CSC10003 – Object Oriented Programming

PROGRESS REPORT

I. Information

Group ID: 11

Group name: Super Mario Bros OOP

Members:

ID	Student ID	Full name	CURRENT Tasks (%)
1	23127438	Đặng Trường Nguyên	**25%**
2	23127144	Đinh Đại Vũ	**25%**
3	23127489	Nguyễn Ngọc Minh Thư	**25%**
4	21126089	Nguyễn Thể Phụng	**25%**

II. Percentage of completion

40% (Complete basic level 1 and main menu)

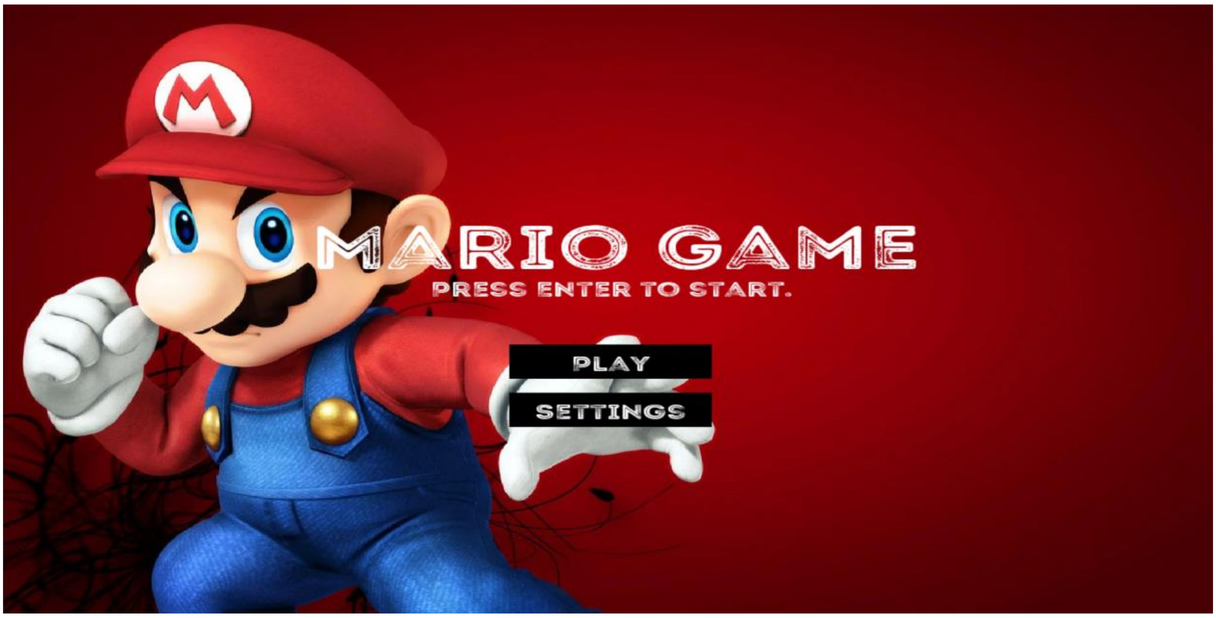
III. Github Link

- https://github.com/trngnneeee/MarioGame

IV. Implemented features

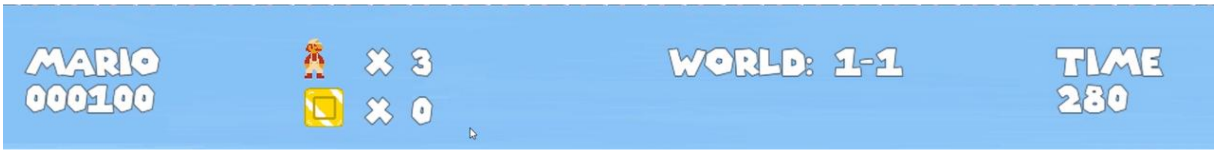
1. Main menu

- **Description:**
  - The main menu includes buttons: Play and Settings.
  - Currently, only the Play button is functional to start the game.
- **Screenshots**



