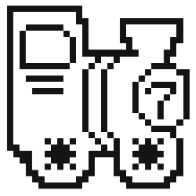
****

****

**Team 3 Design Document**

**Team members:** Ellison Hohmann, Rei Manning, Ryan Parris, John Slater, Danny Spear

**Project Title:** The Purdue Party Game

**Index:**

[**Purpose**](#_ljwe35eyd522) **3**

[**Functional Requirements**](#_ugqvws22hixn) 3

[**Non-Functional Requirements**](#_e72j5cjr1v2h) 6

[**Design Outline**](#_4sa0ktxhqppg) **8**

[**High-level Overview**](#_999zdfgz0lbq) 9

[**Sequence of Events Overview**](#_iwrcwltnnhfa) 10

[**Design Issues**](#_7uotnsqcgqtb) **11**

[**Functional Issues**](#_8rju75ia9rlf) 11

[**Non-Functional Issues**](#_ovtfpo2nmhyo) 12

[**Design Details**](#_d49j833s47m) **15**

[**Class Diagrams and Descriptions of Classes**](#_wx7ssucg5o5x) 15

[**Sequence Diagrams**](#_7xyx5muy0zca) 32

[**Navigation Flow Map**](#_dx6wjljzbkpd) 34

**State Diagram**36

[**UI Mockup**](#_bj2bfofl3hle) 37

# 

# 

# Purpose:

Video games provide a great opportunity to connect with others, have fun, and destress. Players are able to interact with new, creative worlds with unique challenges and interesting mechanics, creating an enjoyable and immersive experience. Party games, designed to be played by multiple players, are a great way to foster connections through friendly competition. Party games are unique in their ability to bridge the gap between the real and virtual worlds.

Our goal is to design an offline, 2D party game based on the Purdue University campus and culture. Targeted mostly towards students, fans, and alumni, The Purdue Party Game will utilize the familiar sights of Purdue to create an immersive world that further connects the players. Played on PC with up to four players, users can use wired controllers to traverse a virtual game board and play fun minigames. Players can earn ‘Boiler Bucks,’ a currency which can be exchanged for ‘Degrees’ that are used to determine the winner of the game.

In addition to in-person players, we will also have CPU players to fill in any missing spots. CPUs will be able to move around the board, play minigames, and earn Boiler Bucks and Degrees just like a normal player. These CPUs will have a variable difficulty setting which determines their skill in minigames, offering options for a more or less challenging experience depending on the users’ preferences.

## Functional Requirements:

Start Navigation

1. As a player, I would like the game to start with a title screen, with buttons for creating a new game, resuming an unfinished game (if applicable - if not, button grayed out), viewing game options and settings, exiting the game
2. As a player, I would like to be able to hear music and sound effects throughout the game and I would like to be able to adjust music volume and sound effect volume in the settings menu
3. As a player, I would like to view a ‘start game menu’ after selecting the ‘new game’ option from the title screen, with options to select the number of rounds in the game, select the number of human players, and to assign a character from a roster of 8 options to all players (human and CPU)

Main Game Board Function

1. As a player, I would like for me and CPUs to be able to roll die/dice in order to progress on the board
2. As a player, I would like for me and CPUs to be able to move around the board, choosing where I/the CPU would want to go
3. As a player, I would like to have the option to view a zoomed-out version of the map (as opposed to the default local view) to see the entire board before my turn.
4. As a player, I would like to be able to view how many Boiler Bucks and Degrees each player has at any time
5. As a player, I would like to be able to save and exit my game between turns so that I am able to return at a later time or exit my game without saving between turns if I do not wish to return at a later time

Cross-Minigame Function

1. As a player, I would like a minigame to be randomly chosen after each round of player turns and for an instructions menu for said minigame to appear so that I know how to play before the game begins.
2. As a player, I would like to be able to choose to play a demo version of a minigame, which does not count for real points, from the minigame instruction menu where after completion, it will take me back to the instruction page and I would like to be able to enter the actual minigame from the minigame instruction menu
3. As a player, I would like to see a screen after each minigame that displays the winner/placements and reward(s) and allows a return back to the game board

Minigame 1: Boilermaker Special Race

1. As a player, I would like to have a train that moves across the screen based on the frequency that I alternate between pushing 2 buttons
2. As a player, I would like “gear-shifts” to occur at random intervals throughout the race that change which 2 buttons need to be alternated between in order to move
3. As a player, I would like for the CPU to move across the screen based on a simulated button frequency that is created using random time intervals, scaled to the level of difficulty of the CPU
4. As a player, I would like the first, second, third, and fourth places of the minigame to be decided based on the order in which players cross the finish line where coins are awarded based on placement

Minigame 2: 11:59

1. As a player, I would like to see a clock on-screen that starts at 11:59:45 and counts up to 11:59:50 before going blank
2. As a player, I would like to be able to push a button to “submit assignment” when I feel I am as close to 11:59:59 as possible without submitting past 12:00:00
3. As a player, I would like the CPU to choose to “submit assignment” at a time randomly generated, but scaled based on difficulty level of the CPU
4. As a player, I would like the first, second, third, and fourth places of the minigame to be decided based on who got closest to the time 11:59:59 but does not go over, with those going over being placed in the latter places ranked on who went the least over where coins are awarded based on placement

Minigame 3: State Street Scramble

1. As a player, I would like my character to begin at one side of University Street, with 10 lanes of traffic (bikes, skateboards, cars) in which vehicles move across the lane at constant speed at different set intervals
2. As a player, I would like to be able to move forward, backward, left, or right in order to move from one side of the road to the other while avoiding vehicles
3. As a player, I would like for the CPU to be able to choose to move forward, backward, left, or right based on the position of vehicles on the road, with non-optimal moves made at a frequency corresponding with the difficulty of the CPU
4. As a player, I would like collision with any vehicle to result in my character or the CPU re-spawning at the beginning point
5. As a player, I would like for the first, second, third, and fourth places of the minigame to be decided based on the order in which characters reach the opposite side of the street where coins are awarded based on placement

Minigame 4: Tipsy Tunnel Trek

1. As a player, I would like to be able to play a maze minigame based on navigating Purdue’s underground tunnels where the maze is randomly selected from a pool of completable mazes and all players start at the same spot within the maze
2. As a player, I would like to be able to move freely throughout the maze, viewing only a local chunk of the entire maze at any given time
3. As a player, I would like for my controls to be altered/inverted at random time intervals throughout the duration of the minigame.
4. As a player, I would like for the CPU to be able to choose a path through the maze using a graph-based traversal algorithm with “incorrect” path choices taken at a frequency corresponding to the difficulty of the CPU
5. As a player, I would like the first, second, third, and fourth places of the minigame to be decided based on who navigates to the exit of the maze the fastest where coins are awarded based on placement

Minigame 5: Time’s Up

1. As a player, I would like to see my character climbing the bell tower at a set pace, starting on the left side, as well as be able to use the joystick to switch my character from one side of the tower to the other in order to avoid obstacles
2. As a player, I would like clocks to randomly generate at increasingly-frequent intervals on either the right or left side of the bell tower that fall towards me at a set speed. If a clock collides with my character, I would like to fall slightly down the tower before my character begins climbing again
3. As a player, I would like the CPU to be able to switch from one side of the tower to the other to avoid obstacles with a certain probability of avoiding each obstacle, with this probability increasing with higher levels of CPU difficulty
4. As a player, I would like the first, second, third, and fourth places of the minigame to be decided based on who gets to the top of the bell tower first where coins are awarded based on placement

Special Board Squares

1. As a player, I would like for me and CPUs to be able to exchange Boiler Bucks for Degrees when passing “graduation” squares
2. As a player, I would like to be able to land on special “trivia” squares and have the chance to answer a Purdue-themed or map-location-based trivia question randomly selected from a pool of questions for Boiler Bucks
3. As a player, I would like for the CPU to be able to land on special “trivia” squares and have the chance to answer a Purdue-themed or map-location-based trivia question for Boiler Bucks, with the CPU choosing an answer randomly with frequency of a correct answer corresponding to CPU difficulty
4. As a player, I would like for I or the CPU to be able to land on “scholarship” squares and gain Boiler Bucks or land on a “student loan payment” square and lose Boiler Bucks

Store and Items

1. As a player, I would like to be able to view a store with items that I am able to purchase with Boiler Bucks during my turn
2. As a player, I would like to be able to purchase and use an item that slows down the roll of the dice so that I am more easily able to choose when to stop the dice in order to get whatever number I desire
3. As a player, I would like to be able to purchase and use an item that allows me to swap places on the board with another player of my choosing
4. As a player, I would like to be able to purchase and use an item that allows me to roll a second die in order to increase the number of spaces that I can move.

End of Game Functions

1. As a player, I would like the game to end once the specified number of turns for each player is done
2. As a player, I would like bonus Degrees to be awarded at the end of the game based on most minigames won, most Boiler Bucks, most trivia questions gotten correct, and/or most “student loan payment” squares landed on
3. As a player, I would like an end screen to display the ranking of the players based first on the number of degrees then secondarily, in the case of a tie, on the number of Boiler Bucks
4. As a player, I would like for there to be transitions between screens to improve the visual flow of the game

## Non-Functional Requirements:

Usability

As a player, I would like for the game to be intuitive and easy to navigate. New users should be able to learn basic controls within 5 minutes of first use.

Video games should be a fun, easy, and relaxing experience. If the user interface of the game is not intuitive or requires the player to spend time trying to figure out how to play the game instead of actually playing, it is unlikely that the player will want to continue the game. Because of this, it is important that our game be easy to navigate, even for casual gamers and new players.

Performance

As a player, I would like my loading times between screens to be under 8 seconds, and for the game to run optimally at 30 frames per second.

Performance is an important aspect of video gaming. In order to keep the player immersed in the gameplay, it is important that loading times between screens (e.g., between game options and main board, between main board and minigame menu) be minimal. This can be controlled by ensuring that only necessary assets are loaded for each screen and that there are no time-intensive algorithms or loops in the creation of each screen.

Scalability

As a player/players, I would like the game to be scalable from 1 to 4 human players, with no negative effect on frame rate and response time.

Party games like ours work best when there are four players. It is important to note, however, that not all players will be in groups of exactly four people wanting to play our game. In order to expand the base of players, our game will be able to support from 1 to 4 human players, with the remaining players being controlled by the computer. This makes our game more scalable to different groups of players.

Relatability

As a player/players, I would like for the game to be reminiscent of my experiences with Purdue culture

While not exclusive to Purdue students and alumni, these groups are ones that are in our intended audience. Because of this, we want the games, trivia, map, and references in our game to be relatable to current or previous students or fans of the university**.**

# 

# 

# Design Outline:

Most functionality of our project takes place within the GameMaker framework and environment. Objects, event detection, functions, GUI updates, and during-execution game data storage can all be implemented within the GameMaker Application component.

In order to obtain user input, 1-4 Player Controller(s) are needed. These serve as devices that take in physical input from the player(s) and send that information into the GameMaker Application. With this information received from Player Controller(s), the GameMaker Application handles most of the system calculations and functions.

