# Angry Birds
Built with PYGAME

**Sparsh Gupta**
**24B1043**

CS104 Project Report

Spring Semester, 2024–2025

---

**Abstract**

This report presents the design and development of **Angry Birds**, a 2D game developed using Python and the PYGAME library. The project is inspired by the classic Angry Birds mechanics and serves as an exploration into basic game physics, collision detection, and gameplay logic. The report discusses the modules used, directory structure, core game features, customizations and challenges faced, with the aim of providing insight into the game's functionality and underlying architecture.

## Contents

## 1 Introduction

*Angry Birds* is an interactive 2D game where players launch a bird using a slingshot to destroy their opponent's tower using the minimum possible number of birds. Developed using PYGAME, the game aims to replicate essential elements of physics-driven gameplay such as parabolic motion and impact dynamics.
The primary goal of the project was to learn and implement concepts related to object-oriented programming, 2D rendering, input handling, and basic physics.

The player can choose between two themes **- Day Theme** and **Night Theme**. The theme selection does not alter gameplay mechanics but adds immersion and visual variety. The game maintains a stable frame rate of 60 FPS (frames per second), ensuring an overall fluid gameplay experience.

## 2 Modules Used

- `pygame-ce` – Core library used for rendering graphics, handling events, audio playback, and game loop management.
- `math` – Provides trigonometric and mathematical functions essential for bird trajectory and slingshot calculations.
- `numpy` – Used for random bird generation.
- `sys` – Handles program exit.
- `time` – Used for time-based interactions, animations and input delay.

## 3 Project Directory Layout

The structure of the project is organized for clarity and modularity:

```
Angry Birds
├── data
│   ├── audio
│   ├── font
│   ├── images
│   ├── leaderboard.txt
│   ├── tower1.txt
│   └── tower2.txt
├── scripts
│   ├── bird.py
│   ├── block.py
│   ├── tower.py
│   └── utils.py
└── main.py
```

- **main.py** – Launches the application, sets up initial game state, controls the game loop, rendering and animations.
- **bird.py** – Contains the Bird class.
- **block.py** – Contains the Block class.
- **tower.py** – Contains the Tower class.
- **utils.py** – Contains utility functions that support core gameplay logic, such as vector and angle calculations, sling action, collision helpers.

## 4 Running the Game

### 4.1 Requirements

To run the game, the following setup is required:

- Python 3.9 or newer
- pygame library

Installation can be done via pip:

```
pip install pygame
```

### 4.2 Execution

Navigate to the project directory and run:

```
python main.py
```

## 5 Game Navigation and Gameplay

### 5.1 Intro Screen

The game starts with an intro screen with a loading bar.

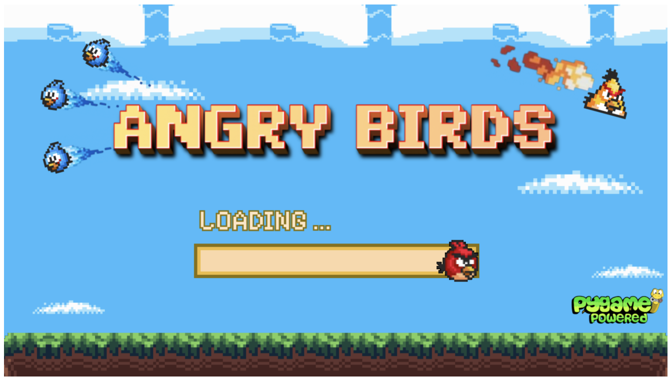


Figure 1: Loading Screen

### 5.2 Main Menu

Once the loading bar is complete, players are presented with a Main Menu interface offering the following options:

- **Play** – Start the game and proceed to entering player names.
- **leaderboard** – Display names of top 3 players for each mode.
- **Settings** – Toggle sound on/off and Adjust gameplay preferences.
- **Quit** – Quit the game and return to the desktop.

These options are accessible using the mouse, and the menu interface is designed to be intuitive and responsive across various screen resolutions.

