DESIGN DOCUMENT (VERSION 1.3)

I. Introduction

Project Name: WORLD'S HARDEST GAME
Group Name: World's Hardest Group
Group Members: Aidan Wong, William Wang
Period: 1

II. Description

This project replicates the game "The World's Hardest Game" (link). The objective of this game is to get to the green area from the spawn area. On some levels, it is required to collect coins before the green goal area is activated and accessible.

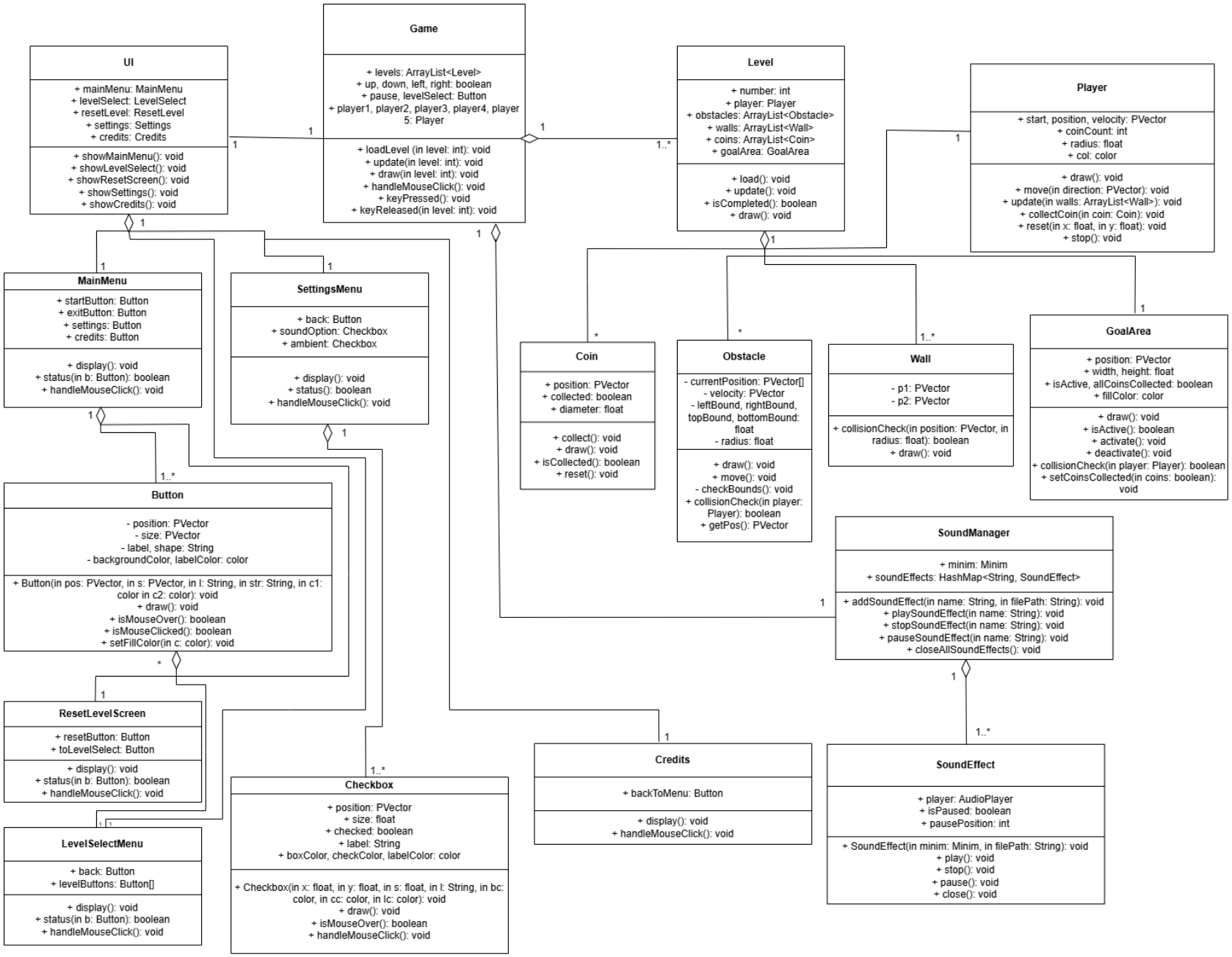
A. Functionalities

1. Player Movement: Control Red Square using WASD or arrow keys (goal is smooth and responsive movement)
2. Levels: Each level has various amounts of coins and unique obstacles → up to 25
3. Goal Area: Green square areas that the player needs to reach to complete the level
4. Obstacles: Blue obstacles have predetermined paths and move, touching them will return the player to spawn (obstacles stay in their current position and continue moving)
5. Walls: The player can collide with them, but can't go past
6. Coins: Yellow coins, when touched by a player, disappear and add to a coin count at the top of the screen (required to collect all for levels that have them)
7. User Interface: Main menu, interactable buttons, level selection, reset level screen, settings menu (for sound, graphics, square color, etc)
8. Sound effects (if extra time)

