Samurai’s Swarm

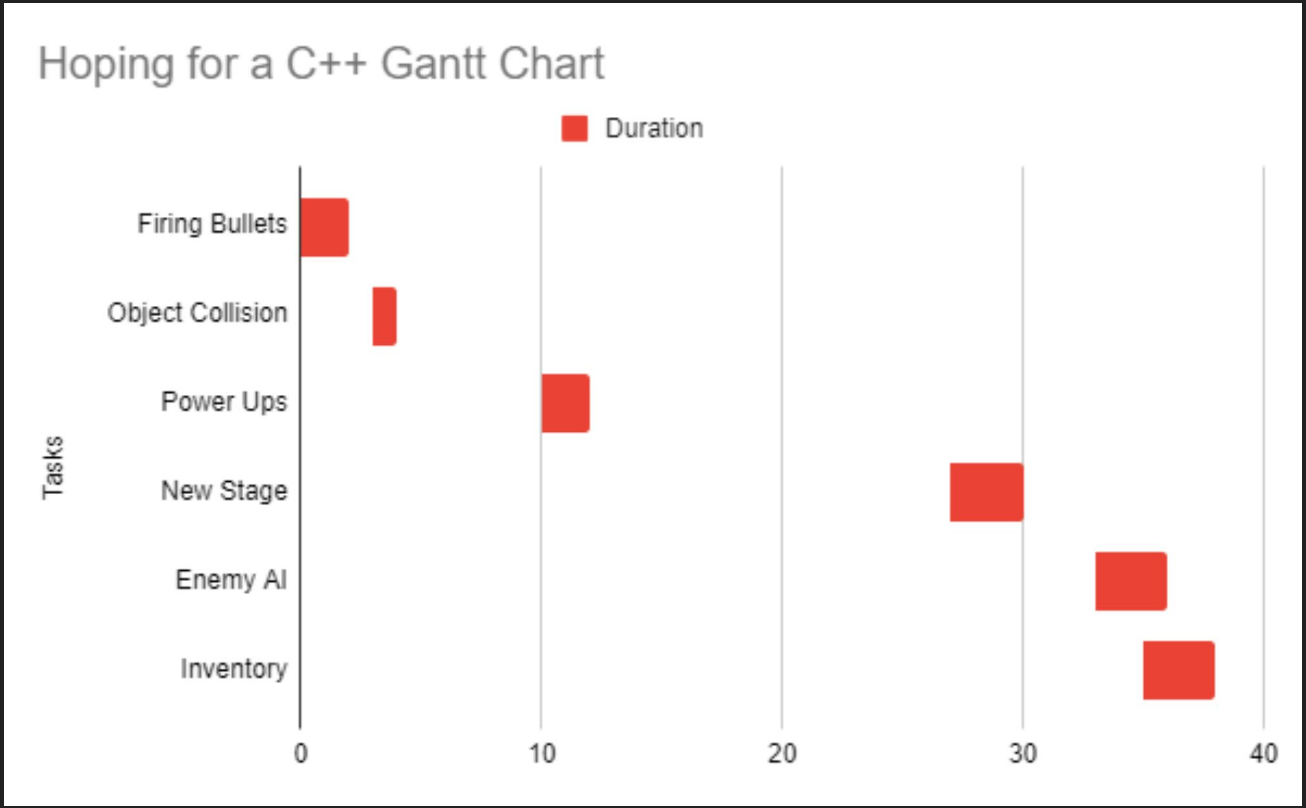
Team: Hoping for a C++

By: Dylan Vannatter,
Justin Rickert,
Emily Linderman, and
Dawson Herner

High Level View

Samurai Swarm is a Dungeon Crawler game that has an infinite number of levels. The goal of the game is to make it to the highest possible level without being defeated. By using power ups, hiding behind obstacles, and dodging enemy attacks the player will avoid losing health. In order to advance to the next level, the player must kill off all enemies in a room. Each level in the game progressively gets more difficult with new enemies as the player advances. How far can you make it before your time is up?

