**Software Engineering**

**Projectile Parry**

**Analysis Document**

**Contents**

**1. Introduction**

**2. Overview**

**2.1 Gameplay**

**2.2 Player Control and Parrying**

**2.3 Player HP**

**2.4 Enemies**

**2.4.1 Standard Enemy**

**2.4.2 Fast Enemy**

**2.4.3 Fire Enemy**

**2.5 Blocks**

**2.5.1 normal block**

**2.5.2 moving block**

**2.5.3 tough block**

**2.5.4 bonus block**

**3. Functional Requirements**

**4. Nonfunctional Requirements**

**5. System Models**

**5.1 Use case models**

**5.1.1 Options**

**5.1.2 Credits**

**5.1.3 Starting Game**

**5.1.4 Checking High Scores**

**5.1.5 Pausing Game**

**5.2 Dynamic Models**

**5.2.1 Sequence Diagrams**

**5.2.2 Activity Diagrams**

**5.3 Object and class models**

**5.3.1 Game app class**

**5.3.2 Player class**

**5.3.3 Projectile class**

**5.3.4 Enemy class**

**5.3.5 Block class**

**5.4 User interface – navigational paths and screen mockups**

**5.4.1 - Main Menu**

**5.4.2 - Game screen**

**5.4.3 - High scores screen**

**5.4.4 - Options screen**

**5.4.5 - Credits screen**

# **1. Introduction**

As group 4 we are developing an original video game project using object oriented programming in Java called Projectile Parry. In this report we will go over the program’s function and mechanics, the functional and nonfunctional requirements, and explain the game using diagrams and models. Models will include use case models, object and class models, sequence diagrams, and activity models. As a group we are planning for our game to have the following features.

- Fast and responsive gameplay.

- Simple and clear graphics

- 3 enemies and 3 types of shots.

- 5 levels

- Score keeping and saving

- 3 difficulty options

- Sound options

# **2. Overview**

2.1 Gameplay

Projectile Parry is an arcade style game in which a player controls a paddle and hits balls in order to destroy blocks and defeat enemies. The game features three types of enemies who have different attacks and movement. The game will have a standard enemy, fast enemy, and fire enemy. The enemies will all attack by shooting projectiles and these projectiles will damage the player if they are hit by them. The attacks can also be reflected back at enemies by the player with correct timing. A player completes a stage when every enemy is defeated and reaches a game over state when their energy is depleted.

2.2 Player Control and Parrying

The player will control a paddle in this game that will move left and right using the ‘left’ and ‘right’ arrow keys respectively. The player will start a level by launching their main projectile with the spacebar. Afterwards friendly projectiles are automatically hit back by the player touching them. When any projectile gets close to the player during a level they can press the spacebar to hit the ball harder. If this is done with correct timing all projectiles will be launched back and do more damage. This also turns enemy attacks into friendly balls.

2.3 Enemies

2.3.1 Standard Enemy

The standard enemy in our game will stay still and take two hits from the player to be defeated. They will fire a standard shot that moves at a normal speed. When parried by the player the shots will turn into a standard ball similar to the players main projectile.

2.3.2 Fast Enemy

The fast enemy will be a moving enemy that will move in a set area left and right across the screen. Every few seconds this enemy will shoot a fast moving projectile that will be harder for the player to parry. When parried by the player they will get a faster moving shot.

2.3.3 Fire Enemy

A fire enemy moves left and right like the fast enemy but not as fast. The fire enemy launches strong fireballs that do a lot of damage if they hit the player. If the player parries the fireballs they become fire shots which burn through blocks without bouncing off them.

2.4 Blocks

The game will have different kinds of blocks.

2.4.1 - Standard block: Will be removed with any hit.

2.4.2 - Tough block: A block that takes two normal hits but only takes one hit from projectiles that the player has hit with good timing.

2.4.3- Moving block: A block that moves left to right in a set path. Takes only one hit.

2.4.4 - Bonus block: Same attributes as the normal block but gives more points.

# **3 Functional Requirements**

**Game Screen Setup:**

1. The game must provide a screen to display the game. The screen should show a paddle at the bottom of the screen, a ball that bounces off the walls and breaks bricks, and enemies that fire projectiles at the player. The game should start automatically when the screen loads after pressing Play in the main menu.

**Paddle Movement:**

1. The game must allow the player to move their paddle left and right using keyboard controls. The movement should be smooth and responsive.

**Ball Movement:**

1. The ball should move randomly in any direction when the game starts. The ball must move continuously until it hits a wall, a brick, or the paddle. Once the ball hits a wall or a brick, it should bounce off in the opposite direction. If the ball hits the paddle, it should bounce off at an angle based on where it hit the paddle.

**Brick Breaking:**

1. The game must detect when the ball hits a brick and destroy it. The game should have different types of bricks with varying levels of toughness that require multiple hits to break.

**Enemy Projectiles:**

1. The game should have enemies that fire projectiles at the player. The projectiles should move in a straight line towards the player's paddle. If the projectile hits the paddle, it should destroy the paddle and end the game.

**Parry Mechanic:**

1. The game should allow the player to parry the enemy projectiles using their paddle. If the player hits the projectile with their paddle, it should bounce off in a different direction towards the enemy, potentially destroying it. Parrying the projectile should require precise timing and skill.

**Score Keeping:**

1. The game must keep score for the player. The player should receive points for breaking bricks and destroying enemies. This score should then be added to a High Scores list viewable in the main menu.

**Level Progression:**

1. Upon clearing the bricks and destroying the enemies in a given level, the player should be informed that they cleared the level, and the game will progress onto the next one.

**Game Over:**

1. The game must end when the player runs out of health or when they lose their paddle. The game should display a message announcing the player's score and give the player an option to restart the game.

**Sound Effects:**

1. The game should provide sound effects for when the ball hits a wall, brick, or the paddle. The game should also have background music during gameplay. The game should have unique sound effects for when the player parries a projectile.

