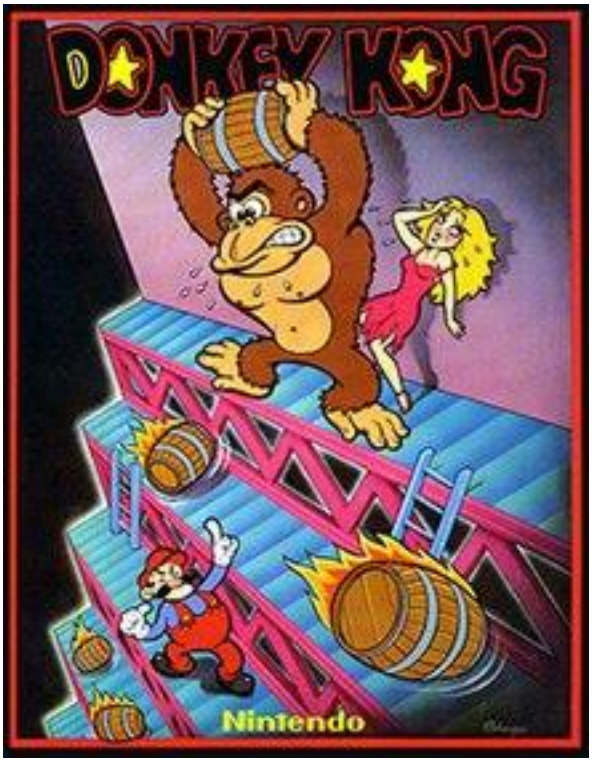
MARIO
VS
DONKEY KONG



RAÚL DÍAZ DEL SER and JUAN CERDÁ BERMEJO

GROUPO 96

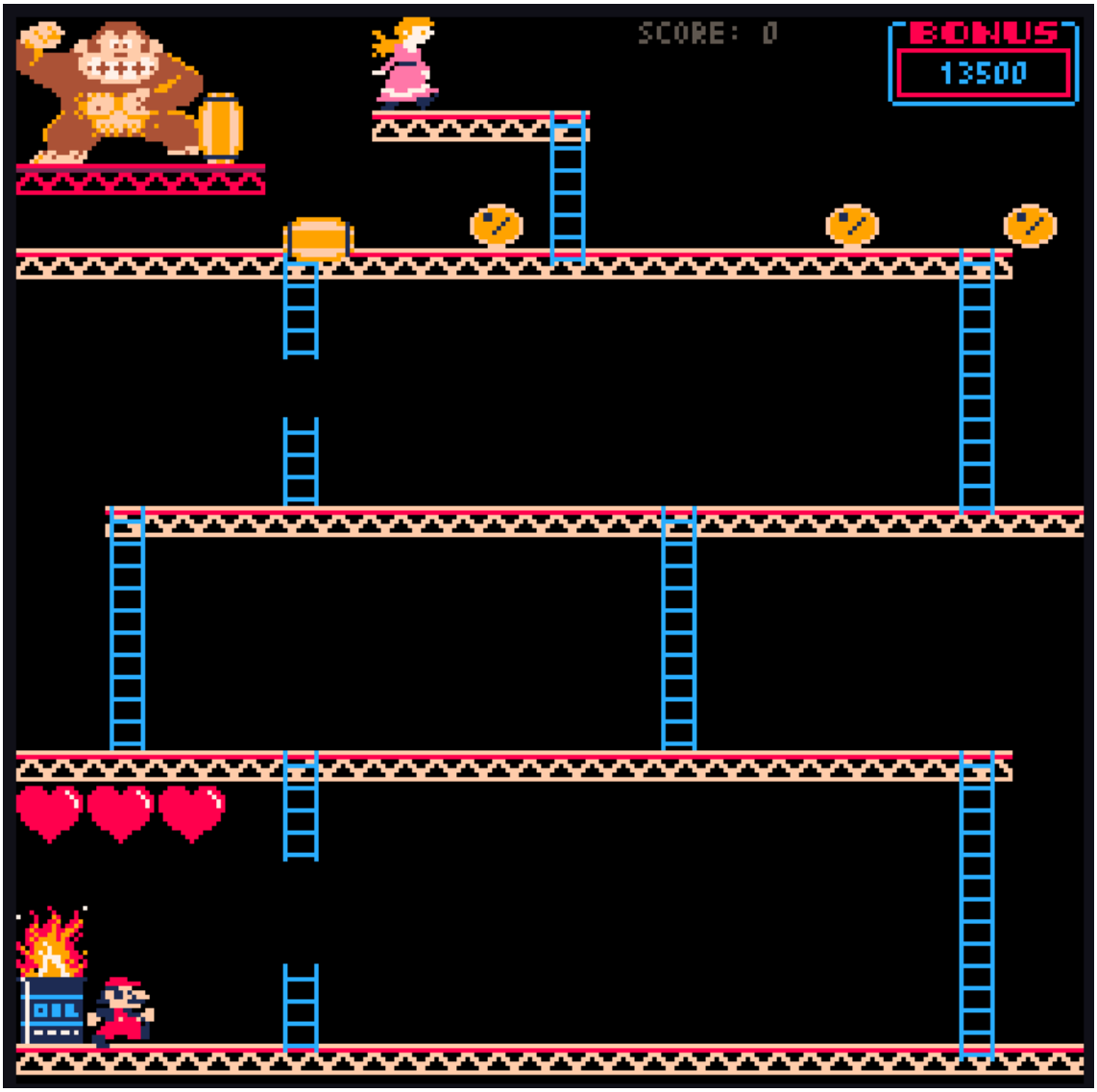
The goal of the project was to create a *"Mario vs Donkey Kong"* video game. In an attempt to do it, we have used the python´s library, PYXEL, which had let us create and design a game from scratch.

