USER MAUAL – PROJECT 1

Penalty Shootout

Team Hemerocallis

Table of Contents

1 PROJECT 1 2

1.1 PROJECT DESCRIPTION 2

1.2 HOW TO PLAY (USER MANUAL) 2

1.3 GAME MECHANICS & LOGIC 3

1.4 TECHNICAL IMPLEMENTATION 3

1 Project 1

1.1 Project Description

This project is a simple interactive penalty shootout game developed using Processing (Java mode).
The player takes penalty shots by aiming and controlling the shot power, while a goalkeeper attempts to save the ball. The outcome of each shot is determined using a probability-based logic, rather than complex physics, to keep the game simple and focused on interaction and decision-making.

The game includes a difficulty selection menu, different goalkeeper behaviors, and basic animations to visually communicate game states such as goal, save, or miss.

Tools Used:

- Processing (Java mode)
- Object-oriented programming (classes)
- Keyboard interaction

1.2 How to Play (USER MANUAL)

Menu Controls:

1 / 2 / 3 → Select difficulty
    1 = Easy
    2 = Normal
    3 = Hard
ENTER → Start the game

In-Game Controls:

Left / Right Arrow → Aim the shot
Up / Down Arrow → Adjust shot power
Spacebar → Shoot the ball
R → Take the next shot (after result)
M → Return to menu
X → Full reset (score and history)

Game Flow:

- Select difficulty from the menu
- Aim and adjust shot power
- Shoot the ball
- View result (Goal / Save / Miss)
- Repeat for multiple shots

1.3 Game Mechanics & Logic

Penalty Shootout Logic:
The ball moves in a straight trajectory toward the goal after the player shoots.
When the ball crosses the goal line, the game checks:

- Whether the ball is inside the goal frame
- The horizontal distance between the ball and the goalkeeper

Probability-Based Outcome:

Instead of physical collision detection, the game uses probability:

- If the ball is close to the goalkeeper, the chance of scoring is low
- If the ball is far from the goalkeeper, the chance of scoring is high

This approach keeps the game fair, readable, and easy to tune.

Difficulty Levels

Difficulty affects:

- Goalkeeper movement speed
- Goalkeeper commitment accuracy
- Minimum and maximum goal probability

This allows the same mechanics to feel different across difficulty levels.

1.4 Technical Implementation

Object-Oriented Design

The game is structured using multiple classes:

- Player
    - Handles aiming, power control, and user input.

- Ball
    - Manages ball movement, trajectory, and goal-line crossing detection.
- Goalkeeper
    - Commits early to a dive direction and performs a dive animation based on the selected difficulty.
- Goal
    - Defines the goal dimensions and checks whether the ball is inside the goal area.

Arrays / ArrayLists

Arrays are used to:

- Store shot positions
- Track goal vs save history
- Display recent shot outcomes

This allows easy extension and visualization of past shots