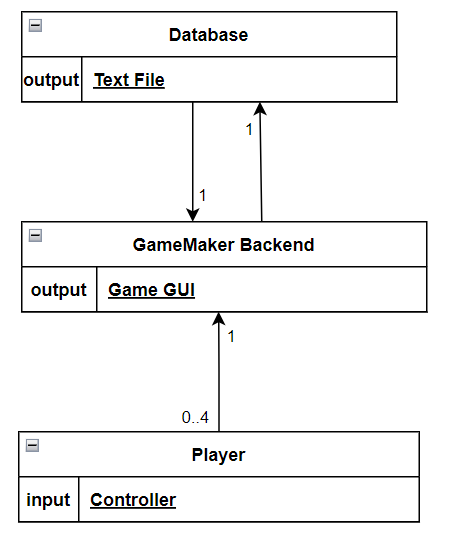
In order to locally save game data between executions of the program, a Database is needed. The conversion of game data into an encrypted file is supported within GameMaker. Whenever requested, the Database will convert and store current game data and statistics into an encrypted file format, storing said file locally on the device on which the application is running. Additionally, the Database will access this local file and send it back to the GameMaker Application when requested in order to restore the application to the state of the saved game.

1. Player Controller
   * Takes input from player’s actions
   * Requires one analog stick and four buttons for all user input
   * Communicates player actions to GameMaker Application
   * 1-4 player controllers supported
2. GameMaker Application
   * Receives data on player actions from Player Controller(s)
   * Receives data on previously saved game from Database when requested
   * Converts player actions to associated character movements/behavior
   * Algorithmically calculates CPU movements/behavior
   * Updates GUI based on player/CPU behavior
   * Displays graphics of game on monitor
   * Keeps track of player/CPU/game information and statistics
   * Sends data on game to be saved to Database when needed
3. Database
   * Outputs encrypted file
   * Receives data on game to be saved from GameMaker Application
   * Converts player/game information and statistics into encrypted file format
   * Stores file locally to maintain game state between executions of main application
   * Converts encrypted file into player/game information and statistics
   * Sends previously saved game data to GameMaker Application when requested

## High-level Overview:

The main component of our system is the GameMaker Application, which receives player action input from 1-4 Player Controllers.

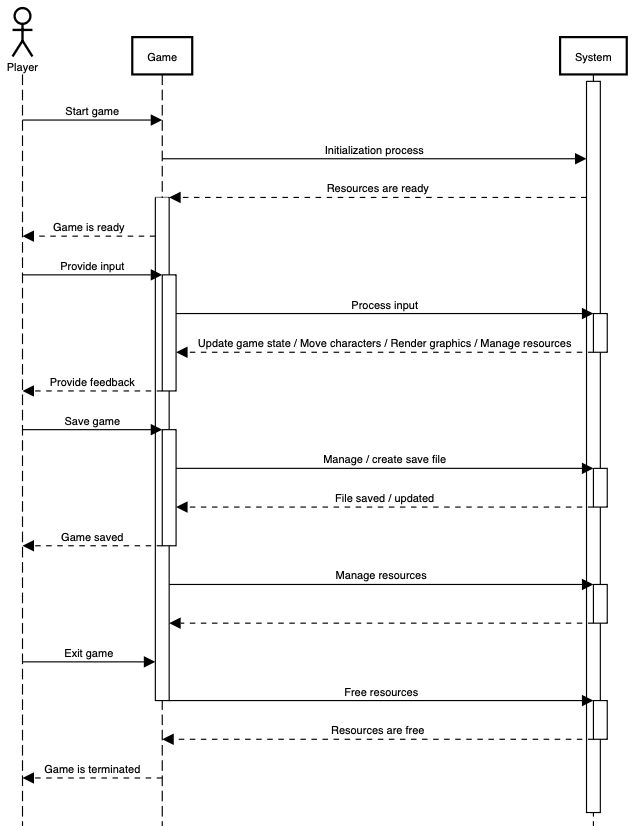
The GameMaker Application also sends data to and receives data from the local Database in order to support local saving of the game.



## 

## Sequence of Events Overview:

The sequence diagram below shows the typical interaction between the player, the game, and the computer it runs on. The sequence is initiated by a user starting the game. After starting the game, the player gives the video game input in multiple forms, such as pressing a button, selecting a specific option, etc. The game code then interprets the input, and sends it to the system if computer resources (memory, storage, etc.) or graphics need to be modified. The game will return the system’s results and its own results to the player. The game will also be sending the requests to the system if it needs more resources allocated for its own processes, and the system will respond accordingly. When the player saves the game, the game triggers the system to create a new file in the game’s filesystem in which to save game data. After the file is created (or modified if it already exists), the system gives a response (positive or negative depending on whether or not there was an error) to the game, which forwards it to the player.



# Design Issues:

## Functional Issues:

1. When do we want players to be able to interact with the game (i.e. when will player controller input be valid)?
   * Option 1: Only during their turn and during minigames, with the first player being the only one to also access title screen, settings, instruction pages
   * Option 2: Any player can interact at anytime
   * Option 3: Any player can interact at any time except when it is not their turn in the game

Choice: Option 1

Justification: The team decided option 1 was the best option to mitigate confusion and accidental selections in the title screen, settings, and instructions pages. Along with that, it enhances gameplay and allows the players to better focus on their turn without other players interrupting. We think having player one being the main person to decide the settings is the best choice so players can decide quickly and start their game as soon as possible.

1. When do we want each player’s current “Boiler Bucks” and “Degrees” to be available to view?
   * Option 1: A player is able to toggle during their turn to view all player’s scores
   * Option 2: After every minigame the score’s can be displayed for every player to see
   * Option 3: We can show each player’s respective “Boiler Bucks” and “Degrees” during the player’s turn

Choice: Options 1 and 2

Justification: The team decided option 1 and 2 were the best to integrate into our game so players were always aware at the start of each round where they stood with their opponents. We also thought it would be beneficial for each player during their turn to be able to view the leaderboard in the event that it influenced what their next steps were in the game and they could not remember where players were after the minigame. We thought these two were the best and that including option 3 would not be the most beneficial way to display the information compared to the two prior options listed.

1. How should the leaderboard be displayed?
   * Option 1: Have the players be sorted based on their current placements, based on who has the most “Degrees” first and who has the most “Boiler Bucks” second
   * Option 2: Have the players remain in the same constant location but mark who is first, second, third, and fourth

Choice: Option 1

Justification: The team decided option 1 was the best way to have the leaderboard be displayed because the player’s would be able to more quickly analyze where they stood compared to their opponents. Although option 2 would make it easier for an individual to find their “Boiler Bucks” and “Degrees”, it would be difficult for them to calculate what place they were in. We felt calculating it for the players and having them look for their sprite would be easier for them than them looking at the same spot but needing to calculate where they stood.

1. How should we handle ties in minigames?
   * Option 1: Have those who are tied share the placement they are in
   * Option 2: Have them roll a die to break a tie
   * Option 3: Have them play a minigame to break a tie

Choice: Option 1

Justification: The team came to the agreement of allowing people who do tie to share the placement they are in. We felt option 2 devalued what was done for them to result in last by having the results come down to luck in a tie. We felt option 3 would make the game take much longer than we felt was ideal. Option 1 maintains the integrity of the work done by players and it allows us to keep the game within a better length from start to finish.

1. What screen ratio do we want our game to be at
   * Option 1: 4:3
   * Option 2: 16:9

Choice: Option 2

Justification: The team decided on option 2 so we can utilize more of the screen for viewing the map and minigames. Although we heavily considered option 1 because it closely fit the aesthetic of a retro game, we found the space we would be missing out on too important. With having to share the screen with 4 players, maximizing how much of the screen is visible gameplay is crucial in creating a better experience for the players.

## Non-Functional Issues:

1. Where do we want to keep the game’s save file?
   * Option 1: Have it save to a server we host
   * Option 2: Have it save locally with the other game files in a standard text file
   * Option 3: Have it save locally with the other game files in a encrypted .sav file

Choice: Option 3

Justification: The team decided the best and most efficient decision was to implement option 3. We found option 1 to be unnecessary as it would require all systems who play our game to need an internet connection while playing. When we have the ability to make our game completely local to the system, we found saving the game locally was the best. Our team also believes that the integrity of the game is important so preventing the save file from being changed was what made us decide option 3 was the best for what we want our game to be.

1. How many frames per second should our game run?
   * Option 1: 30 frames per second
   * Option 2: 60 frames per second

Choice: Option 1

Justification: The team decided option 1 was the better option to fit the overall look and aesthetic of the game. The game is supposed to fit a 2D retro game aesthetic and feel for the player and after researching, doing 30 frames per second was the better choice. Although 60 frames per second is said to provide a “smoother” form of gameplay, it was unnecessary for the types of minigames in which the players would be competing against each other.

1. How many players should we support for our game?
   * Option 1: A maximum of 2
   * Option 2: A maximum of 4
   * Option 3: A maximum of 8

Choice: Option 2

Justification: The team decided supporting 4 players was the most ideal for the type of game we’re designing. Because we are running on computers, it would be unlikely for the user to have access to 8 USB ports, making a max of 8 players really unlikely. Because of that, we believe supporting that wouldn’t be the best use of our resources. We also decided against option 1 due to the lack of competition that occurs if one player is ahead of the second. By allowing 4, it is convenient for the user and allows a competitive and fun environment for the game.

1. What engine should we use for game development?
   * Option 1: GameMaker (GML)
   * Option 2: Unity (C#)

Choice: Option 1

Justification: The team decided to utilize the GameMaker engine for our game’s development. We decided on GameMaker due to its ease in building and designing 2D games. The team also had more experience with GameMaker compared to Unity and because of our comfortability, GameMaker would allow us to make more creative development decisions for our game.

1. How should our naming conventions be for the different coding events as well as sprites, objects, and other built in to GameMaker’s program
   * Option 1: For built in items in GameMaker, we follow AbbreviationForItem\_NameOfItem and we can have variables in coding events fit Pascalcase formatting. An example for an item would be SP\_Player where SP stands for sprite and an example in a coding event could be CharSpeed.
   * Option 2: We could utilize Snakecase for both items in gamemaker and coding events. An example for an item would be sprite\_player and an example for a variable in an event would be char\_speed
   * Option 3: We could utilize Camelcase for both items in gamemaker and coding events. An example for an item would be spritePlayer and an example for a variable in an event would be charSpeed

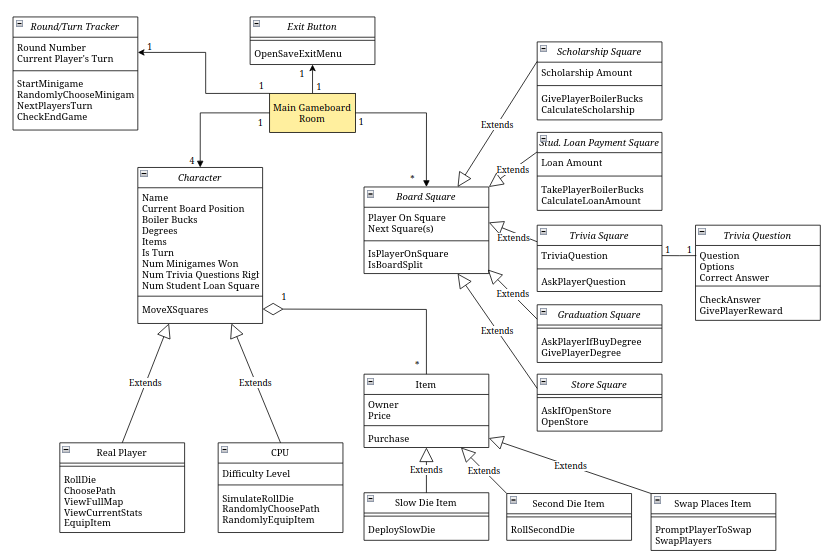
Justification: The team decided option 1 would be the best for our game as it would help keep track of the difference between items and variables contained within the different event functions. Option 2 and 3 are both good options but for the purposes of keeping items built into GameMaker and the functions and variables we build ourselves separate, option 1 would be the best fit.

