Homework 4

Client

Lothar Narins

with Server by

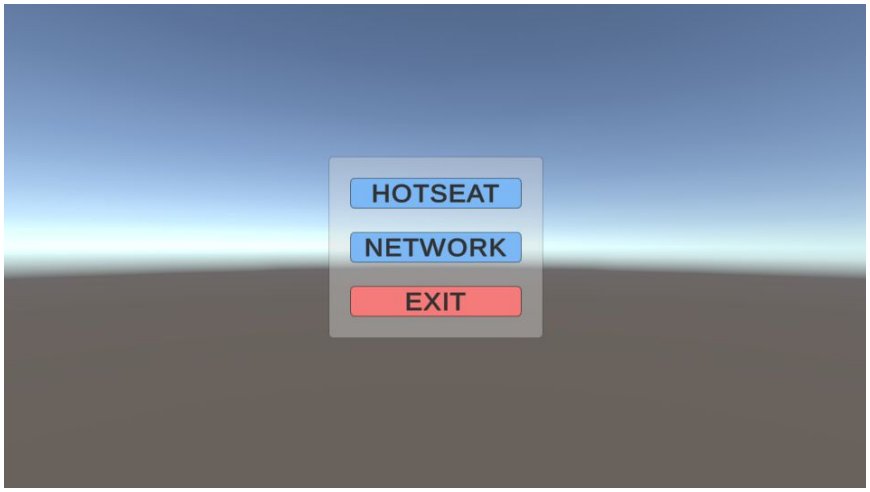
Alexander Beers

San Francisco State University
College of Science and Engineering
CSC 831 Multiplayer Game Development
Spring 2020

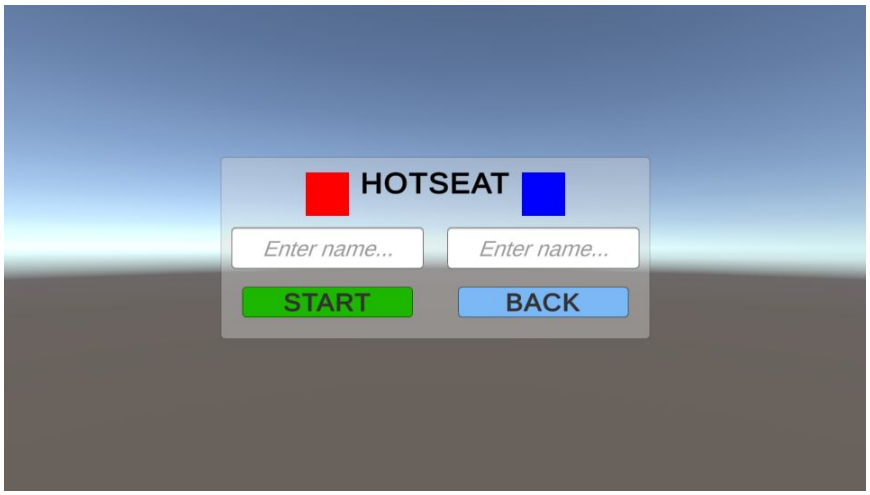
Table of Contents

The Game

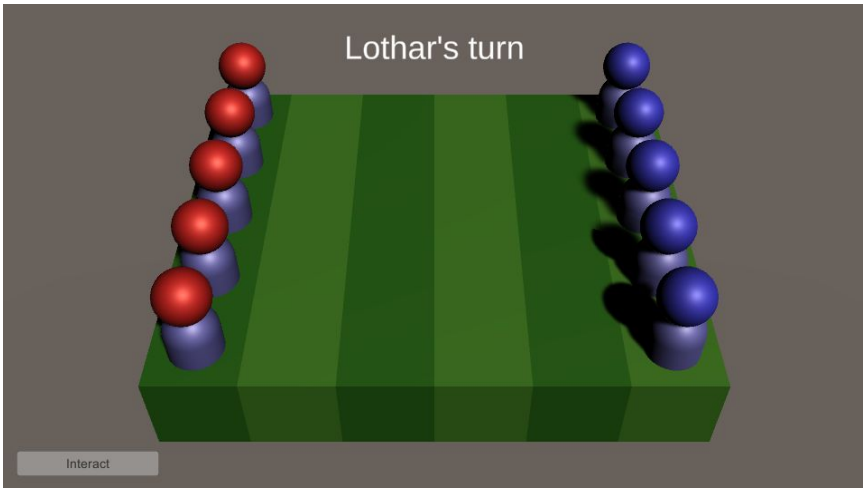
For this homework, I tried to implement a simple game that shares some of the basic features of the Super Dungeon Game, such as being turn-based, moving on a grid, and interacting with enemies. The game starts on a main menu screen seen here:



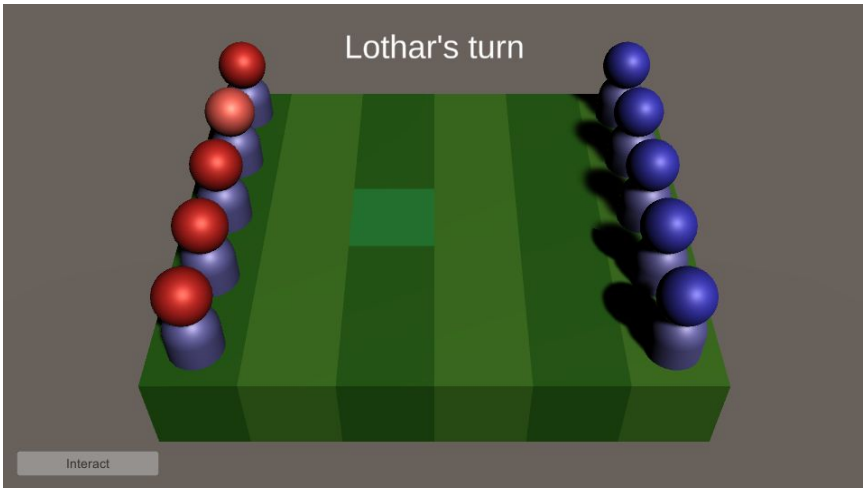
The player is presented with the possibility to do “hotseat” play, which does not require a network and where both players are controlled from within a single client. The other option (besides exiting the game) is to join network play, which supports exactly two people connected to a server, each with their own client. Clicking on hotseat brings one to the hotseat lobby, where the two players on a single client are prompted to enter their names:



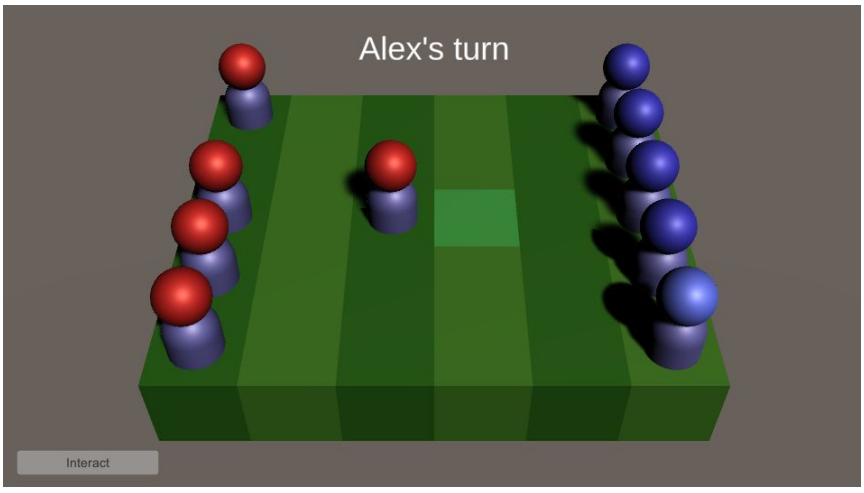
Upon entering names and pressing start, the game begins with player 1, the red player, taking a turn:



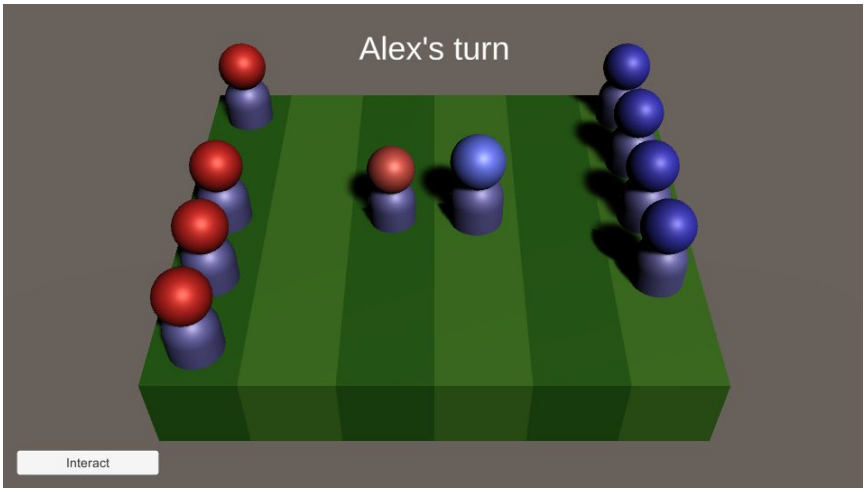
After selecting one of their pieces, the player can move it to any empty square by clicking on it:



At which point it becomes the other player's turn, who can do the same:



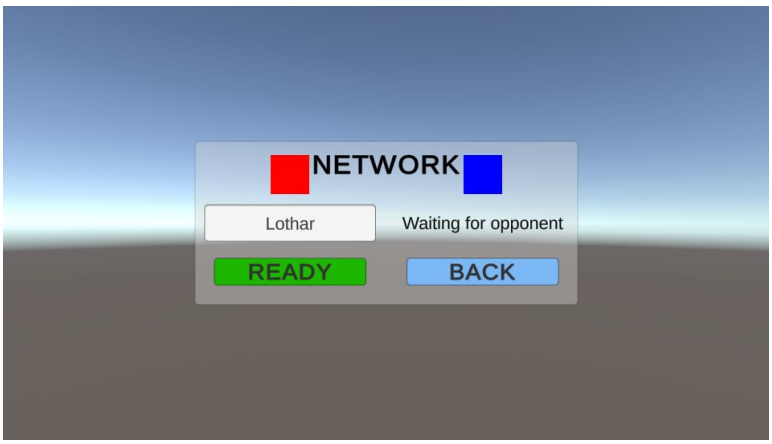
If a player moves a piece next to an opponent's piece, they must click Interact in the bottom left corner, then on an opposing neighbor:



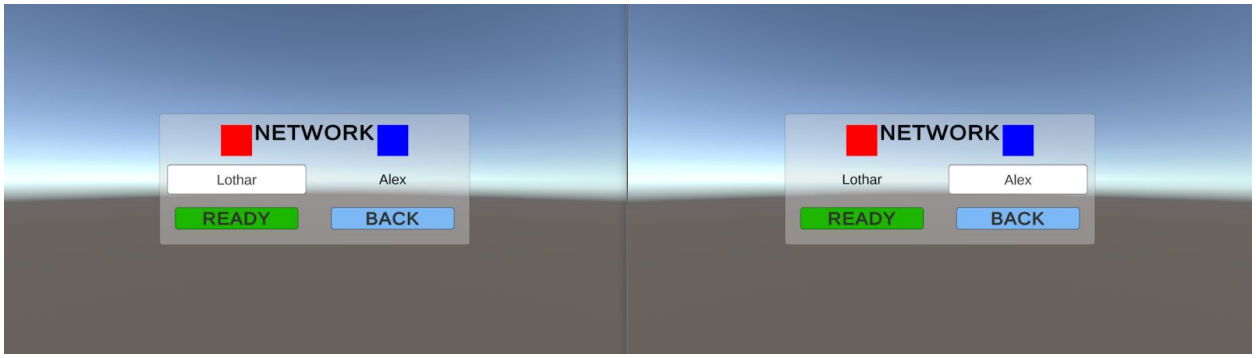
This plays a little animation where the active player's piece grows, while the opponent's piece shrinks. After the interaction, it becomes the other player's turn. The game does not yet have a goal and doesn't end. It's just a demonstration for now.

If the server can't be reached by the game, pressing network play from the main menu yields an error message:

Otherwise, the first player to join becomes player 1 and can enter their name:



Once another player joins, they can enter their name as well, and both clients can see the other player's name:



Once both players click ready, the game starts, and proceeds as in the hotseat version, except each client obviously only controls their own actions, and must wait for the opponent to take their turn.

In order to do network play, the server must be started before any of the clients are started, and must be restarted whenever you want to start a new game.

Protocols

The client and server communicate via the following protocols:

Join: When a player presses the network button on the main menu, the client sends a Join message to the server. The server checks which player slots are taken (player 1 or player 2) and assigns the first free slot to the joining player, returning a status code of 0 if a slot was available, and a status code of 1 if the server was full. If the slot was available, the server also sends the player's slot id (1 or 2), the opponent's slot id (or 0 if no opponent has yet joined), and the opponent's name and ready status. It also sends a SetName message to the other player, if there is one, so as to inform the other player that you have joined.

Leave: When a player presses the back button from the network lobby, they return to the main menu, and the client sends a Leave request to the server. The server sends both players a Leave response with the slot of the player who issued the request. This sets the ready status of the leaving player to false.

SetName: When a player finishes entering their name, the client sends a SetName request to the server containing the string with their name, and the server echoes a SetName response to both players, which contains the id (slot) of the player who issued the request and their new name. As mentioned above, such a response is also sent to the other player when a player issues a Join request.

Ready: When a player presses the ready button in the network lobby, they send a Ready request to the server, which records their ready status and echoes a Ready response to both players that contains the id of the player who issued the request. When both players have clicked ready and received their

responses, the game starts. Note that if a player leaves the lobby, their ready status is set to unready, even if they had clicked ready.

Move: When a player moves one of their pieces during the game, the client sends to the server a Move request containing the number of the piece that they moved and the x and y coordinates (as integers) of the target square. The server echoes a response containing the issuing player's id and that same information to both players.

Interact: When a player interacts with an opponent's piece during the game, the client sends to the server an Interact request containing the number of the piece they used and the number of the opponent's piece they interacted with. The server echoes a response containing the issuing player's id and that same information to both players.

Heartbeat: Every 0.1 seconds, the client sends a Heartbeat request to the server, and the server sends back all the responses it had queued for that player.

The classes defining these protocols in the client are located in the Scripts/Network/Request and Scripts/Network/Response folders. The callbacks that handle the responses are defined in Scripts/UI/MainMenu.cs for Join, Leave, SetName, and Ready, and in Scripts/GameManager.cs for Move and Interact.

Reflection

This project took a lot more time and effort than I had anticipated, and initially I was very overwhelmed with the scope of what I took upon myself to do. I had a lot of fun making this, though! Alex and I had several long Zoom sessions where we talked through the code, and I explained what I needed all the protocols to do. I felt that this project really made me understand how to communicate between game clients across a network through a server. It also was fun to figure out from scratch how to implement highlighting of tiles and selecting and moving characters on a grid, since that was all done for us by the previous group in the Super Dungeon Game.