Due Date: 12 May 2023, 23:59

EHB354E Object Oriented Programming Project

Spring 2023

1. Objective

The objective of this project is to implement the classical Mario platform game using Object-Oriented Programming (OOP) methods and SFML library.

2. Requirements

The project will be implemented using SFML Graphics and Multimedia library (https://www.sfml-dev.org/)

You are recommended to use a Windows machine to do your project. You may use Visual Studio 2017 or later Community version as your build and debug environment. For Visual Studio, download 32-bit version of SFML 2.5.1 (https://www.sfml-dev.org/files/SFML-2.5.1-windows-vc15-32-bit.zip). A sample VC project is provided at Ninova with pre-setup environment for you as well. You may use this project as a starter. If you use this project, make sure you select x86 instead of x64 as build target from drop down box at the top of visual studio window. For Mac and Linux, you need to follow the directions for SFML 2.5.1 for your operating system on this page: https://www.sfml-dev.org/download/sfml/2.5.1/.

3. Project Description

The game graphics will be drawn using a set of sprites. A sprite is a 2D graphics drawn at a particular location on screen. There will be a sprite for each type of visible game element such as Mario, enemies, background objects. The textures for the sprites will be provided by the project.

The game window will be as shown in Figure 1 below.

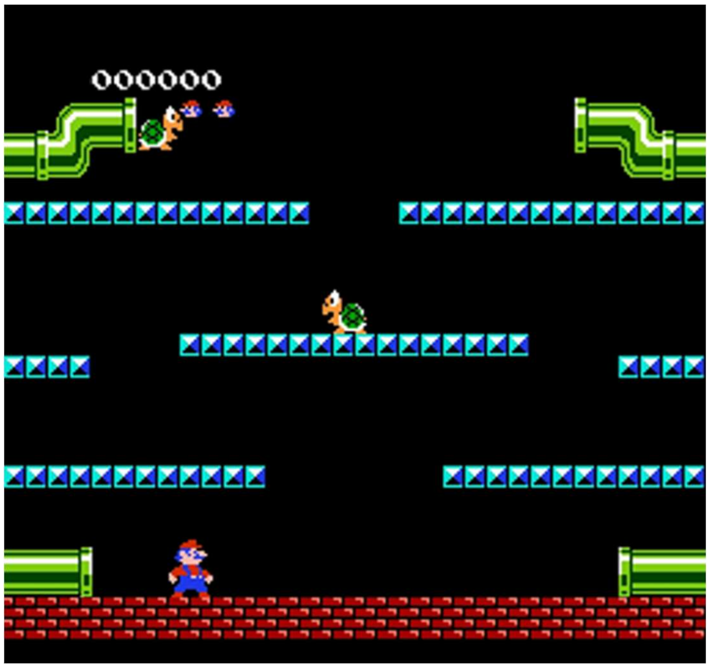


Figure 1. Game window

There will be a menu for the game as shown in Figure 2, which lets the player to start a new game from beginning You are free to design the main menu as you wish. Below image is just an example.

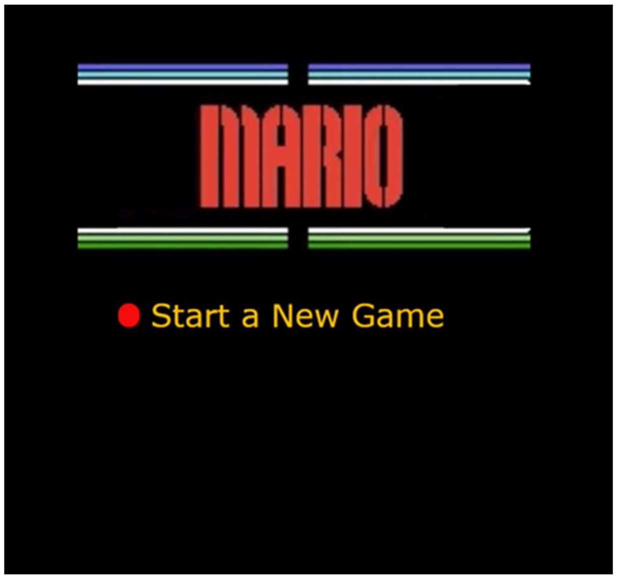


Figure 2. Game menu

3.1 Game Rules

The game will be played by a single player. The player will control Mario character in the game. The turtles will be the enemies.