**Difficulty Levels:**

1. The game should offer different difficulty levels, such as easy, medium, and hard. Each level should change the speed of the ball and the enemies' firing rate.

**Pause and Resume:**

1. The game should provide an option to pause the game at any time, and resume when the player is ready to continue.

**Settings:**

1. The game should allow the player to change settings such as volume, difficulty, and also color scheme.

**Credits:**

14. The game will have credits that are viewable from a button on the main menu. This will display the game developers for users to view.

**User Interface:**

1. The game should have a user-friendly interface that is easy to navigate and understand. The main menu should have clear instructions on how to play and how to navigate through the settings.

**Compatibility:**

1. The game should be compatible with different operating systems such as Windows, Mac, and Linux, and should be able to run on different screen resolutions. (Thanks to Java and LWJGL this shouldn’t be a hitch.)

# **4. Non-Functional Requirements**

**Performance:**

1. The game should be optimized to run smoothly on a range of hardware configurations. It should not consume excessive system resources or cause significant lag or slowdown during gameplay.

**Responsiveness:**

1. The game should respond to user input quickly and accurately, with minimal delay or lag. The movement of the paddle, ball, and projectiles should be smooth and fluid.

**Reliability:**

1. The game should be stable and reliable, with no crashes or freezes during gameplay. The game should save the current level and restore it if the game is closed unexpectedly.

**Security**:

1. The game should not have any security vulnerabilities that could be exploited by malicious users. The game should not access or modify any system files or sensitive data.

**Accessibility:**

1. The game should be accessible to a wide range of users, including those with disabilities. The game should have customizable color schemes to accommodate users with visual impairments.

**Usability:**

1. The game should have an intuitive user interface that is easy to use and navigate. The game should provide clear instructions on how to play, and how to navigate through the settings. The game should also have a help section for new users.

**Portability:**

1. The game should be portable, meaning it can run on different platforms and environments without any issues. The game should be developed using platform-independent technologies and should not have any platform-specific dependencies.

**Maintainability:**

1. The game code should be well-organized and structured, making it easy to modify and maintain in the future. The game should also follow coding best practices and design patterns to improve readability and reduce complexity.

**Localization:**

1. The game should support multiple languages and be easily translatable. The game should have a localization file that can be modified to add new languages or modify existing ones.

**Graphics and Sound:**

1. The game should have acceptable graphics and sound effects that enhance the gameplay experience and should not be distracting or overwhelming for the player.

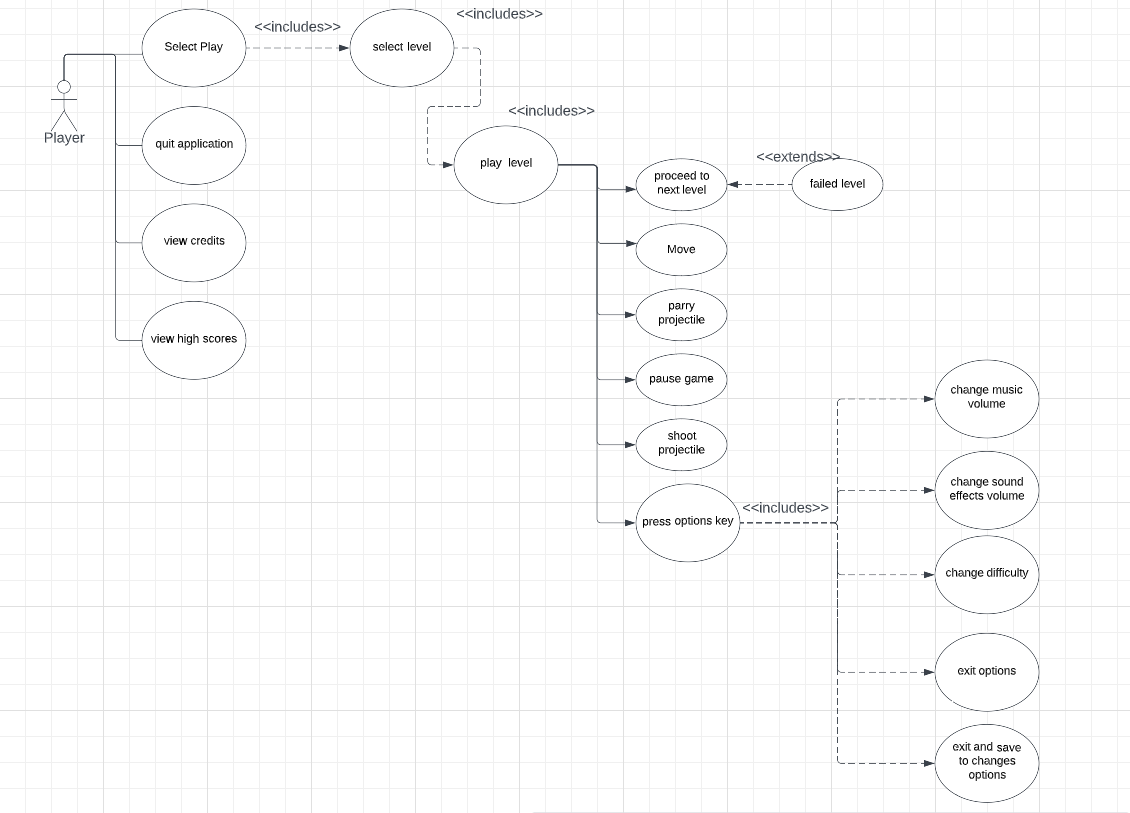
**Compatibility:**

1. The game should be compatible with different input devices, such as keyboards and gamepads. The game should also work well on different screen sizes and resolutions.

**Data Management:**

1. The game should manage player data such as scores, settings, and preferences. The game should store this data in a separate file which is not modifiable by the user (storing the serialized scores).

