Angry Birds/LOTR PvP: A Two-Player Projectile-Based Fortress Battle

Parthiv
Course: CS 104/108

April 29, 2025

Abstract

This report details the development of a sophisticated two-player Angry Birds-inspired game with both classic and Lord of the Rings themes. The project implements newtonian physics simulation, procedural content generation, and complex game state management using Pygame and NumPy. Key features include multi-theme support, dynamic block texture generation, and special ability projectiles. The system demonstrates collision resolution, texture generation, and theme-aware object management.

1 Architectural Overview

1.1 Modules Used

- **numpy**: For efficient numerical computations in physics simulations.
- **math**: For certain functions used in dragging simulation and other places.
- **random**: For procedural generation of textures and projectile selection.
- **time**: For managing game timers and turn durations.
- **pygame-ce**: For rendering graphics, handling input, and game loop management.

1.2 Directory Structure

Below is the directory structure of the project. A summary table with file purposes is provided for reference.

```
main.py
Rohan_The_Refs/
  |-- background/
  |-- dialogue/
  |-- fonts/
  |-- music/
  |-- projectiles/
Gondor_The_Code/
```

```
|-- block.py
|-- button.py
|-- const.py
|-- fortress.py
|-- game.py
|-- inputbox.py
|-- level_manager.py
|-- projectile.py
|-- slider.py
|-- texturegenerator.py
|-- togglebutton.py
```

File/Folder	Purpose
`main.py`	Main entry point
`Rohan_The_Refs/`	Contains all non-Python elements
`Rohan_The_Refs/background/`	Contains the background and catapult images for the g
`Rohan_The_Refs/dialogue/`	Contains audio snippets to be used
`Rohan_The_Refs/fonts/`	Contains fonts
`Rohan_The_Refs/music/`	Contains music
`Rohan_The_Refs/projectiles/`	Contains projectile images
`Gondor_The_Code/`	Contains all Python elements
`Gondor_The_Code/<insert name here>.py`	Contains `<insert name here>` class

2 Execution Protocol

2.1 Requirements

Make sure that all the listed modules are pre-installed, as specified in 1.1. If window is found to be too large or small, navigate to `const.py` and in the first line, change SCREENX's initialization value.

2.2 Launch Parameters

Initiate the game with:

```
python3 ./main.py
```

2.3 Navigation

- Your menu should look like this 1 upon launch.
- For a simple Angry Birds match, enter player names and start directly.
- For the LOTR theme and other settings (e.g., wind impact, recommended 3 or below), navigate to *Settings*.
- In LOTR theme, select the level via *Select LOTR Level* in the settings menu.
- After a game, choose to start a new game or return to the main menu.