**Design Details:**

## Class Diagrams:

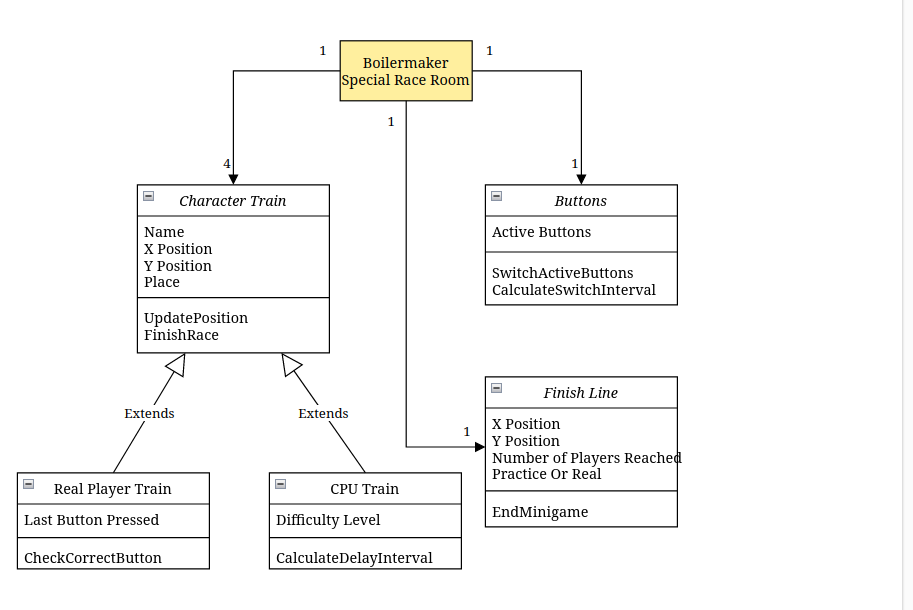
As our system is written in GameMaker Language, classes are a little different than normal object-oriented programming languages like Java. “Classes” in GameMaker Language are Objects, each of which has associated variables and methods. These Objects each belong to a certain Room in GameMaker, meaning that each of our “rooms”/screens have their own set of Objects and associations between those objects. Because of this, we have created a series of modified UML Class Diagrams where each diagram represents the class relationships for one Room. These Rooms are labeled in the diagrams as yellow-shaded boxes.

**Main Gameboard Room:**



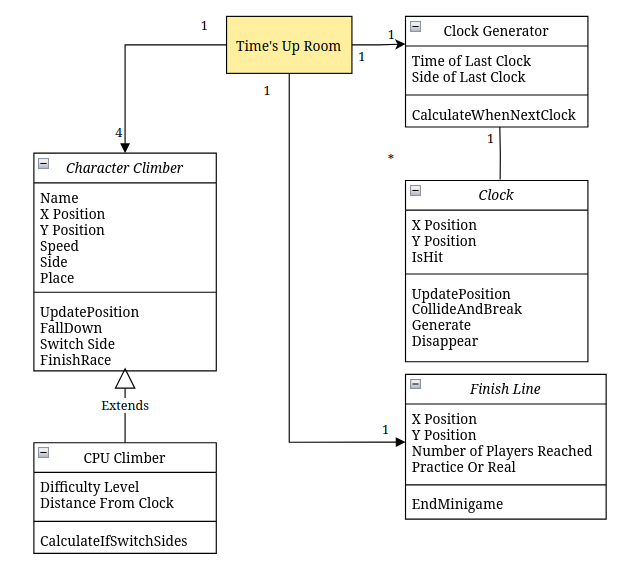
* **Main Board**
  + Round/Turn Tracker
* Keeps track of current player’s turn and when round ends
* Generates random minigame at the end of each round
* Keeps track of number of rounds and when the game should end
  + Character
* Holds character’s stats including name, Degrees, Boiler Bucks, items, and statistics for Bonus Degrees
* Stores and updates character’s position on game board
  + Real Player
* Child of Character, allows human players to roll die, choose to view map, view score, or use items via controller input
* Allows human players to choose direction on board when there is an intersection
  + CPU
* Child of Character, allows CPU to roll die or use items randomly
* Allows CPU to randomly choose direction on board when there is an intersection
  + Item
    - Related to Character, can be equipped to grant the Character special abilities
    - Can be purchased in the shop using Boiler Bucks
  + Slow Die Item
    - Child of Item, causes the die to change numbers slowly, allowing the Character to choose the number of spaces they move
  + Second Die Item
    - Child of Item, grants the Character a second die to roll with
    - The amount of spaces the Character moves is the combined total of the dice
  + Swap Places Item
    - Child of Item, allows the Character to swap places with another Character of their choosing
  + Board Square
    - Places where the players can land
    - Keeps track of a player’s existence on the square and the square(s) that come after it
  + Scholarship Square
    - Child of Board Square, gives a player Boiler Bucks when they land on it
    - Calculates the amount of Boiler Bucks depending on certain conditions
  + Student Loan Payment Square
    - Child of Board Square, takes away a player’s Boiler Bucks when they land on it
    - Calculates the amount of Boiler Bucks depending on certain conditions
  + Trivia Square
    - Child of Board Square, asks the player to answer a Trivia Question
  + Trivia Question
    - Child of Trivia Square, contains data for a multiple choice trivia questions
    - Gives the player a reward if answered correctly
  + Graduation Square
    - Child of Board Square, prompts the player to buy a degree
    - Gives the player a degree if purchased
  + Store Square
    - Child of Board Square, asks the player if they would like to go to the Store
    - Brings up the Store menu if they select yes
  + Exit Button
    - Opens the Save and Exit Menu

**Boilermaker Express Race Room:**



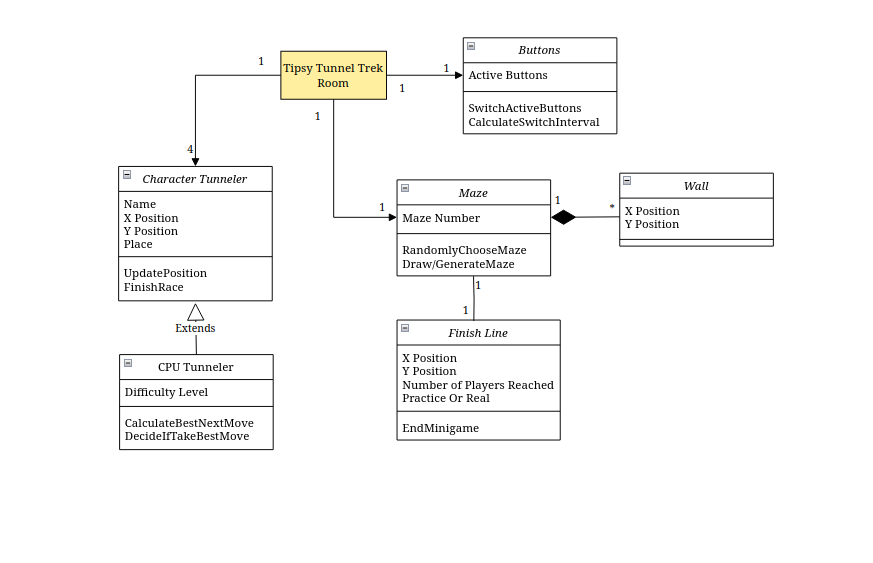
* **Boilermaker Express Race**
  + Character Train
    - Tracks the name of the Character, the x and y positions, and the ranking (1st, 2nd, 3rd, 4th)
    - Updates the position as needed and finishes the race when crossing the finish line
  + Real Player Train
    - Child of Character Train, tracks the last button pressed by a human player, and checks to see if a button press is correct.
  + CPU Train
    - Child of Character Train, keeps track of the CPU difficulty and calculates how fast the CPU should press the buttons.
  + Finish Line
    - Tracks the position of the finish line, the number of players that have crossed the finish line, and whether or not the game is real or a demo.
    - Ends the minigame when enough players have reached the end
  + Buttons
    - Keeps track of the buttons currently in use
    - Switches which buttons are being used based on a calculated interval

**Time’s Up Room:**



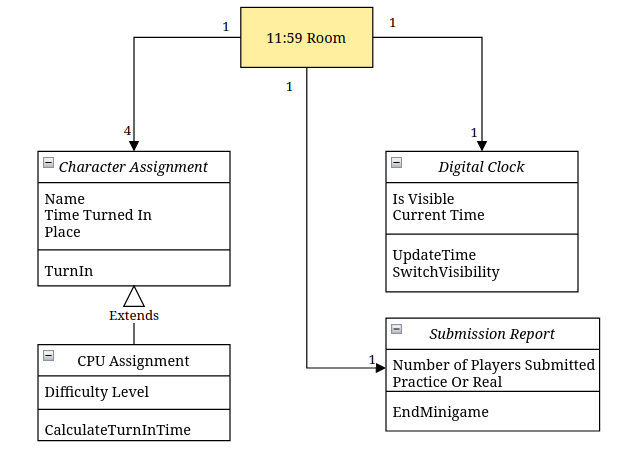
* **Time’s Up**
  + Character Climber
    - Keeps track of Character data, including name, position, speed, side of the tower, and place (1st, 2nd, 3rd, 4th)
    - Updates the player position and side as needed
    - Finishes race when reaching the finish line
  + CPU Climber
    - Child of Character Climber, keeps track of CPU difficulty and distance from nearest clock
    - Calculates when to switch sides
  + Finish Line
    - Tracks the position of the finish line, the number of players that have crossed the finish line, and whether or not the game is real or a demo.
    - Ends the minigame when enough players have reached the end
  + Clock Generator
    - Keeps track of when the last clock was dropped and the side it was dropped on
    - Calculates when to drop the next clock
  + Clock
    - Keeps track of the position of the clock and if it has been hit by the player
    - Generates and removes clocks as needed
    - Updates the clock position and breaks it if hit by a player

