CMU 15-112: Fundamentals of Programming and Computer Science

Anushka Saxena

**Description**

Welcome to Pac-Man, Version 1.12! In the classic arcade game, there is a yellow creature called Pac-Man that the user controls, and the user is trying to collect points around the board and not run into walls. However, Pac-Man is also being chased by four ghosts, each with their own method of following Pac-Man. This take on the game features more interactive capabilities, such as smarter ghosts using different algorithms in the original mode, larger mazes in the side scroll mode, and the ability to map your own walls in creative mode. If time allows, there will be a multiplayer feature that will allow the user to play as a ghost and a physical, arcade version of the original mode. For prototyping, I will be using ovals to represent Pac-Man and the ghosts and rectangles as the walls. Later, I will implement better graphics using sprite sheets and images in the actual game.

**Competitive Analysis**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Maze** | **Ghost** | **Pac-Man** | **Other Game Features** | **Extras** |
| **UC Berkeley** | Normal maze | Normal ghost | User controlled Pac-Man | None | Pygame |
| **Stanford** | Normal maze | Four different algorithms for the ghosts | User controlled Pac-Man | None | Eval function |
| **Andy Sommerville** | 12 different levels | Path-finding | Joystick controlled Pac-Man | Jumping fruit | Cross platform functionality, Pygame |

From personal development to university assignments, many different versions of Pac-Man have been developed from programmers across a wide variety of backgrounds. At the University of Berkeley, there was a class assignment to create Pac-Man using PyGame (<https://inst.eecs.berkeley.edu/~cs188/fa08/projects/contest/Documentation/pacman.html>). This documentation shows that the user created four classes: one class dedicated to the rules of the game, one to the game state of all the objects and players of the game, one to dictate ghost, and one to dictate Pac-Man. The game is command based, so the user will be interacting with the command line. I will not be using PyGame to create my game, as I want to program all my features from scratch, which will ultimately give me more freedom to change the features as I please. However, I will be using multiple classes, and I know I will also be having two separate classes that will dictate Pac-Man and the ghosts behavior. My ghost class will have four subclasses for each of the ghosts, thus involving more specialities for the ghosts.

At Stanford, there was a similar class assignment to create Pac-Man (<https://stanford.edu/~cpiech/cs221/homework/prog/pacman/pacman.html>). However, this assignment mainly focuses on exploring different algorithms to control the ghost based on the state of Pac-Man. This method allows the programmer to explore different path algorithms, and I will most likely be doing something similar with my ghosts. However, out of the four algorithms used in the Stanford assignment, I had only planned on exploring minimax as a possibility to move the ghosts. I am still looking into other algorithms such as Dijkstra's to control the ghosts. This version of Pac-Man also involves an evaluation function for Pac-Man, but in my version, the Pac-Man will be user-controlled.

The last version of Pac-Man that I found online was the most similar to what I want to achieve in my term project, but there are still some differences. Andy Sommerville coded Pac-Man using PyGame (<https://github.com/greyblue9/pacman-python>). He coded 12 different mazes using a maze generator; instead of having 12 static fields, I want to create a similar maze generator that will allow the user to drag and drop walls on the map and then play. His ghosts also use path-finding algorithms, as will mine. However, mine will most likely involve algorithms similar to the one in the Stanford assignment. His game also involves features such as jumping fruits and cross functionality across different platforms. Features like these are important, but for the purposes of this project, I will not be exploring these in the beginning. Finally, he utilized a joystick to control Pac-Man; I will do something similar, but instead of having a joystick control the Pac-Man on screen, I want to create a physical version of the entire board and play Pac-man using a joystick.

**Structural Plan**

There will be seven main classes: SplashScreenMode, PacMan, Ghost, Wall, Points, OriginalGameMode, and MyModalApp. SplashScreenMode is the homepage for the game and will have two different options: single player and multiplayer. For single player mode, there will be three options: original, side scroll, and creative. For multiplayer mode, there will be only one option, original. For now, the OriginalGameMode is the only one implemented; it dictates how the game is played and how the players and objects interact. OriginalGameMode will have several basic functions; for example, the function movePacMan only allows Pac-Man to move in one direction at a time depending on which arrow the user pressed; illegalPlaces defines the areas the Pac-Man nor the ghosts can travel, which are the walls; and drawCoins, which places coins around the board wherever there are no walls. More functions that involve the ghosts interaction with Pac-Man will be implemented over time.

The players are Pac-Man and the four ghosts, which are defined by their respective classes. These classes define the characteristics and how the players are drawn. There are four ghosts, which are subclasses of the Ghost class. Each of the algorithms will be defined within each subclass for the ghost. The objects are walls and points. The Wall class is called by the board subclass to draw the map on the screen; the classes draw the walls themselves. However, the Points class only defines what the points are; each point is drawn within OriginalGameMode, since the points on the screen might change if Pac-Man interacts with those points.

MyModalApp calls the SplashScreenMode and OriginalGameMode and determines which mode the game is in.

All the player and object classes and their respective subclasses will be placed in separate files on a directory. However, all the game modes will be on one file.

**Algorithmic Plan**

My program involves two algorithmically challenging aspects: the ghosts algorithm and the side scroll mode. For the ghosts algorithm, I was originally considering using minimax to control the red ghost; however, looking into it, minimax is more effective in two player games that don’t have to make instantaneous solution. Another algorithm I am considering is Dikstra’ algorithm. The ghosts will begin in the middle of the board, and over time, each ghost is released from the pen. Dijkstra’s algorithm will use the ghost’s current position, weigh each path, and select path with lowest weight based on distance. The algorithm is based on two classes: graph and vertices. If I am using Dijkstra’s, there is a possibility that I will change how everything is drawn and make it table based instead of pixel based. For side scroll, want the program to generate a random board everytime the player chooses side scroll mode. The board will extend to a certain point off the screen, but it will not go on forever. The algorithm would involve randomly adding walls to a list of walls that are drawn to create the board. The points are distributed accordingly without interfering with the walls.

**Timeline Plan**

I am planning on finishing the Original Game Mode by TP1 (Wednesday, November 20) and Side Scroll Mode by TP2 (Wednesday, November 27) . However, the ghosts’ algorithms will be implemented by TP2, as I have to do more research in order to understand which algorithms will work best for them. After MVP, I plan on implementing Creative and Multiplayer mode, as well as creating a physical hardware version of Pac-Man over Thanksgiving Break (Wednesday, November 27 to Sunday, December 1). By TP3, I hope to add features such as background music and highest score, but only if time allows.

**Version Control Plan**

I am using GitHub to backup my code. I will be using a private repository.

**Module List**

I am not using any additional modules besides the ones provided in the course.