2023 Spring

MARIO GAME PROJECT REPORT

Course: EHB354E

Class Lecturer: Tankut Akgül

Class Assistants: İsmail Melik Türker

mehmet soner korucu
040190225

Introduction

This project is about making a 2D Mario game using SFML (Simple and Fast Multimedia Library). SFML is an open-source game design library that is using C++ programming language and object oriented. The project is made by Mehmet Soner Korucu and Fatih Sari. You can also reach source codes in our Github pages by the end of this semester (spring 2023).

Game Details

You can directly run the program by opening the Mario.exe file in debug folder. You can also build and run program in Visual Studio in Windows. If you're using macOS, first, you should install SFML using homebrew. After installing it, open terminal and go to the Mario folder. In Mario folder, type "Make run" command in terminal. (Warning: In macOS, when an object (Mario or turtle) dies, program crushes. This behaviour doesn't happen in Windows. I couldn't fix it yet.)

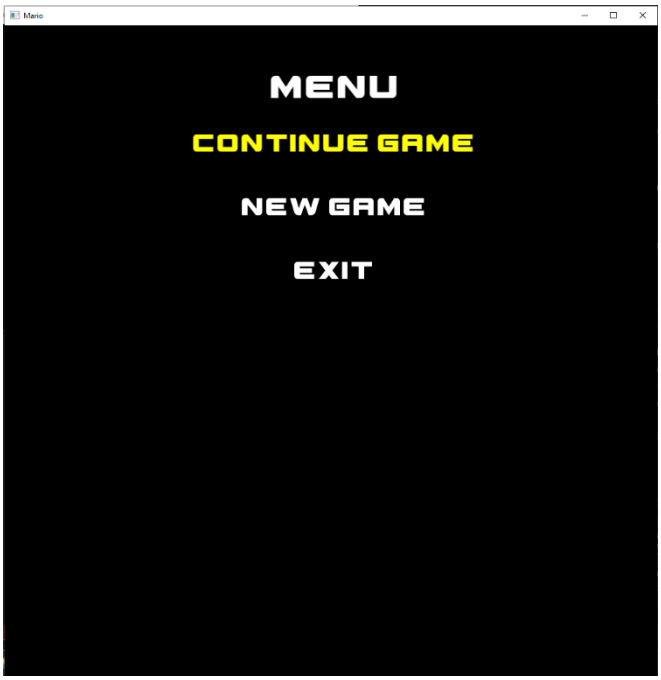
Main Window

In the main window: there are three options:

1 - Continue game

2- New Game

3 – Exit



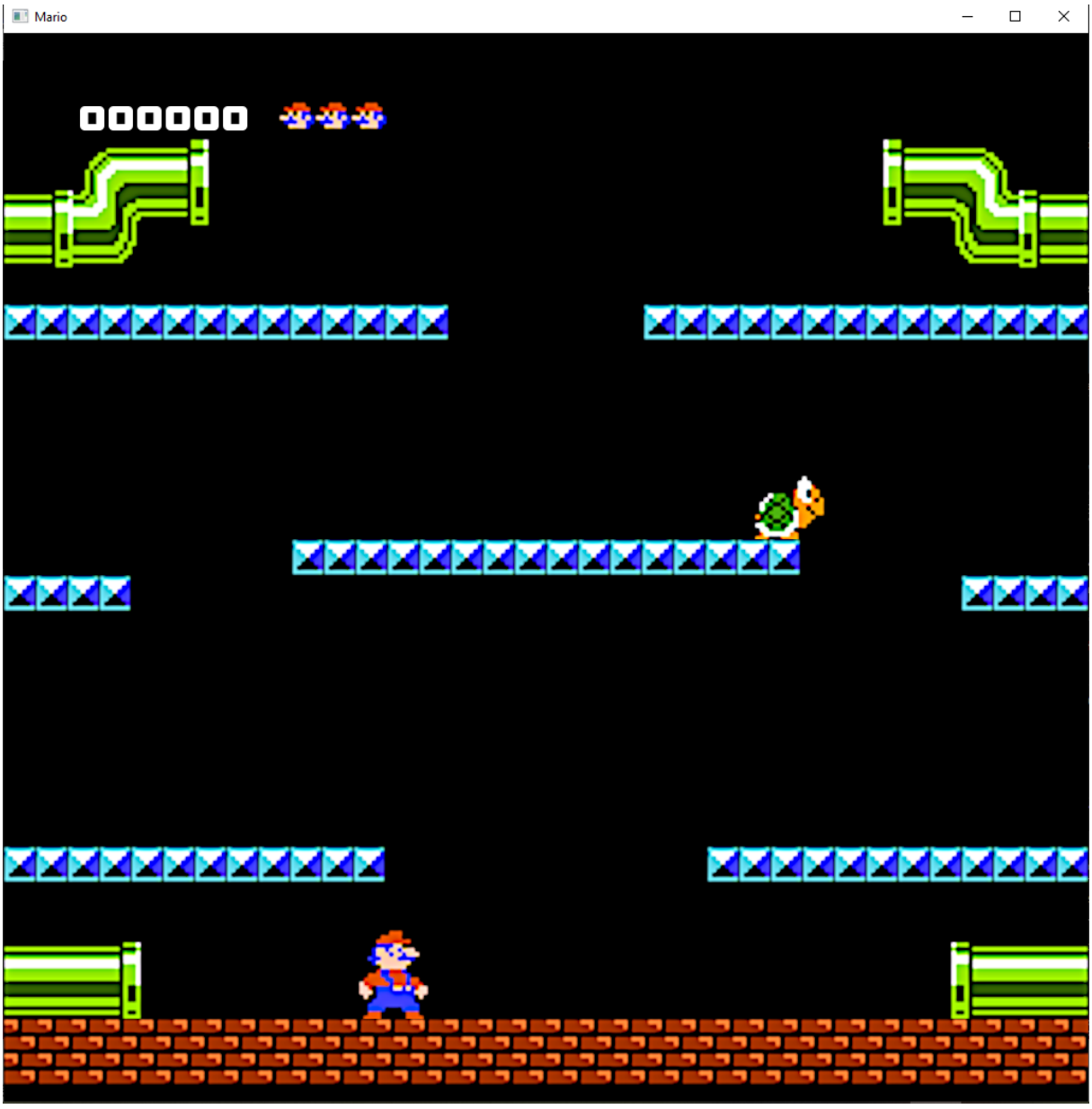
You can navigate between each option using arrow keys and select one of them using enter (return) key.

You can either select continue game or new game in the beginning. Both options will start a new game. In playing state, you can press ESC button to turn back to the main window. You can continue game or select new game. New game option will delete all objects (Mario and the turtles) and start a new game. If you select continue, nothing will change. If you select exit, program will delete all dynamically allocated areas (Mario, turtles, scoreboard) and will return the program successfully.

Game Window

Game window is designed for 1024 x 1024 pixel. You can navigate Mario by using arrow keys. If you press both keys or neither of them, Mario will slide a little and will stop where it is. If you jump and there is a brick on top, he will hit his head and fall back. If there is nothing on top, he will fall back.