# **5 System Models**



**5.1 Use case diagram**

**5.1.1** **Use Case** #**1:** Options

Primary Actor: Player

Stakeholders and Interests

- Player wishes to change the volume because they find the game too quiet or loud or wishes to change the game’s difficulty.

Entry Condition

- Player is at the main menu for Projectile Parry

- Player selects “Options” button from the main menu

Exit Condition

- Player selects the “Exit” button from the options menu.

- Player selects the “Save and Exit” button from the option menu

Flow of Events

1. Player selects the sound setting from the menu.
2. Player changes the sound either up or down
3. Player hits save and exit button to leave the menu with changes applied.

Alternative flow of events

1. Player may want to change difficulty instead
   1. Player selects easy, medium, or hard difficulty options.
   2. Player hits the save and exit button to leave the menu.
2. Player may decide against changing any options.
3. Player hits exit button to leave the menu without making any changes

**5.1.2 Use Case #2:** Credits

Primary Actor: Player

Stakeholders and Interests

- Player wants to look at the developers of the game and their contributions

- Game wants to provide the player with a list of who created the program.

Entry Condition

- Player is at the main menu for Projectile Parry

- Player selects the “Credits” button from the main menu

Exit Condition

- Player selects the “return to menu” option in the credits screen

Flow of events.

1. The system presents the credits screen of the game showing all developers, asset sources, and special thanks.

Alternative Flow of Events

1. Player wants to exit the credits and return to the main menu.
2. Player selects the “return to main menu” option in the credits screen.
3. System presents the main menu screen to the player.

**5.1.3 Use Case #3:** Starting Game

Primary actor: Player

Stakeholders and interests

-player wants to start the game application

Entry Condition

-Player opens up the file location the game application is located in

Exit condition

-Player closes the game application

Flow of events

1.the player opens the file location for the game application

2.the player launches the game application

Alternative flow of events

-player wants to close the game application

1.the player returns to the main menu

2. The player selects the “exit game” option

**5.1.4 Use case #4:** Checking High Scores

Primary actor: Player

Stakeholders and interests

-Player wants to check the list of high scores earned in various playthroughs of the game

Entry Condition

-Player selects the “view high scores” option in the main menu

Exit condition

-Player selects the “return to main menu” option in the credits screen

Flow of events

1.the player enters the main menu

2.the player selects the “view high scores” option

Alternative flow of events

-player wants to exit the “view high scores” screen

1.the player selects the “return to main menu” option in the “view high scores” screen

**5.1.5 Use case #5:** Pausing the Game

Primary actor: Player

Stakeholders and interests

-player is in the process of playing a level and needs to pause the game.

Entry condition

-the player then presses the pause button while inside of a level

Flow of events

1.From the main menu, the player selects a level to play

2.The player presses the “pause” button or the keyboard input for pausing the game.

Alternate flow of events

-player wants to unpause the game

1.the player will press the “pause” button again or re-enter the same keyboard for pausing the game