**Tipsy Tunnel Trek Room:**



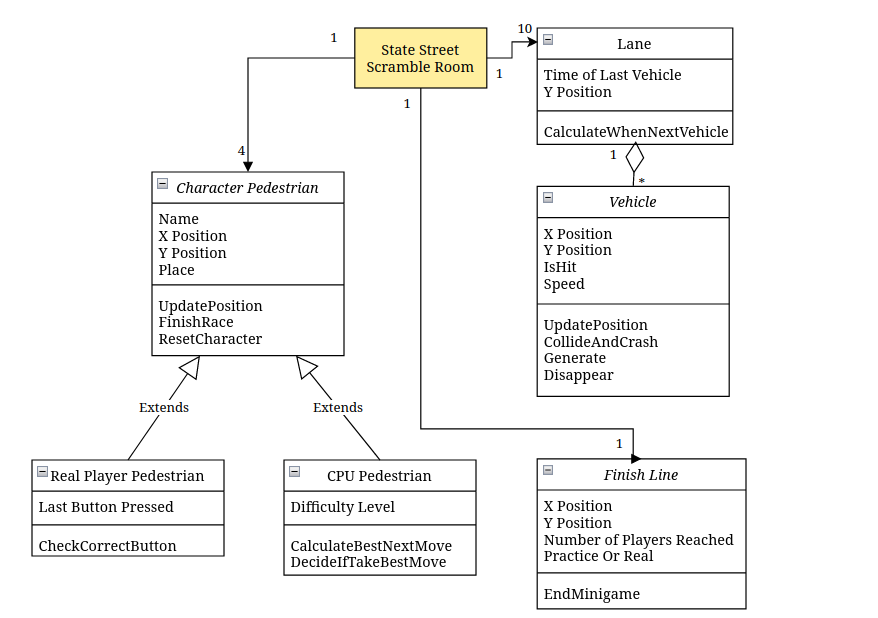
* **Tipsy Tunnel Trek**
  + Character Tunneler
    - Keeps track of Character data, including name, position, speed, side of the tower, and place (1st, 2nd, 3rd, 4th)
    - Updates the player position as needed
    - Finishes race when reaching the finish line
  + CPU Tunneler
    - Child of Character Tunneler, keeps track of the CPU difficulty
    - Determines the best move for the CPU, and whether or not it will make that move
  + Maze
    - Chooses a random maze from a list of potential mazes
    - Keeps track of the maze ID
  + Wall
    - Tracks the position of the wall
  + Finish Line
    - Tracks the position of the finish line, the number of players that have crossed the finish line, and whether or not the game is real or a demo.
    - Ends the minigame when enough players have reached the end
  + Buttons
    - Keeps track of the buttons currently in use
    - Switches which buttons are being used based on a calculated interval

**11:59 Room:**



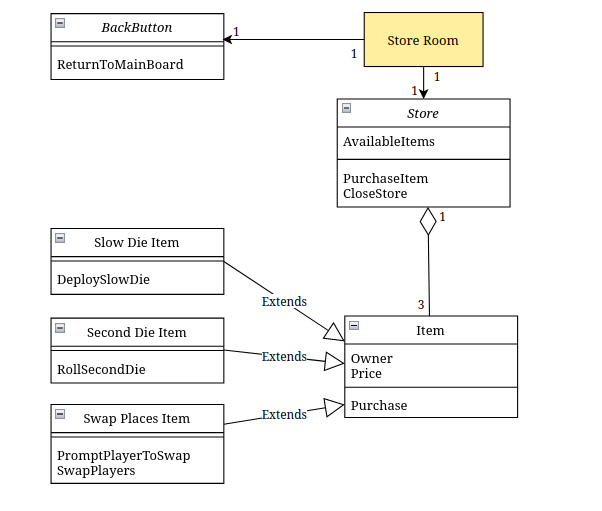
* **11:59**
  + Character Assignment
    - Keeps track of the Character name, time assignment turned in, and place (1st, 2nd, 3rd, 4th)
    - Allows the player to turn in the assignment
  + CPU Assignment
    - Child of Character Assignment, keeps track of the CPU difficulty level and when it will turn in the assignment
  + Submission Report
    - Keeps track of the number of players that have submitted the assignment, and whether or not it was real or practice
    - Ends the minigame when needed
  + Digital Clock
    - A clock which displays the current time
    - Updates the time on the clock and whether or not the clock is visible to the players

**State Street Scramble Room:**



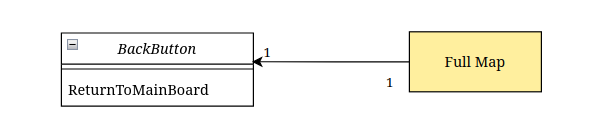
* **State Street Scramble**
  + Character Pedestrian
    - Keeps track of Character name, position and place (1st, 2nd, 3rd, 4th)
    - Updates the player position as needed, resetting their position to the start when needed
    - Finishes race when reaching the finish line
  + Real Player Pedestrian
    - Child of Character Pedestrian, keeps track of the last button the player pressed, and if a button press is correct
  + CPU Pedestrian
    - Child of Character Pedestrian, keeps track of the CPU difficulty
    - Determines the next best move for the CPU, and if the CPU will make that move
  + Finish Line
    - Tracks the position of the finish line, the number of players that have crossed the finish line, and whether or not the game is real or a demo.
    - Ends the minigame when enough players have reached the end
  + Lane
    - Tracks the position of the lane and when the last vehicle was spawned
    - Calculates when to spawn a new vehicle
  + Vehicle
    - Tracks the vehicle position, speed, and if it has been hit by the player
    - Generates and removes vehicles as needed
    - Updates the vehicle position and determines if it has crashed into a player

**Store Room:**



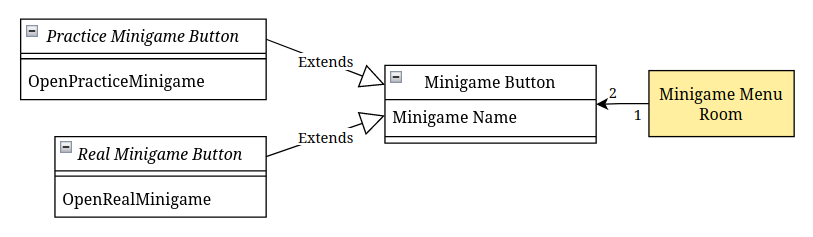
* **Store**
  + Store
    - Tracks which items are available
    - Can purchase items and close the store
  + Item
    - Related to Store, can be equipped to grant the Character special abilities
    - Can be purchased in the shop using Boiler Bucks
  + Slow Die Item
    - Child of Item, causes the die to change numbers slowly, allowing the Character to choose the number of spaces they move
  + Second Die Item
    - Child of Item, grants the Character a second die to roll with
    - The amount of spaces the Character moves is the combined total of the dice
  + Swap Places Item
    - Child of Item, allows the Character to swap places with another Character of their choosing
  + Back Button
    - Used to return to the Main Gameboard Room

**Full Map Room:**



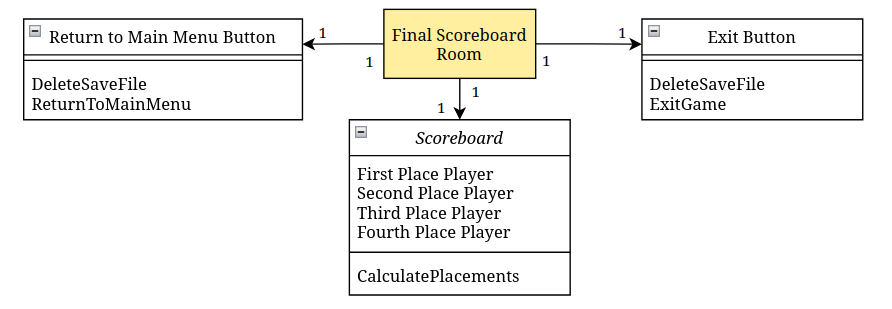
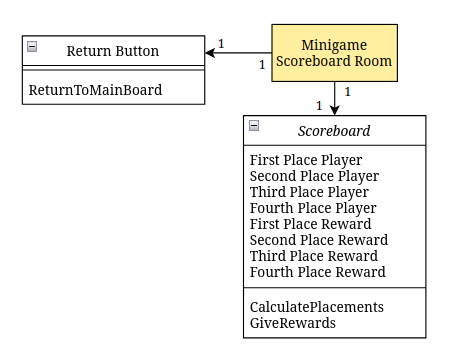
* **Full Map**
  + Back Button
    - Used to return to the Main Gameboard Room

**Minigame Menu Room:**



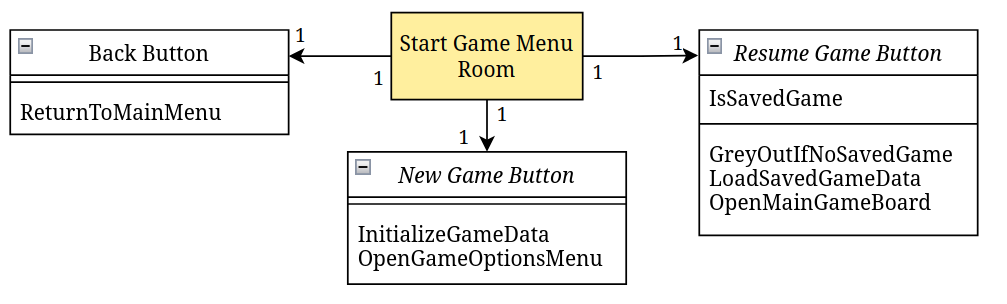
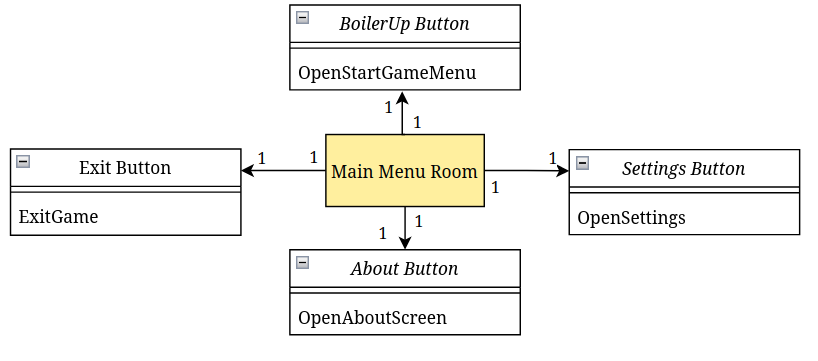
* **Minigame Menu**
  + Minigame Button
    - Contains name of the minigame
  + Practice Minigame Button
    - Opens the minigame in practice mode
  + Real Minigame Button
    - Opens the minigame in non-practice mode

**Minigame Scoreboard Room and Final Scoreboard Room:**



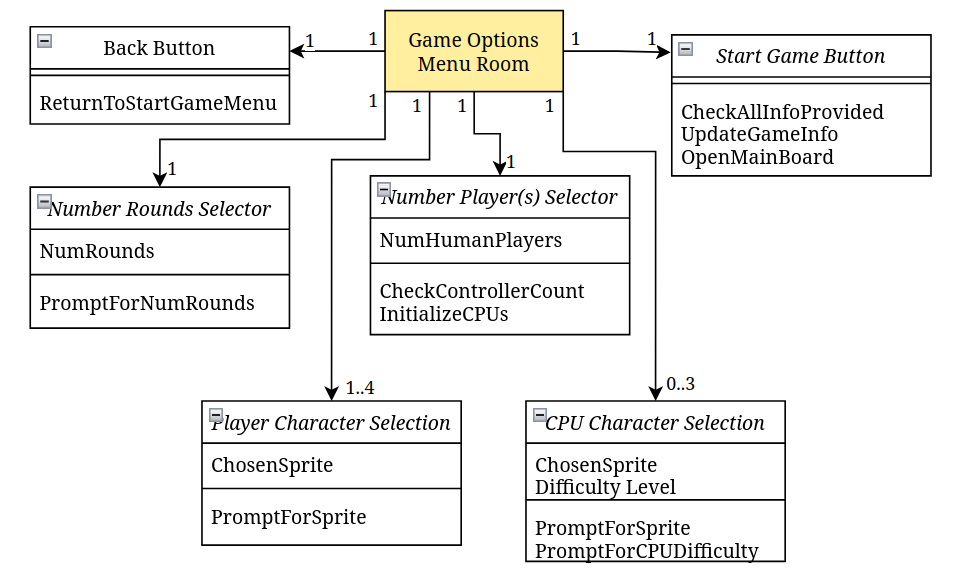
* **Minigame Scoreboard**
  + Scoreboard
    - Calculates and contains the placing and rewards for all players
  + Return Button
    - Returns to the Main Gameboard Room
* **Final Scoreboard**
  + Scoreboard
    - Calculates and contains the placements of each player
  + Exit Button
    - Deletes the save file and exits the game
  + Return to Main Menu Button
    - Deletes the save file and returns to the Main Menu Room

