DONKEY KONG

REPORT

BY: GUILLERMO ANDRES HERNANDEZ BELLO

AND

PABLO MONTERO POYATOS

We were asked to simulate, using Pyxel (an implementation for python for doing retro games), the first level of the game “Donkey Kong”. For simulating this game we have used Object Oriented programming creating a system of classes, attributes, methods and objects to be as similar as possible to the original game.

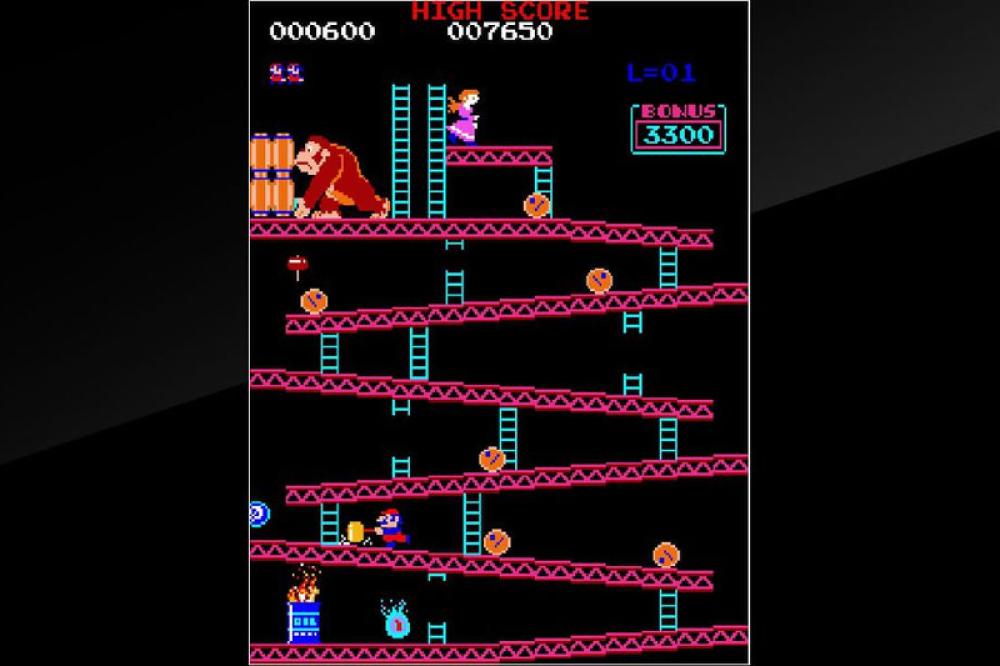
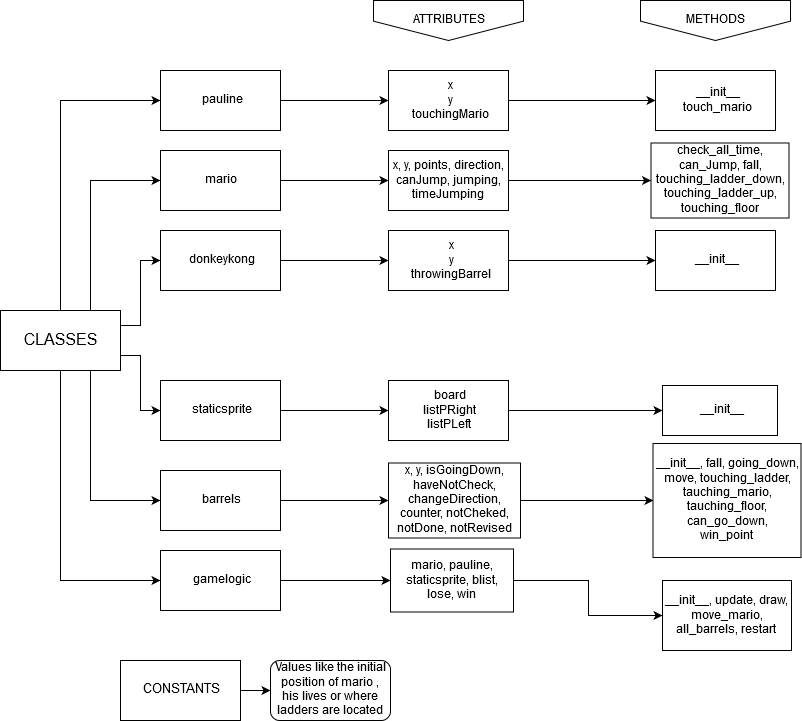
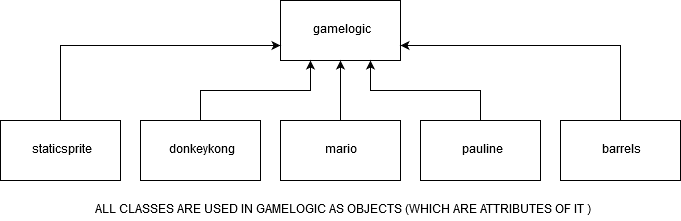


Photo from the level that had to be simulated

Here is a diagram of the files and the classes used in our program:



As we can see 7 files are used where 6 of them are classes of the program. The main program is the class called game logic in which the rest of the classes appear as objects. Platforms and ladders are implemented using the matrix board in the class staticsprite.



