Angry Birds Using Pygame-CE
Development Report

Kashyap Khandelwal
CS-108 Project

Spring 2024–25

Abstract

This report aims to provide a comprehensive overview of the game's architecture, detailing the organization of its various modules and the core functionalities implemented within them

Contents

1 Introduction to the Game: "FeatherFall: Siege of Avaria!"

Developed using Pygame-CE, this game is inspired by the popular mobile game, Angry Birds!
"FeatherFall: Siege of Avaria!" engages two players in a strategic battle. The core gameplay revolves around launching a variety of birds, each with unique characteristics, to dismantle the opposing player's fortresses.

The game features two distinct modes:

- **Quick Game:** This mode offers a fast-paced and direct confrontation. The primary objective is to be the first player to accumulate 200 points by successfully hitting the opponent's structures and the roaming aliens.
- **Basic Game:** This mode offers power-ups which layers can strategically utilize to gain an advantage. The ultimate goal in this mode is the complete demolition of the opponent's structural defense.

2 Modules

The external modules used are:

- `pygame-ce` - The frequently updated pygame community edition version of pygame, which is a set of Python modules designed for writing video games.
- `random` - A module to implement pseudo-random number generators for various distributions.
- `math` - A module providing access to mathematical functions defined by the C standard.

3 Directory Structure

The project directory is as follows:

```
FeatherFall/
├── main.py
├── game.py
├── game_modes.py
├── tools.py
├── classes.py
├── assets.py
└── media/
    └── ...(bird images, block images, background images, power-up icons, font files.)
```

- `main.py`: This module serves as the central entry point for the game. It handles the initialization of the Pygame environment, manages the overall game flow, and orchestrates the transitions between different game states, such as the start screen, the main menu, and the selection of game modes. It acts as the conductor of the entire game experience.
- `game_screens.py`: This module contains functions responsible for displaying static or semi-static screens within the game. This includes the initial start screen that welcomes players, the main menu offering game mode choices and options, informational screens that might provide instructions or context for different modes, and the game over screen that presents the results of a match.

- `game_modes.py`: This crucial module houses the specific implementation and logic for each of the distinct game modes offered. It contains the code that governs the gameplay within both the "quick" and "basic" modes, including managing player turns, handling game-specific rules, and determining win conditions for each mode.
- `tools.py`: This module provides a collection of utility functions that are utilized across various parts of the game. These functions include procedures for the procedural generation of the players' defensive structures, algorithms for detecting collisions between game objects (birds, blocks, aliens), functions for drawing game elements on the screen, and mechanisms for managing the queues of available birds for each player.
- `classes.py`: This module defines the blueprints for the fundamental game objects that populate the game world. It contains the class definitions for entities such as the `Bird` (with its properties and behaviors), the `Block` (representing structural components with health), the `Alien` (as interactive targets), and the `Powerup` (providing temporary gameplay enhancements in "basic" mode).
- `assets.py`: This module is responsible for loading and managing all the external resources that the game relies upon. This includes loading image files for various game elements (birds, blocks, backgrounds, icons), loading font files for text rendering.
- `media`: This subdirectory serves as a repository for all multimedia assets used by the game.

4 Running Instructions

To execute "FeatherFall: Siege of Avaria!" on your system, ensure that you have the necessary prerequisites installed and follow these steps:

1. **Install Python**: The game is developed using Python. Ensure that you have a compatible version of Python installed on your operating system. You can download the latest version from the official Python website (https://www.python.org/downloads/).
2. **Install Pygame**: The game relies on the Pygame library for handling graphics, input, and other game-related functionalities. Open your terminal or command prompt and install Pygame using the Python package installer (pip):

   ```
   pip install pygame
   ```

3. **Navigate to the Game Directory**: Using your terminal or command prompt, navigate to the main directory where the game's Python files (including `main.py`) are located.
4. **Run the Game**: Once you are in the correct directory, execute the game by running the `main.py` script using the Python interpreter:

   ```
   python main.py
   ```

   This command will initiate the game, and the "FeatherFall: Siege of Avaria!" window should appear on your screen, allowing you to start playing.

5 Various Implementations in the code

The codebase for "FeatherFall: Siege of Avaria!" incorporates several key implementations to deliver its gameplay experience:

- **Game Mode Management**: The game features two distinct game modes, "quick" and "basic," each with its own set of rules, objectives, and gameplay dynamics. The `game_modes.py` module orchestrates the logic for each of these modes, providing different challenges and replayability.
- **Physics-Based Launching and Trajectory**: The core mechanic of launching birds involves simulating a basic physics trajectory. When a player drags and releases a bird, the game calculates an initial velocity based on the drag distance and direction. The `Bird` class in `classes.py` has a `simulate()` method that predicts the bird's path, taking into account gravity. The `tools.py` module then visualizes this trajectory to aid player aiming.
- **Collision Detection and Handling**: Accurate collision detection is crucial for determining interactions between the launched birds, the opponent's block structures, and the alien enemies. The `tools.py` module contains functions like `check_block_collisions()` and `check_alien_collisions()` that use Pygame's rectangle collision detection to identify these interactions and trigger appropriate responses, such as damage calculation and score updates.
- **Destructible Environments**: The block structures that players build and attempt to destroy are not static. The `Block` class in `classes.py` manages the health of each block. Upon impact from a bird, the `take_damage()` method reduces the block's health, and the block's visual representation might change based on its damage state, as managed in `assets.py` and drawn by functions in `tools.py`.
- **Power-up Implementation (Basic Mode)**: The "basic" game mode introduces strategic power-ups. The `Powerup` class in `classes.py` defines these enhancements, such as the ability to see the full trajectory of the launched bird or to inflict double damage on impact. The `game_modes.py` module handles the activation and application of these power-ups based on player interaction.
- **Turn-Based Gameplay**: The game follows a turn-based structure, where players take turns launching their birds. The `game_modes.py` module manages the switching of turns between the two players after a bird has completed its trajectory or come to rest.
- **Score Tracking and Game Over Conditions**: The game maintains separate scores for each player, updated based on successful hits on structures and aliens. The `game_modes.py` module monitors these scores and checks for game over conditions, such as reaching a target score in "quick" mode or the complete destruction of a player's structure in "basic" mode, triggering the game over screen managed by `game_screens.py`.

6 Bibliography

References

[1] Pygame Official Documentation https://www.pygame.org/docs/.

[2] Offiicial Pygame CE Documentation https://pyga.me/docs/

[3] Space/Science Fiction Fonts https://www.1001fonts.com/space+science-fiction-fonts.html?page=3

[4] Freepik (A source for some of the game's graphical assets)

[5] Youtube Tutorials https://www.youtube.com/watch?v=i6xMBig-pP4&list=PLzMcBGfZo4-lp3jAExUCewBfMx3UZFkh5&index=2