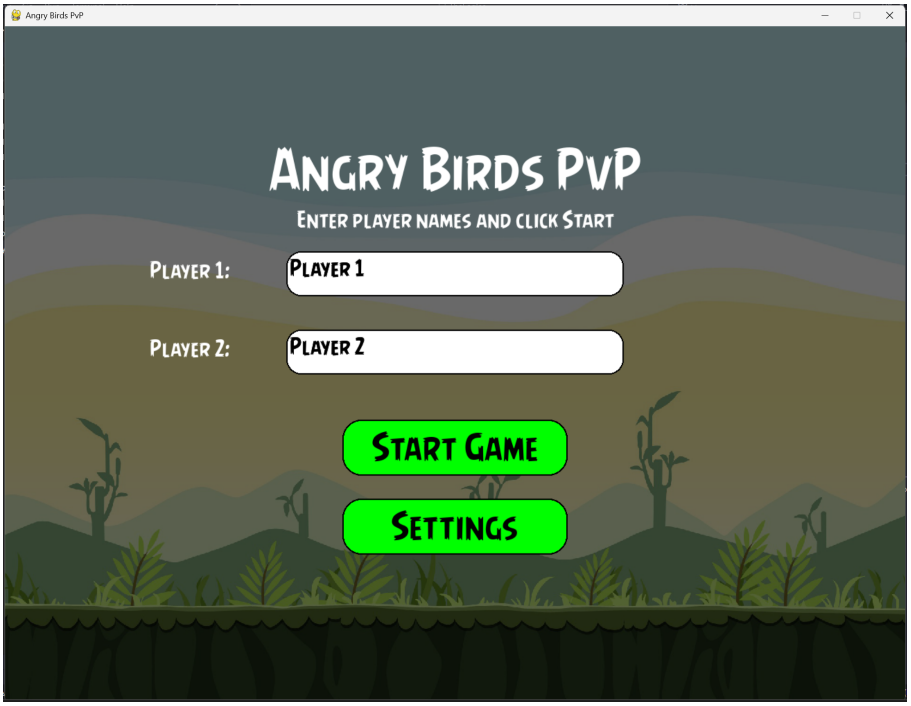


Figure 1: Main Menu Interface

2.4 Control Schema

The input system combines mouse and keyboard interactions. Below is a detailed breakdown:

Table 1: Game Controls

Action	**Input**
Select Projectile	Left-click on projectile in arsenal for catapult loading
Aim Projectile	Mouse drag-and-release; release near catapult base to keep on catapult
Activate Special Ability	Left-click mid-flight before collision (some projectiles bypass this)

3 Core Gameplay Mechanics (Basic Features)

3.1 Gameplay Interface

The interface displays the timer, score, and remaining turns at the top of the screen2. Catapults and available projectiles are shown in the foreground using a custom font [8].

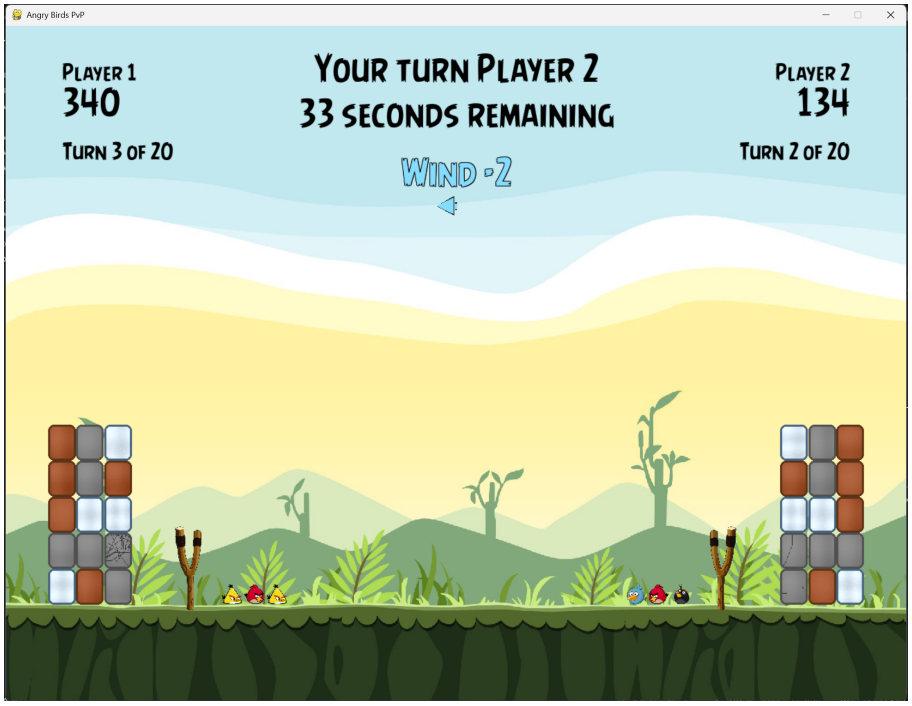


Figure 2: Standard Gameplay Window

3.2 Physics Simulation

The projectile motion system implements Newtonian mechanics with:

$$\begin{cases} v_x' = v_x + w(t) \cdot \Delta t \\ v_y' = v_y + g \cdot \Delta t \\ x' = x + v_x \cdot \Delta t \\ y' = y + v_y \cdot \Delta t \end{cases}$$

where $w(t)$ represents wind force, modeled as a time-varying function with random fluctuations to simulate real-world unpredictability. Collision response uses impulse-based resolution with damping factors of 0.8 and 0.6 in tangential and normal directions, chosen to balance realism and gameplay feel.

