Programming with C/C++ Academic year: 2022-2023, Fall semester

*Programming assignment*

# The AI BipBop

## Programming using SFML

Student name: Dominic Armand Stamatoiu
Student number: 2051167

# 1.Overview of project results

Most of the goals of this project were achieved, apart from a few ones that required more time and programming knowledge. One goal that was not met was the implementation of a “smart” AI, in this solution, there is presented only a brute-force “AI”. Another goal that was not achieved is the implementation of the different angles that the ball could take when hitting the board.

The logic behind the solution was to create classes for the ball, board, and bricks that could interact with each other and then implement gameplay elements, for example, the score, which would increase the playfulness character of the game. The ultimate goal of the solution, the project was to create an AI capable of playing the game, which was partially met.

The biggest challenge was making the interactions between the objects, mainly the collision functions, which were necessary for the game to work. Another challenge was finding the right libraries that would help in building the project. Due to the lack of time, I had to find a library that would make it easier for me to build the interactions between objects. I found the SFML library which focuses on simple to code graphics and objects in C++ and used it in the CLion IDE. I worked on this project for two or three hours a day in the span of two weeks.

The code is organized into three parts. The first part outside the main function was used to create the functions that are responsible for detecting the collisions between objects. The second part is represented by the implementation of the classes at the beginning of the main function. The third part is the code inside the while loop that deals with updating the window, and the graphics.

# 2.Tasks and objectives

The first task is the implementation of the window, board, ball, and bricks. I used Paint, to draw the board, bricks, and ball with the given dimensions, then created classes that use textures, mainly the png files of the drawings. After this, I set the positions that the objects should have on the window.

The second task relates to the implementation of the movement of the ball. I created the movement of the ball by setting the velocity for its x and y axis to change if the ball hits any surface, then made the borders of the window so that the ball does not go outside of it. Then I started working on the collision between the objects so that the ball does not go through the bricks and board.

The third task focuses on the movement of the board. As you can see in the code below, I created two if statements, one in which the board moves to the left on the x-axis if the key “A” is pressed, and one in which the board moves to the right if the key “D” is pressed. Also, I created borders for the board on the left and right of the window so that the board does not go over the border.

```
Vector2f board_pos = board.getPosition();
if (E.type == Event::KeyPressed && E.key.code == Keyboard::A && board_pos.x
!= 0)
    board.move(-20, 0);
if (E.type == Event::KeyPressed && E.key.code == Keyboard::D && board_pos.x +
120 != 640)
    board.move(+20, 0);
```

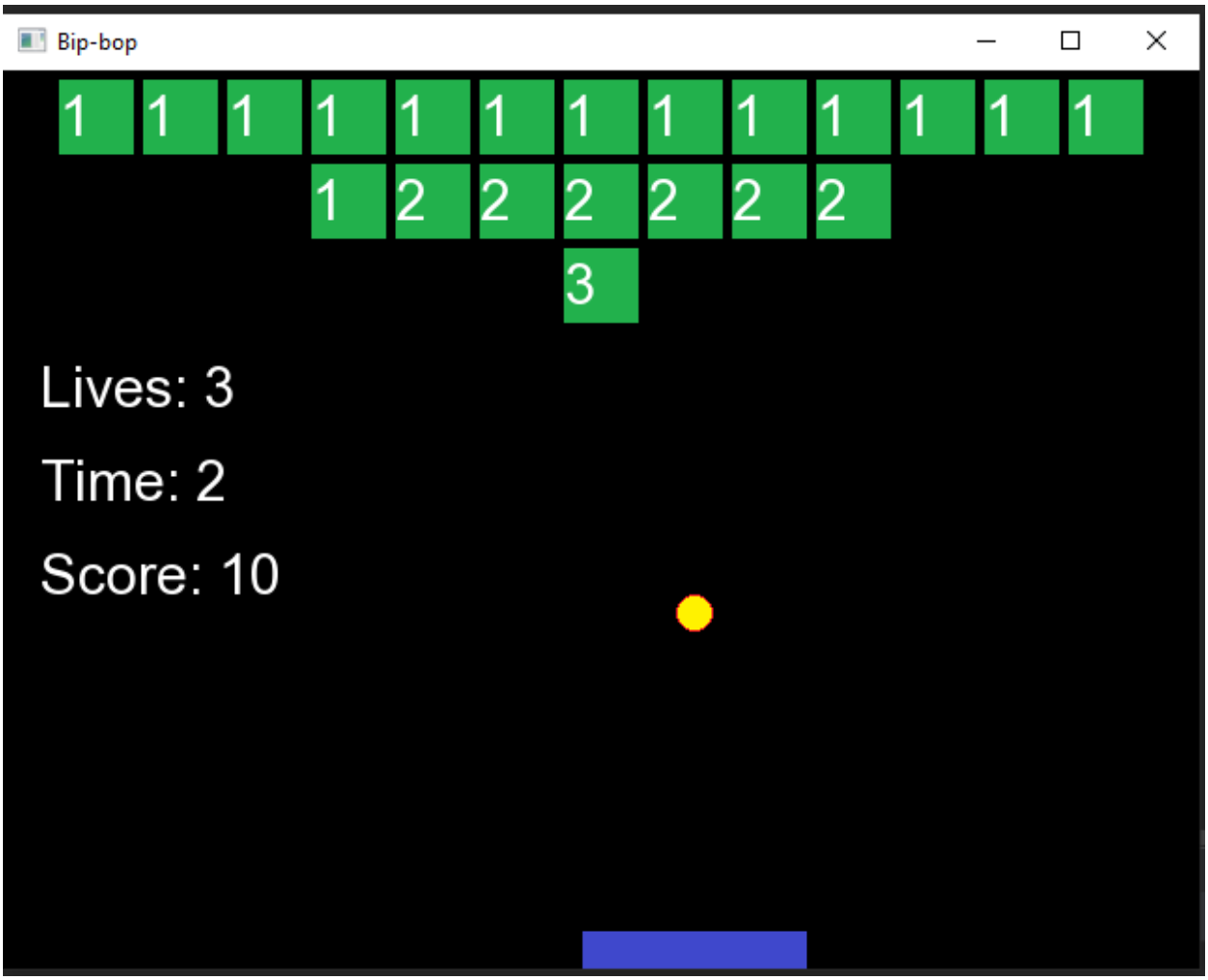
The fourth task is the gameplay aspect of the project. I finished the implementation of the collision between all objects. I created the score of the points, the lives of the player, and the timer that presents the duration of the game in seconds. I also added a text that appears on the window if the player destroys all the bricks and wins or if the player loses all their lives. Then I implemented the integrity of the bricks by adding numbers over each one of them that goes lower and lower if the bricks get hit by the ball. The code below builds the text of the lives of the player on-screen, and the position and color of the text.

```
lives_txt.setFont(my_font);
lives_txt.setString("Lives: 3");
lives_txt.setPosition(20, 150);
lives_txt.setFillColor(Color::White);
```

