Angrybirds(pygame-ce)

CS-104 Project - Spring 2024-25
Vidith Hundekar

Outline

This report is about the game "Angry Birds" a 2Player game. It covers the game's instructions, implementation, game aspects, gameplay description, ideas, references and the challenges encountered during development.

Contents

1 Introduction to my Game

This is a 2Player game, each player tries to destroy opponent's fortress with their angry birds, first to do that with better accuracy WINS.

2 Modules

The external modules used are:

- `pygame-ce` - Version of pygame, Python module designed for writing code for games.
- `Random` - A module that generates random number for various distributions.
- `Time` - A module that provides various time-related functions.
- `Math` - A module that provides many mathematical functions

3 Repo tree

The project directory is as follows:

```
.
        Modules
             MainMenu.py
             PlayGame.py
```

```
                Preferences.py
                Scores.py
          media
                fonts
                images
                sounds
                videos
          data
                path.txt
                LeastTimes.txt
          game.py
          settings.py
```

- `main.py` - The main game loop.
- `game.py` - Contains all the global variables and modules necessary for the game to function smoothly.
- `Modules` - The programs that manage various game parts.
- `media` - Contains all the images, sounds, and fonts used in the game.
- `data` - Contains the path of the maze and the High ScoreCard (Least Time Taken).

4 Instructions to run

4.1 Game Navigation

Note: To ensure easy navigation between various screens, a back button is introduced which smoothly takes you to the previous screen.

4.1.1 Intro Screen

The game starts with an Intro Screen:

4.1.2 Main Menu

After loading, you will be greeted with a Main Menu, from which you can choose to:

- Play
- See the Fastest Solves in each Level
- Customize the Game: Mute or Unmute
- Quit

You can select any of these by pressing on these buttons:

Figure 1: Intro Screen

4.1.3 Game Level Selection

We have three levels of mazes from which you can choose:

4.2 Gameplay description

The game starts, waiting for you to navigate using the arrow keys or [W A S D]. The goal is to reach the door in the opposite corner. The *Score* is measured in terms of the time taken to go to the opposite end: the Lower, the Better!

Various themes can be changed using the *Change Theme* button. The music can be turned off by pressing the Music button.

Some examples of the game screens:

4.2.1 Game Over

On reaching the opposite end, the game ends, and the time taken is displayed:

4.2.2 Fastest Solves

On clicking the Fastest Solves button on the Main Menu, you will see the Least Time taken to solve the various levels of the maze. An example screen:

4.2.3 Preferences

This window enables you to mute the music part of the game. You can do this by clicking on the red music button. If you want the music back on, click the button again.

Figure 2: Main Menu

4.2.4 Quit

On clicking this button, the game ends and the program terminates.

5 Various Implementations in the Code

For the maze generation, I used the *Recursive Backtracking* algorithm. This algorithm is a randomized version of the depth-first search algorithm. The algorithm starts at a random cell, chooses a random neighboring cell that has not been visited, creates a path between the two cells, and moves to the next cell. The algorithm continues until it has visited every cell in the grid. I have modified this algorithm slightly to make the wall size and the path size the same, which makes the maze look more appealing.

For the pathfinding, I used the A* algorithm. The A* algorithm is a pathfinding algorithm that uses a heuristic to determine the next node to visit in a graph. The algorithm uses a priority queue to determine the next node to visit based on the cost of the path to that node and the heuristic value of the node. The algorithm continues until it reaches the goal node or there are no more nodes to visit.

I have used the `heapq` module to implement the priority queue for the A* algorithm.

For the pygame functions, I referred to the official documentation of pygame and pygame-ce, mostly the latter for the updated functions and methods.

5.1 Customizations in the Game

A list of all the special customizations implemented in the game:

Figure 3: Game Level Selection

- Animation is when the player moves.
- Dynamic Background of the Main Menu.
- Music and Sound.
- Themes for the Game.
- High Scores.
- Preferences.
- Back Button for easy navigation.
- Customized Fonts.
- Responsive Buttons.

6 References

1. Pygame Official Documentation
2. Pygame CE Official Documentation
3. Maze Generation Algorithms by `professor-l`
4. A* Algorithm

report/MazeGame.png

Figure 4: Game Starts!

report/MazeGameThemes.png

Figure 5: Game Play

report/GameOver.png

Figure 6: Game Over

report/HighScores.png

Figure 7: Fastest Solves

`report/PreferencesSoundOn.png`

Figure 8: Music On

`report/PreferencesSoundOff.png`

Figure 9: Music Off