

# Final Project *Mario Bros*

## 1. Introduction

This document describes the final project for the Programming course, focused on the development of a Mario Bros game. Students will apply the programming concepts and techniques learned throughout the course to create a fully functional and engaging game.

## 2. Project Objectives

The main objectives of this project are:

- Design and implement a classic Mario Bros game using Python with pyxel.
- Demonstrate mastery of fundamental programming concepts such as loops, conditional statements, data structures, and **object-oriented programming**.
- Gain experience in game development concepts such as character movement, or collision detection.
- Develop problem-solving and critical thinking skills in a creative and engaging context.

## 3. Game Description



A digital version of the famous game 'Game & Watch: Mario Bros. (1983)' must be developed. In this version, Mario and Luigi work in a **bottling factory**, and their mission is to ensure that **packages arrive correctly at the delivery truck without falling to the ground**.

You can play this game online at <https://itizso.itch.io/nintendo-mario-bros>

## Setting and Characters

The game must be played on a single screen where the following elements and characters will be available:

- **Mario:** Located on the right side of the screen, he will be responsible for picking up empty boxes (Conveyor0) and placing them on the first conveyor belt (Conveyor1). Similarly, he is responsible for movements between EVEN-ODD conveyor belts:
  - Conveyor0 → Conveyor1
  - Conveyor2 → Conveyor3
  - ...
  - ConveyorX-1 → ConveyorX
- **Luigi:** Located on the left side of the screen, he will be responsible for placing the finished packages in the truck. In addition, like Mario, he must be responsible for movements between ODD-EVEN conveyor belts:
  - Conveyor1 → Conveyor2
  - Conveyor3 → Conveyor4
  - ...
  - ConveyorX → Truck
- **The Boss:** He is Mario and Luigi's boss. Every time a package falls, he will appear to punish Mario or Luigi as appropriate. In addition, he will add a 'failure' to the brothers' work. He will also appear after each break, to make brothers come back to work.
- **Conveyor0:** This is the first conveyor belt. It carries empty boxes to Mario so that he can deposit them on Conveyor1.
- **Even Conveyor Belts:** These will be the conveyor belts where Mario receives packages and moves them to an Odd Conveyor Belt (of the immediately higher level).
- **Odd Conveyor Belts:** These will be the conveyor belts where Luigi receives packages and moves them to an Even Conveyor Belt (of the immediately higher level).
- **Truck:** It will be waiting for Luigi to deliver 8 packages. Once it has 8 packages, it will go out for delivery. While it is out for delivery, Mario and Luigi will rest, and all conveyor belts will be stopped and the packages will remain in the position where they were. If a package is in the last position of the conveyor belt when the truck leaves, it will disappear when the game resumes.
- **Floors and Stairs:** Mario and Luigi will always be located at the beginning of the game on Floor0. Both characters can go up and down floors using painted stairs.

## Game Mechanics

Mario and Luigi can be controlled to go up or down the stairs and thus change floors. To control Mario, use: ARROW UP, ARROW DOWN. To control Luigi, use W and S. There will be N floors depending on the number of conveyor belts, and if the characters are on that floor, they can automatically (without pressing anything) perform the following movements:

- **Floor0**
  - Mario will move the package from Conveyor0 to Conveyor1
- **Intermediate Floors**
  - Mario will move the package from Even Conveyor to Even Conveyor+1
  - Luigi will move packages from Odd Conveyor to Odd Conveyor+1
- **Floor N**

- Luigi will move packages from ConveyorX (Odd) to Truck

The game will end after 3 errors (lost packages). A package will be lost when Luigi or Mario are not on the correct floor and that package has no one to receive it.

Every time Mario or Luigi make a correct movement (passing a package from one conveyor belt to another), they will be given 1 point.

Every time a truck is completed and goes out for delivery, they will be given 10 points.

There must always be at least one package in play. The minimum number of packages in play will vary (see Difficulty Levels) according to the score obtained.

The conveyor belts will have a determined speed (See Difficulty Levels).