Index:

1. Game:

"Mario vs Donkey Kong" is a videogame developed by Nintendo. It was released on 1981 as an Arcade version. It is one of the first platform games of all times.

The protagonists of the game are Mario, an Italian plumber, Donkey Kong, a gigantic ape, and the princess, Mario's love. Donkey Kong, enraged, took custody of the princess, and Mario's duty is to rescue her from the ape. In an attempt to do so, Mario has to walk through several platforms while dodging and jumping over obstacles.



2. Class Design:

In order to achieve a better videogame as a result, we have created 12 different classes. Most of them are going to be reflected in the actual game, and with a part of those, we will be able to interact throughout the game. Other classes, as "constants", hide information that is useful to carry out the game as well.

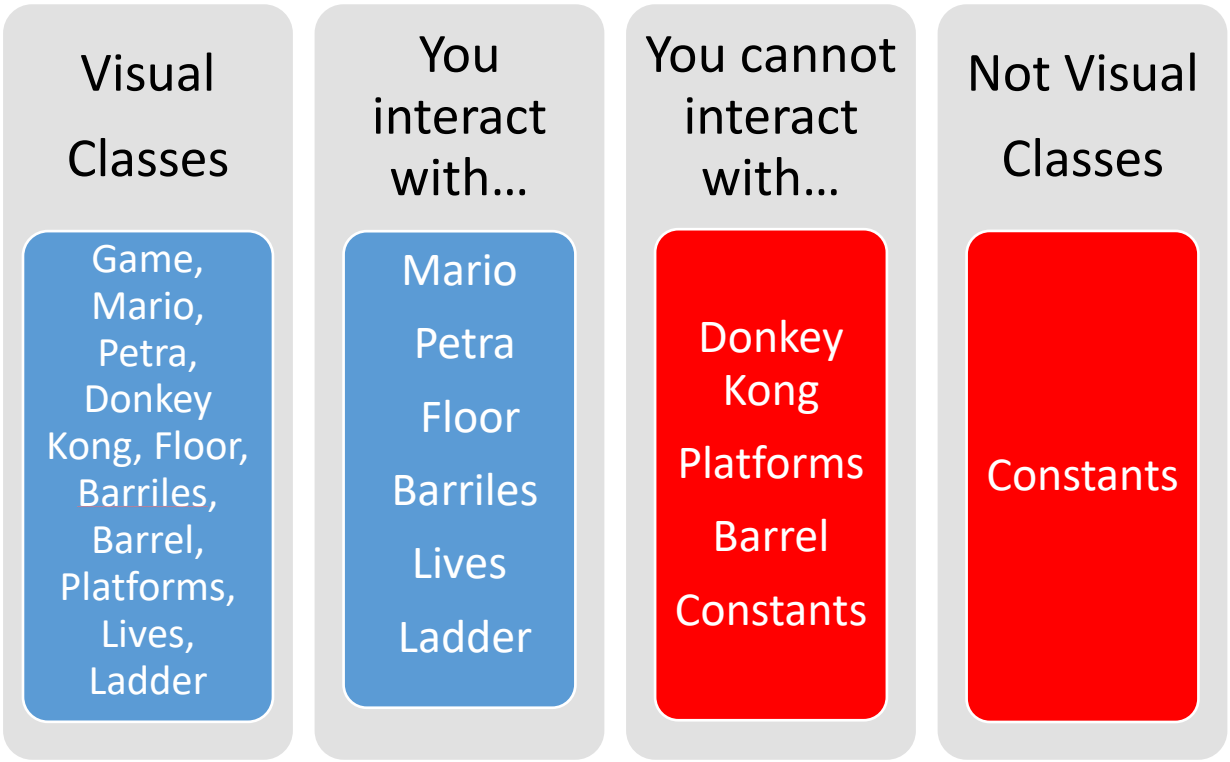


Figure 1: Class classification.

Game: It is one of the most important classes we have created. It contains the videogame itself. It is formed by ***moveMario***, which help us move the main character as we are playing, ***moveBarril,*** which also helps the videogame to move the obstacles that Mario has to dodge, ***update***, which contains information that contains information on which the course of the game depends, and finally ***draw***. This last one has the visual information, that way it is possible for us to have a visual look at the game we have created.

Mario: Contains information about the main character which is controlled by the user. This class contains information as, Mario's

position on the screen, Mario's ability to jump, climb, fall or even die, and last but not least, Mario's lives during the game.

Donkey Kong: This class represents the enemy in the game. It does not affect in the actual game, in other words, you do not interact with it, although it is visually represented in the game.

