Space Expiration

CS M117: Final Project

Members:

Justin Ma

Dan Molina

Arthur Arutyunyan

Tianyue Zhang

Motivation

When we f rst began thinking of ideas for our project, it was easy for us to decide on making some sort of game. As a group, we all love playing games whether they are mobile, console, or computer. We thought about what it is about games that we love to play. The idea of playing with your friends was very important. Connecting to a game with our friends and working together on f nishing a task was fun for us all, so we decided to make a multiplayer game using WiFi connectivity.

As a group, we are all fans of the retro style video games. Our vision was a game that had the complexity as a modern-day video game but also being as visually nostalgic as some of our favorite retro video games. After we had decided on what project to take on, we began building the game on the Unity software that was recommended to us by Justin. Unity is responsible for games like Pokémon Go and Super Mario Run which we all have enjoyed playing.

About Unity

Unity was great for our purposes since it provides an easy to use interface while still allowing programmable changes in the functionality of game objects that are in the game. Unity provides drag and drop features which make it very user friendly when building our game. The Unity engine’s scripting language is C# which all of us grew familiar with in a relatively short period of time since it is very similar to languages like C and C++. Unity’s engine also provides multi platform support which made it easy for us to play the game on our iOS, Android, Mac OS and Windows devices. The library of tutorials also was very helpful in getting our networking to work and be able to achieve our goal of connecting multiple players into one game.

Functionality

Our main goal with Space Expiration was to make a game we could all enjoy as a group while still achieving the networking capabilities that were required from us. We took some of our favorite aspects from old-school, retro games like the arcade style joystick and implemented it into our game. Using a joystick to navigate through the game on a modern-day smartphone or on a keyboard on a computer was something we thought would be a really cool aspect of the game.

With Unity’s easy to use interface, we were able to build a UI that made it easy for us to connect multiple players at the same time to the same lobby. We added a special lobby manager to the game that allows one player to create a server and allows other players to connect to that server very easily as long as they are on the same WiFi connection. The Uss Revaz runs very smoothly even with three players connected all in the same gaming universe since it is optimized really well. Our carefully designed controls also allowed us to build a multiplayer game with very little to no input lag.

Wireless Technology

One of the many prebuilt features unity offers to developers is the choice for implementing multiplayer games. For more advanced games developers can create a dedicated backend for the game using the Network Transport API. Our game did not require this level of sophistication, so instead we used Unity’s premade Network manager that uses Unity’s own game servers.

We came across some problems with the controls of the game due to the multiplayer functionality. Our main problem was one player being able to control movements of other players. Using Unity’s library of tutorials, along with our experience with networking, we were able to solve this issue by writing a script that controlled which player was in control at any time throughout the game. This script blocked other players in the game from controlling anyone else's ship.

Another issue we had was connecting to the same lobby when testing the game. We were able to see the lobby created by one player but when we tried to join, we would receive a client error. After doing further research on what this error was and what caused it, we realized that it was the WiFi connection. We were testing our game on the UCLA WiFi that was giving us problems when trying to connect. To solve this problem, we all connected to a mobile hotspot and were able to connect to the same lobby and f nally into the game.

Implementation

Our game is broken up into two Unity scene components. The f rst scene, lobby, is the network manager hub. Most of the GUI used is actual premade with Unity, but we added a few cosmetic changes and enhancements that f t in with the overall gameplay. For the actual backend of the network manager we had f rst authorize network access through our unity accounts in order to enable the use Unity's multiplayer servers. After that we changed some of the basic conf gurations scripts to enable up to three players, and allow a single player to start and play their own game. The last change needed was linking the the actual game scene to the lobby manager script, so that once all players are connected the actual game could start.

The game scene is the heart of the project, where all of the actual game play takes place. The game scene starts out as a blank canvas with just the joystick in the bottom right corner. Then our game scripts create the space background and generate all the asteroids, collectables, and snake boss. Then each player is assigned a ship prefab that is mapped to there controls.

Once everything is setup the actual game begins. The boost script is responsible for taking input from the joystick and causing the ship to move. In the background all the other non environment generation scripts periodically will update, handling events such as collisions and item pickup.

Team Workf ow

The actual code for the game was split evenly throughout all four members of our group but everyone also contributed to the game in where they had the most experience. The games visuals were made by Tianyue, who created many of the design elements that are seen throughout the map. Justin brought game development experience to our group which helped us all in creating our f rst game. Dan and Arthur both worked together, with the help of Justin’s experience in Unity, to bring the controls to life, along with the networking capabilities

Game Art

A crucial part of every game is graphics. We decided on a retro pixel art aesthetic as a homage to the early arcade and computer games of asteroids and space monsters. Not content to simply borrow online graphics used in a hundred other games, we drew all pixel art assets for the game ourselves.

To give the game a unified color scheme, with the simplicity of early games with their limited color representation, we worked off a public 16-color palette designed by DawnBringer. Using such a limited color range for the darkness of space, the light of galaxies, and all the reds, greens, and blues in between required creative utilization of color theory. The function of these art assets within a game presented additional constraints: the player had to understand the roles of what they saw within gameplay.

Important objects had to appear more vibrant and notable to the player, “good” objects had to appear brighter than “bad” objects, and so on.

Therefore, the harmless but annoying asteroids were done in drab shades of gray, while the dangerous space worm was designed to be eye-catching scarlet with pops of bright white, yellow, and orange, shaded with deep violet to make it appear darker overall. In addition to color, their relative danger levels were denoted by their contours: the asteroids were rounded and lumpy, while the space worm was covered in spikes.

In contrast, the players and collectibles primarily used bright, wholesome colors, with only small accents done in gray. Shading was done with saturated colors as well, and the limited color palette encouraged us to choose them in sometimes surprising ways. For example, the brightest highlights on the green and red collectibles were actually a pastel yellow that harmonized strikingly with the other hues.

The background needed to provide visual interest while not distracting from the action from the game. Therefore, the stars, planets, and galaxies on it were very small, and used darker and more desaturated hues than one would expect. For example, all the stars were done in grays, brown, and dark purple, nothing lighter than a midtone, but look brighter due to contrast with the darker background. The only small touches of bright color in the background were in the galaxies, but we made sure that the galaxies still appeared relatively desaturated overall, to keep them from looking too overwhelming.

All in all, the art for this game required considerable application of both general design principles and considerations specif c to the project.

Demo Video Link:

https://youtu.be/lgOhS3oxDA8