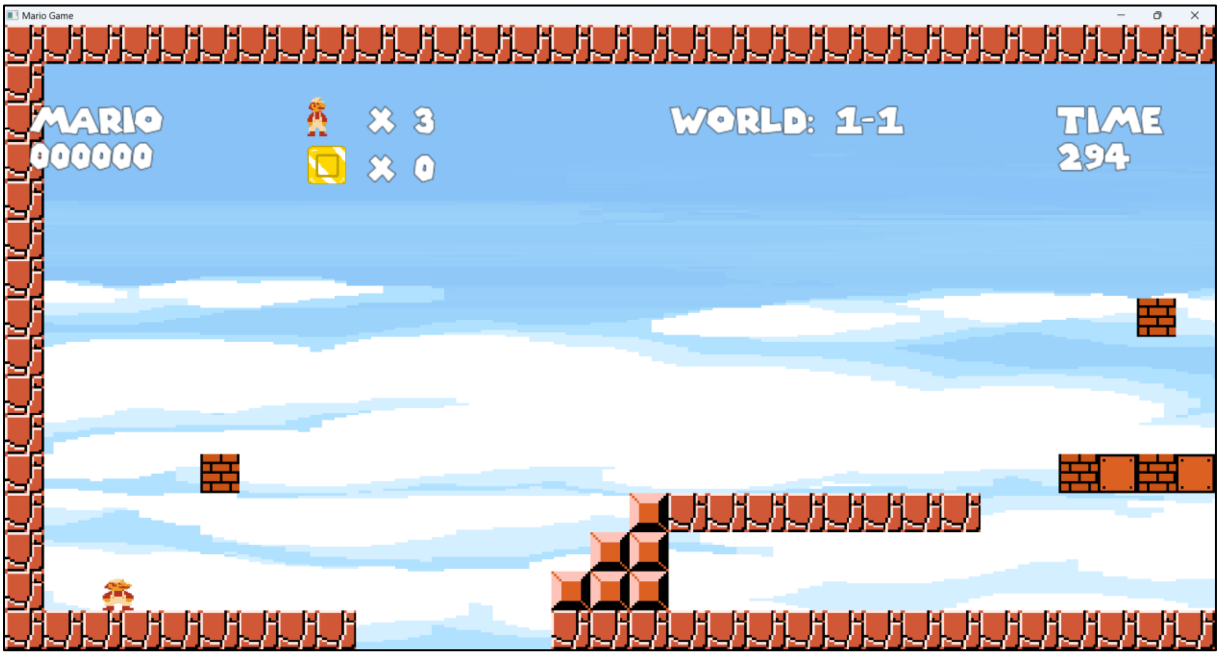
2. UI for counting points, coin, Mario’s life, and game time

- **Description:**
  - **Points** that Mario get is updated and display into window and only reset when Mario if out of lifes.
  - Each game, Mario have 3 **lifes**, when Mario is dead, the game is reset at the beginning of the game. If out of life, the game will be over and switch into Menu
  - **Coin counter** have been added, but Coin haven’t been added to game
  - **Game time** is set at default 300 seconds, if the time is running out, decrease the Mario lifes.
- **Screenshots:**



