Michael Ward

A01755332

CS 3430 Final Project

28 April 2015

Final Project: Pac-Man

Abstract

The classic game of Pac-Man was implemented in Python by Michael Ward for his final project in CS 3430.

Introduction and motivation

For my final project, I made Pac-Man in Python. This is a fully functioning version of the classic two-dimensional game, with all four monsters: blinky, pinky, inky, and clyde. There is the option for one or two players to play the game, playing as Pac-Man and Miss. Pac-Man in two player mode.

My motivation for this project was an interest in Pac-Man. I had started creating a game of Pac-Man in Visual Basic while in high school but never had the chance to finish. I wanted to rewrite the classic game in Python for my final project in this class.

Detailed Problem Description

I had several problems to face while working on this project. The first of them was setting up a maze with collision detection to prevent the Pac-Man from moving through barriers. The next problem was making the Pac-Man move with the movement pattern used in the original game. When a command is typed, the Pac-Man will change direction, unless the desired direction is blocked by a wall. If this is the case, that direction will be remembered and he will turn the desired direction as soon as possible. This is very useful in making sure the Pac-Man does not waste any time on turns but can be rather difficult to implement. The next challenge facing me was creating the four monsters and the logic behind them. Whenever a monster hits Pac-Man or Miss Pac-Man, the monsters and players will reset

and one life will be lost. The final step is to fill the maze with food, and provide a means to level up once all the food has been eaten by the player or players.

Problem Solution & Implementation

My first problem was creating the walls and collision detection between the player and the walls. At first I planned on using pygame's colliderect method to test for a collision between the player and the wall. However, this proved to be useless as the player always has at least one side in collision with a wall, but the only side that matters is the side in the direction that the Pac-Man is moving in. I therefore used the rect.top, rect.left, rect.right, and rect.bottom variables of the Pac-Man and compared them to the equivalent values of the barriers, looping through a list that contained all the barriers each turn to see if the pac-man had run into any of them. Implementation of collision detection for the food and monsters was much easier, as the side that collides does not matter, and therefore colliderect could be used.

The monster logic was fairly difficult. I had designed my own maze in order to make my game slightly different from the classic pac-man. However, this meant I had to come up with my own routes for the monsters. I gave each monster a specific route to loop through. They each start at the center of the board, wait a pre-determined amount of time and then move from the center of the board to their route and proceed to loop through it. I finally added food to the maze using two for-loops to distribute it evenly throughout the board. I randomly set one in fifty food pieces to be larger than the others, and this is the food that makes the ghosts edible. Whenever pac man or miss pac man eats one of the larger pieces of food, all the ghosts will freeze in place and start flashing white and blue. While the ghosts are in this state, a collision between the ghost and player will result in a death of the ghost, instead of the other way round. When a ghost dies, it moves back to the center and proceeds back to its loop after a short wait. All the monsters who have not been killed by the player will remain in this mode for a

random amount of time before returning back to their normal state and proceeding throughout their loop. I also implemented a level system, where the player can level up once all the pieces of food throughout the board have been eaten. With each level the player gains, the monsters become slightly faster, making the game more challenging. The players have a total of three lives, and one is lost every time pac man or miss pac man collides with a monster that is not in freeze mode.

Conclusion

Overall, I have greatly enjoyed working on this project. The project has been very challenging, but also very rewarding, and has been a great experience putting my new Python skills to use. There is not much need in explaining the game, since it functions exactly as the classic Pac-Man game does. However, working on this project has exposed me to many steps of real-life software development and helped me improve my programming skills. I hope you will enjoy playing the game as much as I enjoyed writing it!

Note: The game itself is in the game.py file within my submission. The code includes multiple classes but I put it all in one .py file for simplicity's sake. I hope this is okay.