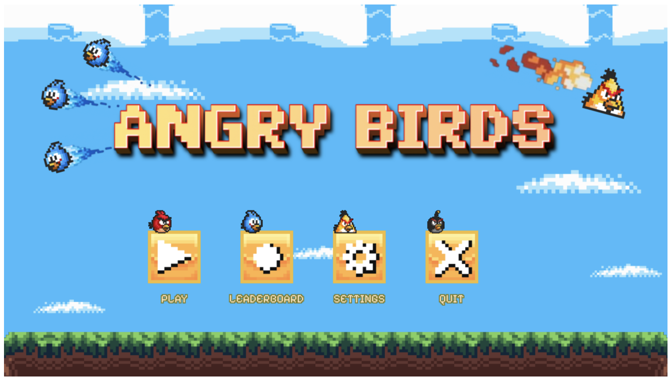
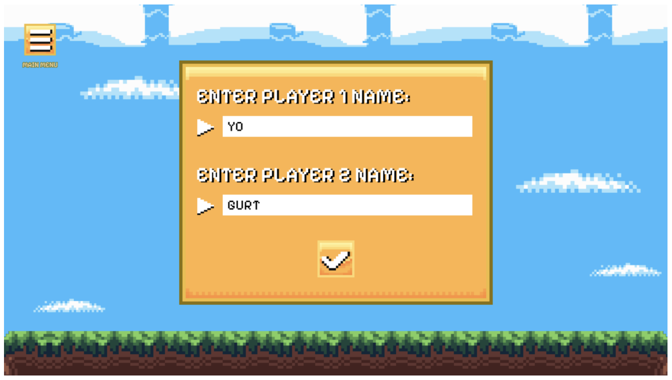


Figure 2: Main Menu

## 5.3 Enter Player Name

On clicking the play button, the players are presented with an interface for entering their names.



ENTER PLAYER 1 NAME:
YO
ENTER PLAYER 2 NAME:
BURT

Figure 3: Enter Player Name Screen

## 5.4 Mode Selection

After entering player names, the players have to choose among two game modes - One tower mode, Two tower mode.

## 5.5 Gameplay

The player is taken to the selected game mode, where the birds are loaded onto each slingshot. The next bird for each player is generated randomly. To aim, the player clicks and drags the bird away from the slingshot origin. As the drag vector increases, the stored launch velocity increases proportionally, mimicking a stretched elastic band. Once released, the bird follows a parabolic trajectory. The player which destoys the other player towers first is declared the winner and his/her leaderboard ranking is calculated using the number of birds used to destroy the tower. *Trajectory dots* to see the path of the bird before launching and *wind* can be enabled in **Settings**.

Upon impact with a block, the bird's velocity changes according to realistic physics logic. The player which plays first is decided randomly using numpy's random module. A tower consists of one or two columns of blocks with 10 blocks in each column. The health of each block is 100.

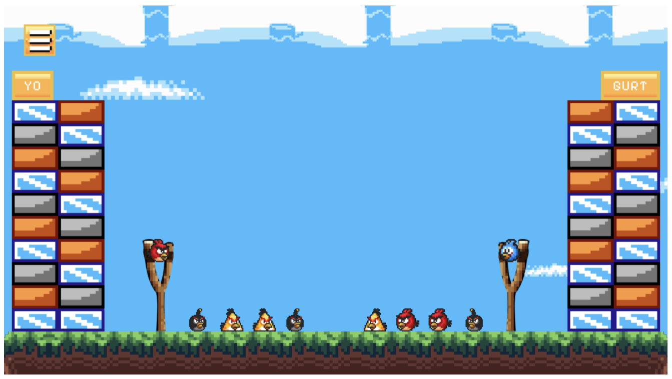


Figure 4: Main Gameplay (Two Tower Mode)

**Birds without Special Abilities:**

- **Red Bird** – It does balanced damage (50) to each type of block.
  Damage - Ice: 50, Wood: 50, Stone: 50
- **Blue Bird** – Damage - Ice: 50, Wood: 25, Stone: 25
- **Yellow Bird** – Damage - Ice: 25, Wood: 50, Stone: 25
- **Black Bird** – Damage - Ice: 25, Wood: 25, Stone: 50

**Birds with Special Abilities:**

These abilities can be triggered mid-flight by pressing a key (e.g., Spacebar).