3.3 Projectile Generation

Initially, both players receive the same set of 3 unique projectiles. Subsequent projectiles are randomly generated and may differ between players.

3.4 Block Damage System

- Blocks feature health points(HP) based destruction with visual degradation.
- Damage follows specifications with randomized noise, based on block and projectile type,i.e. Chuck does more damage against wood, Bomb against stone and Blue against ice.
- Damage decreases with projectile bounces (except for *Legolas* and *Blues*).

- For *Chuck* and *Aragorn*, damage increases with activations.
- For *Bomb* and *Gimli*, damage decreases with explosion distance.

Note: While in LOTR theme, in case you don't know which block corresponds to which of the initial wood/ice/stone archetypes, you can always look at the outline of the block to know for sure. In that case, damage done is the same when a projectile that replaces Chuck/Blue/Bomb collides with a block that replaces Wood/Ice/Stone

3.5 Activation

For most projectiles, clicking mid-flight before collision triggers a unique effect, fulfilling the *Unique Projectile Effects* requirement:

Projectile	**Activation**
Red	None
Chuck	Increases speed and damage on activation
Blues	Splits into three instances
Bomb	Explodes, damaging nearby enemy blocks

Table 2: Bird Abilities

3.6 Win Condition

1. A player wins by destroying all opponent fortress blocks.
2. If turns expire with both fortresses standing, the higher score wins.
3. Further, if scores are tied, a tie is declared.

Note: The first-player advantage is acknowledged, akin to chess. To mitigate this, "Play Again" causes the players to switch sides.

4 Advanced Features

4.1 Lord of the Rings Theme Integration

The theme system dynamically swaps assets, physics parameters, and projectile effects.

4.1.1 Levels

Includes 3 unique battle locations with themed textures:

- *Helm's Deep*: Theoden vs. Saruman - Increased wind turbulence
- *Minas Tirith*: Gandalf vs. Witch King - Standard physics
- *Mordor*: Aragorn vs. Sauron - Enhanced gravity field

Figure 3: LOTR Themed Gameplay

4.1.2 Projectiles

Traditional Angry Birds are replaced by LOTR characters:

1. *Frodo*: Charming but harmless (replaces *Red*)
2. *Aragorn*: Mysterious Ranger (replaces *Chuck*)
3. *Legolas*: Dexterous Elven prince (replaces *Blues*)
4. *Gimli*: Natural Sprinter (replaces *Bomb*)

4.1.3 Projectile Effects

LOTR projectiles have unique effects:

Character	**Behavior and Abilities**
Frodo	None
Aragorn	Activates twice, can activate even after one collision; speeds up horizontally with increased damage
Legolas	Jumps in the air, backward if collisions have happened, backward if not. Supports activation after up to 2 bounces
Gimli	Explodes with greater damage than *Bomb*

Table 3: Character Behaviors and Abilities

4.2 Procedural Texture Generation

4.2.1 Texture Generation

Textures combine soft gradients and noise over base RGB colors for natural variations (e.g., stone, wood). Styles like lava apply glowing effects, output as Pygame surfaces with smooth transitions.

4.2.2 Crack Generation

Cracks are randomized branching lines, with complexity increasing quadratically by intensity. Rendered with anti-aliased lines for realism, cracks can be incrementally added to textures.

4.3 Additional Advanced Features

4.3.1 Score

Scores, visible below usernames increases when:

1. Damage is dealt to opponent blocks (By an amount equal to damage dealt)
2. For breaking opponent blocks (By a fixed amount)

4.3.2 Timer

A timer at the top of the screen turns red near expiration. Players must complete turns (selection, aiming, flight) before it runs out, or the turn is forfeited.

4.3.3 Wind

Wind, shown below the timer with text and a directional arrow, varies in color and strength. Players can wait for favorable conditions, but be mindful of the timer!

