Contents

[Psuedocode 1](#_Toc348790763)

[Player Mode 1](#_Toc348790764)

[Builder Mode 2](#_Toc348790765)

[Pregame Menus 3](#_Toc348790766)

[Control Manager 3](#_Toc348790767)

[Exit Game Menu 3](#_Toc348790768)

[Task Assignment 4](#_Toc348790769)

[John 4](#_Toc348790770)

[Jesse W. 4](#_Toc348790771)

[Luke 4](#_Toc348790772)

[Tyler 4](#_Toc348790773)

[Interfaces 5](#_Toc348790774)

[Graphics 5](#_Toc348790775)

[Networking 5](#_Toc348790776)

[AI 5](#_Toc348790777)

[Game Logic 5](#_Toc348790778)

[Flow Charts 6](#_Toc348790779)

# Psuedocode

## Player Mode

Move Player

{

Check passed in value

If value is “left”, move left

Call Engine function for moving left

If value is “right”, move right

Call Engine function for moving right

If value is “up”, move up

Call Engine function for moving up

If value is “down”, move down

Call Engine function for moving down

}

Attack with Player

{

Check passed in value

If value is “left”, attack left

Call Engine function for attack left

If value is “right”, attack right

Call Engine function for attack right

If value is “up”, attack up

Call Engine function for attack up

If value is “down”, attack down

Call Engine function for attack down

}

## Builder Mode

 Build Tower:

When the player clicks on a button to build a tower, they will have control to then click on the map in any valid position and have a tower built there. We will call a function from game logic that will essentially build the tower, or not build the tower if it is in an invalid position.

We will be using SFTML for the graphics so we will be able to get the position of the mouse on the screen and then convert it to coordinates that will work with the in game map. This will also allow us to create the buttons on the HUD and simply give them their own ID’s so they can be monitored easily.

Spawn Creep:

When the player indicates that they want to spawn a creep, by pressing on the particular creep’s button, they will also have to select which lane they want the creep to travel on. This will essentially be done the same as building a tower. We will have a function to call from game logic and it will validate against the players resources to make sure they can afford the particular unit. Again, we will use the buttons in SFTML and just monitor these so we know which creep is being spawned.

## Pregame Menus

Get Button Input

{

Check input against list of buttons

If team select, report to engine and return

If start game report to engine and return

Report to manager to swap to flagged game mode

If mode select report to engine and return

Mark new game mode as flagged

}

## Control Manager

Context Changer

{

load input class

add menu, builder, and player context objects and have an input function associated with each of them

variable current\_context set to player context by default

On Click menu context

current\_context swapped to menu context

return value 10

On click builder context

current\_context swapped to builder context

return value 20

On click player context

current\_context swapped to player context

return value 30

}

## Exit Game Menu

Generic Menu Function

{

load menu class

add item "Return to lobby" with Return to lobby function

add item "Exit Game" with a exit function associated with it

add item "Resume Game" that closes the menu

make a thread or call a listener that pretty much listens in when the user clicks on a button

if the user clicks on any of the buttons

call the function that associated with the button

}

Return to lobby function

{

pop up message box asking the user, "Are you sure you want to return to lobby?"

if user clicks "yes"

exit out of the game and change to lobby menu context

if user clicks "no"

close message box

}

Exit Game function

{

pop up message box asking the user, "Are you sure you want to exit the game?"

if user clicks "yes"

exit game program

is user clicks "no"

close message box

}

# Task Assignment

## John

* Player Mode

## Jesse W.

* Builder Mode

## Luke

* Manager
* Exit Game Menu

## Tyler

* Pregame Menus

# Interfaces

## Graphics

* A shared header defining RECTs intended to mark onscreen button locations
* A checker function to notify when an onscreen button has been pressed

## Networking

* N/A

## AI

* N/A

## Game Logic

* Function for moving Player character
* Function for attacking with player character
* Function for building a minion
* Function for building a tower

# Flow Charts