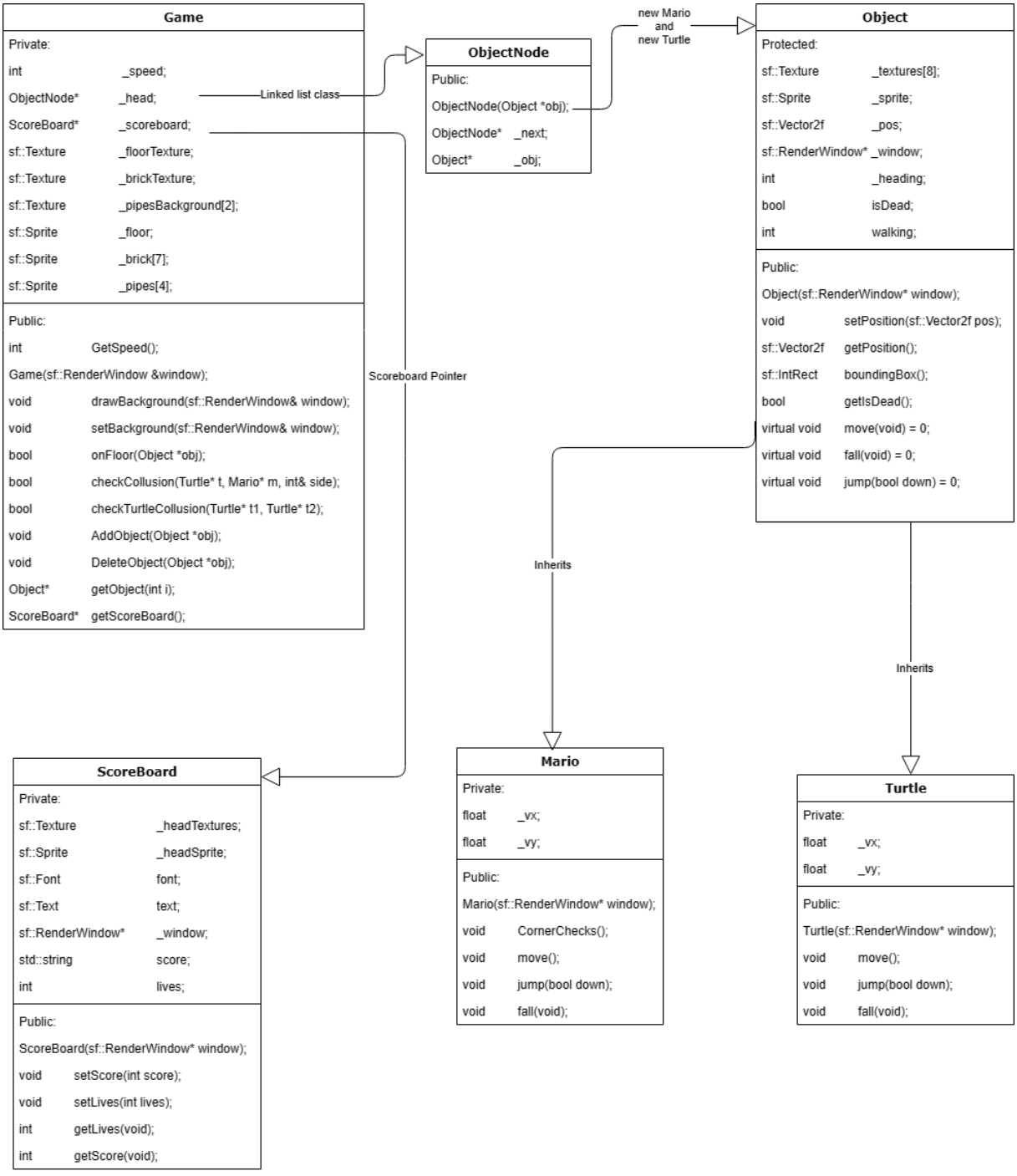
Whenever Mario kills a turtle, turtle starts disappearing from windows by falling and you gain 100 score. On the other hand, if a turtle kills Mario, he will respawn again and you will lose a heal point. If Mario dies 3 times, you will lose the game. If Mario reaches 500 points, in other words, kills 5 turtles, you will win the game and You Win screen will appear on screen and eventually game menu will appear.



Team Info

I mainly coded Mario, Object and ObjectNode classes while my teammate Fatih mainly coded Turtle and Scoreboard. Game class and main loop are coded by me and my teammate together.

