, and number of baby rabbits rescued.

Introduction

In our app, we were hoping to create a typical survival game with a unique twist. We wanted to make a game that was visually appealing with a beautiful nature aesthetic that can, in a sense, be therapeutic to play, and 3 dimensional with an immersive perspective that really makes you see from the lens of a rabbit. Through our project, we hope that users would find a fun game that gives all the exciting elements of the challenge of Pac-man with the feel of Minecraft.

We approached the app development from this lens, hoping to have both the “arcade” feel whilst simultaneously being immersive. For example, we used a hexagonal grid layout with random Perlin noise to generate a realistic nature aesthetic. Whilst other survival games may have a hexagonal layout such as Mouse Trap as shown below, it lacks the three dimensional aspect/camera perspective that really allows for an immersive experience. While some games

have a layout with a realistic nature layout, they can either be aesthetically bland with either a grid layout or lose the “arcade” feel that a hexagonal tile layout could offer. Each refresh generates a new map that helps retain the “arcade” and explorative aspect of our game.



Methodology

We implemented our project backend completely in Javascript and used HTML/CSS to develop the front end. We used packages from the THREE.JS framework, such as WebGLRenderer for rendering, Clock for updating the time, ObjectGeometry for creating the tiles, Vector3 for controlling the positions of the assets, and PerspectiveCamera for controlling the camera angle. We also imported the SimplexNoise package from CDN Skypack to generate our beautiful randomly generated terrain. For our object files, we searched for 3D files, particularly in the FBX format to be consistent with our asset loading. These were found on Turbosquid, a website that allows users to purchase or use assets generated by others.

We decided to deviate from the starter code that was offered to us as we found a repository that used Perlin noise to generate a similar looking hexagonal tile layout. We chose to build off of this repository instead as the basic implementation for the tiles were developed here. This starter code was taken from Domenicobrz’s github in the link attached.

While this gave us a head start, the map that we started with was very limited in many ways. First of all, the aesthetic and fbx objects that were used were different from the ones that we were hoping for, and therefore we had to acquire, load, and finetune our code to support assets that too would be procedurally generated. In addition, the tiles were not linked with each other at all, meaning that without our implementation, there was no way to know which tiles were next to each other and thus potential animals (rabbits, baby rabbits, and fox) could traverse through the map, and there was no way to know which tiles were “valid” tiles in the sense that there were land tiles as opposed to water tiles or tiles covered by trees. All of this was implemented by us on our own to create a map that could support the mechanics for the animals that were acting on the map. To overcome these hurdles, we first kept track of each of the tiles in a map and used an offset hexagonal coordinate system to keep track of neighboring tiles, thereby allowing us to have a map that supports traversal. We also labeled tiles as traversable or non traversable depending on if it was a water tile and only allowed movement to tiles that were traversable.

In addition, the starter code was not camera friendly, with the camera angle being fixed even when the map got larger and larger. The camera angle did not support first person or birds eye view following the rabbit, all features that we had to implement.

Next we needed to generate the moving assets randomly for the rabbit, baby rabbits, and foxes. We did this by finding all the tiles that were valid and spawning all the objects randomly by drawing from these tiles. After doing so, we needed to implement the mechanics of the asset movements for the rabbit, baby rabbits, and foxes. For the rabbit, we were deliberating different options to control the rabbit’s movement. One approach was to use keyboard direction keys such as the WASD keys. A limitation with this approach was that there were six adjacent tiles for every tile, meaning that there ought to be 6 keyboard keys that were roughly in the shape of a hexagon to perfectly control the rabbit. Unfortunately, this is not a standard way of moving around in games typically, and so a user would likely not be comfortable with moving around in this way. Another idea was to use the trackpad and the clicker to click onto adjacent tiles that the rabbit would like to jump onto. However, a limitation of this was that this would

require a lot of manual click and mouse movement, which would be unideal for the user. Therefore, we decided to go with our approach which was to use the left and right keys to rotate the direction of the rabbit and the spacebar to jump forwards onto the tile that is directly in front.

