Turn-Based PvP Fortress Game in Pygame

salluri vivek
Roll Number: 24B0912

April 29, 2025

Contents

1 Introduction

1.1 1.1 Overview of the Project

This project is a two-player, turn-based fortress destruction game inspired by Angry Birds, implemented in Python using Pygame. Players take turns launching projectiles to destroy each other's fortresses, with different block and bird types affecting gameplay strategy.

2 Modules Used

2.1 2.1 Standard Modules

- `pygame`: For graphics, input handling, and game loop.
- `random`: For randomizing block types and bird choices.
- `sys`: For system exit.

2.2 2.2 Custom Modules

- `config.py`: Stores global constants and file paths.
- `ui.py`: Utility for rendering text.
- `assets_loader.py`: Functions for loading and resizing images.
- `fortress.py`: Fortress generation, rendering, and collision.
- `helpers.py`: UI screens, player name entry, and game helpers.
- `mechanics.py`: Handles physics for dragging and projectile flight.
- `main.py`: Entry point and main game loop.

3 Directory Structure

3.1 3.1 Folder Layout

```
angrybirds_project/

 main.py
 config.py
 ui.py
 assets_loader.py
 fortress.py
 helpers.py
 mechanics.py
```

```
assets/
   backgrounds/
      main_menu.png
      player_interface.png
      gameplay.png
   catapults/
      left_catapult.png
      right_catapult.png
   projectiles/
      red.png
      chuck.png
      blues.png
      bomb.png
   blocks/
       wood.png
       stone.png
       ice.png
```

3.2 3.2 Asset Organization

Assets are categorized into folders for backgrounds, catapults, projectiles, and blocks to simplify management and modifications.

4 Running Instructions

4.1 4.1 Software Installation

Install Python 3 and ensure the Pygame library is also installed in your Python environment.

4.2 4.2 File Setup

Organize all files and folders as specified in the directory structure above. The ‘assets‘ folder must be in the same directory as the Python files.

4.3 4.3 Game Execution

Navigate to the project folder and execute the main game file to start the game. This will take you to the main menu.

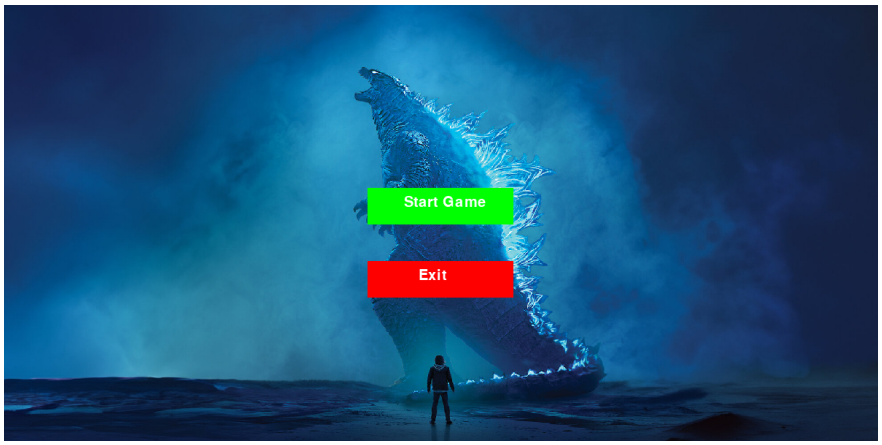
4.4 4.4 Controls

Use the mouse to drag and release projectiles and click on menu options. The keyboard is used to input player names and confirm selections with the Enter key.

5 Basic Features Implemented

5.1 5.1 Main Menu

Start and Exit buttons with dynamic background visuals.



5.2 5.2 Player Name Entry

Sequential input for both players with custom backgrounds.

5.3 5.3 Gameplay Mechanics

- Randomized fortresses with blocks (wood, stone, ice) represented by monsters.
- Drag-and-release physics for projectile launches.
- Health bars for blocks.
- Monster/projectile selection and switching.

5.4 5.4 Win Screen

Displays winner and allows exiting the game.

6 Advanced Features and Customizations

- Dynamic Fortress Generation: Randomized 3x4 layouts enhance replayability. - Scalable Visual Assets: All blocks, projectiles, backgrounds, and catapults use scalable images for a polished user experience. - Health Visualization: Shrinking health bars provide instant feedback; blocks vanish at zero health. - Modular Design: Logical module organization ensures clarity and scalability.

7 Project Journey and Key Learnings

- File Organization and Asset Loading: Centralized configurations addressed file path and loading issues. - User Interface Development: Built intuitive UI systems with clear visual feedback for user actions. - Turn Management and Collision Logic: Enhanced gameplay reliability through robust data structures. - Debugging Strategies: Overcame common issues like index errors via conditional checks.

8 References

8.1 8.1 Documentation and Online Guides

- Pygame documentation: https://www.pygame.org/docs/
- Reddit: https://www.reddit.com/r/pygame/comments/279oiw/so_how_do_you_organize_your_pygame_game/
- Python Forum: https://python-forum.io/thread-6431.html
- Real Python: https://realpython.com/videos/structuring-your-game/
- StackOverflow: https://stackoverflow.com/questions/71173349/my-file-structure-for-a-p