**Main Menu Room and Start Game Room:**



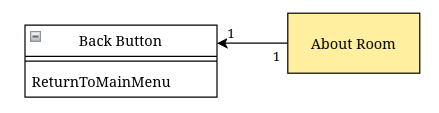
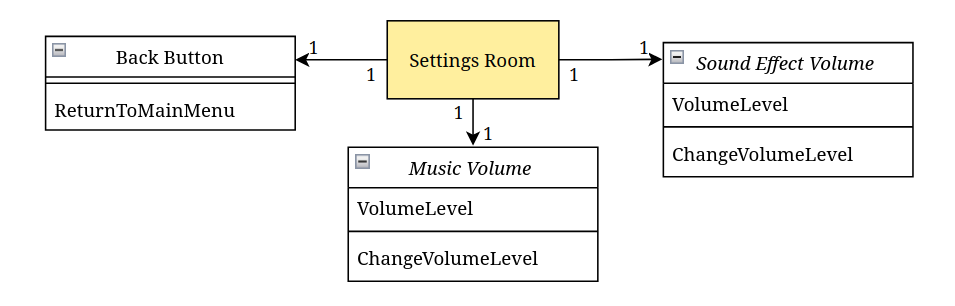
* **Main Menu**
  + Boiler Up Button
    - Opens the Start Game Menu
  + Settings Button
    - Opens the Settings Menu
  + About Button
    - Displays the About Screen
  + Exit Button
    - Exits the game
* **Start Game Menu**
  + New Game Button
    - Initializes data for the game and opens the Game Options Menu
  + Resume Game Button
    - Keeps track of if there is a saved game
    - Button is greyed out and does not work if there is no save game data
    - Otherwise loads the saved game data and opens the Main Gameboard Room when pressed
  + Back Button
    - Returns to the Main Menu Room

**Game Options Menu Room:**



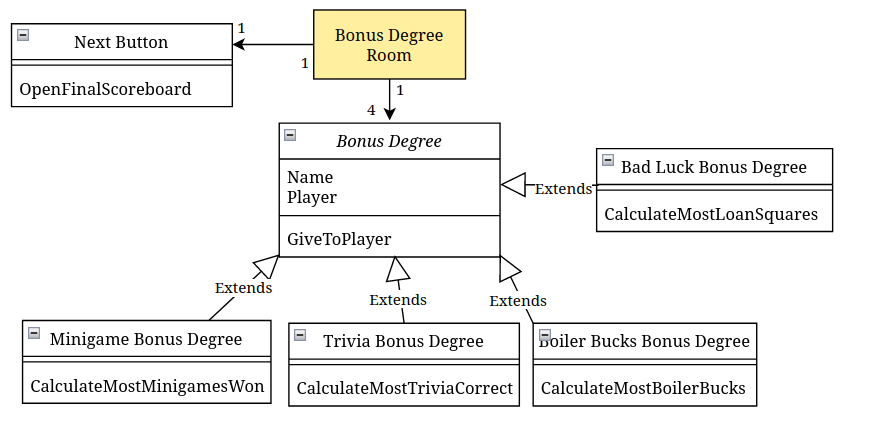
* **Game Options Menu**
  + Number Rounds Selector
    - Prompts for the number of rounds and stores the result
  + Number Players Selector
    - Checks the controller amount to determine the number of human and non-human players
    - Initializes CPUs as needed
  + Player Character Selection
    - Prompts character to choose a sprite and stores the result
  + CPU Character Selection
    - Prompts for CPU sprite and difficulty and stores the result
  + Start Game Button
    - Checks that all information has been provided
    - Updates the game info and opens the Main Gameboard Room
  + Back Button
    - Returns to the Start Game Menu

**Settings Room and About Room:**



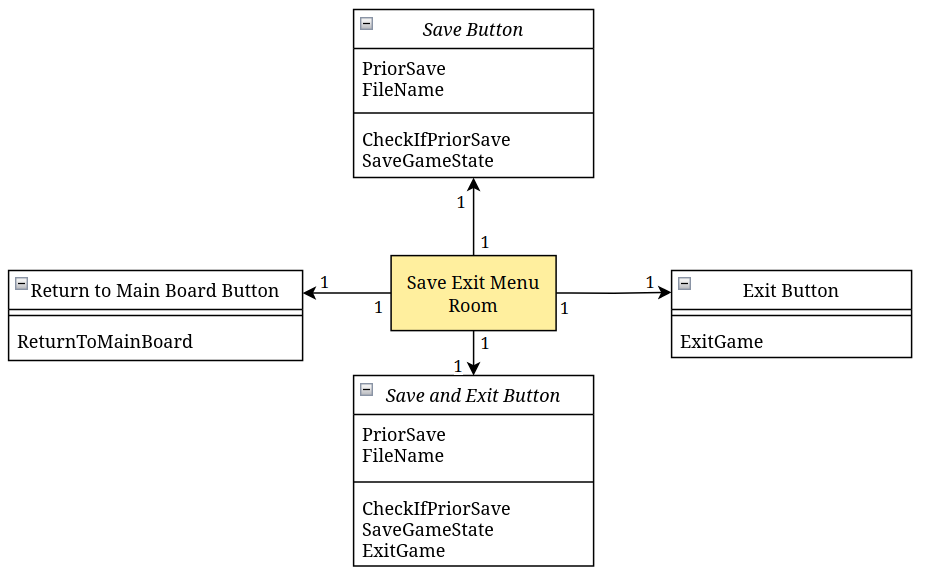
* **Settings**
  + Music Volume
    - Allows users to adjust music volume
  + Sound Effect Volume
    - Allows users to adjust SFX volume
  + Back Button
    - Returns to the Main Menu
* **About**
  + Back Button
    - Returns to the Main Menu

**Bonus Degree Room:**



* **Bonus Degree**
  + Bonus Degree
    - Contains the name of the bonus degree and the player who earned it
    - Gives the player an additional degree
  + Minigame Bonus Degree
    - Calculates who won the most minigames
  + Trivia Bonus Degree
    - Calculates who answered the most trivia questions correctly
  + Boiler Bucks Bonus Degree
    - Calculates who earned the most Boiler Bucks throughout the game
  + Bad Luck Bonus Degree
    - Calculates who landed on the most loan squares
  + Next Button
    - Opens the Final Scoreboard Room

**Save Exit Menu Room:**

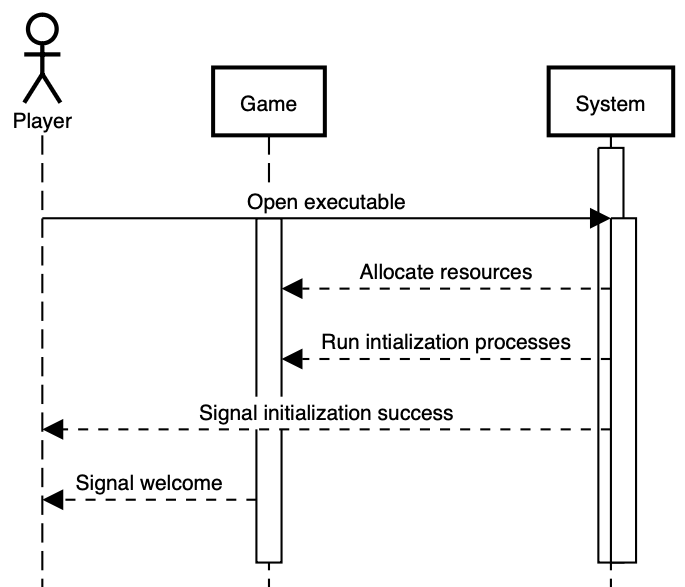


* **Save Exit Menu**
  + Save Button
    - Saves the game state, overwriting a prior save if needed
  + Exit Button
    - Exits the game
  + Save and Exit Button
    - Saves the game state, overwriting a prior save if needed
    - Exits the game
  + Return to Main Board Button
    - Returns to the Main Gameboard Room

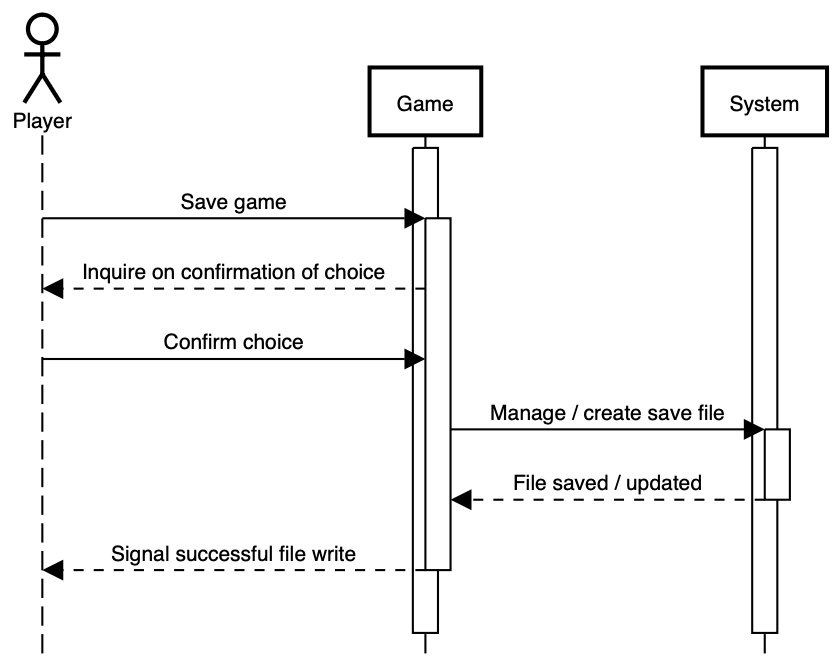
## 

## Sequence Diagrams:

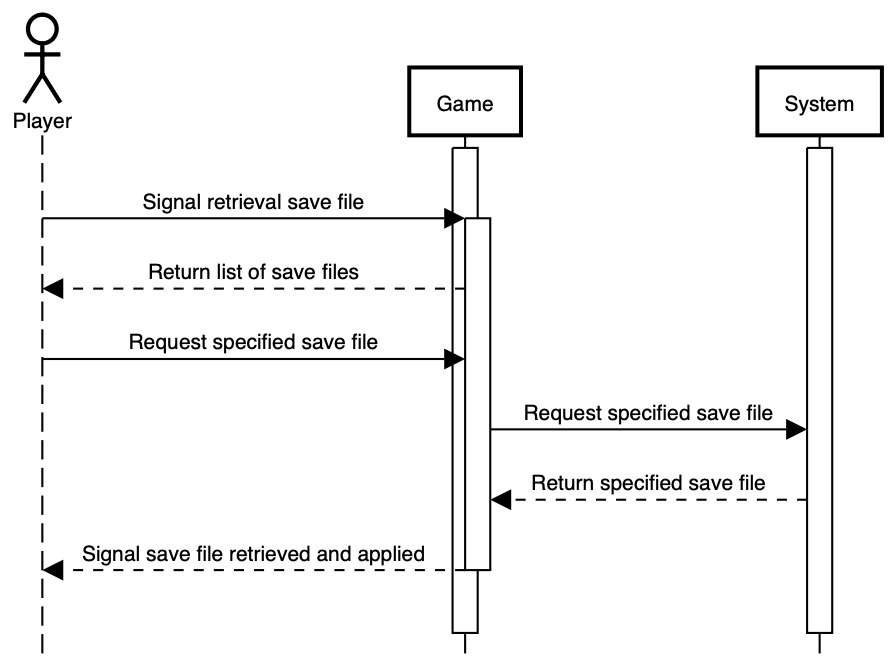
1. Sequence of events when the game is started