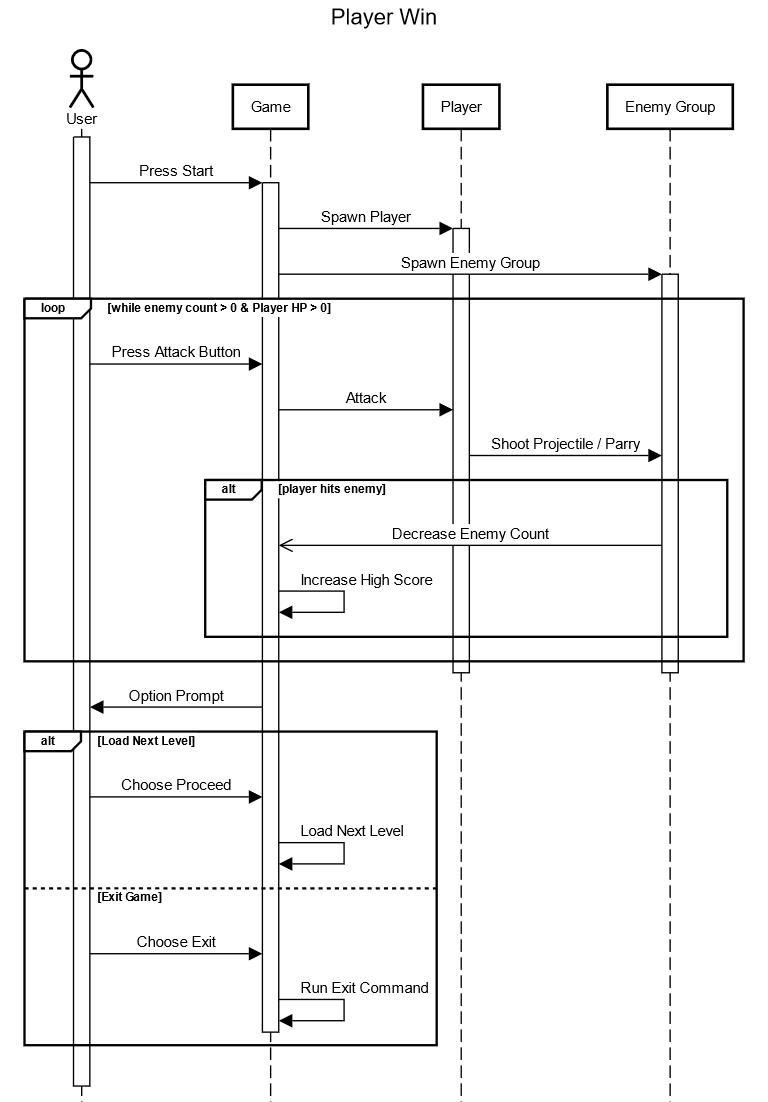
**5.2 Dynamic Models**

**5.2.1 Sequence Diagrams**

**5.2.1.1 Player Win**

**Scenario:** Playing a stage of the game

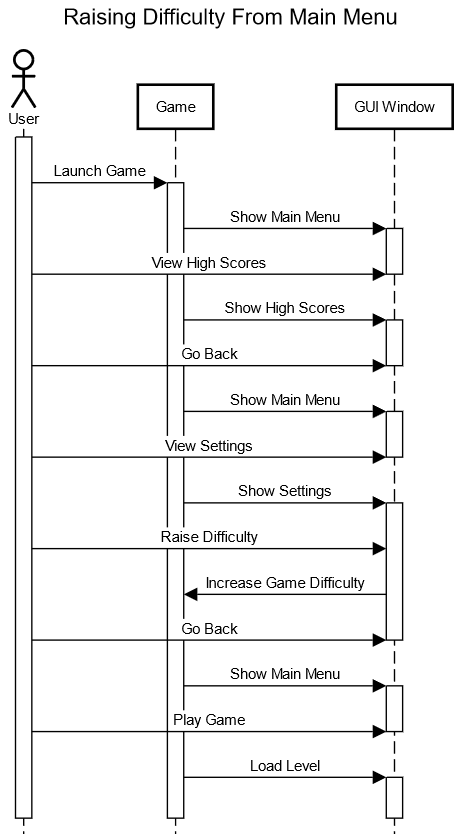
User wants to play the game. They press start and the game initializes spawning in the player using the player class and the enemy group and obstacles using the enemy and block classes. The player starts playing the game by hitting the attack button which launches the first shot. As the player hits enemies the game decrements its enemy count. When the enemy count hits 0 the player will win the level and the game will update its high score if needed before prompting the player with the option to either continue to the next level or quit.

****

**5.2.1.2 Checking Scores and Raising Difficulty**

**Scenario:** User enters high score and settings screens

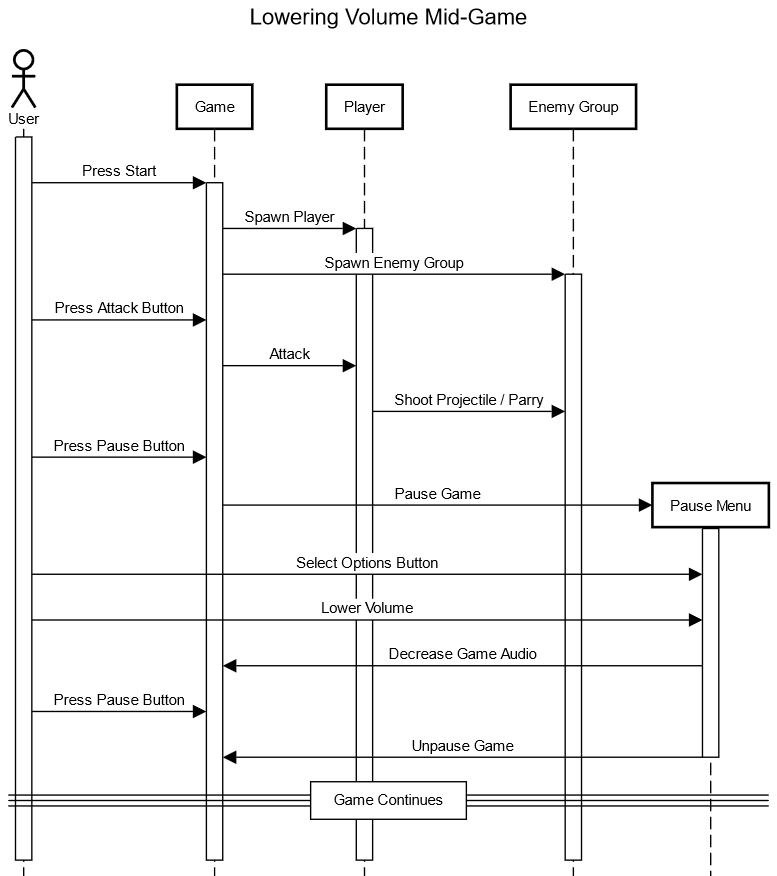
User wants to view the high scores of previous players of the game, so they select the high scores option from the main menu. The GUI Window shows the high scores to the user who then exits the menu which has the game return the main menu to the user. The user then hits the settings option to view the games settings. They hit the difficulty button to change the game’s difficulty which causes the GUI Window to update the game. The player hits back again causing the GUI Window to return the main menu again. The user then hits play game causing the game program to load a level

****

**5.2.1.3 Lowering Volume Mid-Game**

**Scenario:** The user pauses the game to change the volume.

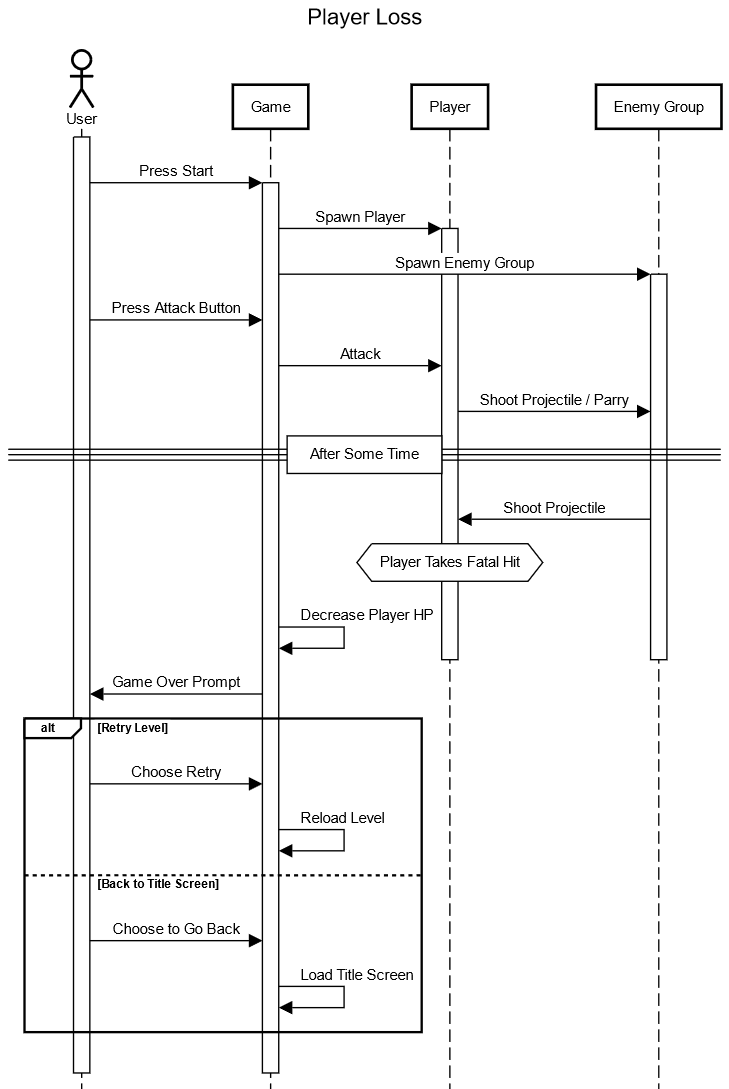
After the user has started the game and the player and enemy group has been spawned in the player hits the pause key to bring up the pause menu interface. From this the user presses the options button which brings up the volume panel from the games GUI class. This tells the game to decrease its audio. The player then presses the pause button again which continues the game.