Petra: This is the princess class, in our case we decide to call the princess Petra, hence the name of its class. The information which it contains is her position on the screen, which our protagonist Mario has to reach to win the game.

Barriles: "Barriles" are the main obstacle that the player has to dodge to rescue Petra. The class we have created, defines as how the barrels move along the floor, in a clockwise direction, or to the left, or even how fast they fall from one floor to another.

Floor: The floors are the most basic piece of the map. The player will only be able to move in a direction if standing on a floor and so will the barrels. The objective of the game is to get to the top floor therefor its essential.

Ladder: During the game Mario has the ability to climb ladders. In this case, the player will interact with this class throughout the game.

Barrel: This class is merely decorative. It contains its position x and y represented on the axis.

Constants: As we have seen before, ***"constants"*** is the only class which belongs to the "Not visual classes", in other words it is an abstract class. It is one of the most important classes we have created, it contains predefined information. This class was useful if during the course of the project we wanted to change something that affected our game. For instance, at the beginning we decided to give a value of width and the height of the screen of 160, later on we realized it was not enough, so we decided to change to 240 both.

Also, it contains some other useful information, such as, Mario's position on the screen, the number of his lives, etc. that we have

used in the "***Game***" classes, that way we were able to develop a better game.

3. Methods:

The most relevant methods are those linked to the movement of Mario and the barrels. These functions are essential so that the game works and can be played.