B. Required Libraries

1. Processing Core Library: Preinstalled, to code the bulk of the features
2. Sound Library: Sound effects for player movement, collision, etc
3. ControlP5: For creating the GUI (buttons, sliders, text fields)
4. Collision Detection Library: Basic collision detection, used for more efficiency/optimization

III. UML


Game
+ levels: ArrayList<Level>
+ up, down, left, right: boolean
+ pause, levelSelect: Button
+ player1, player2, player3, player4, player 5: Player
+ loadLevel (in level: int): void
+ update(in level: int): void
+ draw(in level: int): void
+ handleMouseClick(): void
+ keyPressed(): void
+ keyReleased(in level: int): void
UI
+ mainMenu: MainMenu
+ levelSelect: LevelSelect
+ resetLevel: ResetLevel
+ settings: Settings
+ credits: Credits
+ showMainMenu(): void
+ showLevelSelect(): void
+ showResetScreen(): void
+ showSettings(): void
+ showCredits(): void
Level
+ number: int
+ player: Player
+ obstacles: ArrayList<Obstacle>
+ walls: ArrayList<Wall>
+ coins: ArrayList<Coin>
+ goalArea: GoalArea
+ load(): void
+ update(): void
+ isCompleted(): boolean
+ draw(): void
Player
+ start, position, velocity: PVector
+ coinCount: int
+ radius: float
+ col: color
+ draw(): void
+ move(in direction: PVector): void
+ update(in walls: ArrayList<Wall>): void
+ collectCoin(in coin: Coin): void
+ reset(in x: float, in y: float): void
+ stop(): void
MainMenu
+ startButton: Button
+ exitButton: Button
+ settings: Button
+ credits: Button
+ display(): void
+ status(in b: Button): boolean
+ handleMouseClick(): void
SettingsMenu
+ back: Button
+ soundOption: Checkbox
+ ambient: Checkbox
+ display(): void
+ status(): boolean
+ handleMouseClick(): void
Coin
+ position: PVector
+ collected: boolean
+ diameter: float
+ collect(): void
+ draw(): void
+ isCollected(): boolean
+ reset(): void
Obstacle
- currentPosition: PVector[]
- velocity: PVector
- leftBound, rightBound, topBound, bottomBound: float
- radius: float
+ draw(): void
+ move(): void
- checkBounds(): void
+ collisionCheck(in player: Player): boolean
+ getPos(): PVector
Wall
- p1: PVector
- p2: PVector
+ collisionCheck(in position: PVector, in radius: float): boolean
+ draw(): void
GoalArea
+ position: PVector
+ width, height: float
+ isActive, allCoinsCollected: boolean
+ fillColor: color
+ draw(): void
+ isActive(): boolean
+ activate(): void
+ deactivate(): void
+ collisionCheck(in player: Player): boolean
+ setCoinsCollected(in coins: boolean): void
Button
- position: PVector
- size: PVector
- label, shape: String
- backgroundColor, labelColor: color
+ Button(in pos: PVector, in s: PVector, in l: String, in str: String, in c1: color in c2: color): void
+ draw(): void
+ isMouseOver(): boolean
+ isMouseClicked(): boolean
+ setFillColor(in c: color): void
SoundManager
+ minim: Minim
+ soundEffects: HashMap<String, SoundEffect>
+ addSoundEffect(in name: String, in filePath: String): void
+ playSoundEffect(in name: String): void
+ stopSoundEffect(in name: String): void
+ pauseSoundEffect(in name: String): void
+ closeAllSoundEffects(): void
ResetLevelScreen
+ resetButton: Button
+ toLevelSelect: Button
+ display(): void
+ status(in b: Button): boolean
+ handleMouseClick(): void
Checkbox
+ position: PVector
+ size: float
+ checked: boolean
+ label: String
+ boxColor, checkColor, labelColor: color
+ Checkbox(in x: float, in y: float, in s: float, in l: String, in bc: color, in cc: color, in lc: color): void
+ draw(): void
+ isMouseOver(): boolean
+ handleMouseClick(): void
Credits
+ backToMenu: Button
+ display(): void
+ handleMouseClick(): void
SoundEffect
+ player: AudioPlayer
+ isPaused: boolean
+ pausePosition: int
+ SoundEffect(in minim: Minim, in filePath: String): void
+ play(): void
+ stop(): void
+ pause(): void
+ close(): void
LevelSelectMenu
+ back: Button
+ levelButtons: Button[]
+ display(): void
+ status(in b: Button): boolean
+ handleMouseClick(): void

