***Team Cilantro***

***EC327 Project Documentation***

***PacMan 2.0 - The multiplayer classic everyone’s been waiting for…***

**Contents:**

* Overall Summary of the Program
* Inputs
* Outputs
* Relationship between the files
* Relationship between the classes, data structures, etc.
* Interesting core algorithms
* Functions of interest
* Challenges
* Marketability
* Why this was a good project and deserves 15% of your grade

**Overall summary of your program**

This program creates a multiplayer PacMan game where both pacman and the ghosts are controlled by users. As in traditional pacman, if PacMan eats all the food on the board, PacMan wins, while if one of the ghosts catches PacMan by touching him, the ghosts win. PacMan has powerups spread across the map that allows pacman to eat the ghosts for a short amount of time. Our program utilizes the OpenGL Utility Toolkit, or GLUT for the graphics interface of the game. Original code was adapted from Patricia Terol, GitHub username [patriciateroltolsa](https://github.com/patriciateroltolsa).

The user controls were added in a similar way to how the PacMan is controlled. In order to do this, we first had to comment out the code for the random ghost movement and then copy and paste the code structure of PacMan with ‘wasd’ movement. This was then mapped to keys ‘tfgh,’ ‘ijkl,’ and ‘1234’ for 3 other people to fit on a keyboard. Due to space constraints, we chose to leave the 4th ghost out because fitting 4 people on a keyboard (PacMan and 3 ghosts) seemed like the maximum we could add.

The powerups are placed at the corners of the screen as those are harder areas to reach, which allows PacMan to reach harder areas with a reward of getting a powerup and being able to eat the ghosts. The powerups are coded into the program as big red dots. These red dots are drawn using the same algorithm as with the food except with different values to make it larger and a different color. When a powerup is eaten, the power mode variable (our power-up variable) is set to a positive value and serves as a flag to tell the program when to make the ghosts vulnerable and for how long the powerup mode lasts. The variable stores an integer value because it serves as both a timer and a flag (boolean) since it evaluates as “true” when it is positive and “false” when it is negative. This is essential to the code since GLUT does not function with time so this introduces a pseudo-time factor.

**Inputs**

One of the biggest challenges we tackled was making the single-player game to a multiplayer game. PacMan 2.0 can be played between four players. The following are the controls user inputs to play the game:

*To control Pacman use A (left), D (right), W (up), and S (down).*

*To control Ghost 1 use F (left), H (right), T (up), and G (down).*

*To control Ghost 2 use J (left), L (right), I (up), and K (down).*

*To control Ghost 3 use 1 (left), 2 (right), 3 (up), and 4 (down).*

*To start or restart the game, press the space key.*

**Outputs**

The output of PacMan 2.0 runs in a separate window and has 4 states: welcome screen, gameplay screen, winning screen, and losing screen. The welcome screen is first triggered when the .exe file is run. To access the gameplay screen, the ‘space’ bar is pressed. If PacMan wins (eats all the food), the winning screen displays and the user can press the spacebar to replay the game. Otherwise, if PacMan loses (touches a ghost in non-power up mode), the losing screen displays and the user can press the spacebar to replay the game.

The specifics of the output include colors and display functions. PacMan is a circle with a mouth that can rotate based on direction. The ghosts display as their usual colors except when PacMan eats a red powerup food, then the ghosts all turn white for a short period of time. During this short period of time is when PacMan can eat the ghosts.

**Relationship between the files**

We have one .cpp file which incorporates all our functions, game commands and graphics for the game. It made the troubleshooting process much easier and also seemed convenient since several people were working on the same code. Moreover, the source we used for making the default game was all in a single file too.

**Relationship between the classes, data structures, etc.**

The PacMan code did not utilize classes, however, it did use data structures to organize the coordinates of the game board and graphics displayed on it. The coordinates were organized using some type of data structure, whether it be a heap float array, a vector, or a deque.

The coordinates of the border walls, and all the obstacles were done through the use of vectors. Subsequently, the food and power up coordinates were set up using a deque. These coordinates were then passed to multiple display functions that are used for loops and the OpenGL libraries to print each static item. Examples of these display functions are: drawFood and drawLaberynth.

The moving components of the game that included data structures were the ghosts (called monsters in the code). These are float pointers to a float array in the heap. The monsters needed to be pointer arrays because they are all able to take user input and refresh their location using the same updateMonster function.

Since there is only one PacMan, he could be defined using regular float variables to which a separate function that controls the rotation of the graphic was used.

**Interesting core algorithms**

An interesting algorithm that we edited into the program was how the red food dots are displayed on the screen. First the point size has to be set using a glPointSize function. The points are drawn using the GL\_POINTS function in the OpenGL library. The last specification has another function glColor3f() to specify the color of the point in (red, green, blue) parameters.

These red dots are then printed at each respective corner using a for loop and coordinates set globally in the beginning of the program via a static deque<float> called powerup.

**Functions of interest**

Multiplay Functionality: Unlike traditional Pacman, the game features three controllable ghosts in addition to PacMan.

Powerups: While large dots that allow PacMan to turn the tables on the ghosts are in the original game, implementation of this feature in our multiplayer version presented a new challenge.

**Challenges**

Before beginning the project and after a session of brainstorming the team came up with a list of specifications we wanted to include in the game. They are as follows:

1. Make the game multiplayer.
2. Add Power Ups for PacMan to eat the ghost and win (the original code we acquired was quite basic with no powerups at all, the powerups currently in the game are created by us)
3. If PacMan eats the ghost, they are not generated. We just delete the ghost objects because we want PacMan to have some sort of advantage over real-time user ghosts.
4. Speed Controls (PacMan moves faster than the ghosts, with the powerup)
5. Make changes to the Visuals of the game
6. Add Secret channels to the game, exclusive for PacMan.
7. Give users the ability to decide the number of players ranging from 1 to 5.
8. Stretch goal- make it an Android App.

We achieved most of our laid out goals except giving users the ability to decide the number of players and adding secret channels. The team definitely does understand how to achieve the last two goals and had an algorithm aid out for the same. However, we did not have enough time to accomplish these since we also spent the first week working on a different project topic (Facial Recognition) which we decided to switch from. In the end, we wanted to ensure we were able to present and deliver the best quality of whatever we had done and decided to focus on presentation and marketing. Thus, I’d say the main challenges we faced was the limited time availability and making the code both Mac and Windows compatible. More details on this are included in the README file.

**Marketability**

This game is for users of all ages, especially those who enjoy arcade games and are fans of the classic PacMan game. The graphics of the game are colorful to appeal to the young audience but still subtle. Moreover, it is designed with extremely convenient and self-explanatory controls where one does not need prior knowledge of video games or even the computer.

**Why this was a good project and deserves 15% of your grade**

Word on the street is that Professor Densmore loves arcade games. Team Cilantro thought what better than making the 2.0 version of the greatest classic arcade game!

*Pac-Man* was a widespread critical and commercial success, leading to several sequels, merchandise, and two television series.

Working on this project, all the team members imbibed great skills that made us all better coders. First of all, we learned the use of different libraries that made the code work both on Mac and Windows. We ran through the original PacMan code we acquired from Patricia Terol’s GitHub step by step to understand the purpose of each line in the code and understand how and where we could make our modifications. This not only introduced us to different behaviors and attributes of C++ but also helped us put in use all the concepts we’ve learned in class this semester to use.