## Game Objective

The objective is to get the highest number of points before having 3 failures.

## Difficulty Levels

The game will have several difficulty levels. Each level will have rules that will make it more difficult or crazy.

	Easy	Medium	Extreme	Crazy
Mario Controls	Arrow Up: Up Arrow Down: Down	Arrow Up: Up Arrow Down: Down	Arrow Up: Up Arrow Down: Down	Arrow Up: Down Arrow Down: Up
Luigi Controls	W: Up S: Down	W: Up S: Down	W: Up S: Down	W: Down S: Up
Belts	5	8	10	5
Conveyor Speed	Always 1x	Conveyor0: 1x Even Conveyors: 1x Odd Conveyors 1.5x:	Conveyor0: 1x Even Conv.: 1.5x Odd Conveyors: 2x	Conveyor0: 1x Random (1-2) for each conveyor
Minimum Number of Packages	Starts at 1. Increases by 1 every 50 points	Starts at 1. Increases by 1 every 30 points	Starts at 1. Increases by 1 every 30 points	Starts at 1. Increases by 1 every 20 points
Truck	Eliminates one failure every 3 deliveries	Eliminates one failure every 5 deliveries	Eliminates one failure every 5 deliveries	Does NOT eliminate failures

## 4. Project Requirements

The project must meet the following requirements:

- **Functionality:** The game must be fully functional, allowing the player to control Mario and Luigi and move packages from Conveyor0 to the truck.
- **Code quality:** The code must be well written, organized, and commented, and must use object-oriented programming.
- **User interface:** The game must have a clear and intuitive user interface, showing the bottling factory as well as the described characters.
- **Game design:** The game design must be attractive and challenging, incorporating elements of the classic Mario Bros game.
- 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 can change these images for their own. It is recommended not to spend too much time on this, as what will be marked is the correct implementation of the game using the object-oriented programming paradigm, and not its visual appearance. The first 3 groups that upload their pyxres or png graphics with the game images to Aula Global will receive an extra 0.1 points, provided that the final score of the work does not exceed the maximum of 2.5 points.
- **Music and sound effects:** sound effects can be included (when moving a box, when going up or down a floor, when the truck leaves, when punishing Mario / Luigi for a failure, when a box falls, etc.)

## 5. Project Deliverables

Students must submit the following final products:

- Source code: the complete and well-commented source code of the game.
- Executable game: a playable version of the Mario Bros game.
- Documentation: a document that includes:
  - A description of the game design and implementation.
  - A user manual explaining how to play the game.
  - A reflection on the challenges faced and lessons learned during the project.

It is recommended to follow the steps below, although each group of students can choose their own implementation strategy, without weekly control or monitoring.

### Sprint 1: Objects and graphical interface

- Create a class for each main game element: **Characters, Conveyor, Truck, Package**, etc.
- All the **behavior logic** of each entity must be contained in its corresponding class. Avoid including game logic in the main program (penalty if it occurs).

- Design the **graphical interface of the scenario**: a single screen with the conveyor belts, floors, stairs, and truck.
- Implement a **score counter and error counter (failures)** visible during the game.
- Define the **basic data structures** that will manage the state of the conveyor belts, packages in transit, and the difficulty level.

✓ For this first sprint, it is enough to visually represent the elements (even if they do not move). It is advisable to design the classes thinking that the game will have different difficulty levels.

### Sprint 2: Mario and Luigi Movement

- Implement the **vertical movement control** of the characters:
  - Mario: **Arrow Up / Arrow Down** keys.
  - Luigi: **W / S** keys.
- Each character must be able to **go up and down floors** using the stairs, respecting the upper and lower limits.
- Graphically display the **current position of each character** (Floor0, Floor1, etc.) on the screen.

✓ The objective of this sprint is to have fully controllable characters within the environment.