Likewise, the baby rabbits had to be implemented to follow alongside the rabbit upon being rescued. This was implemented by randomly selecting a tile for the baby rabbit to walk on that was adjacent to the rabbit as the rabbit moved. A limitation to this came due to the fact that the rabbit was allowed to walk on watery tiles while baby rabbits were not allowed to, and as a result, there are cases when the rabbit was not adjacent to any tiles that a baby rabbit could move to. We decided to get around this issue by making the baby rabbit stay in place in the cases when the rabbit went into water, until the rabbit returned back to terrain, in which case the baby rabbit would appear next to the rabbit again. While not ideal, this implementation made sense as one may realistically imagine that baby rabbits would generally be following the rabbit and find a land path to the rabbit even if the rabbit enters into a watery area as long as the rabbit is within range of the baby rabbit.

Furthermore, the foxes were implemented to jump to tiles every second in a path that brings them closer to the rabbit to simulate its “pursuit” of the rabbit. Upon contact with the rabbit, the rabbit’s score would go lower, and upon contact with the baby rabbit, the baby rabbit would die and disappear, making the rabbit’s score go significantly lower. There were two considerations for an implementation of the path searching algorithm for the fox: A* path finding algorithm or just a greedy, “closest tile” heuristic algorithm. For the sake of performance and simplicity, and to add a degree of randomness that simulates nature, we went for the latter implementation.

In addition, we implemented animations, such as the animals when they jumped from tile to tile. We also created a burrow that effectively ends the game, and a score that is a function of the number of babies rescued that survived, time taken, and rabbit’s health. Other html pages were added such as a start screen with instructions and an end screen showing your score and allowing you to restart the game.

Discussion and Conclusion

After implementing our project, we wanted to make sure that our game hit our objective, which earlier was described as one that gave the excitement and “arcade” nature of a survival game such as Pacman but had the immersive and aesthetically pleasing feel of Minecraft.

We asked friends and strangers to assess our game in terms of aesthetics and understandability. Feedback was generally positive, and the areas that were suggested for improvement were ultimately incorporated. These include changing the mechanics of the rabbit’s movement from WASD to hexagonal, changing the camera angle and giving camera angle choices, and changing the user interface to add a start and end screen. Most users commented that the visuals were very pleasant, and that there is even almost an addictive feel to the game because of the combination of the visuals as well as the game idea.

Generally, it seemed as if our approach was indeed promising and hit the objectives that were imagined at the start, albeit not perfectly. We still see areas in which our project could be improved. For example, further work that can be done is to more rigorously generate the grid so that all the valid tiles are part of one connected component. We didn't want the animals to be able to walk on water tiles, but at the same time, the rabbit had to have the ability to traverse "islands" where baby rabbits by chance randomly spawned. In our implementation, we chose to meet in the middle by allowing rabbits to hop across water tiles but foxes and baby rabbits were not able to. A more advanced algorithm could perhaps ensure that the tiles were all connected, and none of the assets could go into water.

Further contributions could also include more animations for when foxes or bear traps came in contact with the rabbits. For example, in Minecraft, when the user hits a pig, there is a sound effect and a visual that appears onto the screen. Extensions could include animations such as these as well as other assets like waterfalls that beautify the scenery, or berry bushes that give health. Additional assets are always welcome, as long as they do not overcomplicate the game.

Ultimately, we feel as though we obtained our goals, and it was an incredible learning experience for us. We want to thank and give a special shout out to Professor Felix, all the TAs and UGAs, and Edward for being so helpful throughout our time working on our project. The invaluable feedback we received throughout the course and exciting materials we learned and were able to grapple with formed the very foundation to work on a project that we can sincerely say we are proud of.

Works Cited

https://github.com/Domenicobrz/Threejs-in-practice/tree/main/three-in-practice-2
https://www.hoodamath.com/games/mousetrap.html