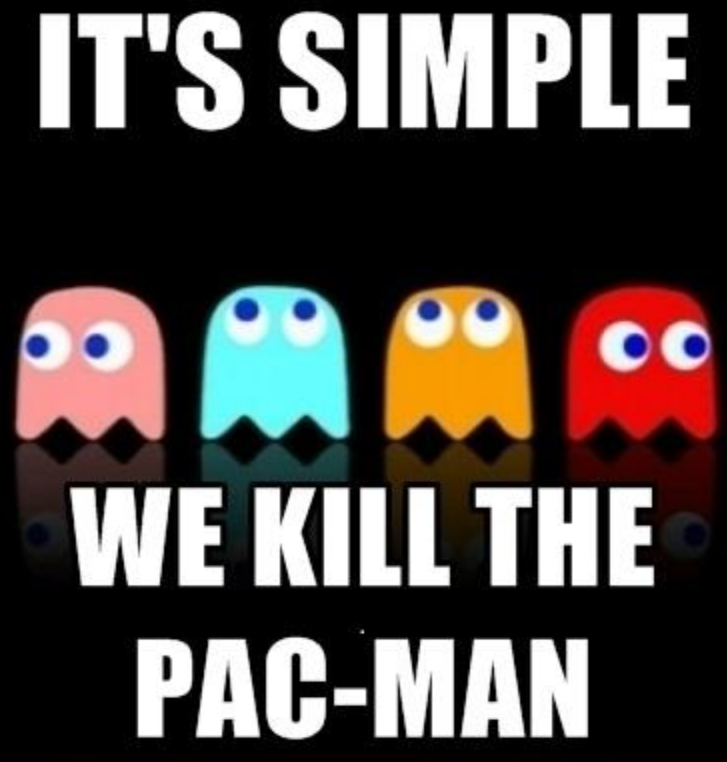
IT'S SIMPLE
WE KILL THE
PAC-MAN

Ryan Tick and Ryan Moran — CSE 30332 – Programming Paradigms

Final Project README — Due: 05/07/2015

Overview: For our game, we thought that it would be cool to do a multiplayer version of PAC-MAN. If only one person is connected, than that player would take control of PAC-MAN. All of the ghosts at this point are simply AI controlled moving randomly. If more people join, however, more ghosts become human controlled. Four additional players can join our game. That way, each additional player would control a ghost. At max networking capability, one player is controlling PAC-MAN and four other players are controlling different ghosts.

How to Run the Game: In order to play our game, each player has to have the code on their computer. First, the PAC-MAN player has to be logged into student02.cse.nd.edu. We recommend you SSH into student02.cse.nd.edu with the -YC flag. The command would be as follows: *ssh netID@student02.cse.nd.edu -YC*. If more players want to play, then each respective player should (for consistency), SSH into one of the student machines. It is alright if all of the ghost are on the same student machine as each other, or as PAC-MAN. For testing purposes, we would also SSH into student01 as the ghosts, again streaming the -YC flag. The command would be: *ssh netID@student01.cse.nd.edu -YC.*

The way that we set up our Twisted networking, PAC-MAN needs to be running first before the ghosts can join. Therefore, run PAC-MAN.py and then wait a little bit before running redGhost.py, blueGhost.py, orangeGhost.py, and pinkGhost.py. To be safe, wait until the ghosts have exited their center cage before running the ghost.py files. If that explanation is too long for you, follow these rules:

1) SSH into *netID@student02.cse.nd.edu -YC* if you are PAC-MAN.
2) SSH into *netID@student01.cse.nd.edu -YC* if you are any of the ghosts.
3) Run pacman.py from student02 machine first.
    a. This player is the PAC-MAN character.
4) Wait 10 seconds.
5) Run any of the ghost.py files.
    a. These players are the ghosts.
6) Enjoy!

Gameplay: The basic controls are very simple. Regardless if you are PAC-MAN or the ghosts, the controls are the same. Use the arrow keys to move. The goal, if you are PAC-MAN, is to eat all of the dots while avoiding the ghosts. Another way to win, would be to eat all of the ghosts! The goal if you are one of the ghosts, is to run into PAC-MAN if he hasn't eaten a big pill; if he has, run away! You will turn blue and PAC-MAN will be able to eat you. This last for a set amount of time (about 30 seconds), and you will blink from blue to the original color of your ghost when time is almost up.

We wanted to add more strategy to the game. Therefore, there is no respawning. PAC-MAN has one life and all of the ghosts have one life each. If PAC-MAN eats you, and you are a ghost, you are done for the game; the other three ghosts have to attack PAC-MAN. If you are one of the ghosts, and run into PAC-MAN, then PAC-MAN dies and the ghosts win! Start a new game (by rerunning all of the files again as described above) to play again. Additionally, the ghosts cannot see each other! Only, PAC-MAN can see all of the ghosts. This makes it so that all of the ghosts cannot gang up and kill PAC-MAN instantly.

Tips: From writing the code and debugging our game, we have come up with a few tips from playing so much. Use them if you'd like.