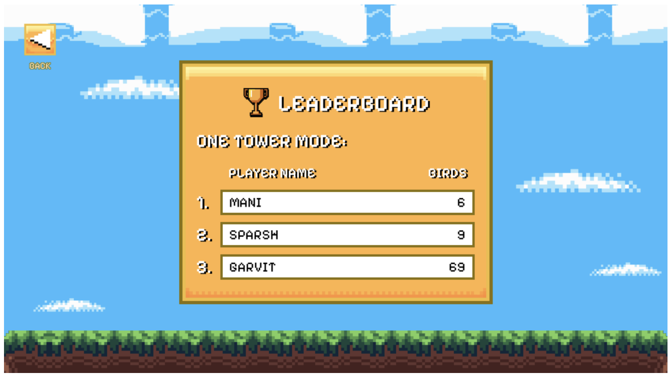
- **Blue Bird (Splitter)** – Splits into three blue birds mid-air, allowing a spread attack ideal for fragile or widely spaced blocks.
  Damage - Ice: 50, Wood: 25, Stone: 25
- **Yellow Bird (Speed Boost)** – Accelerates sharply forward when activated, increasing its impact damage significantly.
  Damage - Ice: 50, Wood: 100, Stone: 50
- **Black Bird (Bomb)** – Explodes mid-air, dealing area-of-effect damage ideal for damaging blocks within a short radius.
  Damage - Ice: 25, Wood: 25, Stone: 25

## 5.6 Game Result

If one player tower is completely demolished, winner's name is displayed on the screen and added to the leaderboard depending on the number of birds used.

## 5.7 Leaderboard

Each mode maintains its own independent leaderboard system, each lists the names of the top 3 winners along with the number of birds used.



LEADERBOARD

ONE TOWER MODE:

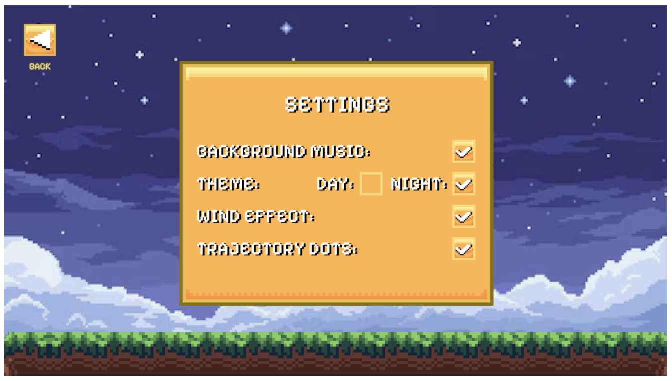
| | PLAYER NAME | BIRDS |
|---|---|---|
| 1. | MANI | 6 |
| 2. | SPARSH | 9 |
| 3. | GARVIT | 69 |

Figure 5: Leaderboard for One Tower Mode

## 5.8 Settings

This allows the user to set various game parameters according to their preferences.

- **Background Music** – Allows the user to turn on/off background music.
- **Theme** – A visual toggle that allows users to switch between a **Day Theme** and a **Night Theme**.
- **Wind Effect** – Introduces lateral wind forces influencing the trajectory of launched birds.
- **Trajectory Dots** – To aid aiming, the game includes an optional **Trajectory Guide** system that visually predicts the bird's initial path after launch.



SETTINGS

BACKGROUND MUSIC: ✓

THEME: DAY: ☐ NIGHT: ✓

WIND EFFECT: ✓

TRAJECTORY DOTS: ✓

Figure 6: Settings

## 5.9 Quit

Players can quit the game by clicking on the quit button.

# 6 Customizations

- Themes - Day Theme and Night Theme
- Wind - Alters the bird's horizontal velocity
- Special abilities - Blue Bird (Splitter), Yellow Bird (Speed Boost), Black Bird (Bomb)
- Bird Sounds - Bird launching and special ability sound effects.
- Game Modes - One Tower Mode and Two Tower Mode

- Guiding Trajectory Dots - To aid aiming (can be enabled/disabled in *Settings*)
- Dynamic Block Damage Representation - Four distinct sprites that update at 25% health intervals.
- Cloud Animations - Clouds move periodically if wind is enabled.

## 7 Challenges Faced

- **Physics Accuracy:** Ensuring realistic projectile motion and collision response without external engines like Pymunk.
- **Bird Launch Smoothness:** Fine-tuning drag mechanics and velocity calculations to eliminate launch jitter.
- **Collision Detection:** Handling bird collisions with blocks and based on which side of block was hit, changing velocity vector of bird.
- **UI Responsiveness:** Creating a responsive interface that transitions cleanly between menus, gameplay, and end states.

## 8 References

- Pygame Documentation: `https://www.pygame.org/docs/`
- Pygame CE Documentation: `https://pyga.me/docs/`