Mario will stand still, will be able to run in the right and left directions or will be able to jump. When standing still, you need to use mario1.png as the texture. While running, you will be switching between textures mario2.png, mario3.png and mario4.png to produce the run animation. While jumping, you will use mario6.png texture. Since in all of these textures, Mario is facing left, you will need to flip the textures if Mario is facing in the opposite (right) direction.

Mario can jump straight up in place or it can jump diagonally if he was running at the same time. By jumping, he will be able to reach to upper floors shown in blue. So, he can climb up this way. Mario may run off the edge of a floor he is standing on. If this happens, he doesn't die, but drops to the floor below.

Turtles will be running in the game until they die. When they are moving, you will be switching between textures turtle1.png, turtle2.png and turtle3.png to produce the run animation.

The turtles will be dropping from the top left or top right pipes onto the topmost floor. They will fall down off the edge of current floor they are moving onto a floor below. When they reach the bottommost floor, they will move towards the pipes on the bottom left and right corners of the game. They will go into those pipes and will be teleported to the top pipes from which they will drop again. If two turtles meet, they will pass by each other.

Mario will start with 3 lives total at the very bottom floor. If at any point in time Mario touches a moving turtle from left or right, he will fall straight down from the floor he is currently standing on, will fall off the screen and will die. You can use mario7.png texture while Mario is falling down.

If all of Mario's lives are exhausted, the game will be over and a “Game Over” text should be overlayed on screen. Then, the game will return to the main menu. Mario's current lives will be shown under the scoreboard with Mario headcount.

Mario can beat a turtle if he steps over the turtle by jumping over it. If this happens, the turtle will fall straight down from the floor it is currently standing on, will fall off the screen and will die. You can use turtle5.png texture when the turtle is falling down in this case.

There will be a total of 5 turtles. Initially, they will not be on screen. They will drop from the top pipes one after the other. As the game time elapses, they will start moving faster making the game more difficult.

Mario will gain 100 points for each turtle it beats. If he beats all the turtles, “You win!” text will be overlayed on screen and the game will return to the main menu.

3.2 Game Controls

Mario will move left and right using the left and right arrow keys on the keyboard. He will be able to jump by using the up arrow key. Pressing left or right and up arrow at the same time, Mario will jump diagonally. You need to implement proper jumping physics here just like the bouncing tennis ball exercise we did in lecture.

3.3 Classes

You are expected to implement the following classes with given attributes and methods at least. But, as long as you stick with OOP design rules, you may add extra attributes and methods depending on your design:

1. Game Class

This class will be the top class for the game. It will hold the game state with at least the following: Scoreboard, a linked list of objects including all the turtles and Mario. It should have the below methods at least:

```
    void drawBackground(sf::RenderWindow& window);      //Draws the background objects
    bool onFloor(Object *obj);                          //Checks if object is touching a
                                                        floor surface
    bool checkCollusion(Turtle *t, Mario *m, int& side);//Checks if Mario has hit a turtle
                                                        and from which side
```

2. Object Class

This class will keep attributes and methods for the base object class. Mario and Turtle classes explained below will derive from this class via inheritance. Below are minimum required attributes and methods.

<u>Attributes</u>:

```
    sf::Texture textures[8];    //Textures of Object (there are 8 textures total for Mario
                                and 6 textures for Turtle)
    sf::Sprite sprite;          //Sprite for the object
    sf::Vector2f pos;           //Current position of object on screen
    sf::RenderWindow* window;   //Pointer to render window
    int state;                  //Current animation state of the object (one of animation
                                states)
    int heading;                 //Facing direction of object
```

<u>Methods:</u>

```
    Object(sf::RenderWindow *window);   //Constructor
    void setPosition(sf::Vector2f pos); //Sets position of the object
    sf::Vector2f getPosition();         //Gets position of the object
    sf::IntRect boundingBox(void)    //Returns the bounding rectangle of the object texture
    void draw(sf::RenderWindow &window);//Draws current the object texture to screen
    void move(void);                    //Abstract method that will be overridden
    void fall(void);                    //Abstract method that will be overridden
    void jump(bool down);               //Abstract method that will be overridden
```

