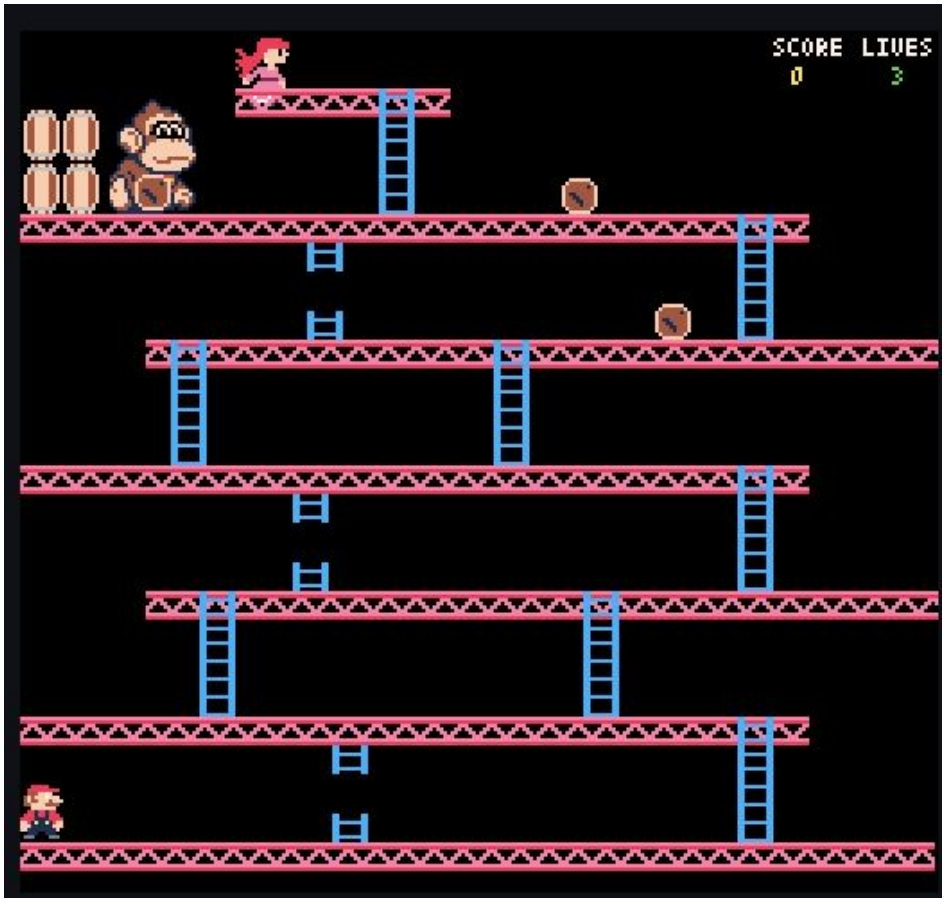
Programming Project report

Donkey Kong

Javier Asensio Gómez & Maria de la Corte Arnau
Group 89
Computer Science



Abstract

This project tries to emulate the game Donkey Kong. After following the steps marked by the project sprints, we have achieved to successfully program a fully functional version of the game. The code has been designed taking into account the techniques learned in class such as object oriented techniques or encapsulation

Maria de la Corte Arnao 100405979
Javier Asensio Gómez 100405835

INDEX

Maria de la Corte Arnao 100405979
Javier Asensio Gómez 100405835

1. **FINAL PROJECT**

The final project for Programming consists on programming the game Donkey Kong.
For this game we had to follow the established rules of the game, which were provided to us in a PDF.
We have created 8 classes to make the program more organised and to clarify it.
The most relevant sprites for these project were:

1. Generating the objects and its graphical interface.

2. Generating Mario's basic movements: Make Mario move right and left, taking into account that platforms are non horizontal. Allow him to go up and down using ladders. Also allow him to go down at the end of platforms.

3. Generating the barrels, which must go down by the platforms and have a 25% chance to descend by a ladder. Once the reach the start point they will disappear.

4. Basic game; When Mario jumps over a barrel it gets 100 points added to the score. If Mario is touched by a barrel it loses a life, dies and starts again. If there are no remaining lives the game ends.

5. Add new functionalities to your program: music, sound effects, hammers, highest scores table, entry animation, etc.

Maria de la Corte Arnao 100405979
Javier Asensio Gómez 100405835

2. DIFFERENT CLASSES.

The first challenge we had to get through in order to start this project was to decide all the different classes our project should have in order to make the game run.
We have created 8 classes in order to have a cleaner code and to have the program well organised.

- Board Class:

  The Board Class is the skeleton of the project. It is the class designated to store all the data we have from other classes and make the game work. This class is divided in 3 different methods:

- __init__: Which initialises all of the object that are going to be in the board
- update : It acts as a loop until the user presses Q (exiting the game).
- draw : This method draws all of the items in the board game by adding the sprites on the right position

- Constants Class:
  In this class we inserted all the initial values, constants and positions of all the elements of the game. Also the different images that were going to be drawn in the board. For instance, the initial position for Mario is at the 1st platform on the left.

- Plat Class:
  In this class we wrote how did the initial values of the platforms should behave (be bigger than 0 in the case of the positions of x and of y), and we could use an image. We knew we were going to create a tuple list at the “board Class” were we were going to insert all the different positions of the platforms

- Ladders Class:

  In this class we wrote how did the initial values of the ladders should behave (be bigger than 0 in the case of the positions of x and of y), and we could use an image. We knew we were going to create a tuple list at the “board Class” were we were going to insert all the different positions of the ladders

- Mario Class:

  This class is the one that controls Mario and its behaviour. Here we can find Mario's movement, the methods to check whether we should use the gravity or not, the methods that check it's position in relation to the barrels...