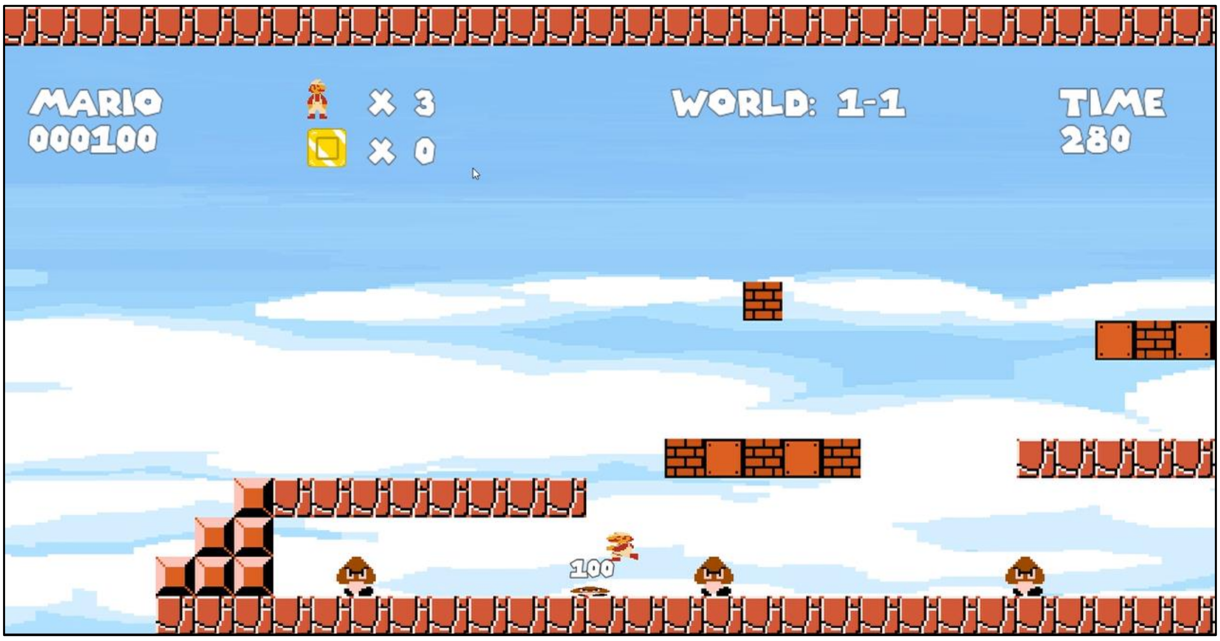
Level 1 - Basic Movement System

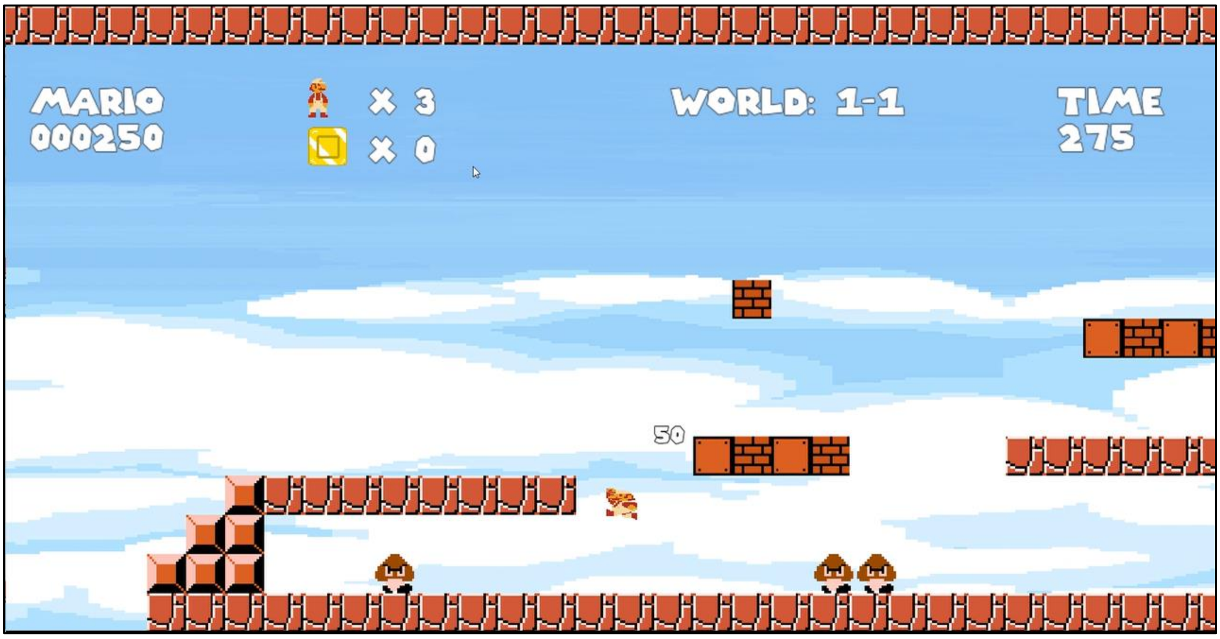
- **Description:**
  - Mario can **walk** and **jump** with basic animations.
  - Added **Animation** for walk and jump, splitted when Mario change movement direction.
  - Added sound effects for the **jump** action.
- **Screenshots:**