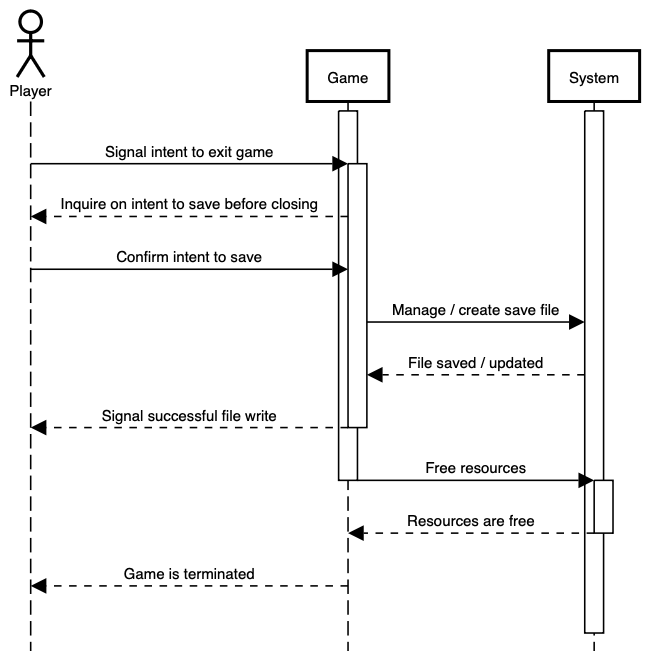
1. Sequence of events when the game is saved



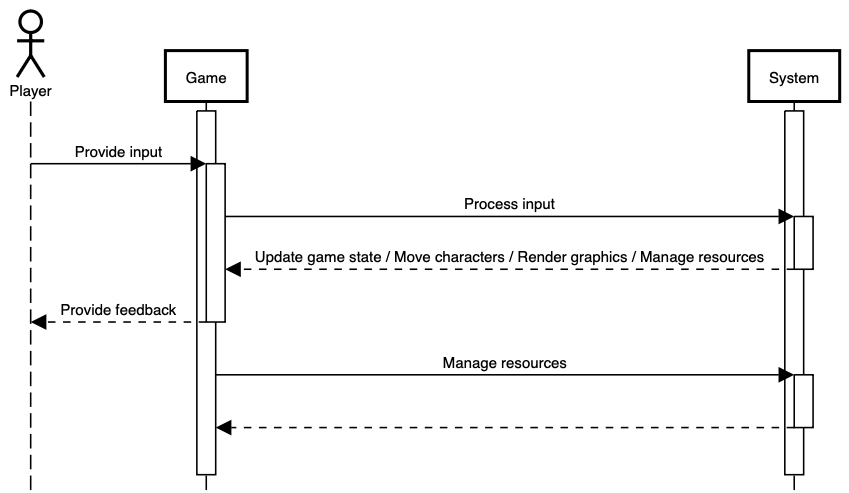
1. Sequence of events when a save file is retrieved



1. Sequence of events when the game is shut off



1. Sequence of events when the game is played



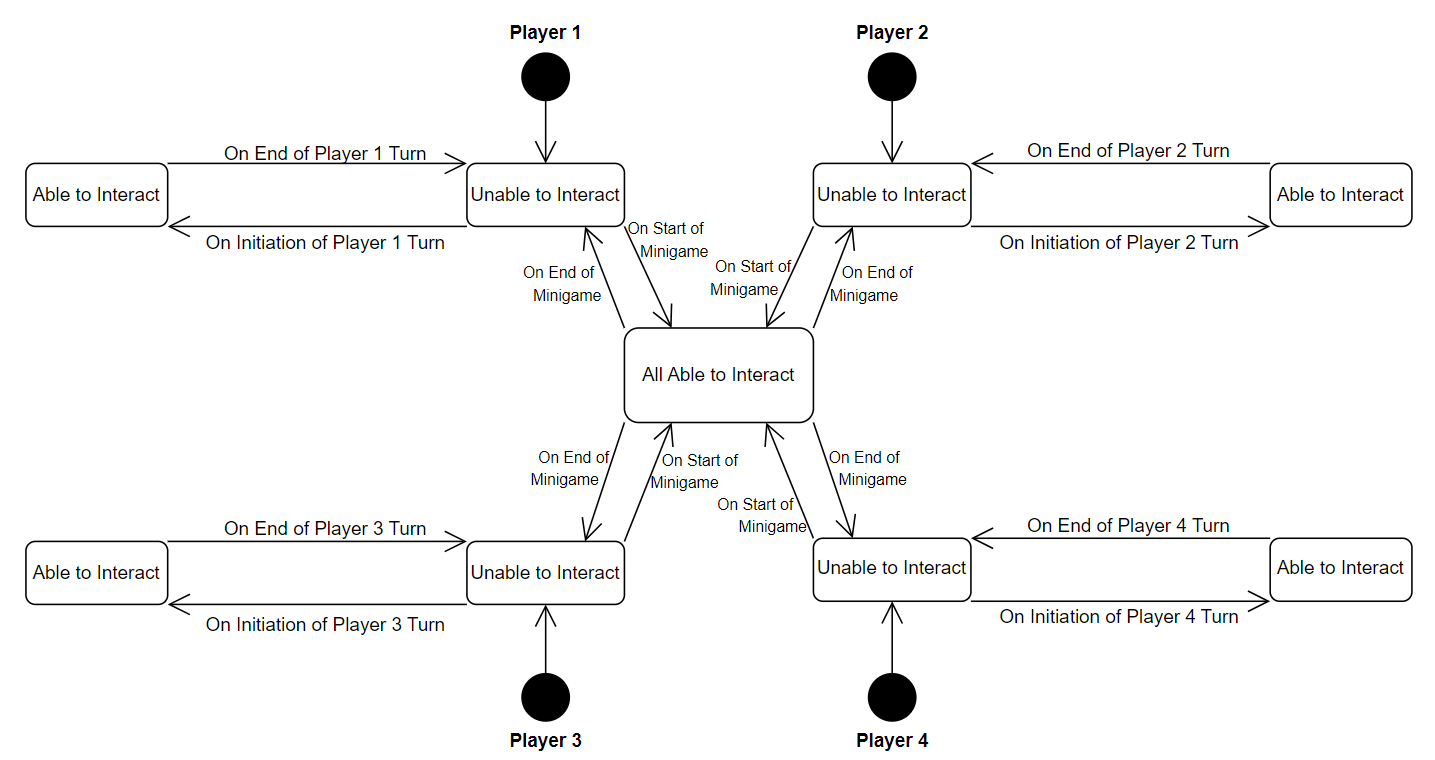
## 

## **Navigation Flow Map:**

## 

The Navigation Flow Map depicts how the screens of our application are navigated by the user. Each box represents a different screen that the user may see throughout the game. Arrows connecting these boxes represent the ways in which a user can transition from one screen to another, with labels depicting how the user initiates the screen transition. The gold box is the screen that the user starts on upon launching the application. The arrows pointing to **SYSTEM EXIT** represent ways in which the user can exit the game without interfering with the saving system.

## State Diagram:



This State Diagram depicts the interaction ability states of all four players during the main gameplay (Main Game Board, Store, Minigames) of our application. Each player can either be in a state in which they are Unable to Interact with the game, meaning that controller input will be ignored, or Able to Interact with the game, meaning that controller input will have an effect. All players will begin on initial entry to the Main Game Board in the Unable to Interact state. On the initiation of each player’s turn, the player will switch to the Able to Interact state until the end of their turn, when they will return to the Unable to Interact state. On the start of a minigame, all players will switch to an Able to Interact state until the end of the minigame, when they will return to the Unable to Interact state.

## 

## 

## UI Mockup:

The below images are based off of the navigation flow map included above

* Title Screen
  + Boiler Up is a button to continue with setting up and starting the game
  + Settings is a button that will take you to the settings screen
  + Exit is a button that will exit the application
  + About is a button that will take you to the about screen