First, we´ll take a look into the movement of the barrels. The function that summarizes the movement is *moveBarrel*, and this is the function that will be processed in the update. Inside this function we have a group and conditions and subfunctions that make the barrels move perfectly and interact with the rest of the map (floors and ladders). We can specifically find the functions *move* and *fall* which are defined in the class "barriles"; both of these functions update the position of the barrel either up, down, right or left.

Secondly, we have two basic functions for Mario: death and *moveMario*. *Death* is a simple function that whenever Mario gets hit by a barrel resets his position to the beginning. Another simple function is the function *fall* which simply checks if there's a floor right below Mario and if not than Mario falls to the floor. *moveMario* is the function that encloses the conditions and subfunctions for the movement of Mario. The return of this functions also depends on the keys entered by the player. Inside the function we can also find *move* (dependent on a direction), and *jump* and *unjump*(They automatically connect and together make Mario jump); these three functions are all defined in the class "Mario".

4. Relative algorithms:

The most relevant algorithms are update, draw and run. All of them are located in the main class "game". Run is the base function that executes update and draw in a loop so the game can function.

In the function update we run all the other functions that are necessary for the game to work and the elements to move and interact. The function draw is where we paint the elements in the screen, its necessary to be able to see the game.

Other algorithms that we found useful are some of the ones from the pixel library and random library. We used *frame_count* to be able to time the showing of elements in the screen. We used *playm* to introduce music to the game. We used *btn* to check when the player presses a key. We used *blt* to draw on the screen. And we used *randint* to introduce probabilities to our game.

5. Work:

We have been able to carry out the project successfully with a lot of work. This project has not been easy for us, we felt a little lost but as the days went by, we started to catch on and also started to improve the game.

Everything has a starting point, to be able to do the project we had to find out how we could draw the characters on the screen. After we have done it, it would be a little easier to develop the videogame since we have a visual look of it.

We managed to draw everything in place after a couple of practical classes. The next step was to make Mario move to the right or to the left. In order to do it, we played around with Mario's class, that way we were able to modify its position during the course of the game. Also, we had to take care of Mario going upstairs and downstairs and the barrels' movement. In an attempt to achieve it, we made the code pixel by pixel. Even though we knew it was not the best way. After we were able to finish all the object movements, we decided to move everything into lists, that way we could keep the code as simple as possible.

As we explained before how each element influences in the game, for us the hardest part was the barrels' movement. We had some issues, moving them into lists, but after numerous tries, we were able to accomplish our goal of creating a "cleaner" code.

After we structured of the game, we proceeded to animate the character, that way we would give a better visual look to our game. To make this possible, we have played around with *pixel.frame_count*, applying the method to each character, it helped us to make the character do a random movement after a certain time.

Other extra features we have added, were the winning screen, where you can see how Mario has rescued Petra; the song, which we invented, and the game over screen. This was not strictly necessary, but we wanted to make it as close to the real game.

6. Conclusions:

It has been a long road to achieve the result of our project, thanks to it we have learn a lot about object-orientated programming and python's library "*PYXEL*".

We could say that we are not aware of what we have done, we have created a "*Mario vs Donkey Kong"* doing it in our way. No body says it has been easy, but thanks to each other we have been able to go through any problems we have had during this month. Both of us have contributed ideas, and knowledge, but we have worked as a team.

As time went by, we have solved any problem we had, letting us make the game we have presented. We have learnt a lot about python's ability and mostly from each other. We are really happy with the work we have done.

One useful page we have visited during the course of the project was https://pypi.org/project/pyxel/, . It has helped us to understand better how "PYXEL" works and to get rid of any doubts we had.

7. Personal Comments:

After finishing the game, we can say that even with the occasional frustration that working on this project produced, we have really enjoyed the experience and feel that we have learned a lot.

Before this project we could have never imagined that we would be able to build an arcade game from scratch. Even though we had the most basic code done (by pixel), we feel like we have outdone ourselves.

The only negative part about the project we can think of is the time it takes but we agreed to think that it was time well spent.