3. Level 1 - Interaction System

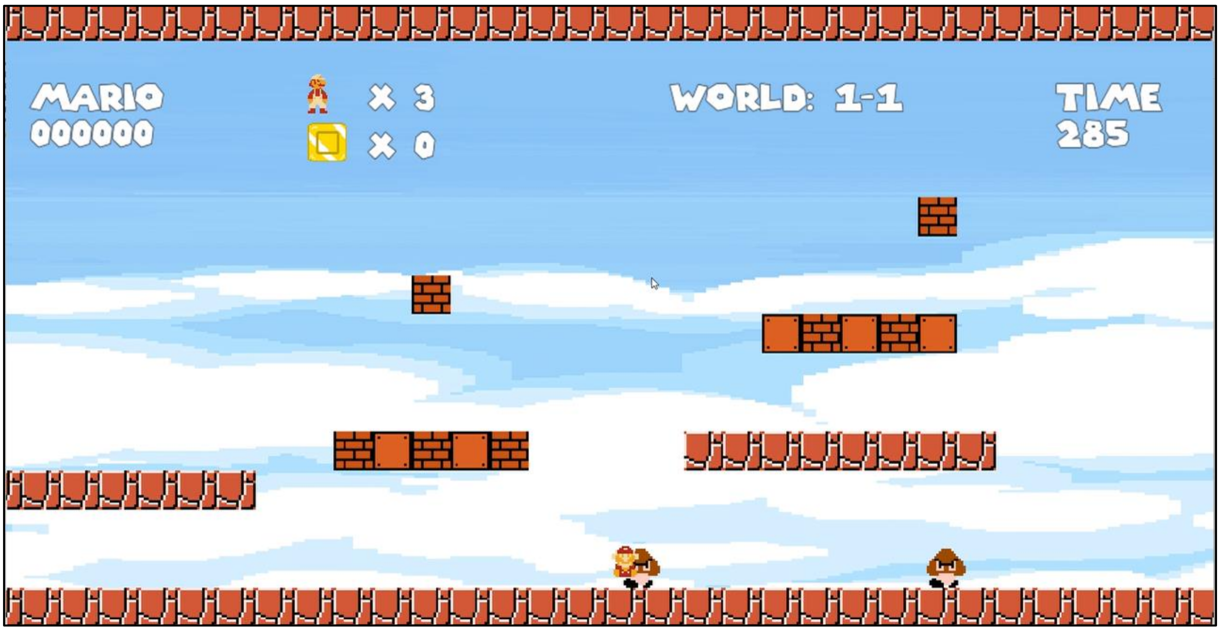
- **Description:**
  - Mario can break the brick to gain **50 points** per hit.
  - Blocks are designed and placed based on a predefined **map layout**, creating the game environment.
  - Mario can jump on **Goombas** to defeat them and gain **100 points**.
- **Screenshots**

