Kronos

A Virtual Reality Roguelike

Reily Stanford
Union City, Tennessee
rstanfo1@utm.edu

Enrique Tejeda
Martin, Tennessee
enrgteje@ut.utm.edu

ABSTRACT

Kronos is a virtual reality video game with roguelike elements in which time is falling apart at the seams. It falls on the protagonist, Kronos, to restore the timeline to its chronological order. Along the way, the hero is met with obstacles to overcome to achieve this goal. Kronos has pixelated 2D art sprites and textures in a 3D world that is inspired by games like DOOM (1993) and Paranautical Activity. The game is a first-person shooter where you are traveling between a mismatch of time periods in order to restore the original order of time. Kronos is built using Unreal Engine 5 and a Procedural Dungeon Plugin created by BenPyton. The behaviors of the enemies within the game, the item effects, and the weapon mechanics are created using Unreal Engine's Blueprint system, which is a visual scripting language. Each floor is set in a different time period with various elements about each period feeling out of place. This can be seen in the enemies and objects within the level.

Author Keywords

Authors' choice; of terms; separated; by semicolons; include commas, within terms only; this section is required.

1. INTRODUCTION

Kronos is a virtual reality game that is set in the first person. The art style is similar to DOOM (1993) in that the enemies are pixel-art 2D sprites that always face you. The game is a shooter where you are tasked to bring the timeline to its original state. Someone has disrupted the chronological order of events leading Kronos, the protagonist, to fix the timeline by any means possible. There are three levels that each correspond to a time period where you can easily tell that time has been altered. This is apparent when you see enemies that do not match the current time period the level is set in.

Mechanics

A primary mechanic that the game has is procedural generation of level layouts. The game offers three methods of movement: teleportation, head-based locomotion, and hand-based locomotion. With teleportation, you move by pointing your hand to a specific location and pressing the movement button. The other option, locomotion, moves like a standard video game in which you hold the movement button to move in your desired direction. Each level is concluded by defeating a boss enemy that is stronger than the enemies you have been fighting on the current floor. There will be multiple weapons, items that affect your player's stats, and a variety of enemies on each floor.

2. TECHNICAL SPECIFICATIONS

The project is built with Unreal Engine 5 [5] and we used Unreal's Blueprints, a visual programming language, to create the logic of the game. For complex functionality that would be too visually complex to be built with Blueprints, we create with C++ [NOTE: WE HAVEN'T MESSED WITH C++ AS OF YET]. We use a plugin called Procedural Dungeon Plugin [2] that places the rooms into a playable layout. To plan our roadmap we used Milanote [7]. The assets we have in our project come from Humble Bundle [3], itch.io [6], and Mixamo [1]. As for source control, we used Git LFS [4], which is used for large files on github, and the built-in source control for Unreal Engine.

3. PROJECT DEVELOPMENT

Throughout the development of Kronos, we have split responsibilities and tasks so that we were not both working on the same feature to maximize efficiency. This has led to the both of us having different experiences when creating our game.

Reily's Experiences

One of the first things that I started messing with in Unreal Engine was adding models and proper collision to those models in the built-in example world in the engine. I quickly learned the basics for adding props to the world. I then decided to tackle my first real addition to our game, locomotion movement. Unreal Engine's built-in VRPawn object is created with a teleportation-style movement instead of the smooth movement you would expect from a standard 3D video game. Adding the option of smooth locomotion to the game allowed me to learn a lot about the Blueprints scripting language. I was able to learn about events, adding different elements to the viewport that aid with detection in the world that aren't visible to the player, and different custom collision descriptions. In the process of adding this, I ran into one major problem. The default collision provided by Unreal led to the player being moved while holding an item due to the addition of smooth locomotion. Once the collision had been sorted, it was time to move to the next feature, enemies. Adding enemies allowed for me work with sprites and flipbooks to make an animation for our 2D enemies. It also gave me a chance to experiment with behavior trees to allow for the AI to function on its own. Prior to the decision to use behavior trees, I was creating the behavior entirely through Blueprints. This caused a lot of issues, especially when it came to handling animations. Occasionally, the enemy would have specific timings on animation changes that would stop the animation from being able to change again. Alongside enemies, item spawning and collection was worked on. This gave me some experience with random generation along with making some simple sprites myself.

Enrique's Experiences

When we first decided to create a VR game, I immediately began researching on how to import and use a low-poly weapon. It was overwhelming at first because I had never developed with any game engine, but it got easier as I read Epic's documentation on the Unreal Engine and watched YouTube videos on the subject. The hardest part of getting a weapon to work was finding a decent muzzle fire animation. I stopped creating weapons as soon as I built one so that I could focus on world design.

Procedural Generation

In order to make a rogue-like game, we needed Kronos to have procedural generation. We both wanted the generation of levels to be similar to how The Binding of Isaac's system works. This type of generation places levels together randomly until a valid dungeon is created. A valid dungeon is when there is a start level, an end level, and a number of levels that connect the start and end level together. Instead of creating a system that places the levels for us, I decided to search if Unreal had a built-in system for this. Unreal did not have a free plugin for what we needed, but I did find a free plugin on YouTube called Procedural Dungeon Plugin by BenPyton. This plugin had exactly what we were looking for. The plugin allowed you to set the minimum and maximum amount of levels between the start and finish since you would want to have a cap on the amount of rooms on a level, otherwise the plugin would be adding levels forever. Because this plugin is not provided by Unreal, it was difficult to get it working with our project since the plugin was built using a third person character, while Kronos uses a VR character. This was problematic since it was difficult trying to find out how to teleport the player to the generated level. In addition, the generated level were only showing the first two rooms so I spent hours trying to troubleshoot what was going wrong on my end. I checked the documentation, the Blueprint I modified, and the example game the plugin provides to compare Blueprints to see why I could only see the first two rooms. A day goes by and I decide to check the plugin's issues tab on GitHub and found a ticket that had a similar problem. The author of the plugin resolved their issue and I was able to fix mine by disabling a single setting called "Occlusion Culling" in the plugin's settings.

REFERENCES

1. Adobe. Mixamo. https://www.mixamo.com/, 2022.
2. BenPyton. Procedural dungeon plugin. https://github.com/BenPyton/ProceduralDungeon, 2019.
3. Humble Bundle. Sfx and music for your games. https://www.humblebundle.com/software/sfx-and-music-for-your-games-software, 2022.
4. GitHub. Git large file storage. https://git-lfs.github.com/, 2014.
5. Epic Games Inc. The Unreal Engine. https://www.unrealengine.com/en-US, 2022.
6. itch.io. itch.io. https://itch.io/, 2022.
7. Milanote. Milanote. https://milanote.com, 2019.