Programming Project report

Super Mario Bros

Martín Ranea Marina and Alejandro Taboada Esteban
Group 97
Data Science

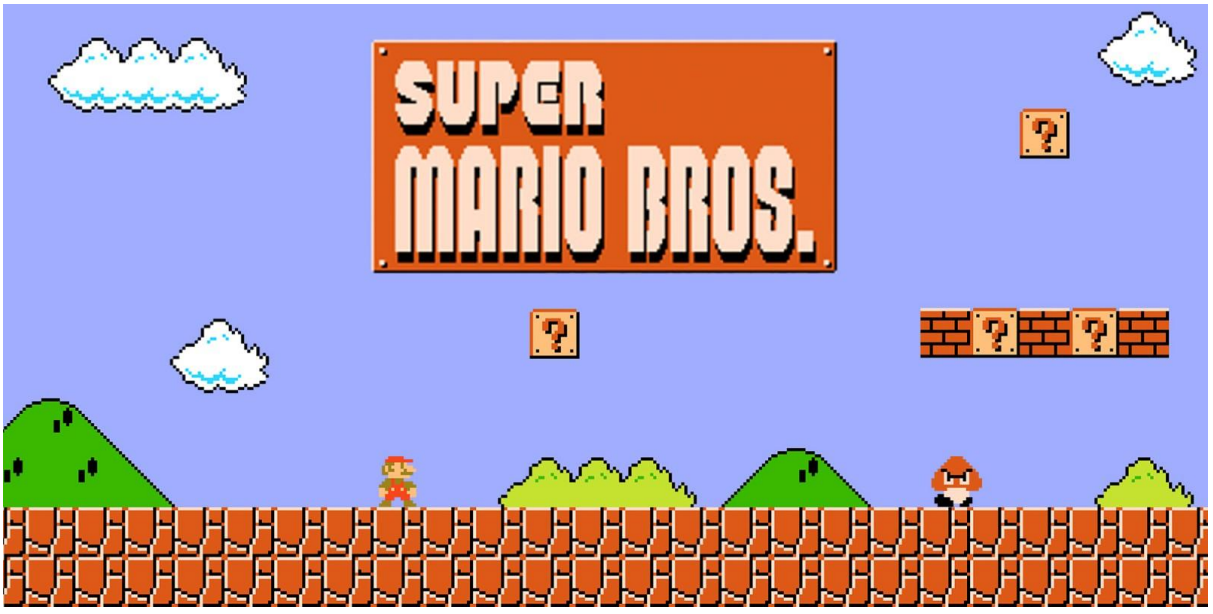
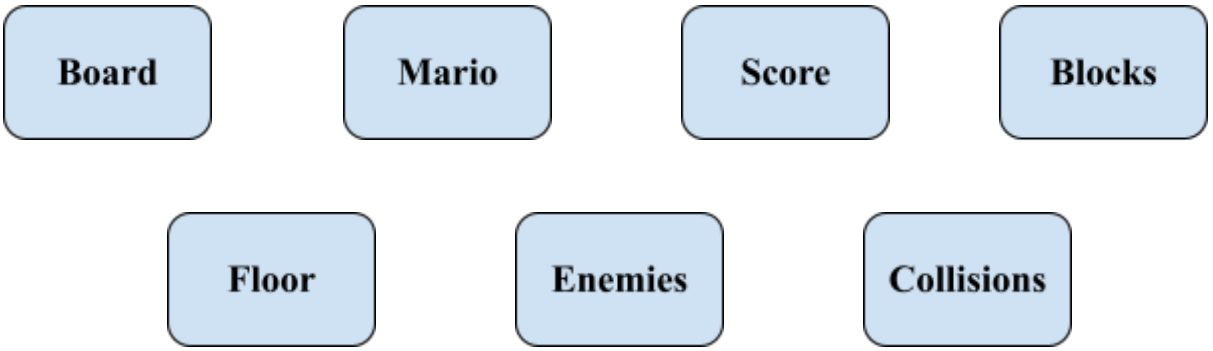


Table of contents

Classes Design

For our final project we have created **7 classes** to organize our code better. From these classes, there are ones that are more important and others that are less important, but all have a meaningful function.