5 Ideas, Inspirations and References

1. Blocks are procedurally generated with unique textures, this idea was introduced by an AI and so is *some* of the code for the texture generation.
2. An audio system for Angry Birds [4] and LOTR [5] music, plus one-liners from Yarn [6], was initialized but not fully developed due to time constraints. The idea was given to me while rewatching the LOTR movie trilogy with a friend, and the website Yarn was suggested by a friend from Discord. Future plans include character-specific dialogue during activation.
3. Screen-shake effect, inspired by a Reddit post [3], was extended from LOTR to the base game.
4. The various projectiles and leaders/figureheads in the game are actually part of a cursor collection[7] which were of really high quality, and is what gave me the idea to do an LOTR theme in the first place (apart from my love of the franchise).

5. LOTR fonts Aniron [9] and Ringbearer [10] were sourced for thematic UI.
6. The Angry Birds images for the catapult[12] and the background[11] were taken from the internet.
7. The LOTR images for the backgrounds and the Catapult itself was AI-generated, though inspiration was taken from Tolkien's world.
8. Unimplemented: *Frodo* invisibility effect could temporarily hide the projectile and hit unexpected blocks, adding strategic depth.

6 Development Challenges

6.1 Modularization

As my first large-scale project, modularizing code was challenging due to inexperience. I addressed this by separating game logic into `game.py` and UI into `button.py`, improving maintainability.

6.2 Theme Switching

Early theme transitions caused inconsistent asset swaps. I resolved this by implementing some more functions in the `Game` class like the `reinitialize_ui_elements()` function.

6.3 Time Constraints

Deadlines limited full development of features like one-liners and Frodo's invisibility. I prioritized core mechanics over audio, learning to focus on a minimum viable product first in future projects.

6.4 Key Learnings

This project taught me advanced Pygame rendering techniques and physics simulations. I also gained insights into balancing feature scope with time management, crucial for larger developments.

7 Conclusion

This project, to me, showcases advanced game development through manually computed physics, dynamic themes, and procedural content. The architecture supports future expansions with additional modes and themes, optimized via efficient collision detection and image rendering.

8 LOTR Dictionary

For those unfamiliar with LOTR, below are characters used as projectiles or leaders:

(a) Frodo

(b) Aragorn

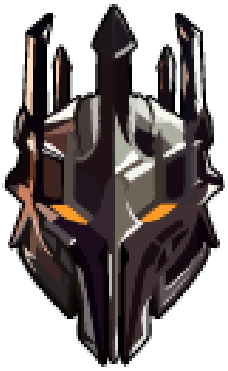
(c) Legolas

(d) Gimli

(e) Gandalf

(f) Theoden

(g) Saruman



(h) Sauron

(i) The Witch King of Angmar

Figure 4: LOTR Characters in the Game

References

[1] Pygame CE Documentation. https://www.pygame.org/docs/ (Core library for game rendering and input)

[2] NumPy Reference Guide. https://numpy.org/doc/stable/ (Physics simulation calculations)

[3] u/jcsirron in https://www.reddit.com/r/pygame/comments/q5g6x6/how_do_you_make_a_screen_shake_in_pygame/ (Inspiration for screen-shake effect)

[4] Angry Birds BGM: https://youtu.be/uQeychfPYVA?si=OqMc8rbjcY5_hJoe (Background music for classic theme)

[5] Lord of the Rings BGM: https://youtu.be/FrWuCPgsp_c?si=GPzuva99Gkt5iamp (Background music for LOTR theme)

[6] Yarn: https://getyarn.io/ (Source for potential character one-liners)

[7] LOTR Themed cursor collection: https://sweezy-cursors.com/collection/lotr/

[8] Angry Birds Font: https://www.dafont.com/angrybirds.font (UI font for classic theme)

[9] Aniron Font: https://www.fontspace.com/aniron-font-f2247 (Thematic font for LOTR UI)

[10] Ringbearer Font: https://www.fontspace.com/ringbearer-font-f2246 (Thematic font for LOTR UI)

[11] Catapult: https://stickpng.herokuapp.com/img/games/angry-birds/angry-birds-catapult

[12] Angry Birds background: https://angrybirds.fandom.com/wiki/Angry_Birds_Wiki?file=Angry_birds_background_by_gsgill37-d3kogmx.jpg