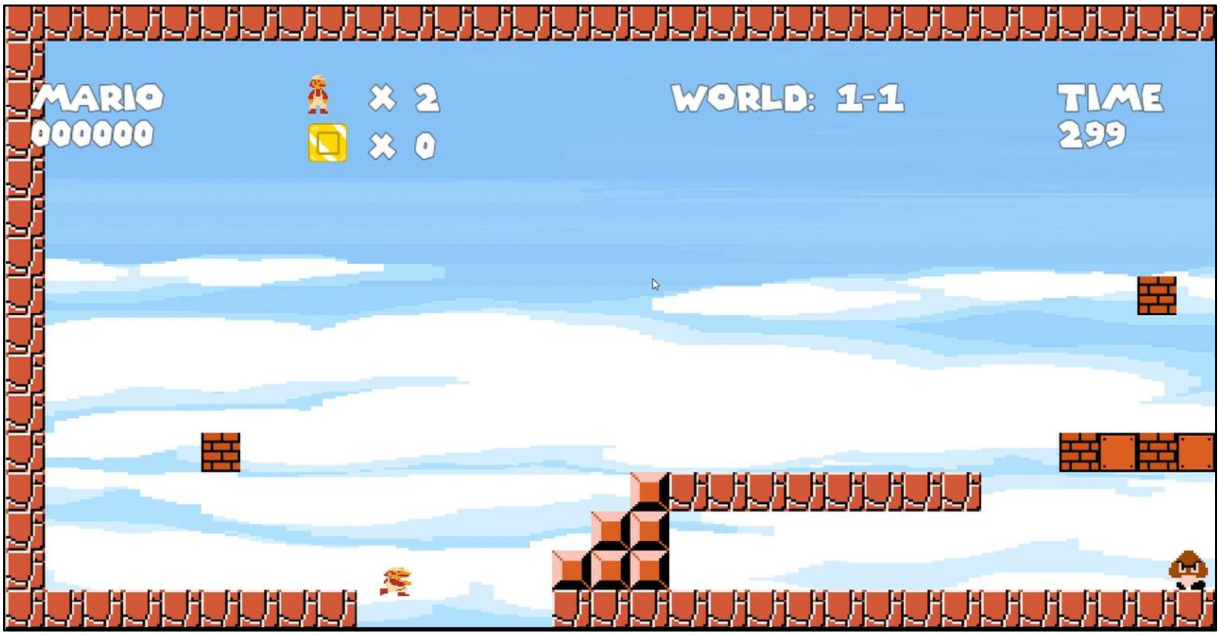




4. Level 1 - Death System

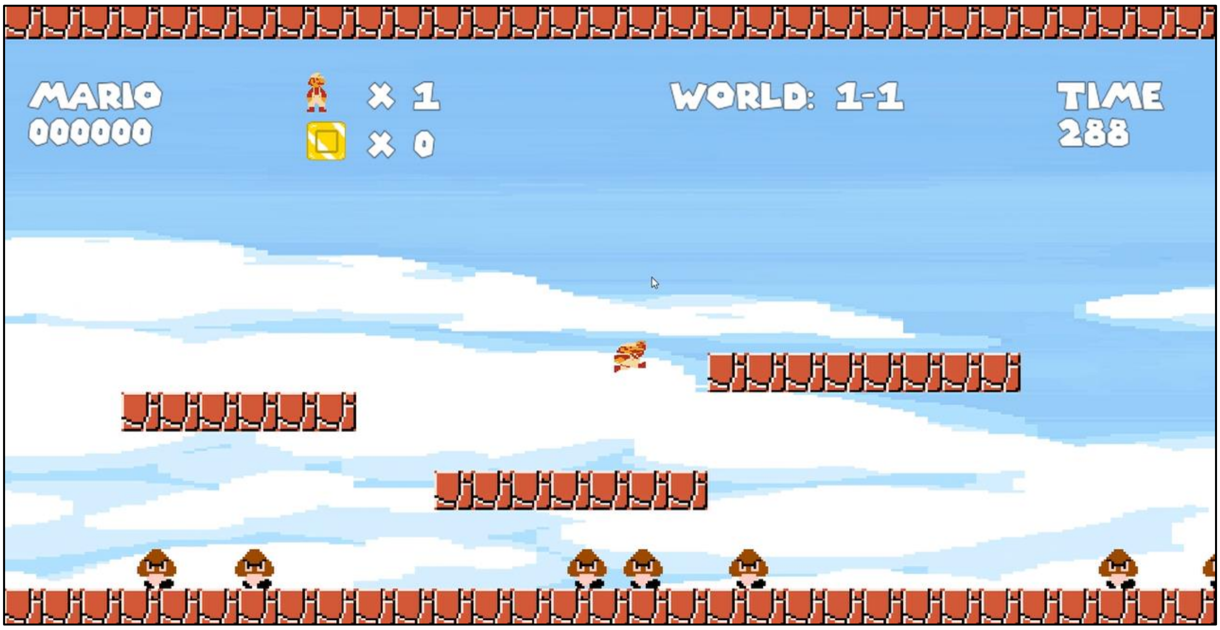
- **Description:**
  - Mario dies when:
    - Touching **Goombas**.
    - Falling into a pit.
  - Upon death, the game is reset to the beginning
  - If Mario is out of lifes, the game is over
  - Added sound effects if the game is over and turn back to the Menu
- **Screenshots**