- Barrels Class:

  This class controls the barrels behaviour, the direction it is going, if it is on a ladder or the direction it should take after falling to the next platform.

- Princess Class:

  In this class we check if the positions of the princess are the correct ones and we add a method to make her "scream" the word help randomly.

2.1 MOST RELEVANT FIELDS & METHODS.

In The Board Class:

- **Update** is one of the most important methods, because it is updating the game every time. All the actions that a sprite might have to do in a frame are inserted in this method.

- **__init__**: it initialises the values of Mario: the barrels, the princess, Mario, the platforms, the ladders and Donkey Kong.

- **Draw**: Draws the objects on the board. Sometimes we have to use loops in order to draw all the elements inside a tuple such as the ladders or the platforms.

- **Throw**: Creates a list, and adds to the list the barrel.

In Mario Class:

- **move**: Depending on the introduced movement ("right", "left", "up", "down" or "space") it will move in that direction only if its possible.

- **lostLife**: If subtracts a life from Mario every time the method is invoked.

- **scoring**: Adds 100 to the total score every time the method is invoked.

- **climb**: Tells if the user is pressing up or down. If Mario is at the position of a ladder, climbing will return True, otherwise climbing will return False.

- **atPlat**: Returns true if the position of Mario is standing right on top of a platform.

- **atPrincess**: Returns True if the position of Mario is at the platform where the princess stands.

- j**ump**: Method that allows Mario to jump.

- g**ravity**: Decreases in 1 the position y of Mario.

In The Barrels Class:

- **atLadder**: Does a random number between 1 & 4, then it Checks if the barrel is at the position of a ladder. If that is the case and the random number is 1, inLadder will return True.

- **move**: It controls the movement of the barrels. If it gets “right” as input, the barrels X will increase in 1. If it gets “left” as input, the barrels X will decrease in 1.

- **atPlat**: Checks if the barrel is right on top of a platform and returns True if so.

- **directionAfterFall**: Assigns the desired direction of the barrel depending on the platform they are at.

- **gravity**: It increases in 1 the position Y of the barrel, depending on the position of the barrel, it will fall to the right or to the left.

- **gravityForLadder**: Subtracts 1 to the position of the barrel when it gets to the ladder position.

2.2 MOST RELEVANT ALGORITHMS USED.

- pyxel.text(X, Y, ("TEXT"),colour), This algorithm inserts the text inserted at the desired position.

- pyxel.blt(X, Y ,*IMAGE, Colour): Inserts the object into the board(if we put colkey=0.)The back of the object will be transparent.

- pyxel.load("assets/gameboard.pyxres"): Loading the pyxel images we had created.

3. DESCRIPTION OF THE PROJECT.

First we will talk about the order in which we decided to do this game, what and why did we introduce in the different classes we created. And Finally we will annex the extra work we have done and the parts which we weren't able to perform.

3.1 WORK PERFORMED

We managed to achieve most of the sprites, trying to make them as short and plain a we possibly could. The game starts right away, whenever Mario is hit by a barrel it goes back to its initial position and losses a life. Whenever Mario jumps a barrel he wins 100 points that are added to the total score. If Mario reaches the princess platform the game ends, and there appears a text saying "YOU WON". If Mario loses all of its lives, it appears a text saying "GAME OVER". Either way, the user is able to restart the game if he/she wants by pressing the key "enter" (all values will be restored) and it is also capable to quit the game by pressing the key "q".

1st Sprite: We had to create an object for each of the game elements, with the appropriate fields inside. We were able to successfully complete it.

2nd Sprite: Make Mario move right and left, taking into account that platforms are non horizontal. Allow him to go up and down using ladders. Also allow him to go down at the end of platforms. We were able to successfully complete it.

3rd Sprite: Make Donkey Kong to throw barrels (use a static Donkey Kong for this sprint). Barrels must go down by the platforms and will have a 25% chance to descend by a ladder. Once the reach the start point they will disappear. We were able to successfully complete it.

4th Sprite :Implement the basic game: Mario can jump over barrels, getting 100 points when doing it. If he is touched by a barrel he loses a life, dies and starts again. If no lives, the game ends. Animate Mario, barrels and Donkey Kong. We were almost able to fully complete this sprite. We didn't implement the animation of the barrels or the animation of Donkey Kong.

3.2 WORK NOT PERFORMED/EXTRA.

The extra work performed, was that whenever the game was done, you had the possibility to restart again by pressing "enter" or quit the game by pressing Q.
We also added limits to Mario, so that it could not get out of the board.

We were not able to do the change of images, for example for the animation of the barrels.

5. CONCLUSIONS

All in all, our work performed has been done in several weeks and although we have had some difficulties, we have been able to overcome most of them.

We both have been working in this game, at first we worked in class trying to perform the exercises that the teacher told us to do. For instance, the first classes we had to attend for the final project (the pyxel Examples) and the basic Mario movements. After this classes, we decided to work together and separately in the different sprints that we had to complete for our project to be completed.

We believe this project helped us improve our programming skills. It has also been helpful to completely understand how is really programming using the object oriented techniques and what we are going to have to do in a near future.

Finally, this has been a project that has helped us to improve and we have considered it a challenge for ourselves. Although all of the problems we have had, we've been able to successfully fix them and to learn from our own mistakes.

Maria de la Corte Arnao 100405979
Javier Asensio Gómez 100405835