CS437 Final Project

Planning

**Group Members**

Trevar Ladd

Madelyn O’Dell

Ernie Yarbrough

Contents

[Current Project Plan Summary 3](#_Toc498455829)

[Language Choice 3](#_Toc498455830)

[Game Timeline Progression (In-Order) 3](#_Toc498455831)

[Game Variables 3](#_Toc498455832)

[‘Global’ Game Variables 3](#_Toc498455833)

[Current Objectives 4](#_Toc498455834)

[GUI Transition Diagrams 4](#_Toc498455835)

[Data Transition Diagrams 4](#_Toc498455836)

[Game Screenshots 4](#_Toc498455837)

# Current Project Plan Summary

As part of the CS437 final project our group decided that a scalable multiplayer game should be made. We plan to make a multiplayer game that moves players through a sequence of mini-games. As the game progresses, players will move on to new mini-games with some sort of temporary storage being used to track who is winning each round. At the end of the round, the players will be presented with their rankings.

## Language Choice

We decided to use JavaScript (JS) as the programming language to complete this task. With a short amount of time and a group environment, we decided that JS would be a good choice. This will be portable and easily compatible between group members’ environments.

Currently the plan is to use different game engines for this which are going to be JavaScript libraries. We would be able to use varying game engines for different levels. The main screens will be simple and switch between the various game levels.

# Game Timeline Progression (In-Order)

|  |  |
| --- | --- |
| Game Stage | Description |
| Landing Page | The landing page will serve as a main menu. From the main menu players can then enter a lobby that will pair players up. |
| Game Lobby | The game lobby will serve to link players with other players. Players will wait here until there are enough players in the room to begin the game. At that point a countdown timer will start and display to all players notifying them of the start of the game. |
| Mini-Game Randomize | An array of mini-game references can be kept. Each time a new mini-game is randomly chosen, it will mark that game in the array making it so that one game can not be chosen more than once. If all games in the game array have been taken, the game will move to the high score screen |
| Mini-game Start (recursive) | There will be a short amount of time before the game actually starts. This screen can be standardized for all games. This screen will display any instructions that would be needed for the game so that all players know what to do. This may only stay on the screen for about 10 seconds. |
| Mini-game (recursive) | <running through minigame> |
| End Scoring Screen | Will display player’s scores and who won |
|  |  |
|  |  |

# Game Variables

## ‘Global’ Game Variables

These variables will be for a single game session. After players are assigned to the session, all of these values will be initialized for the first time. Some variables will be initialized with previous input and be variable such as number of game rounds (how many mini-games per session) and how many players should be in the current session. Making these input values will help to tailor the game session to how the current players would like it.

|  |
| --- |
| Array of player identification |
| Number players |
| Number rounds |
| Mini-game randomization tracker array |
| Array to hold player’s game points |
|  |
|  |
|  |

# Mini-Game Planning

Below are ideas for games. Not all games will be implemented but it will serve as a list to pull from when a developer is ready to create a new game. All games should return a point value with an optimal point number of 100 and worst case 0. This means that if there are 10 games, the total possible points would be 1000 points. Points should reflect how well a player did against the other and in some games will be equally distributed to the players in the game.

## Tic Tac Toe

Description: Traditional tic tac toe for players to engage with on a 3x3 board

Number of Players: 2

Goal: Achieve 3 in a row

Challenge: Trick the other opponent and get 3 in a row

Win Conditions: 3 in a row, diagonal, or column

Point System: Suggest an odd number of rounds. Handle tie conditions with another game, coin flip, trivia question, or “accuracy tap”

## Pong

Description: traditional pong game

Number of Players: 2

Goal: Get the pong ball to bounce on to the other player’s end past their block

Challenge: Timing

Win Conditions: Ball passes other player X amount of times

Point System: Points can be rewarded based on number of goals made and returned to master game controller

## Competitive Block Smash

Description: Just like a typical block smashing game but with two players. There would be no collisions between the two player’s balls or pieces, but will run at the same time in the same environment. The key would be to get more bricks smashed before they run out than the other player.

Number of Players: 2

Goal: Smash most bricks

Challenge: Timing, and angle to beat the other players

Win Conditions: Bricks are all gone and player has most smashed bricks

Point System: Points can be rewarded for each brick smashed suggest 100 bricks and so players can get between 0 and 100 with a balance between the two equaling 100

## Mini-Golf

Description: A top-down view of a mini-golf course with possibly 3 or so holes. Players take turns with no collision between their ball and the opponent’s ball. They will set the velocity, and angle that they would like their ball to be hit in.

