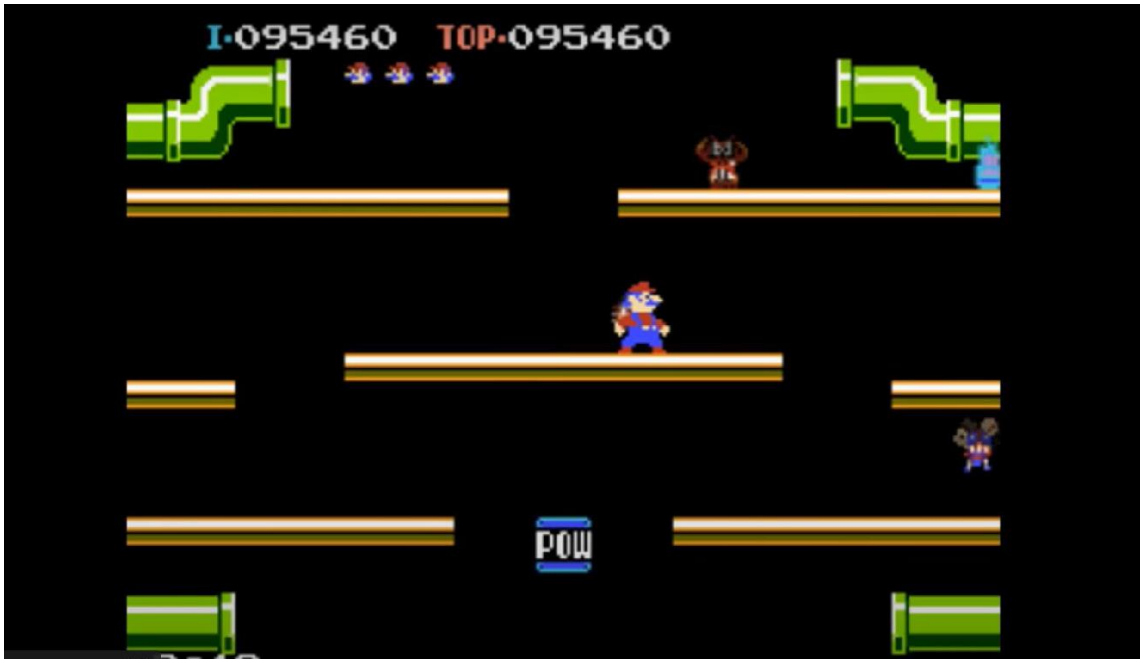
Final Project: Mario Bros
Programming
2023 - 2024

For the final project, a simplified version of the 1983 Mario Bros game will be implemented. The game consists of a series of stages, and the goal is for Mario to defeat all his enemies in each one of the stages.

1. Introduction to the game: Basic Rules



Each stage has a different screen. The screen of each stage is made up of several platforms. There are pipes in each corner through which enemies can go up and down. There is also a block labeled "POW" in the center, we will explain its function later.

Regarding movement, Mario can move left and right and jump. Both Mario and the enemies, if they go off the screen on one side, will return on the other.

Mario loses a life if he is hit by any of the moving enemies. Once he has lost all of his lives, Mario dies and the game ends.

There are different types of enemies. To defeat turtles, crabs, and flies, you must flip them on their backs and then kick them. To flip enemies, you have to hit the platform right underneath them. You can also hit the "POW" block from below, and in this case, the enemies that are touching the ground or a platform will flip onto their backs, so you can then kick them. The "POW" block will disappear after being used three times.

Some considerations to bear in mind:

- If we hit an already flipped enemy, they will get up.

- You cannot leave an enemy flipped on their back for too long before kicking them, as they will get up and get angry, increasing their speed and changing color.

Icicle spikes and fireballs disappear by hitting the platform right below them; there is no need to kick them.

Next, let's describe the different types of enemies that come out of the pipes and with which we will have to fight:

Turtles (Shellcreeper): A single hit is enough to flip them over.

Crabs (Sidestepper): Two hits are necessary to flip them over.

Flies (Fighter Fly): They move by jumping, so to hit them they must be touching the platform.

Icicle Spikes (Slipice): They turn the central platform and the two lower ones into slippery ice if they reach the center of them. They self-destruct in the freezing process. By hitting the platform right underneath them, the Slipices disappear. To prevent the platforms from freezing over, Mario must destroy them quickly. When starting a new phase, all platforms are defrosted.

Fireballs, which can be red or green. The green ones have wavy horizontal movement and disappear after a short time. The red ones are a bit slower, bounce, and do not disappear until they are destroyed. By hitting them when they touch the platform, they disappear.

The score increases according to the number of coins Mario collects and the number of enemies he has defeated.

There are several websites where you can play this game. For example:

https://www.classicgames.me/mario-bros-classic.html

You can find a complete video of the original game at the following link:

https://www.youtube.com/watch?v=nKqlPvslQ4A

2. Work to be Developed

The game must be programmed, including its graphical interface. The graphic design will be a collaborative task in which the entire class will create the graphics as pyxres or png files using the pyxel graphics editor and creating a common repository in Aula Global.

Each group of students may change these images for their own. It is recommended not to invest too much time in this since what will be valued is the correct implementation of the game using object-oriented programming paradigm, and not its visual appearance.

The first 3 groups to upload their pyxres or png graphics with the game's images to Aula Global will receive an extra 0.1 points, provided that the final score for the assignment does not exceed the maximum of 2.5 points.

