Programming Journey

During our journey we learned a lot about the pygame library and its very useful functions. We also learned a lot while programming something without any preconditions just from a “plane paper” on. Another nice thing we encountered throughout our journey was the usefulness of Github. Working with a repository on Github helped us working separately on the project without having to discuss what was already done.

Now onto our journey (This is a broad description of our journey and the challenges we faced):
After we read more about the pygame library we started with the basics of our project. That was our first challenge: starting with a plain sheet without any instructions. We decided to first write a broad outline on a piece of paper stating which things we had to implement first and which rather at the end. This helped us organizing our thoughts and made it easier to start. After that we implemented the pygame window in which we wanted to play our game and the end function to close the window.

Then we created a yellow dot on our screen as our Pacman. There was our next challenge namely the move function. For the move function we had to get the x and y values to change our current position. To solve that problem, we made a vector class to implement a vector with a start position and some useful calculations. With this class moving our Pacman and getting its position made it easier for us. With that we added move functions with which we could move our Pacman by pressing the arrow keys.

Next, we went on and designed our game layout with paths and bricks. We did that by making a txt file with 0s where the bricks are supposed to be, 1s where the coins are and 2s where there is nothing at all. This layout is saved in the 1.txt file. This file had to be read in and changed into a numpy array. Our map class handles the current state of the map which is saved in the array. We added some pictures to where the 0s and 1s are in the array and then our map was done. Our intention of using textfiles was that you can easily add new levels if you like without having to change any existing code.

We wanted our Pacman to move through our maze without going over the bricks and there was our next challenge, our Pacman often appeared to be on the walls instead of on the path. We did not consider the tiles on which our Pacman moved so we checked in the getDirection and validDirection functions if we are exactly on a tile(entry of our array which contains the map) and not between two tiles. This was done with the if-condition col%TILESIZE == 0 and row%TILESIZE == 0, so you can only change the direction if it is valid and when you are exactly at an entry of the underlying map array. We also had to adjust our speed so that TILESIZE % SPEED = 0, because without this you may not be able to navigate exactly on the underlying array structure, which leads to wrong movement or may even result in a dead end where you cannot move pacman at all.

After that we implemented a function updateValues with which we could make our Pacman “eat” our bonbons thus if Pacman touches a bonbon it will disappear.

In the following we created our enemies. This was another challenge for us because first we only had one class namely the ghost class for everything. It managed the movement and the creation of the ghost. This was not the easiest way since we had to created more than one ghost hence, we decided to make another class called ghostgroup in which we implemented all our different enemies with their different colors.

In the next step we implemented all the end functions like winning or losing. You win if you eat all the coins, and you lose if you get hit by an enemy three times. For the count of the remaining lives, we made two functions: looseLife(function) so if you hit an enemy you lose a life and alive(function) which

checks if you have more than 0 lives. Additionally, we also made a function which resets Pacman's start position every time it hit an enemy and has more than 0 lives.

At the end there were only some details left which would improve our game like showing some text if you win or if the game is over. Another detail was the moving mouth of Pacman which was our last challenge. The movement of Pacman's mouth was way too fast so we had to find a way to make the pacman animation slower than our in game FPS. We solved this by implementing another variable in the Pacman class which makes sure that the Pacman is three times slower than the normal game updates. After figuring out this last challenge our Pacman game was ready to be played and our project was done.