**CLASSES:**

* **Gamelogic:**

This is the main program class where the rest classes are attributes of this class. Here we control the general functions like checking every frame which buttons are pressed, restarting the game or creating barrels as objects randomly and appending them into the barrel list. Also in this class will appear every command related with Pyxel such as printing on the screen the sprites (made with the pixel editor) or showing the points the player has.

* **Staticsprite:**

In this class, we will have every element of the game that is static. The most important attribute of this class is the board which is the base of our program. With this 224x269 matrix, we will control what we have in a certain position like ladders or platforms that will be used in methods that for example make Mario fall or the barrels move. This matrix is composed of 0 (there is nothing in that position), 1 (platform), 2(ladders that both Mario and barrels can use) and 3 (broken ladders that only barrels can use).

* **Donkeykong:**

In this class, we create Donkey Kong defining its position and say if it is throwing a barrel (for being able to animate it).

* **Pauline:**

In this class, we create Pauline defining its position and checking if it is in contact with Mario to finish the game and make the player win.

* **Mario:**

Mario class is one the most important ones, here we define all the Mario movements (go right/left, go up/down a ladder, jump and fall), we check the lives it has, to finish the game or not, and we also have a value for the points it has.

In this class, most of the methods are related to the movement of Mario like “touching\_floor”, that makes Mario not fall, or “touching\_ladder\_down” or “touching\_ladder\_up” that lets Mario use ladders.

* **Barrels:**

This is also one of the essential classes. Here we check everything related to the barrels, like their movements (the position, the direction they move, whether they use ladders, if they can fall or etc…). In the class, most of the methods are used to control how barrels move, but there are also functions used for other proposes, like checking if they are in contact with Mario or detecting if Mario is jumping over them to give the player points.

**MOST RELEVANT ALGORITHMS:**

The most relevant algorithms are mainly in the classes Gamelogic, Staticsprite, Mario and Barrels. Something essential is the attribute “board”, which is by far the most important variable in our program. With that, we have been able to set all the statics sprites and define a lot of functions, related to them, like the contact with the platforms or with the ladders.

It is also important to consider all the functions related to movement that appear in both Mario and Barrels. These functions are mostly Booleans that give conditions to be checked before letting the objects do something, like going down a ladder or falling. The rest of them just change the position of x or y depending on what we want the object to do. Without these methods, the game would not have any sense because nobody wants to play a “not moving” game.

All in all, the importance of the class gamelogic must be highlighted, as it is where everything is created. Without it, we would only have a bunch of classes that are not created as objects so they will not have any purpose. Indeed, every Pyxel command used appears in this class, so without this we would not see anything of our program as well.

**PERFORMED WORK:**

For being able to end the project on time, we have been following an every week schedule and doing more or less each sprite on time, by meeting two or three times a week, apart from the practical classes we have.

The first week we made a general idea of the classes that we were to use and we also created the board matrix. One of the problems we found was that our matrix was 224x269 so the floor would not be a line of 1s or the ladders would not be a column of 2s, they would be a bunch of them, so we kept in mind that in the future we would have to work with ranges and not with certain positions. We used loops for printing everything, with jumps of 8 numbers for not superposing the sprites.

During the second week, we had to make Mario move and fall so we took as an example the program of the circle that moved controlled by the keyboard, which was given to us. We implemented some functions to control if Mario was touching the floor or touching a ladder to define whether he could move in certain situations. The biggest problem we came across, at this part, was when Mario arrived at the top of a ladder, he could not continue. This happened because we were checking the position of Mario’s feet and the stairs ended in a platform, so he could not pass through the platform. For fixing this we added a reference also eight pixels, more or less, bellow his feet and he finally could pass. So we ended up checking two points, his feet and eight pixels below them.

The third week was about the barrels, where we reused some functions of Mario like falling and moving side to side. This week we had two remarkable issues which were changing the barrel’s direction after it had gone down (because it changed direction before going down and did not fall), and generating them randomly but not creating two in an small period of time. Luckily, we finally were able to solve these problems with some frame counts and the errors were fixed.

The last week, we have not had big problems apart from counting the points and animating Donkey Kong to throwing barrels. We include barrels in a list and delete them perfectly, there was no problem of respawning Mario and animations worked well, the problem was winning points when you were hit by a barrel.

When you hit a barrel, you enter in the range of the barrel that we introduce to win points so you win points hitting it, so we had to change that range and include a condition that says that you can only win points if you are jumping.

**CONCLUSION**

Finally, we have been able to finish the game overcoming a lot of problems by solving a big amount of small details that were wrong and a lot of indentation errors. The work has been long and hard but we feel happy about having made our first game.

It has been a good simulated experience of what will be demanded from us, in the future, as professional programmers and it has been fun doing it due to it being a game and not another type of program.

In conclusion, we are happy about our work and we hope you have fun playing it.

**PERSONAL COMMENTS:**

Honestly, we have found that doing videogames is not that easy as we thought it would be. Sometimes it has been frustrating and exasperating, but we have enjoyed it we things slowly started worked.

We think Pyxel is a good tool to start with Object-Oriented programming and it is cool to have learned a bit of how to use it.

Summing up, we have learned a lot from doing this game. Not only about programming but also about how to deal with frustration, making things work in a difficult moment or working in pairs, so it has been really advantageous to make this project.