Board

This class is the most important class as it contains the main code of the videogame. Inside the Board class there are two main functions, called draw and update, which are the ones in charge of making all of the other classes work. The draw function makes everything appear on screen by drawing them with the use of pyxel, while the update function is the one that makes the actual game function by moving all of the sprites and activating the collisions between them.

Mario

This class contains all the information about Mario and how the character moves and jumps through the map. It is important to mention that Mario can either be Big Mario or Small Mario and could be moving to the right or to the left, so depending on which Mario it is and which direction it is going to, the sprite changes. In addition, we decided to create the jumping action by using gravity, which also appears in this class.

Score

In the score class, we can find everything related to the marker of the game; score, lives, coins and time remaining. We used **import time** to make the time count downwards from 400 and when it reached 0 the game finished. We also included a “game over screen”, which appears if the time remaining or the lives remaining reach 0, and a “game winning” screen, which appears if you reach the final flag, containing the final score, lives and coins.

Blocks

This class does not contain much, but it is very important as it contains all the types of blocks, including the final flag and the mushroom, and their sprites. We also use this class to assign a width and height to each block, as there are some blocks such as the pipe that have a wider width than other blocks.

Floor

This class is not as important as other classes, but is needed due to the fact that we decided not to make holes in the floor where Mario could fall into. For that reason we created the floor as a border which Mario could never surpass.

Enemies

This class is similar to the Mario Class, but for the two types of enemies we have decided to implement in the game. As both enemies have the same behaviour, we created the same move function for both of them but subtracting 8 pixeles to the y coordinate of the koopa troopa, as it is 8 pixeles higher than the goomba. The game instructions mentioned that the probability of an enemy being a koopa troopa must be 25% and a goomba 75%, therefore we created a random randint from 1 to 4, and if the number was equal to 1, the enemy would be a koopa troopa, and if it was any other number, it would be a goomba. For that reason, in this class we stated that depending on the number randomly generated, the sprite was either a goomba or a koopa troopa.

Collisions

This class was one of the most important classes of the project, and the one that contained the most lines of coding. The main reason for it was that it contained the lists of blocks and enemies and all the interactions and collisions between them and mario. This means that we had to study all the different conditions that could appear in the game, which included that mario could be small or big, enemy could be koopa troopa or goomba, all the different types of blocks and the final flag.

Main

Main is not an actual class, but it is very important as it is in charge of running the whole program, assigning which pryxes document has been used and also shows the dimensions of the screen.

Most Relevant Methods

move - This method allows mario to move and change sprite, depending on whether it is moving to the right or to the left and if it is Big Mario or Small Mario. In addition, we decided to implement a sprint function which increments the velocity of mario by pressing the shift key.

jump - In this method, mario is allowed to jump along the whole map. At first, we made a basic jump with no acceleration but as we progressed through the project, we decided to create a gravity variable, to improve the program.

interactions - This is one of the most important methods and the one which we had the most complication, as it involves a lot of coding. This method includes all the possible interactions that may exist between mario and any kind of block, including normal blocks, mushrooms and the final flag.

die_mario and ***kill_mario*** - These two methods are very important, as it allows mario to interact with the enemies. In die_mario, enemies can kill mario by simply touching Small Mario horizontally, or making Big Mario go back to Small Mario. On the other hand, kill_mario allows mario to kill enemies by jumping on top of them.

show_screen - This method is related to everything that has to do with the marker, it makes the score, lives, time remaining and level appear on screen, and makes the time remaining decrease from 400.

update and ***draw -*** As mentioned above, these two methods are very important, as they are in charge of allowing all the movements of the game to function, and every sprite to appear on screen.

Performed work

To start with the project, we created the main class with all the necessary to run the program, and the board class with the width and height attributes, and the update and draw methods, essential for running the program.

