**Final Project: Donkey Kong**

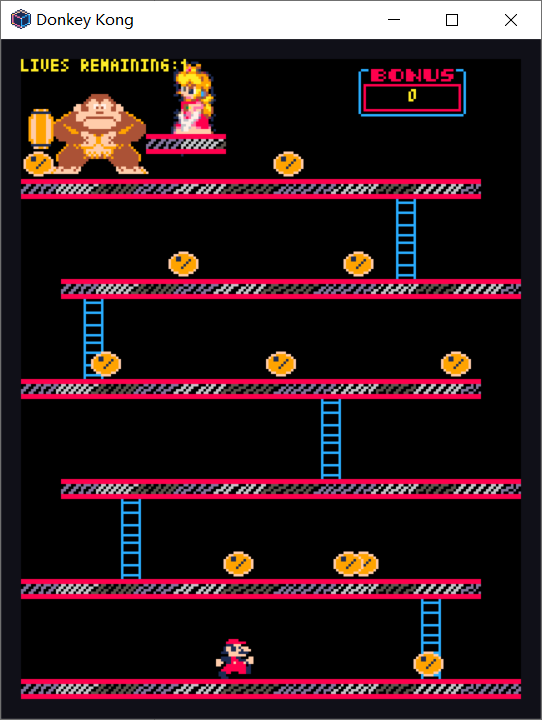
**By Zijun He & Aya Benhariz**

**Group 96**

**Data Science and Engineering**

**Abstract**

This project is to regenerate the game Donkey Kong using Pyxel by following instructions given by professors. Skills and Knowledges learnt from the class will be implemented in the project, such as: functions, loops, objected oriented programming.



**Table of Contents**

* Class Design----------------------------------------------------------- 3
* Relevant Fields and Methods---------------------------------------- 4
* Relevant Algorithms-------------------------------------------------- 4
* Description of work--------------------------------------------------- 5
* Conclusion------------------------------------------------------------- 6
* **Class Design**

We created three files to store our classes. All the classes are private. A file named constants was created to store all the constants. All classes are able to be called and used to create object in the main program file: jump game.

1. **Jump Game**

This class contains the main thread of this game. Inside it we find the update method: it will keep checking every frame of the game to see if the condition inside it is fulfilled. Inside the update, there are two conditions; one executed when the game is waiting for the user to press the key Q to quit and the others are Mario and barrels movements. The *marioMovements* method is the main control to move Mario, and *barrelManagment* method controls the barrel movements during the game. Finally, we added a draw method which allow us to have all our sprites on the game screen. The floors, barrels and ladders are drew using for loops and stored in lists.

1. **Mario**

We have made a special class for Mario in a new file because it’s the main character and the work is more complex, we are calling in the init method for the attributes x, y, lives, saltando and times. Then we created three methods *jump*, *notJump* and *move* in order to allow Mario do the basic movements as instructions.

1. **Donkeykong**

This file contains the class donkeyKong with attributes x and y.

1. **Characters**

This file stores 6 classes that are necessary to show on the game screen. They are: princess, ladders. Floors, livestitle, scoreBox and barrels. The class princess, ladders and floors only have attributes x and y; the class livestitle has attributes x, y and a changeable attribute number; the class scorebox has an attribute called point besides attributes x and y; and the class barrels has attributes x, y and direction and a function of movement. Class scoreBox handles x and y coordination and point which the number of points that Mario will accumulate every time he jumps over a barrel and that is going to be updated in *barrelManagement* by adding 100 points each time. Livestitle is a class that handles x and y coordinates and number. Number represents the number of lives and that is going to be updated in *barrelManagement* every time a barrel hits Mario by subtracting by one.

1. **Constants**

An entire class for constants and predefined variables. Some important coordinates like the princess, Mario, donkey kong, ladders, floors, platforms, number of lives, or the board width and height are declared here. This is useful if we want to change a constant of the game and if we want to call them in another class, because we only have to modify it in one place.

* Relevant fields and methods

The most relevant methods are related to the movement of Mario and the barrels. There are 4 different methods in the two classes to move them: *move* from the barrels class and *move2* from mario class, and *jump* and *notjump* from mario class. These functions are vital for the game, as they are used constantly and have to be as optimized as possible.

*move2* and *jump* instantly move Mario by pressing the buttons “Left”, “Right”, “Up”, “Down” and “Space”. we have elaborated loops that made Mario move in the right path by restricting the x and y. Then x and y are replaced by the others given to the method each time we make Mario move.

*notJump* makes Mario fall back to the ground after he jumps by setting a difference in the pixel.frame\_count, so without pressing any button he will be back to the floor.

*move* from the barrels class makes the barrels move automatically. We don’t have to press any buttons. Here we elaborated an algorithm that tells the path that the barrels have to follow during the game and the gaps and ladders where the barrels will fall down to the next floor by using loops and lists calling from the constants file.

* Relevant algorithms

The most relevant algorithms in this project are the two movement functions created in the main file jump game: *marioMovements* and *barrelManagement*.

*MarioMovements* links the pressing of buttons of “Left”, “Right”, “Up” and “Down” with the *move2* method from class mario and the button “Space” with the *jump* method. So that Mario can move in the direction that players want as they press a corresponding button. It also helps to indicate the direction of *jump* and *notJump* methods: when players press the button “Space” while pressing down the button “Right”, Mario will jump to the right and falls to the right; when players press the button “Space” while pressing down the button “Left”, Mario will jump to the left and falls to the left.

*BarrelManagement* first creates up to ten barrels in the screen and indicates them to move along the floors and fall from the gaps and the ladders. It also allows barrels to be removed when they get to the starting point of Mario so that it can generates new barrels. It also indicates that when a barrel hit Mario, he will go back to the starting point and loses one live. And every time Mario jump over a barrel, he will obtain one hundred point.

* Description of work

Here we are going to descript how our game work. Mario starts on the ground floor and he has to reach the top to fight with Donkey Kong and save the princess. Mario can walk to the right or to the left as players press corresponding buttons. He can jump only while walking in a direction, so players have to hold on to the button “Right” or “Left” when press button “Space”. Mario can climb up and down the ladders to reach the higher floor.

In his journey, Donkey Kong throws barrels on the top floor and these barrels will roll down along the floors and fall from the gaps. They also have a 25 percentage of probability of falling down from the ladders. Mario should avoid being hit by these barrels, otherwise he will lose one of his 3 lives and go back to his starting point. When he loses all his lives, the game ends. If he avoids hitting by the barrels by jumping over them, he gets 100 points. These points will be accumulated even if he loses lives, till the game ends.

* Conclusion

It’s undeniable that thanks to this project we have learnt more about python and object-oriented programming, we had to go deeper and explore all the helpful tools that offer us and discover how object-oriented programming facilitate a lot and organizes a complex work. It was easy to get informed about python since it’s a well-documented programming language whenever we had mistakes and some syntax errors, or when we had an algorithm idea and we had to translate it into the programming language.

However, our obstacle was pyxel which was the hardest part for us, since we weren't familiar with graphical programming languages and there was no help available nor a small guide, the discovery of pyxel was an adventure for us. At the end we could achieve it thanks to the help of the professors and the team effort. We discovered that it’s a good tool, not a complex one, so it is accessible to everyone and we feel that we could do more if we had deeper knowledge and guidance into this language. No doubt this made us being more mature and ready for the programming world.

This game can be improved in the future when we get to know more knowledge about python and pyxel.