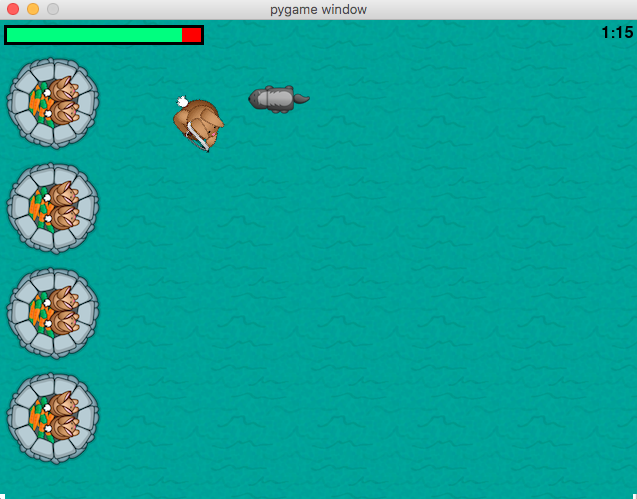
TAP TAP REVOLUTION



## Anthea li, period 6

My final project, “Tap Tap Revolution,” is a spoof of the popular game “Dance Dance Revolution,” but adapted for the regular computer keyboard. The game features left, up, down, and right arrows on the screen and arrows of the same directions that animate up the screen, speeding up as the game goes on, and the player must press the arrow keys at the correct time when the background empty arrows line up with the animated arrows. Depending on how close you are to pressing the arrow keys at the correct time, you gain points. The player is allotted thirty seconds to get as many points as they can before time is up.

“Tap Tap Revolution” is programmed in the language Python, or more specifically, the cross-platform set of Python modules designed for writing video games called pygame. The goal at the beginning of this project was to create an arrow-key based game that would be made multiplayer through the use of the Python implementation of the event-driven realtime server socket.io. However, due to a huge time crunch and my underestimation of the time and effort needed to complete my goal, sadly I did not end up accomplishing everything. In the end, I was only able to complete the game portion.

I started off my project by following a guide for a simple Python game written with pygame that I found online in order to expose and introduce myself to the language, to dip my toes into the water before diving in. The introduction game was fairly simple, it was a "protect the base" kind of game with a primary player that the user controls and enemies that needed to be killed.  From this sample game, I learned the basics of pygame such as how to construct the background and the key elements of the game window, how to "animate" certain elements sufficiently, as well as how to add a timer.

After completing the introduction game, I began the process of coding "Tap Tap Revolution."  I first gathered all the images I needed, such as the background image and arrows.  Next, I constructed the base of the game window, the parts that don't move throughout the whole game.  The next step was to animate the arrows across the screen.  The first approach I attempted was to create a plethora of variables for each characteristic of each arrow.  Halfway into this approach I realized that the code was going to be ridiculously messy and practically unreadable to another person.  My second approach was creating a class called "Arrow," and included instance variables for its direction, x-value, y-value, and transparency.  The transparency variable dictates whether the arrow was a moving arrow or a background arrow that stayed stationary.  Using a class to create Arrow objects was extremely useful; it condensed all the random variables that were previously declared for every arrow individually.

The next step was to make an arrow disappear when its respective arrow key is hit.  Theoretically, this should be quite simple with the usage of event listeners, but what I thought would be easy turned out to be the most time consuming and difficult part of the whole project.  With my background knowledge of event listeners from Java GUI, I easily constructed an event listeners portion for the arrows, as well as attempted using the pygame.Rect class to simulate collisions. However, this concept proved to be a bit too complicated and was not worth the trouble, so I resorted to using x and y values to test collisions. I soon found out that the arrows weren’t disappearing from the screen when the respective arrow key was pressed. Eventually, I figured out this was due to my use of a for loop to remove arrows from inside the iterating list of arrows. However, since Python doesn’t have a list iterator that goes by index, nor does it have an Iterator class that works fine when removing things, I resorted to the use of a list comprehension, native to Python. This was an almost entirely new concept of list iteration to me. A list comprehension is a line of code that creates a new list that only retains the items that meet a certain condition. For example:

list[:] = [x for x in list if (x % 2 == 0)]

When I replaced the for loop with a list comprehension, the initial problem was solved only to reveal a new one. This time, when an arrow key was pressed, the arrows with the same direction all disappeared instead of the topmost one. This bug was the hardest to resolve and took me days to figure out. My first approach was to modify the pixels that tested the collisions, which didn’t change anything. I tried about a million different other things such as using different iteration techniques and retrying the pygame.Rect method, all of which either brought up new bugs or didn’t change anything. Finally, I decided to try using one array to store all the arrows and then one array for each of the four directions of arrows. For some reason, this seemed to work rather than just accessing the direction attribute of each arrow, and I still am not sure why.

The last steps were to generate a pause function, a timer, a score counter, and a “Game Over” screen. Most of these were quite easy, however I did have a bit of trouble with the “Game Over” screen. I had at first attempted to bring up the “Game Over” words outside of the loop, which didn’t work because when the “exitcode” variable I used to keep track of whether the timer was up or not was checked inside the loop, there was no way to exit the loop and have the “Game Over” words show up on the screen for more than one frame, since things that are printed onto the window have to be reprinted with every iteration of the main loop. I succeeded when I checked the “exitcode” variable as well as bring up the game over screen inside the loop.

If given the chance, I would not have spent so much time on the introduction game. I spent more than half of the allotted time following the introduction guide, which was unnecessary because many of the concepts I used to create “Tap Tap Revolution” I had either already learned or could easily Google to quickly get the gist of. If given more time, I would have created a beginning screen where you can choose the difficulty level and music, as well as include a scoreboard showcasing the highest scores achieved followed by the player’s username.

The most valuable thing I learned from this project was the basics of Python, as well as the basics of game programming. I’ve been interested in game programming for quite a while now, mainly propagated by my love for video games, and this knowledge will surely be useful in my future. Python is known as a very popular language due to its graphics capabilities and simplicity, and learning the language will be the beginning of my own computer science initiative, the starting point of my future explorations of the programming world that has not been covered in the computer science classes at school. This project has no doubt impacted my plans for the future, for the insight it has given me into the game programming world has piqued my interest in the field enormously.