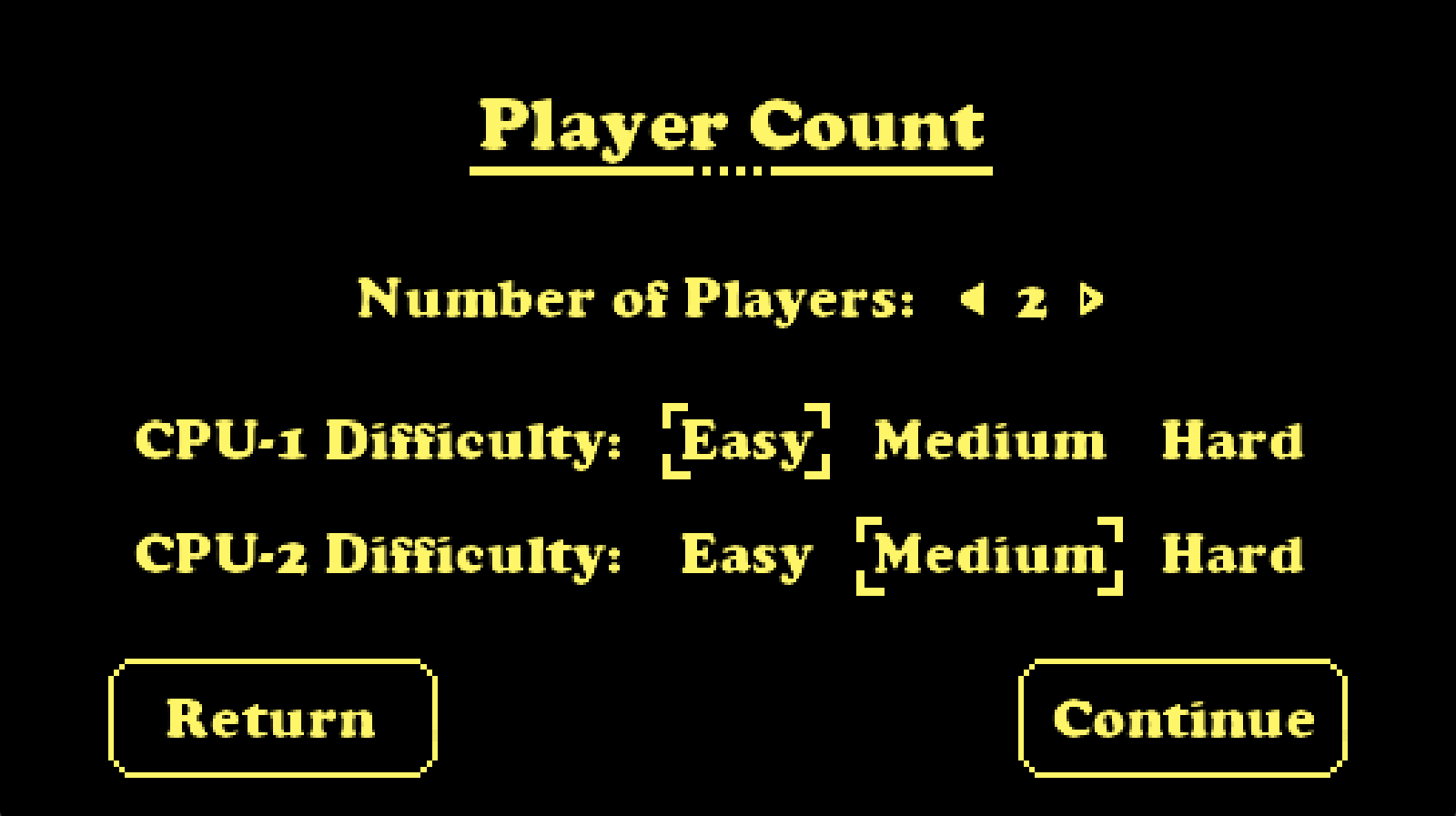
* Settings Screen
  + Return is a button that will take the players back to the title screen
  + The music and sfx settings allow the players to change the respective types of volumes from the game



* About Screen
  + Has a return button that will take the players back to the main title screen



* CPU Settings Screen
  + Return is a button that will take you back to the title screen
  + Continue is a button that will take you to the game setup settings
  + Number of players can be changed and based on that value, up to 4 CPUs will appear on the screen where their difficulty can be selected



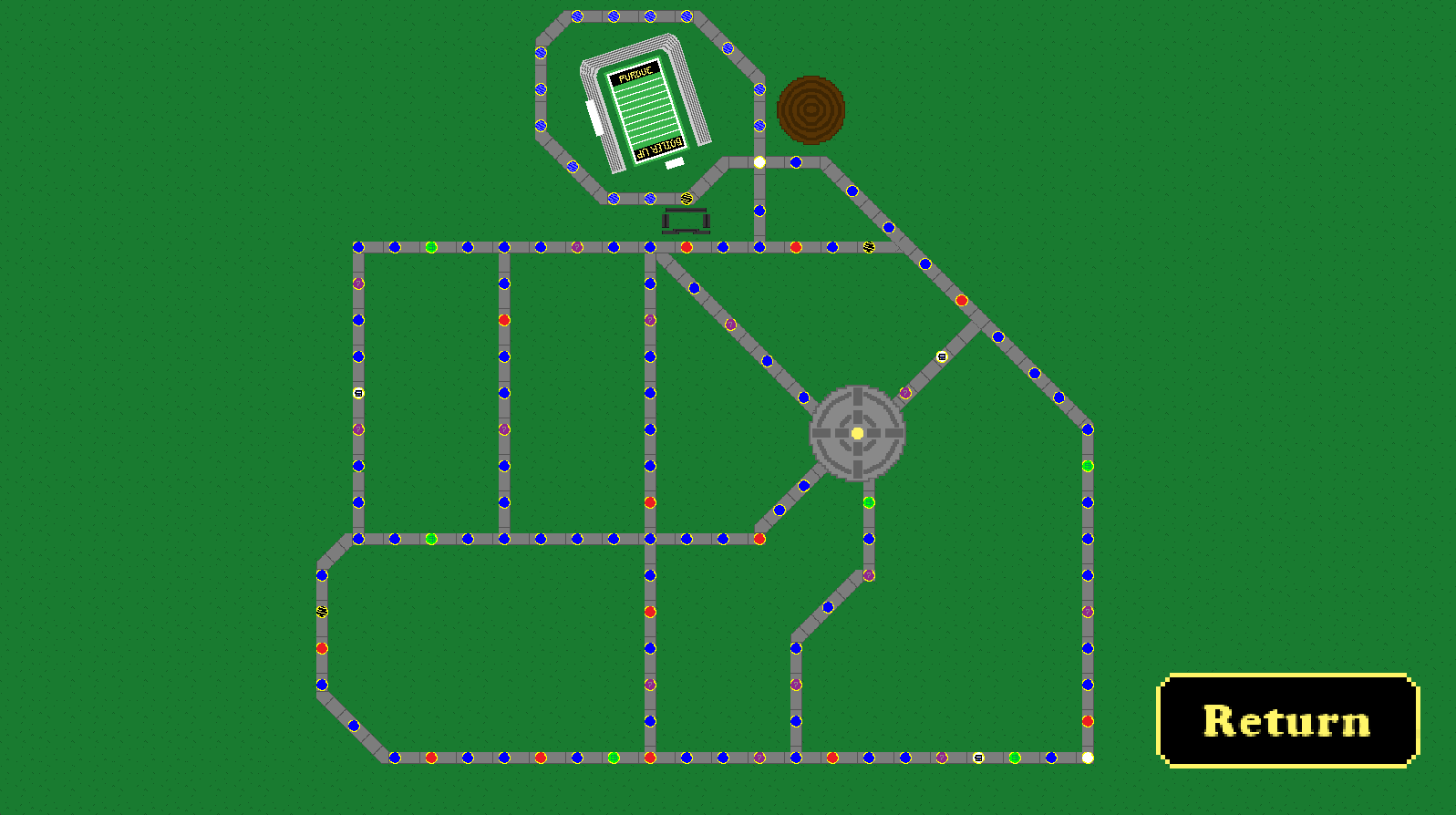
* Game Setup Screen
  + Return is a button that will take you back to the CPU settings screen
  + Continue is a button that will take you to the start of the game



* Local View of Map
  + Roll Die is a button that allows the player to begin the movement phase of their turn
  + Use item is a button that will allow a user to use any items they may have
  + View map will toggle the screen to a full overview of the map so the player can plan how they want to use items or move



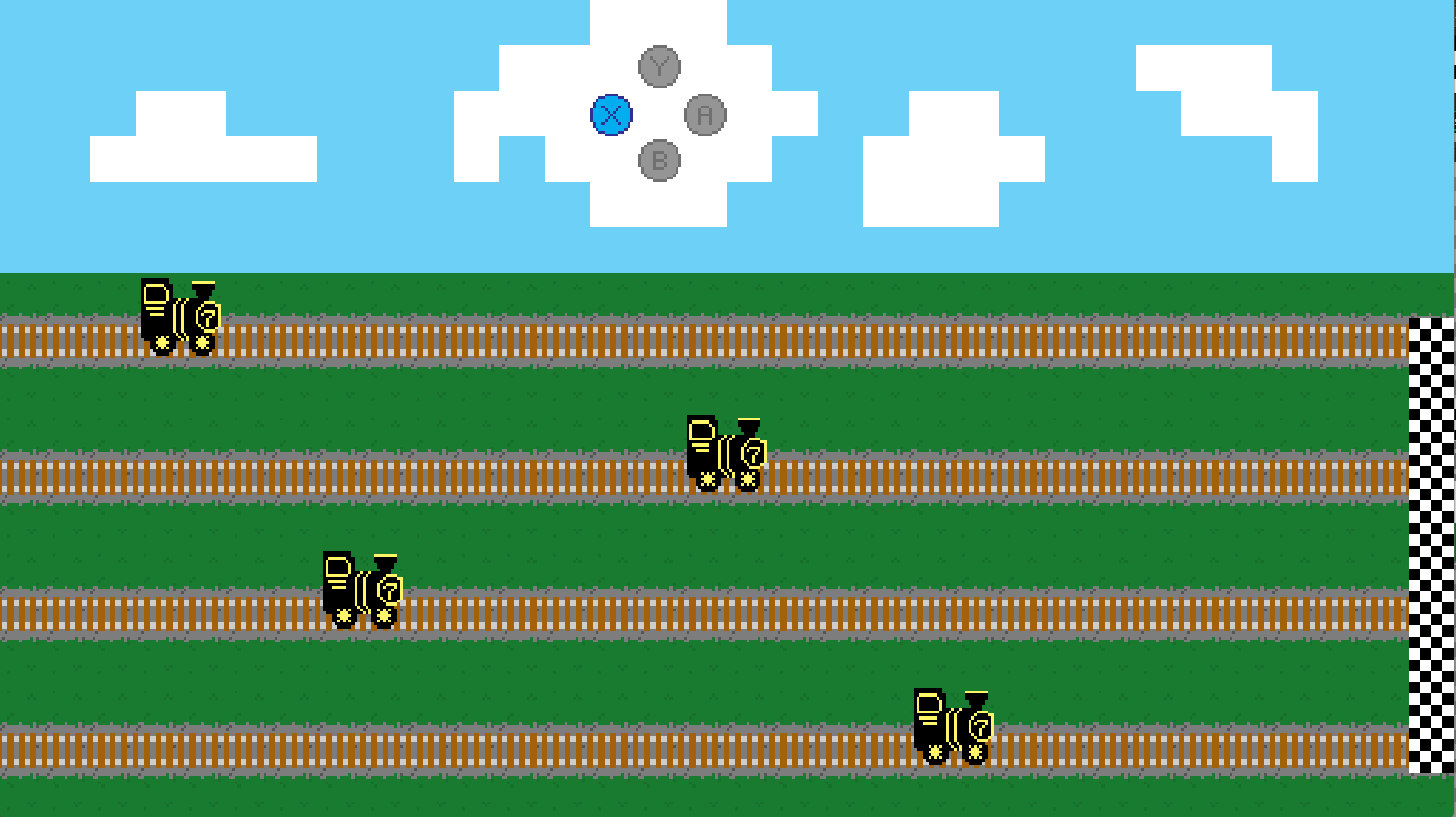
* Main Map
  + A rough overview of the map that as all icons visible to the players
  + Return is a button that will take the player back to the local view of their map