5. **Level 1 - Enemy System**

- **Description:**
  - **Goombas** move back and forth autonomously within the level.
  - Added collision between each **Goombas**, the **Goombas** turn backward if touch each other
- **Screenshots**



V. Specific techniques

OOP Implementation

- **Encapsulation**:
  - Private member variables are used with getter and setter methods for controlled access.
- **Inheritance:**
  - Class Goombas is inherited from class Enemy

Design Patterns Applied

- **State Pattern**:
  - Also used for managing game states: MainMenu, Playing, and GameOver.
- **Prototype Pattern**:
  - Used for creating repeated game objects like **Goombas**, **blocks**, and **coins** by cloning predefined prototypes.

Design Patterns Applied in the Future

1. **Singleton Pattern**:
   - Will be implemented in a **GameManager** class to manage game state globally, ensuring only one instance is active.
2. **Factory Pattern**:
   - Will be used to handle the creation of different types of enemies (e.g., **Goombas**, **Koopas**) through an **EnemyFactory** class.
3. **Observer Pattern**:
   - Planned for managing game events efficiently, such as:
     - **Collision events**: Detecting interactions between Mario and objects.
     - **Item collection events**: Updating the score and triggering sound effects when collecting items.
4. **Strategy Pattern**:
   - Will be used for varying behaviors of enemies or gameplay mechanics, such as different AI patterns for enemies or movement styles for Mario.
5. **Decorator Pattern**:
   - Will allow dynamic enhancement of objects, such as adding power-ups to Mario (FireMario, BigMario) or providing unique abilities to enemies (e.g., **faster movement** or **resistance to certain attacks**).
6. **Builder Pattern**:
   - Will simplify the creation of complex levels by using a **LevelBuilder** class to assemble maps, enemies, and items step-by-step.

Game Engine & Graphics

- Using **SFML** to render graphics.
- **Sprite Animation System**: Used for character animations.

- **Tile-based Map Loading**: Implemented to structure and dynamically load the game environment.

VI. References

Super Mario Bros. (1985) Full Walkthrough NES Gameplay [Nostalgia]

New Super Mario Bros. DS HD - Full Game 100% Walkthrough

https://www.youtube.com/watch?v=aCq7P0e4cv0&list=PLlnvVTSJ0XwdnquTl8y5xvsY4aka_8h8H

https://en.wikipedia.org/wiki/Object-oriented_programming

https://gameprogrammingpatterns.com/

https://www.sfml-dev.org/documentation/2.6.1/