### Sprint 3: Packages movement

- Implement the **package flow** on the conveyor belts according to the established rules:
  - Conveyor0 generates empty boxes for Mario.
  - Even Conveyors → controlled by Mario.
  - Odd Conveyors → controlled by Luigi.
- Graphical representation of the packages must change according to the belt they are on.
- When a package reaches the end of a conveyor belt:
  - If the corresponding character is on the correct floor, **they automatically pick it up** and pass it to the next conveyor belt.
  - If not, the package **falls** and a **failure** is recorded.
- Implement the **automatic movement of packages** based on the **speed of the current level**.

✓ At the end of this sprint, the game should have a basic functional simulation (without levels or scoring).

### Sprint 4: Scoring system, failures, and end of game

- Implement **scoring**:
  - +1 point for each package correctly delivered to the next conveyor belt.
  - +10 points for each completed truck (8 packages delivered).
- Manage the **failure counter** (3 failures = game over).
- Implement the **truck logic** and character rest:
  - When it receives 8 packages, it goes out for delivery.
  - During delivery, the conveyor belts stop temporarily. If a package is at the last position of the conveyor, it is deleted.
  - Upon return, activity resumes.
- Display a **game over message or animation** when 3 failures are exceeded.
- Complete **Boss's visual effects**: he will appear when a package falls, after rest...

✅ From this sprint on, the game should be playable from start to finish.

### Sprint 5: Difficulty levels and final adjustments (optional)

- Incorporate the **difficulty levels (Easy, Medium, Extreme, Crazy)** according to the rule table:
  - Conveyor belt speed.
  - Minimum number of packages in play.
  - Failure elimination rules for truck delivery.
- Adjust the interface to display the **current level**.
- Allow changing the level from the menu or before starting the game.
- Add **optional sound effects**: conveyor belt movement, error sounds, etc.
- Implement **high score tables** or **local competitive mode**.

✅ The final objective is to have a complete, fluid, and fun game, maintaining the retro aesthetic of the original.

## 6. Resources

Students will have access to the following resources:

- Course materials and lessons.
- Online resources and tutorials.
- Programming language documentation.
- Instructor support and guidance.

This project provides students with a valuable opportunity to apply their programming skills in a creative and challenging context. By developing a Mario Bros game, students will gain a deeper understanding of game development concepts using object-oriented programming and improve their problem-solving skills.

## 7. Evaluation

The final project will be graded between 0 and 2.5 points. Each sprint from 1 to 4 will be graded with 0.5 points. The report will represent the remaining 0.5 points. The optional part will be graded with an additional 0.5 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 game execution and adequate adherence to the rules established in this document.
- Quality of the implemented code.
- Exclusive use of the content 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 the following section.

The following penalties will apply, if applicable:

- 0.25 points if insufficient comments are included. All methods and classes must be correctly commented and annotated. Comments must also be included within the code of the different 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 poor class design, if lists are not used to represent packages, etc.
- 0.25 points if unnecessary attributes are included, public attributes that should be private, attributes created outside the init method, or if properties are not used where they should be.
- 0.25 points if there is code outside the classes (with the exception of a module to include constants and the main program) or for having methods that are too long (as a general rule, a method should occupy a single screen without the need to scroll).

**Plagiarism will be strictly penalized:** both those who copy and those who are copied will be automatically excluded from continuous evaluation, in addition to any administrative measures that the University may apply. Two projects can be considered copied even if their degree of similarity is restricted to the code of a single method. Automatic plagiarism detection tools will be used.

## 8. Submission Rules

The following rules are **mandatory**. Failure to comply with them may result in the task not being evaluated.

The final project must be carried out in groups of 2 students. Each group must upload a zip file containing the report and the source code to Aula Global before the deadline. The name of the zip file must follow the following format: "initials1-initials2.zip" (for example, if the students are Lucía Pérez Gómez and Juan García Jiménez, the file name must be lpg-jgj.zip).

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

- Cover page, index, 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, unimplemented parts, and additional functionalities performed.
- Game user manual with a description of how to play.
- Conclusions:
  - Final summary of the work carried out.
  - Main problems encountered.
  - Personal comments, evaluation, and aspects to improve the final project.