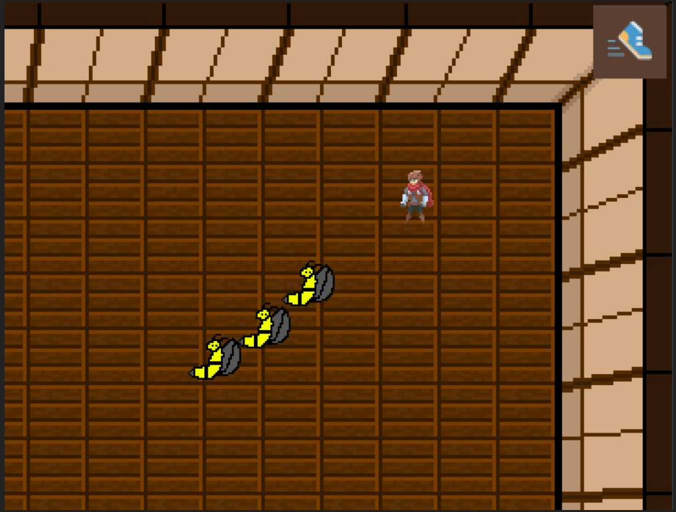
Project Timeline



New Inventory System

Added an inventory system that allows the user to pick up power ups, toggle between them, and activate them.

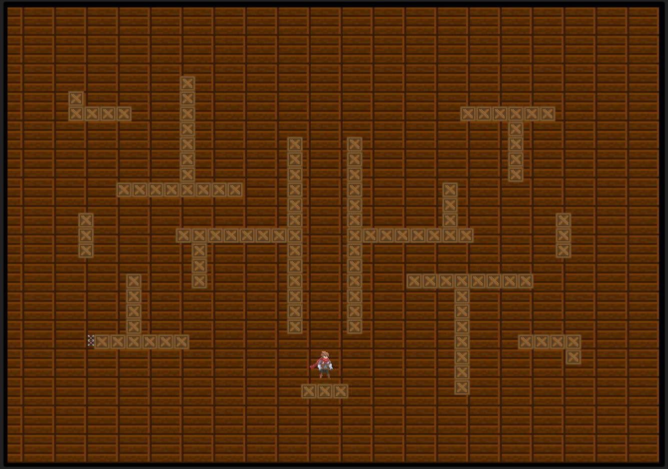
- Most power ups are inventory-able: 🔥 👟 🛡
- Others are not: ❤️ 🔑



New Room Generation and Pause Capability

Room generation has been improved, with new patterns of obstacles being added for a total of 3 unique patterns which each have the potential to be in the room.

A basic Pause Screen has also been added, allowing for the user to pause and unpause the game with the escape button and quit with ‘Q’.

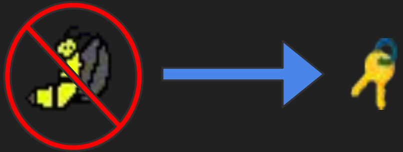


New Bullets and Door Unlocking

Fire Power:

This is a new power-up that last for 10 seconds. During this time, the player will be able to shoot at the enemies without a delay.

Drop Key:



This is a new feature that will drop a key when the last enemy in a room is killed. When the player picks up the key, a door is opened to a new room. The player must then enter through this door to move onto the next level.

New Enemies and Combat

The Ranger:

- This enemy fires projectiles at the player. We needed a way to discourage the player from being able to train our melee enemies and take pot shots so this was the answer.