If you're PAC-MAN:

- ✓ Eat as many ghosts as you can early on. This greatly increases your chances of survival. Then you can worry about eating all of the dots.

- ✓ There are two ways to win the game. You can either eat all of the dots, or eat all of the ghosts. Choose wisely.

- ✓ We made it really easy to move. If you know which direction you want to go next, hold the direction you want to go. If you are moving down a long tunnel going right, and know you want to go up at your next opportunity, hold the up arrow key and you will move up the next time you can. This will allow you to navigate the board more efficiently.

- ✓ You can see all of the ghosts, but the ghosts cannot see each other. Use this to your advantage.

If you're the ghost:

- ✓ The person controlling PAC-MAN will have no idea how many players are connected/controlling ghosts. If a player is not connected to a specific ghost, then that ghost will be really dumb and just move in random directions. Use this to your advantage. Blend in with the dumb ghosts, and attack when you see fit.

- ✓ Unlike PAC-MAN, the ghosts can stop moving. PAC-MAN will continually move in one direction until he hits a wall. The ghosts can stop in the middle of lanes. If there are a lot of humans controlling ghosts, use this to your advantage. It is easier to corner PAC-MAN this way if you can form a perimeter. To make it harder to corner PAC-MAN, you won't know where the other ghosts are though!

Effective use of the PyGame library: We use a lot of the things that we talked about over the semester. We blitz a lot of objects to the screen, depending on different actions. We use different classes, and tick on those classes to accomplish different tasks. For example, PAC-MAN dies in his tick. This is similar to making a gif. We also looked into text in the PyGame library and we were able to blitz the surface to the screen to keep track of score. We use sound, and have sound commented in our code, but it does not run on the student machines. We did it to demonstrate our knowledge of sound. We have a lot of different images and sounds that we use throughout points in our game, and you can see them under their specific image/ or sound/ directory.

Effective use of the Twisted library: We use the Twisted library in an effective, straight forward manner. Pacman essentially acts as a central hub to which all of the ghost players connect. So for the game to begin, the person desiring to play as PAC-MAN must first start their game, pacman.py, on the student02 machine. Then once PAC-MAN has their game started, ghost players can connect to PAC-MAN from any other student machine. Upon initial connection of a ghost player, the PAC-MAN game, which also functions as our central server, sends all of the pertinent information to that ghost player, such as PAC-MAN's location. Once this initial data has been communicated to the ghost player, the ghost machine and the PAC-MAN machine continue to communicate with each other throughout the course of the game, updating each other whenever one makes a positional change on the screen. The receiver of this new data then ensures that its own gamespace objects reflect the reported changes.

Scale and Complexity of Project: We chose a very large scale project to replicate. It took a while just to get all of the dots blitzed to the correct location of the screen. We also spent a while bounding PAC-MAN to the board so that he could not go through walls. Once all of the basics were set up, we added scoring, made the dots disappear, made PAC-MAN munch (animation), and made the ghosts' eyes move in the direction that they are moving (even on their life counter at the top of the screen). Once we developed the game in its entirety, we added networking. Combined, we probably spent 60+ hours working on this game. We had a lot of struggles along the way, but also had a lot of success and think that we chose a very appropriate project. We are happy with the way our game turned out. We had two different GitHub accounts during the course of the project (the first one had to be abandoned because we ran into problems when we chose to upload a 15 minute sound file). Once we ran into this problem, we copied all of our code, including a .gitignore file for that sound file, into a second repo. That link was the one we used for the rest of our project. Both links are below:

1) https://github.com/rtick/Paradigms_Final
2) https://github.com/RyanAMoran/NewParadigmsProject

Combined on the two repos, we had close to 100 commits. We have five different python files needed to run the game to full extent totaling to about 5300 lines of code.

Known Bugs:

- Lag can cause a few bugs in our program. The more ghosts connecting, the more lag time. For instance, if a ghost runs into PAC-MAN on one machine, he might die on that specific machine but his position might update so that he does not die on another machine for another ghost. If we had more time, to fix this, we could send more information from our central hub to our clients (ghosts).

- If you click buttons on the keyboard during our start (loading) screen, then you get a pickling error and the game will not run. Do not touch any buttons during the start screen. Restart the game to fix this.

- If you start a ghost before pacman.py has time to initialize everything, you might get an error causing the game to not run properly. Restart the game to fix this.

If We Had More Time:

- We would add fruit that would randomly appear to the screen for PAC-MAN to eat. We had this image in our /image directory, but never got a chance to implement it.

- We would restructure our code so that there is a Ghost class instead of 4 different ghost files. This would allow the players to choose what ghost they want to play as.

- We would give each ghost certain advantages and disadvantages.

- Add more levels, and more than one life for each character.

- Add a better start screen, a winning screen (either for ghosts or PAC-MAN), and a high score screen.

- We would add an option for an in-game tutorial.

- Find a solution for running sound on the student machines. We might do this by bettering our networking.