Later, we created a file with all the constants needed for the program, because it will simplify a lot of things and make the process easier. We also created a simple Mario class with simple attributes, its x and y position and its direction. We also created the background and found a way to make it move, with a variable in its y position.

Furthermore, we created a movement for Mario, with two different speeds, differentiated by the sprint parameter. Mario's movement can go in two directions, right or left, that have to be introduced when using the function. We decided that the best way to manage the movement of Mario, the background and the future blocks was putting all the movements in the board file, in order to coordinate them.

To count the points and coins obtained we created a score class that would store the obtained points and coins, the lives left and the time remaining. At first, making the time run was very difficult for us because we had never worked with time in python, but we solved it by creating three different variables and using time.time() . We created a method to show all this on the screen, and two different methods in the case that Mario won or lost. These methods consisted of showing a victory or loss screen.

Then, we created a jump method, consisting of a speed and a gravity that constantly reduces the speed. This speed affects the y position of Mario when spacebar is pressed, and when Mario arrives to the ground because the speed is negative you can press spacebar again to jump one more time, We found it very good and realistic, but we had one problem, and it was that we had not created the blocks to interact with yet, we had to set the jumping method with a top y, the one of the floor, so that Mario could never be under the floor.

After that, we made a block class to characterize all our blocks. To use the class, you have to put the x and y coordinates of the block., and its type. Depending on the type, a different block will be drawn. To develop the collisions we created a new class, and inside it we created a huge list with all the blocks of the level, by importing the block class.

At first we really struggled making the collisions, because it was a completely new concept, but we started drawing all the possible cases where Mario touches a block, and we were able to finish it. Once we created the collisions that changed the position of Mario when he touched a block a new big problem appeared: when Mario was over a block he could not jump and when he moved outside it he did not fall. After hours of thinking we added to the jump a variable that allowed jumping and we activated it when Mario was over a block, and to make it fall we created a method that made Mario fall when he did not have a block below.

As we saw that the floor, which was in the background, moved a bit slower than the blocks, we created a floor class and a list of the floor blocks, and a function to show them and move them while the blocks were moving.

To make the level a bit funnier we started creating enemies. We created a class for them, with their x and y position as parameters, and a number from 1 to 4 that would decide their type. We created a list with all the enemies randomly selecting their type, and we created a movement method for them.

Making the collisions for the enemies was pretty easy, because we already had the ones for Mario, and we got inspired by them. We also created different interactions between Mario and the enemies, killing them when Mario jumps on them and subtracting a life to Mario when he touches them on the sides.

Then we started making some important interactions, like making some blocks give a coin or a mushroom when Mario collides with them from below. We also created the big Mario and activated it when he touched a mushroom, and added new functions to it like breaking blocks or, instead of losing a life when crashing into an enemy, becoming small again.

Finally, we put all the functions and methods into the board class, in the draw and update methods, tested it a lot of times and fixed some details. To do this last think it was very useful to reduce the fps down to 5 or 10, watch carefully what failed, and go fix it.

Conclusion

During this project, we have learned a lot about python and have had a lot of fun working through the project. Working together has made us unleash our greatest potential and helped us solve any problems we had to face.

Pyxel has been a very interesting and fun tool to use. Having to draw different sprites which then had to be implemented into the game has been a very entertaining task. However, one of the things we would have liked to do if we had more time was to add animations to Mario and the enemies. Due to the amount of exams we had during these past weeks, we were not able to make koopa troopa drop a shell when killed, which then moved right and left if mario jumped on top of it. There are many more things that we would have liked to add, which is why we do not have a doubt that we will keep improving the game just for satisfaction and pleasure.

It has been a great experience making a game we have all played at least once in our lives. When we were first told that we had to program the actual game of super mario bros, we could believe the fact we had to do such an exciting project, but also thought that it would be very hard to do. However, with a lot of teamwork and time, we managed to complete the project.

Finally, it is important to mention that our programming level has improved a lot during this project. Both of us began the course with no knowledge of programming at all, as we had never programmed before, and have learned a lot.