****

**5.2.1.4** **Player Loss**

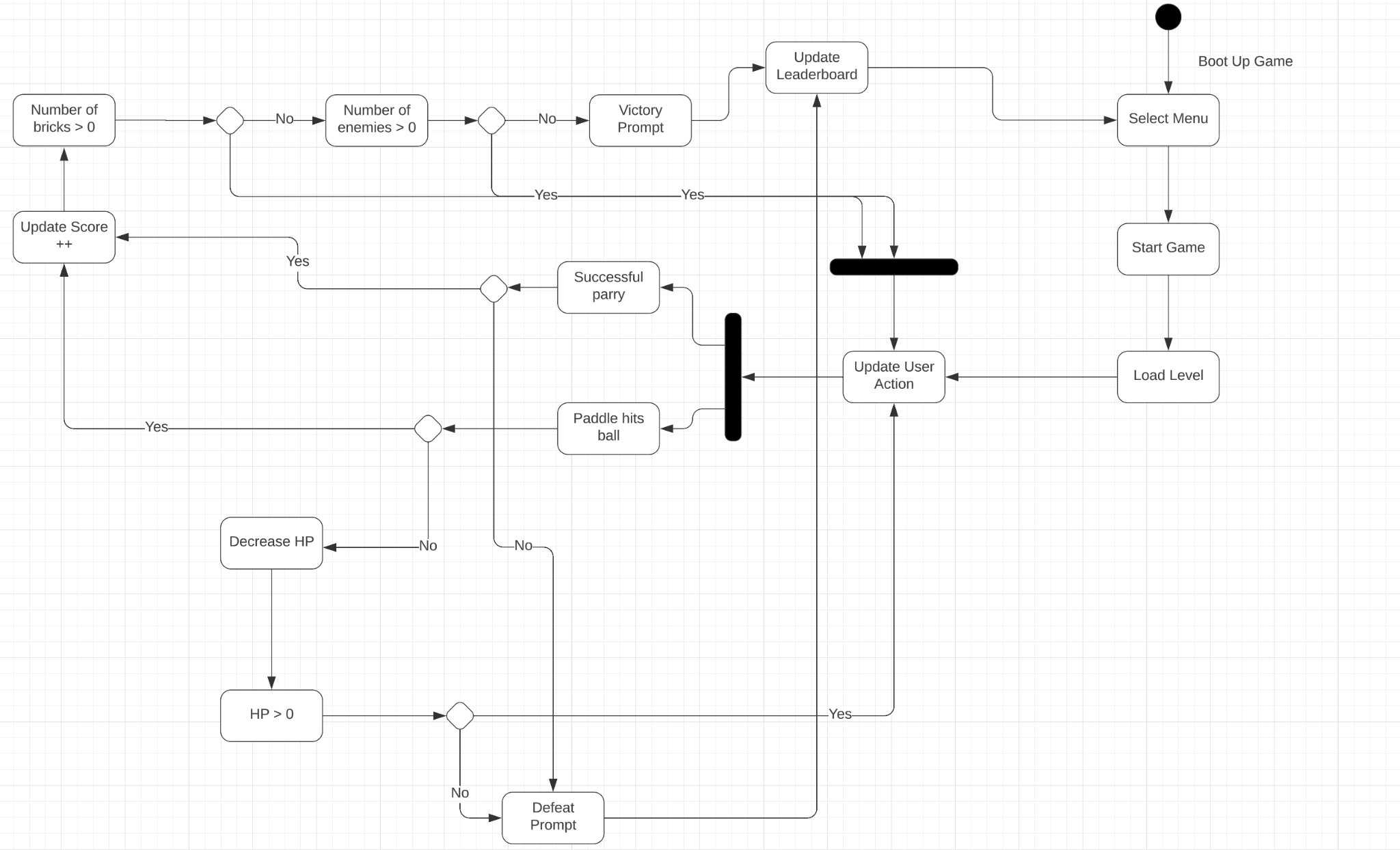
**Scenario:** The user fails at the game’s objective.

After the game has been initialized and the user has been playing for some time the enemy group fires a shot which hits the player. This causes their health to run out and causes the game to display the game over prompt using the GUI class. The user then can choose retry causing the game to reload the level or choose to quit which has the game load the title screen.