IV. How Does It Work?

When the program is first run, the user will be presented with a main menu screen. The main menu screen will have a "play" button that brings the user to a screen with all the levels. Clicking a level will put the user into that specific game level. The main menu screen will also have a button for "settings." Once the button is clicked, the user will be brought to a screen where the color of the square can be changed, options for sound can be adjusted (enabling and disabling), and any other graphics options (such as low/medium/high). After the player has chosen a level, there will be one square that the player can control using the WASD keys or arrow keys. The game's goal is to reach a goal zone (indicated by a green area) while collecting any available coins on the map. Some maps may not have coins, but it is required to complete the level in the levels that do have coins. In each level, there will be obstacles that the player has to dodge to reach the goal zone. Upon touching the obstacles, the player will be returned to the spawn area. If the player wants to quit the game in the middle of a level, pressing the escape key will lead to a menu where the player can choose to restart the level or return to the main screen.

V. Meeting 1 Update: Functionalities/Issues

Our group is currently working on the classes that will work together to create the game, so currently, there is no coherent “game”. Our game plan is to make all the elements we need for the game and combine them for the final product.

A. Current Functionalities

1. Button Class: Base class used for GUIs
    a. Currently, it can detect if the mouse is over it and if it is clicked or not.
2. Coin Class: Object that the player has to collect to unlock the goal area
    a. Currently, if a mouse is hovered over the coin and clicks, the coin will disappear and return the coordinates (of the coin)
3. Goal Class: Object that the player must get to to complete the level
    a. Currently, if a mouse is hovered over it, it will disappear and output “Goal Reached”
4. Player Class: Object that represents the player (square)
    a. Current functionalities right now are basic movement and collecting coins
5. Obstacle Class:
    a. Current functionalities right now is moving around and checking for collision.
6. Wall Class:
    a. Current functionalities right now are adding rectangles onto the map, which will function as walls.

B. Issues

Some issues we encountered include:

- Coin, Button, Obstacle, Player Classes: Aligning the objects (to each other) correctly on the screen with coordinates
    - Solved by trial and error, where we continuously edited and tested the code
- Button Class: Differentiating the color of the text and the buttons [Still working on a fix]
- Player Class: Handling multiple inputs at once to allow for diagonal movement [Still working on a fix]

C. Goals for the Next Meeting

1. Completing the following classes: UI, Game, MainMenu, ResetLevelScreen, LevelSelectMenu, CheckBox, SettingsMenu, SoundManager, SoundEffect, Settings
2. Begin linking the classes together: Make the player class work with coin, obstacle, wall, and goal area classes; Put together a main menu with function buttons

VI. Meeting 2 Update: Functionalities/Issues

Our group has completed most of the individual classes for the game. Our current functionalities list will include all the new classes we have created, and will only have functions from the previous functions if they were changed. All the issues from the previous meeting were solved except for the diagonal movement (we plan to look for a solution after everything has been completed due to its difficulty).

A. Current Functionalities

1. Button Class: Improved so it clicks and works like a button (instead of system outputs of true and false)
2. Player Class: Added movement methods to player class instead of driver
3. Obstacle Class: Improved so the player resets upon collision
4. Level Class: Combines coin, obstacle, wall, goal area, and player classes into one class
5. Checkbox Class: Interactive box that turns green when clicked and back to white when not clicked
6. MainMenu class: Barebones, has 2 buttons, one to close the game, the other goes to the game screen
7. SettingsMenu Class: Only has one option to enable/disable sound using checkbox class
8. Settings Class: Actually controls the settings (currently turns sound on and off)
9. SoundManager Class: Manages all sound files (sound effect class)
10. SoundEffect Class: Controls individual control over sound files
11. ResetLevelScreen Class: Barebones, has 2 buttons: one to reset the level (reset player position and coin count), and one to go back to the main menu
12. LevelSelectMenu Class: Barebones, only has buttons to traverse through the levels

B. Issues

Some issues we encountered include:

- Level Class: Classes didn't recognize each other and causes errors

    - Solved by creating a driver class
- Sound Classes: Sound library not working as intended and raising errors about the location of the sound file (issues with playback) - Not solved yet
- Unable to complete UI and Game classes due to time constraints and the classes not being developed enough to handle the way these classes link them together

C. Goals for Next Meeting/Submission

1. Complete the game → make it so all classes work together to form the game
2. Complete UI and Game classes

VII. Log

We split up the work of the whole project into two parts, interfaces and the game mechanics. Both of us worked on each of them, but Aidan focused on the actual game mechanics, while William focused on making the interface. This would also be how we structure our video presentation. When we finished all the classes, Aidan designed 2 of the 5 levels, and William designed the rest. We both contributed to making the driver file.

Aidan's Functionalities:

- Game
- Level
- Player
- Coin
- GoalArea
- Sound Manager
- Sound Effect
- Checkbox
- Wall

William's Functionalities:

- UI
- MainMenu
- SettingsMenu
- LevelSelectMenu
- ResetLevelScreen
- Credits
- Button
- Obstacle