Number of Players: 2 or more

Goal: Get a low number of puts and beat the other player’s put count

Challenge: This will challenge players abilities to look at things and judge how the ball will bounce

Win Conditions: Less puts than the other player

Point System: Points can be a formula that awards a bouns for winning, and then points out of 100 based on a ratio. For example if one player had 20 puts, and the other 10, the points may be 70 and 30.

## Pastry Panic

Description: A puzzle game that allows for a user to select a cell, and once the cell is selected, shift the row/column up or down/right or left. This will be similar to how a rubix cube works when matching colors. Data will be stored in a two dimensional array, using basic for loops to shift values (we will keep the N down by limiting the size of the game board to only 25 elements).

Number of Players: 2 or more

Goal: Get the highest score in comparison to other players

Challenge: This will challenge a person’s matching capabilities and quick reflexes

Win Conditions: If a row/column wise match occurs for one of the 5 states, points are given based off of which states are matched.

Point System: Points are given based off of which states are matched (going from lowest score to highest: donut, cookie, pie, croissant, cake)

## Virtual DDR

Description: A reaction time based game based off of DDR. This game will have four states (up, down, left, right), these states will go up the game screen to the top, if the user hits the correct button (this uses the arrow buttons of course) while in a certain range, points will be generated, if the button is hit outside of the range, no points will be given

Number of Players: 2 or more

Goal: Get as many points as possible

Challenge: Reaction time will be tested between players

Win Conditions: If the user has the highest point total when the timer expires, they will win

Point System: Every button hit in the correct location will generate a point.

## Monster Battle

Description: Battle game that uses a list of monster objects with specific stats and abilities, allowing for a single player to choose three from the list. Once the monsters are chosen for both players, the players will face off with one another, trying to defeat each other’s monsters.

Number of Players: 2

Goal: Defeat the other player’s monsters, if time runs out, be the player with the most surviving monsters/monsters with the highest health

Challenge: Strategically using the correct moves, and being able to guess your opponent’s moves

Win Conditions: Defeat all of the other player’s monsters, if time runs out, have the most monsters, or if the amount of monsters is the same, have the monster with the highest health.

Point System: This game isn’t point based, but winning this game will give the winner points towards the overall game.

## Running Man

Description: Infinite run game, where the player gains points based on how many obstacles they dodge/jump over. This game will have two basic functions: jump and duck. The user will be able to jump over certain obstacles and duck under other obstacles.

Number of Players: 2 or more

Goal: Avoid the largest amount of obstacles

Challenge: Reflexes are challenged here

Win Conditions: If when the timer runs out and you have the highest score you win.

Point System: Diverting each obstacle will give one point to the total score

## <GAME TITLE HERE>

Description: <INSERT A DESCRIPTION>

Number of Players: 2 – if we go outside of 2 we need to explain how we will do this

Goal:

Challenge:

Win Conditions:

Point System:

## <GAME TITLE HERE>

Description: <INSERT A DESCRIPTION>

Number of Players: 2 – if we go outside of 2 we need to explain how we will do this

Goal:

Challenge:

Win Conditions:

Point System:

## <GAME TITLE HERE>

Description: <INSERT A DESCRIPTION>

Number of Players: 2 – if we go outside of 2 we need to explain how we will do this

Goal:

Challenge:

Win Conditions:

Point System:

## <GAME TITLE HERE>

Description: <INSERT A DESCRIPTION>

Number of Players: 2 – if we go outside of 2 we need to explain how we will do this

Goal:

Challenge:

Win Conditions:

Point System:

## <GAME TITLE HERE>

Description: <INSERT A DESCRIPTION>

Number of Players: 2 – if we go outside of 2 we need to explain how we will do this

Goal:

Challenge:

Win Conditions:

Point System:

## <GAME TITLE HERE>

Description: <INSERT A DESCRIPTION>

Number of Players: 2 – if we go outside of 2 we need to explain how we will do this

Goal:

Challenge:

Win Conditions:

Point System:

# Current Objectives

# GUI Transition Diagrams

# Data Transition Diagrams

# Game Screenshots