**5.2.2 Activity Diagrams**

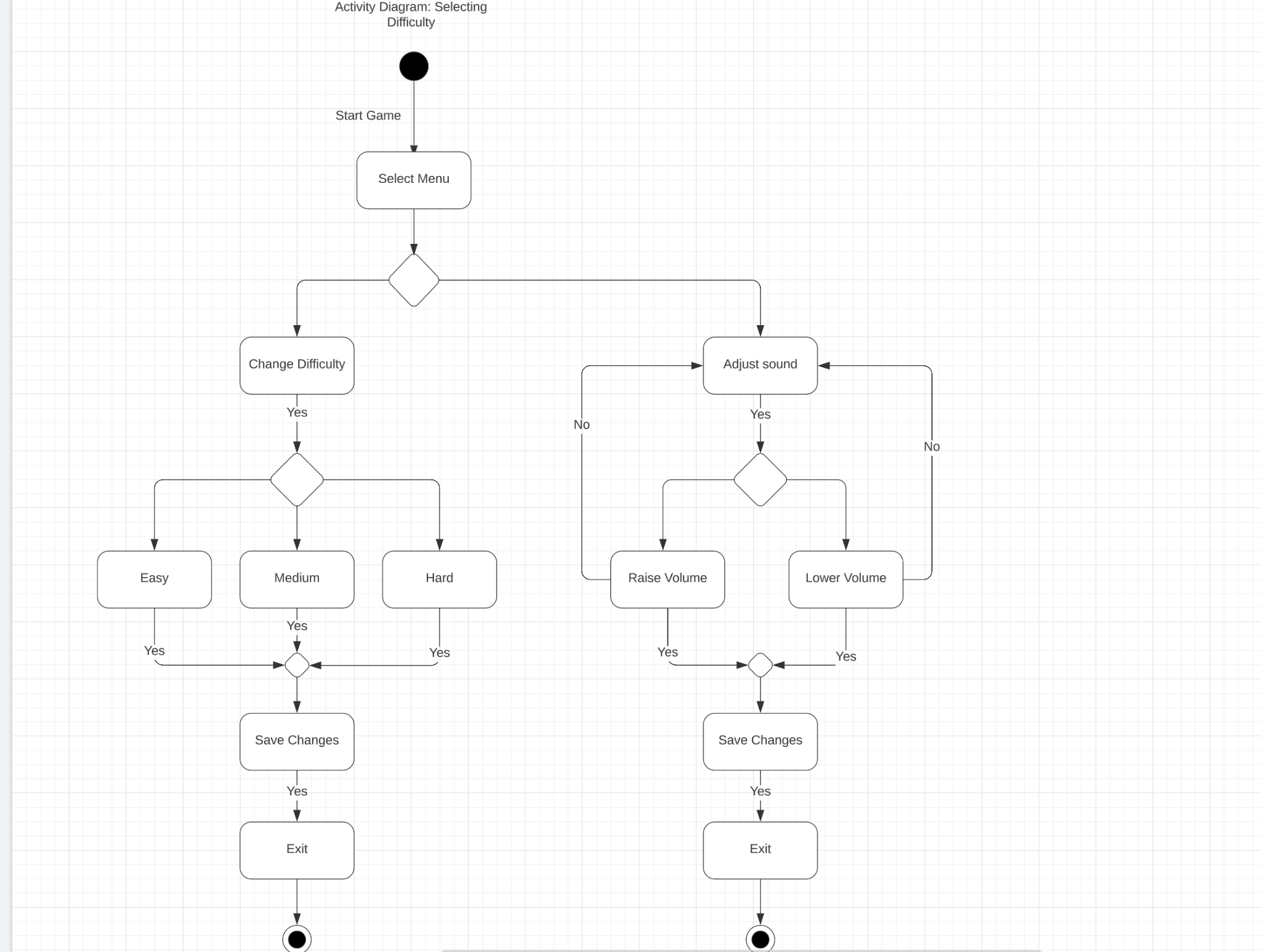
**5.2.2.1 Playing the game**

****

**Description**

The user begins by booting up the game and is presented with the Menu screen. From here the user can choose between 3 prompts, changing the difficulty, adjusting the sound level and starting the game. Once the user chooses to start the game, the user must load the desired level and must begin playing. The user must move his paddle. Upon moving his paddle, several actions take place. He may move the paddle so the ball releases bounces and hits a brick or an enemy. Hitting a brick will increase the score of the game, hitting an enemy will decrease the enemy’s HP. Once all bricks have been cleared and all enemies have reached the HP of 0, the user has beat the level. Once the user has beat the level, a victory prompt appears, the leaderboard is updated, and the user exits the game back to the main menu. The enemy is also able to shoot projectiles and if the paddle is hit, the user loses the level, a defeat prompt is displayed on the screen, and is then sent back to the main menu. The user is able to parry enemy projectile’s, bouncing off the projectile and dealing damage to enemies and breaking blocks. Parrying enemy projectiles and hitting enemies increases score.