The fifth task focuses on the AI agent. In the code below, you can see that the brute-force “AI” is activated only if you set the value of the AI variable to 1. Otherwise, the player will be in control of the board movements. The brute-force algorithm makes the board follow the ball by making sure that it's always under it. I also introduced two if statements to make sure that the board does not go over the window border.

```
int AI=1;

if(ball_pos_center_x > board_pos_center_x && AI==1)
{

    if(board_pos_center_x+ 60 <640)
        board.move(10, 0);

}

if( ball_pos_center_x < board_pos_center_x && AI==1)
{
    if(board_pos_center_x-60 >0)
        board.move(-10, 0);
}
```

The figure below shows the overall game, the window, and all of its components explained above.




# 3.Challenges

## 3.1. Challenges solved

One of the biggest challenges was the programming of the collision between the ball and the bricks. For the first iteration of the project, in which I used classes pre-built to encode the dimensions of the objects, a collision method or function did not exist, and it was too complicated to rewrite code about the already-made classes. A solution that I later found was that there exists a class called “Sprite”, that works with textures and takes as input png files, so I had to rebuild the project from scratch. It has a collision method for detecting if two objects are colliding. I first created a bool function that returns true if two Sprites are colliding, but it only worked effectively for the collision between the board and the ball. I made four additional bool functions, that checked if the ball touches a square from the bottom, left, right, or top so that it can change the x and y velocities accordingly. It works, but it still has a problem that was not

fixed. If the ball moves to the square from one of its corners, the ball just goes through the square, not colliding with it.

Another challenge was finding a suitable library and IDE to work with my laptop. I spent about two whole days trying to find something that could be used to build the project. In the end, I found the SFML library and the CLion IDE that paired beautifully.

### 3.2. Challenges addressed but not solved

One such challenge was the implementation of the AI agent. Due to the lack of time and expertise, I managed to only program a brute-force algorithm that would allow the board to follow the ball. It solves some goals of the project but not the overall goal which was to build an AI agent that takes its own decisions.

### 3.3. Challenges not solved (neither attempted)

The challenge that was not attempted at all, was the programming of the different angles that the ball could take when hitting the board. Firstly, because of the lack of time, and secondly, because I did not understand the math behind choosing the proper angle, by the direction and speed of the ball.

## 4.Discussion and future work

In this project, there is no creation of new classes which is why there are no multiple files that make up the code. An improvement to the current version would be to maybe try and create new classes from the already built SFML classes by inheritance. In a way, it would look tidier but at the same time, it would take unnecessary time. Overall, the code is not very efficient, but I also think that it is a good enough solution. The execution of the program is instant and it does not lag. One of the problems of SFML is that you have to be careful with the code that locates the downloaded library and to have all the paths accessible. But, this library can be used for multiple types of games and has easy-to-learn methods and classes.