To implement the game correctly it is essential to have done a good design of the classes to use. Before writing any code, be sure that your class design is correct by discussing it with your lab teacher. Do not start any code implementation tasks until the design has been approved by your teacher.

Once the class hierarchy has been defined, implementation will be an incremental task, completing the game's functionalities step by step. It is recommended to follow the following steps, although each student may choose their own implementation strategy, with no weekly control or tracking of it.

2.1 Sprint 1: Objects and Graphical Interface

Create a class for each game element with the appropriate attributes. All game logic must be implemented within each class. Including more code than necessary in the main program will be penalized.

Create the level: Position Mario, platforms, and pipes. At least 4 stages of the game must be implemented. Although for the initial sprint it is enough to implement a single stage, when designing the classes it is a good idea to consider that several stages need to be implemented. Add a scoreboard with the score obtained during the level, add the life counter also. Create a list with the enemies, which will not initially appear, but will appear progressively. At least 30 enemies of all types must appear throughout the game.

2.2 Sprint 2: Basic Movements of Mario

Make Mario be able to move to the left or right. If he reaches the edge of the screen, he will appear on the other side. Also make Mario able to jump and get onto platforms.

In this sprint, you can use a static Mario; it is not necessary to animate the movements of hands and feet when walking or jumping.

2.3 Sprint 3: Enemies

Make the enemies appear and implement their basic movements. You can make them appear randomly or following the original pattern of the game. In this sprint, we will not implement the Slipices or icicle spikes nor the fireballs, which will appear as an extra.

In this sprint, it is not necessary for Mario to interact with the enemies, nor to animate their movements.

2.4 Sprint 4: Basic game

Implement the basic game:

- When a platform is hit under an enemy, the enemy reacts as specified for each type, flipping over or disappearing. If a flipped-over enemy is kicked, it is destroyed.
- If too much time passes before kicking a flipped-over enemy, it gets up, increases its speed and changes color.
- If Mario comes into contact with a moving enemy, he takes damage and will lose a life. If Mario loses all lives, the game ends.
- Hitting the POW block causes enemies touching the ground or a platform to flip over and lie on their backs, so you can then kick them.
- Implement animations for Mario and the enemies in this sprint.
- Coin collecting will increase the coin counter and the score. Destroying an enemy will also increase the score.
- When all enemies have been destroyed, the game ends.
- Implement the different screens of the game.

2.5 Sprint 5: Extra elements (optional)

Implement the full functionality of the Slipices or icicle spikes that allow freezing platforms. Implement also the fireballs. Update Mario's movement accordingly.

Create a bonus screen where the player can obtain extra points by collecting coins in a specific period of time.

Add new features to the game: music and sound effects, a high score table, etc. Verify with the lab instructor before doing so in order to receive appropriate guidance.

3. Submission rules

The following rules are **mandatory**. Failure to comply may result in the assignment not being evaluated. The final project must be carried out in groups of 2 students. Each group must upload to Aula Global a zip file containing the report and the source code before the deadline. The zip file name must follow the following format: “initials1-initials2.zip” (for example, if the students are Lucía Pérez Gómez and Juan García Jiménez, the file name should be lpg-jgj.zip).

The report, in **PDF format**, shall contain a maximum of 10 pages, which should include at least:

- Cover page, table of contents, and a brief summary of the document.
- Description of the designed classes, including the main attributes and methods.
- General description of the main algorithms used.
- Description of the work carried out, functionalities included, parts not implemented, and extra functionalities carried out.
- Conclusions:

- Final summary of the work done.
- Main problems encountered.
- Personal comments, evaluation, and aspects to improve on the final project.

4. Evaluation

The final project will be valued between 0 and 2.5 points. Each sprint from 1 to 4 will be valued with 0.5 points. The report will be valued with the remaining 0.5 points. The optional part will be valued with other 0.5 extra points, with 0.1 points for each relevant functionality added, although the maximum score of 2.5 points will never be exceeded.

The following aspects will be taken into account when evaluating the final project:

- Appropriate class design.
- Correct functioning of the game and proper adherence to the rules set out in this document.
- Quality of the implemented code.
- Exclusive use of the contents and principles taught during the course.
- Inclusion of comments in the code.
- General description of the implemented code in the report, according to the rules specified in section 3.
- Compliance with the submission rules (Section 3).

The following penalties will be applied, where applicable:

- 0.25 points if not enough comments are included. All methods and classes must be properly commented on and noted. Comments should also be included within the code of the various methods.
- 0.5 points if methods are not used, if repetitive code is included instead of loops, if good programming practices taught during the course are not followed, for a poor class design, if lists are not used to represent enemies, if breaks are used, etc.
- 0.25 points if unnecessary attributes are included, public attributes that should be private, attributes created outside the init method, or not using properties where they should be.
- 0.25 points if there is code outside of the classes (with the exception of a module to include the constants and the main program) or for having methods that are too long (as a general rule a method should take up a single screen without needing to scroll).

Plagiarism will be strictly penalized: both those who copy and those who are copied from will be automatically excluded from continuous assessment, in addition to possible administrative sanctions that the University may enforce. Two projects may be considered as copying even if their degree of similarity is restricted to the code of a single method. Automatic plagiarism detection tools will be used.

The final project will be evaluated through an oral presentation in which both students must defend and present their work to the teacher.