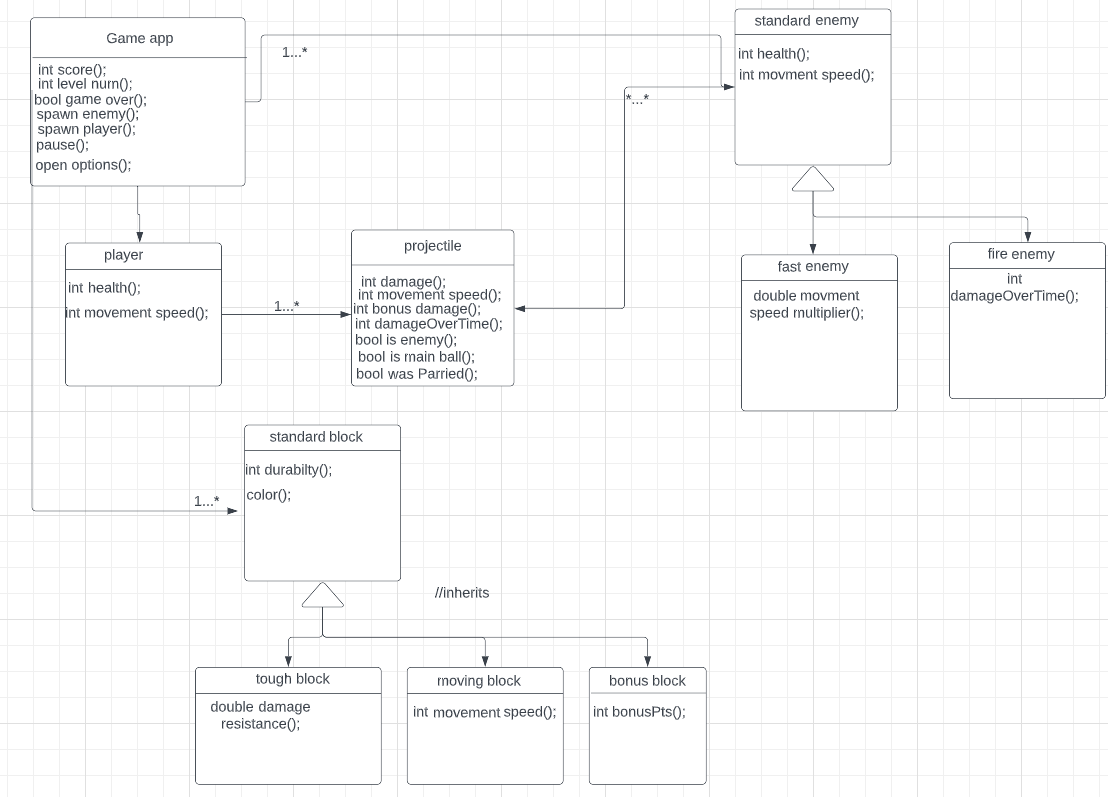
**5.2.2.2 Change Difficulty**

****

**Description**

The user begins this process by starting the game. The user is presented by the menu and has two options to choose from. The user may choose to change the difficulty or adjust sound. If the user chooses to change the difficulty, he has 3 options to choose from. The user can choose the easy difficulty, the medium difficulty, and the hard difficulty. The user will be asked to save the changes and is then prompted to exit back to the main menu. If the user wants to adjust sound, he can choose to raise the volume and lower the volume. The user is then asked to save his changes and is then prompted to exit back to the main menu.

**5.3 Class Diagrams**

****

**5.3.1 Game app class**

This class will handle most of the environment the game takes place in. It will be able to spawn in the player and enemies. It will also be able to pause an ongoing game and open the options menu. There will also be ways to keep track of the level the player is on and a way to view the current score, as well as a way to cause a game over if the player’s health reaches zero.

**5.3.2 player class**

This will hold values associated with the paddle the player is controlling. There is a health value and a movement speed value.

**5.3.3 projectile class**

This class will handle the projectiles that the players and enemies shoot at each other. There is a base damage that will vary depending on whether they belong to the player or an enemy. This base damage can be increased by a damage bonus (via a parry) or shot by a fire enemy (whose projectile can become the player’s if parried). The player will have a main ball that they must keep on screen or else they will take damage.

**5.3.4 enemy class**

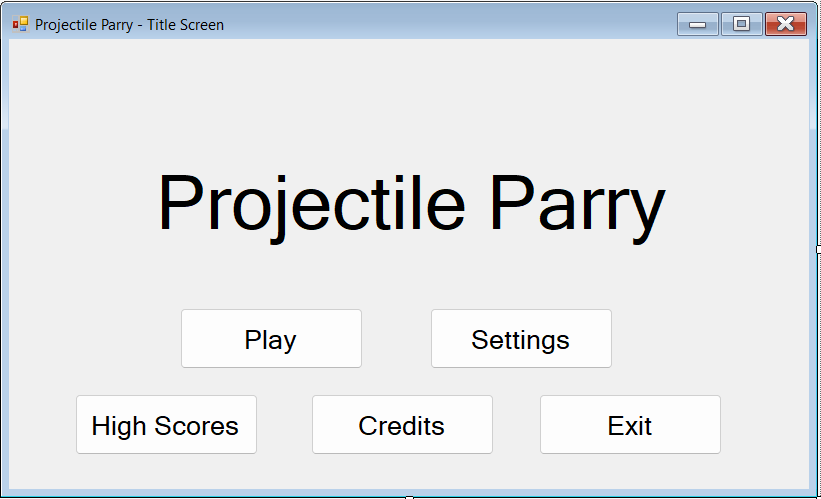
This class will handle the various enemies that are encountered in the game. There is a generic “enemy” class that has a health and movement speed attribute. There are enhanced enemies that are derived from the enemy class. Fast enemies will have a movement speed multiplier and fire enemies will fire projectiles that deal additional damage over time if the player allows themselves to get hit by them.