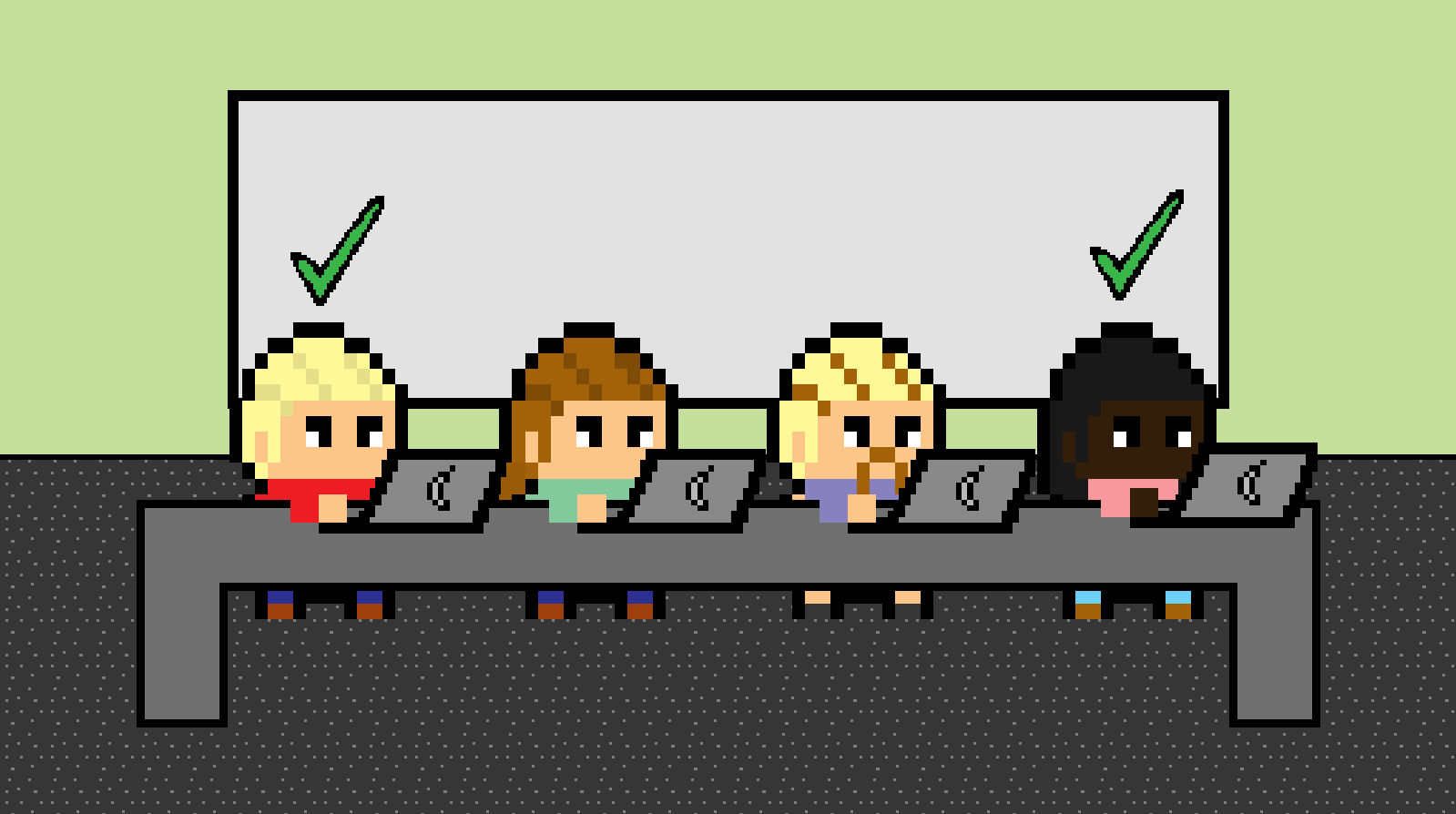
* Sample Instructions Screen
  + Play game is a button that will take the players to the minigame the instructions are describing
  + Play demo is a button that will take the players to the minigame the instructions are describing but when completed, will have the minigame return to the instructions page and it will not contribute to the players’ score



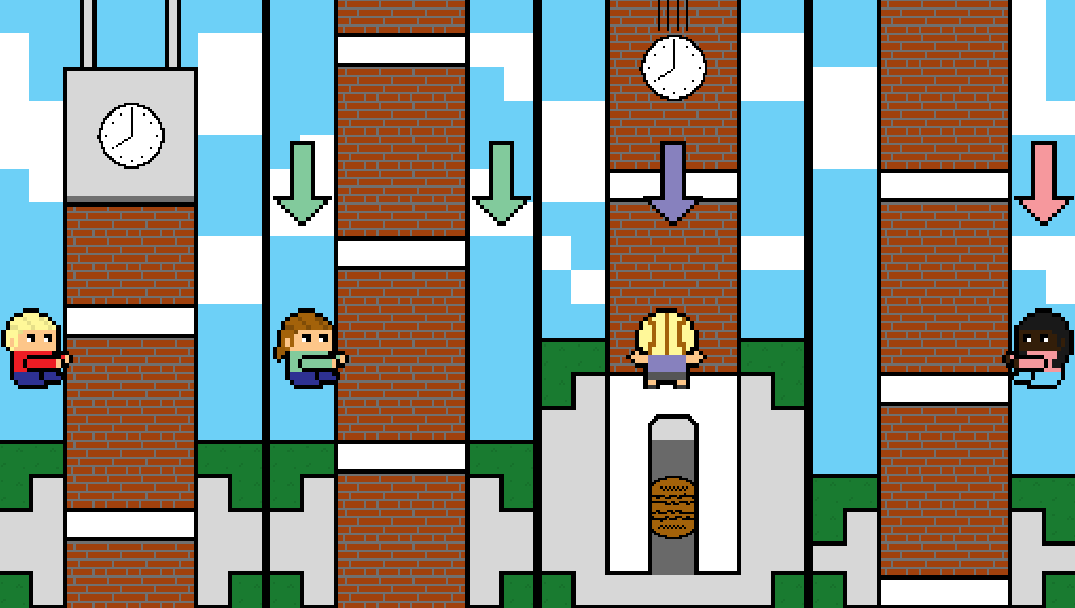
* Boilermaker Express Minigame
  + Each train is assigned to a player in the game that moves at the rate they alternate between the two buttons currently in play
  + The buttons depicted in the clouds will flash to show which two buttons the players will need to alternate from in order to get their trains to move
  + When a player crosses the finish line at the end, we will know where the players are placed



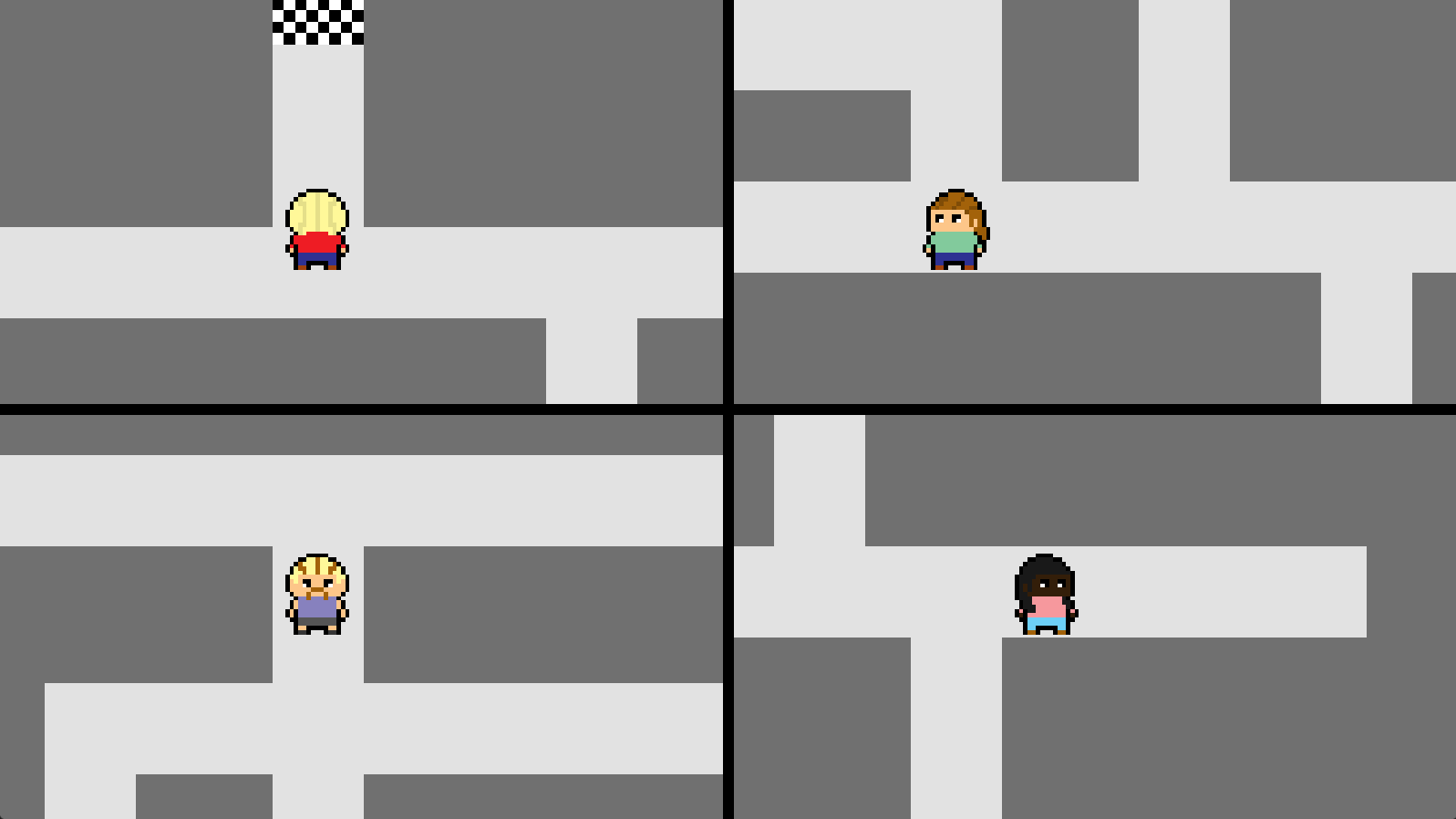
* 11:59 Minigame
  + Each player with be able to see a timer that counts to the turn in time, 11:59:59 on the board behind, with it disappearing after a few seconds, leaving players to guess as close as they can to the turn in time
  + Player presses a button to turn in their assignment, getting signified with a check mark



* Time’s Up!
  + Each player will, at the start of the game, begin automatically climbing up the Bell Tower
  + The player will be able to be on the left, center, or right side of it, and can change sides with the control stick
  + Clocks can fall on the left, center, or right side of the clock tower, and up to two can be on screen at once
  + Players hit by the clocks will fall down slightly but will begin climbing again
  + First to the top wins



* Tipsy Tunnel Trek
  + Each player will be placed in the same randomly selected maze, with the goal of escaping first
  + Each split screen section is focused in closely on each playable character to prevent too much of the maze from being seen
  + Players use the analog stick to move left, right, up, or down



* State Street Scramble
  + Players are able to move left, right, up, and down to avoid traffic and get to the other side



* Store Screen



* Minigame Result Screen
  + Continue is a button that will allow players to return to the map and remain gameplay and start their individual turns for the round
  + Players will be displayed in the respective places based on how they did in the minigames. Their sprite, current “Boiler Bucks” and “Degrees” will be displayed in that space



* Final Result Screen
  + Main menu is a button that will take the players to the title screen so they may begin a new round if they would like to
  + Exit is a button that when pressed, will exit the application
  + Players will be displayed in the respective places based on how they did throughout the game. Their sprite, ending amount of “Boiler Bucks” and “Degrees” will be displayed in that space