The Boss:

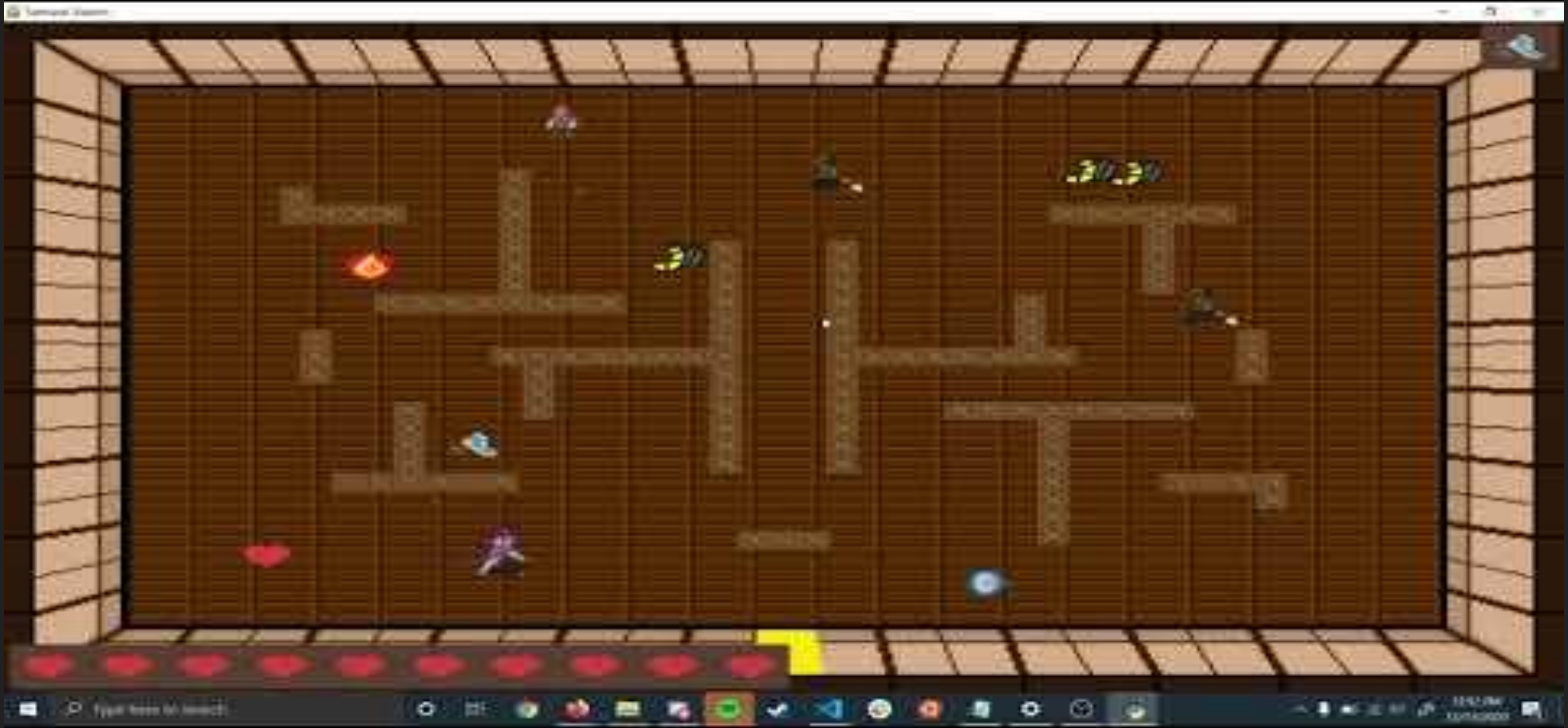
- We obviously needed a boss to pop up every now and again. We wanted to give a 1v1 feel so in these stages it's just you evading the bosses onslaught of projectiles.

Testing Strategies

Our strategy going into testing was to look at what the core of our game was an assuring that the functionality we envisioned was working properly and expected.

To do this we used two different methods:

1. For easy testing that could be automated in a mocked version of our game we used pytest.
2. For things that needed more specific and specialized testing, we developed a log file that logged values and movements that we could review after the game ran to assure we were getting the values we expected.


Type here to search