Class Implementation



Game Class

```
class Game
{
private:
	int	_speed;
	ObjectNode	*_head;
	ScoreBoard	*_scoreBoard;
	sf::Texture _floorTexture;
	sf::Texture _brickTexture;
	sf::Texture  _pipesBackground[2];
	sf::Sprite	_floor;
	sf::Sprite	_brick[7];
	sf::Sprite	_pipes[4];

public:
	int		GetSpeed();
	Game(sf::RenderWindow &window);
	void	drawBackground(sf::RenderWindow& window);
	void	setBackground(sf::RenderWindow& window);
	bool	onFloor(Object *obj);
	bool	checkCollusion(Turtle* t, Mario* m, int& side);
	bool	checkTurtleCollusion(Turtle* t1, Turtle* t2);
	void	AddObject(Object *obj);
	void	DeleteObject(Object *obj);
	Object *getObject(int i);
	ScoreBoard*	getScoreBoard();
};
```

This is the main class for game. It includes head of the ObjectNode pointer which holds objects of the game, scoreboard pointer which points to scoreboard class and sprites and textures of floor, brick and pipes. We added bricks and pipes into game class since they are not specified specifically.

ObjectNode Class

```
class ObjectNode
{
public:
	Object *_obj;
	ObjectNode *_next;
	ObjectNode(Object *obj);
};
```

This is a simple class. Its Constructor takes an object argument , sets obj pointer to object argument and sets next pointer to nullptr.

Object Class

```
class Object
{
protected:
	sf::Texture _textures[8];
	sf::Sprite _sprite;
	sf::Vector2f _pos;
	sf::RenderWindow* _window;
	int	_heading;
	bool	isDead;
	int	walking;
public:
	Object(sf::RenderWindow* window);
	void	setPosition(sf::Vector2f pos);
	sf::Vector2f getPosition();
	sf::IntRect boundingBox();
	bool	getIsDead();
	virtual void move(void) = 0;
	virtual void fall(void) = 0;
	virtual void jump(bool down) = 0;
};
```

This class is the base class of Mario and Turtles. Its attributes and methods are listed above. Walking is used for Turtle class.

Mario Class

```
class Mario : public Object
{
private:
	float _vx;
	float _vy;
public:
	Mario(sf::RenderWindow* window);
	void	CornerChecks();
	void	move();
	void	jump(bool down);
	void	fall(void);
};
```

This class is written by me. I redefined move, jump and fall functions.

Constructor sets its argument to the window attribute of the base class. Constructor sets textures and position.

CornerChecks is a helper method. At the end of each loop, before the end of jump function, Cornerchecks method checks if the position of Mario is out of the game scope or inside the pipes. If it is, method fixes the position of Mario.

Move method moves Mario to the left and right. If you press left or button and mario is facing opposite direction, method will turn Mario top pressed direction. If Mario is already facing pressed direction, it will move Mario. After you release the key or press both of it, mario will slide and when his speed becomes 0, he will stop.

Jump method moves Mario to up and down. If Mario has no Vertical speed and you press up arrow key, method will set Mario's texture to jumping texture and set Mario's speed. The Down parameter is used for if Mario is intersecting with any brick or floor. If he is not touching any floor and has upward speed, he will rise above with downward acceleration (. If Mario intersects with a brick and has upward speed, he will hit his head and starts falling. Else if he doesn't' intersect with any brick and has a downward speed, he will free fall. While falling, if he intersects with a brick or floor, he will stop and method will set Mario's speed to 0. While Mario has 0 speed but stops intersecting with a brick, he will start free fall.

Fall Method is used when Mario is killed by a turtle.

Discussion

We start and finished the project in 1 week. In start, We work together on discord and write the main parts of the code such as class header files, bricks, pipes and main loop. After finishing main parts and printing game screen, we splited the tasks. Personally, I mostly faced with collision problems. I tried different algorithms and tried many different algorithm implementations. In the end, I've created an algorithm that works as fine as possible. Another problem was because of my mistakes. After implementing Mario and object classes, I tried to implement the ObjectNode class. If I had implemented it before I implemented Mario class, It would be easier. But still, I handled that problem too. According to my tests, there is no leak problem if you exit the game by selecting exit option. The game would have been better if I could prevent vibration of the mario (he vibrates while it stands still) and implement a better slide effect. Also, I couldn't handle vertical collision. When Mario hits a brick horizontally, he passes through the brick and rises.

Overall, this project was a good exercise to learn OOP fundamentals. I am very happy that we worked hard and made this game by our owns.