**5.3.5 block class**

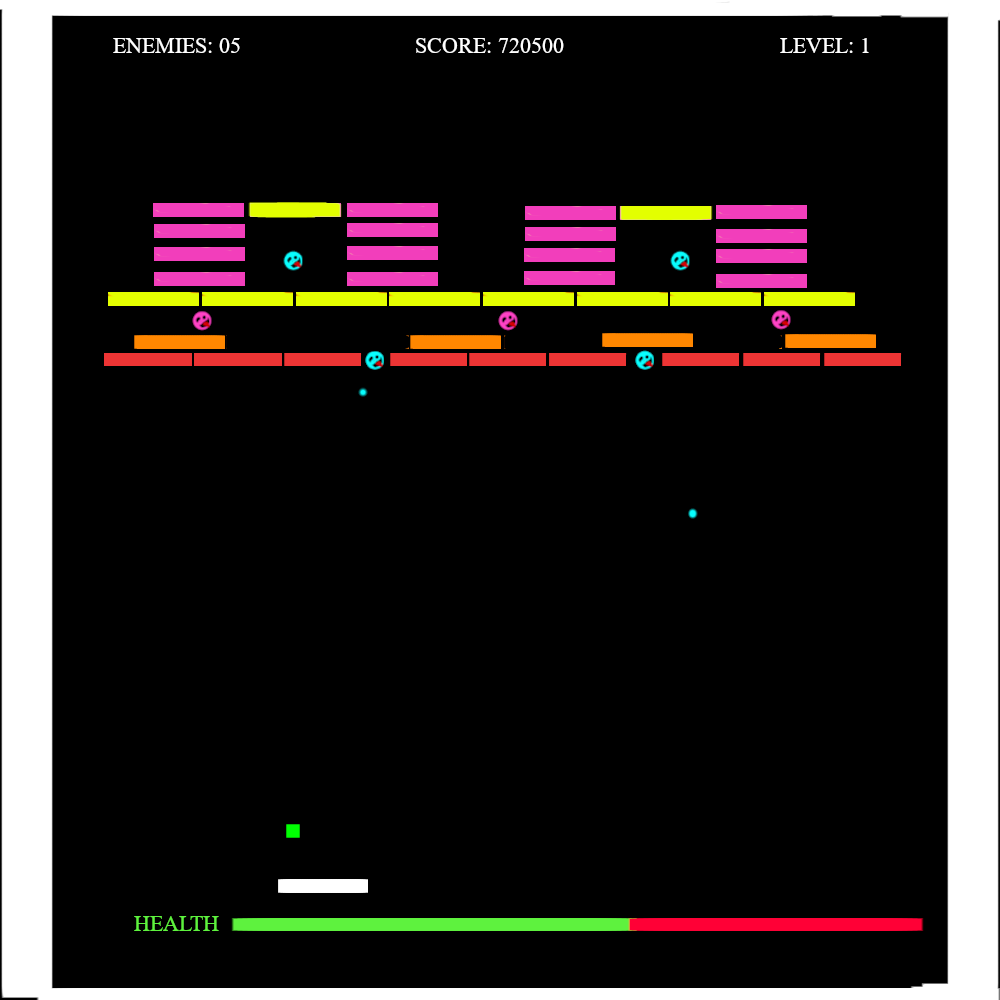
This class will handle the various blocks that are present in the levels of the game. The player is required to destroy all the blocks (via hitting them with projectiles) alongside defeating all the enemies. There is a generic “block” with a durability value and a color. The color of the block will indicate what type of block it is. There are other blocks that are derived from the block class. Tough blocks have a damage resistance value which a projectile’s damage to it is multiplied by a value (ex-0.50). There are moving blocks that move around the screen and there is a bonus block that rewards bonus points for their destruction.

**5.4 Screen Mock-Ups**

**5.4.1 Main Menu:** From this title screen players can select different options such as starting the game, closing the program, changing the settings, and viewing high scores.

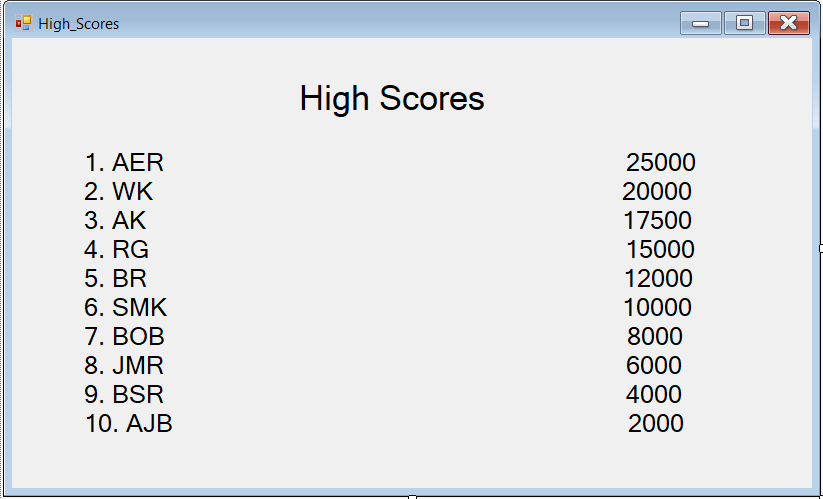
**Figure 5.4.1**

**Game Screen:** The screen the player sees in playing the game. Info shows the health, enemies, score, and which level the player is playing.



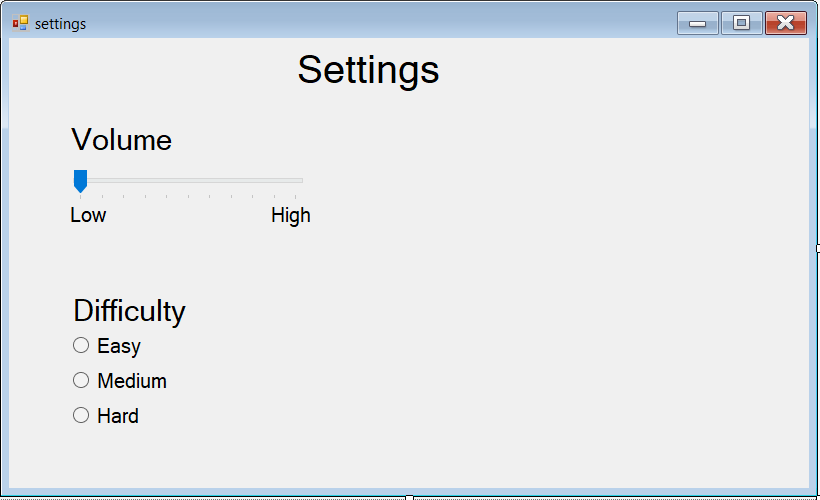
**Figure 5.4.2 Game Screen**

**High Scores:** The player can view the top ten highest scores of players in the game.



**Figure 5.4.3 High Scores**

**Settings:** The screen that enables the player to change the volume and difficulty settings



**Figure 5.4.4 Settings Screen**

**Credits:** The user can view the names of those that developed the game here.

****

**Figure 5.4.5 Credits Screen**