2. Mario Class

Attributes:
```
    float vx;                           //Horizontal speed
    float vy;                           //Vertical speed
```

Methods:
```
    Mario(sf::RenderWindow *window);     //Constructor
    void move();                         //Moves Mario left or right
    void jump(bool down);                //Makes Mario jump (if down is true, Mario jumps
                                          down off the edge of a floor, otherwise he jumps
                                          up)
    void fall(void);                     //Makes Mario fall straight down when he dies
```

3.Turtle class

Attributes:
```
    float vx;                           //Horizontal speed
    float vy;                           //Vertical speed
```

Methods:
```
    Turtle(sf::RenderWindow *window);//Constructor
    void move();                     //Moves turtle left or right
    void jump(bool down);            //Makes turtle jump (if down is true, turtle jumps
                                     down off the edge of a floor. Down being false can be
                                     optionally used for fly animation of turtle when it is
                                     hit from underneath by Mario)
    void fall(void);                 //Makes turtle fall when it dies
```

4.ScoreBoard class

Attributes:
```
    string score;                //Current score
    int lives;                   //Remaining life count for Mario
```

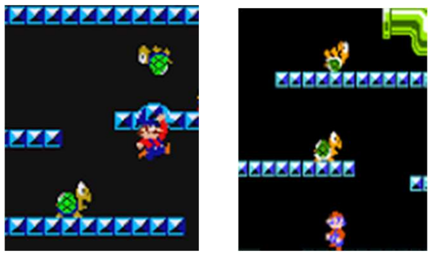
Methods:
```
    void setScore(int score);        //Sets the score
    void setLives(int lives);        //Sets the remaining lives
    int getLives(void);              //Gets the remaining lives
```

4. Bonus

There are three bonus opportunities in this project.

1. You can implement a slide effect for Mario when Mario transitions to stop state from running state, changes direction suddenly or when he hits the floor after jumping up. You can use mario5.png texture for the sliding effect. This bonus is 5 points.
2. For the second bonus, you can add a collusion effect for turtles. If you implement this bonus, then if two turtles meet against each other, they won't pass by each other, but they will first get surprised for a split second and then will both change directions. You need to use turtle4.png in order to animate a turtle getting surprised. This bonus is 10 points.
3. In order to implement the third bonus, you need to add a new way for Mario beating a turtle. For this bonus, Mario can hit the turtle from underneath as shown in Figure 3a. When this happens, the turtle will fly and will fall to floor upside down sitting on its back (Figure 3b). It will stay in this position for 8 secs. Then, when Mario hits the turtle while it is flipped over, the turtle will fall off the floor and will die. If Mario cannot hit the turtle

within 8 secs, the turtle will correct itself and will keep moving faster as it has been aggravated. This bonus is 15 points.



(a) (b)

Figure 3. Mario hitting a turtle from below and flipping it over

5. Project Deliverables

Your submission should include a zip file containing a project report and the contents of your project folder. Each student must submit a separate report written by herself or himself alone. If your project folder is too large for ninova, please delete the contents of the hidden .vs folder in your project tree before compressing your project. To be able to see this folder, you will need to enable show hidden folders option from file explorer.

1. You need to add comments to your code. Uncommented code will get partial credit. Be reasonable with the number of comments you add. Do not try to comment every line.
2. DO NOT submit files separately. Put them into a compressed zip archive and submit the zip archive. Name your zip archive with your team's name such as awesome_team.zip
3. Include your project report in pdf format in your submission (one for each team member).
4. **Only one student** from each group should submit the project. **Do not make duplicate submission.**
5. **Sharing code among project teams is NOT allowed. Doing so will cause credit cut.**

Below is the template for the project report.

Introduction
Briefly describe the project goals here.

Team Info
Describe team members and their roles during the completion of the project here.

Implementation
Here, you should describe the classes you implemented. Point out the object-oriented programming features you used in your implementation.

Try to use block diagrams, flow charts and any other visuals that will help explain your algorithm to a reader who does not know anything about your project.

Discussion
Briefly describe the problems you faced in the implementation of your project and how you could improve it.