BLG252E Object Oriented Programming Project 2

1. Objective

The objective of this project is to improve the 2D shooter game environment of Project-1 using Object-Oriented Programming (OOP) methods and SFML library. The game will add a second soldier, bullets and more interactivity with various objects on a plain ground.

2. Requirements

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

If you use a Windows machine, you may use Visual Studio 2017 Community version as your build and debug environment. For Visual Studio 2017, download 32-bit version of SFML 2.5.1 (https://www.sfml-dev.org/files/SFML-2.5.1-windows-vc15-32-bit.zip). Refer to lecture 6 recording for the environment setup with Visual Studio on Windows. 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/. Let me or the course TA, Halil Durmuş (durmush@itu.edu.tr), know if you need any guidance for Mac/Linux.

3. Project Description

There will be two soldiers in the game. The soldiers can fire bullets in 4 possible directions (up, right, left, down) but not diagonally. Player 1 will use arrow keys to move and the enter key to fire bullets. Player 2 will use buttons A, W, S, D and space to move left, up, down and right, and fire bullets, respectively.

To be able to process simultaneous key presses, you should look for both KeyPressed and KeyReleased events. The keyboard issues a KeyPressed event when a key is pressed and issues a KeyReleased event when a key is released. In your implementation, the keys should stick until a corresponding KeyReleased event is received for that key. This way multiple key presses can be handled simultaneously.

The bullets will travel on straight paths. When a bullet hits a sandbag, the bullet will disappear but the sandbag will remain on screen. When a bullet hits a barrel, both the bullet and the barrel will disappear. If there is a player behind the barrel or sandbag that gets hit, the player will not be impacted. When a bullet hits a player, both the bullet and the player will disappear and the player will be spawned at another random location in the game. The player who did the successful shot will get her/his score incremented.

You will implement a scoreboard and display it at the bottom of the screen at the center. Each player will have a total number of successful shots displayed on the scoreboard. When a player reaches 10 total shots, that player will win the game. A text will appear in front of the screen telling “Player X wins, start over? (Y/N)” where X can be either 1 or 2 depending on which player won the game. If one of the players presses Y button, the game will restart. If the player presses the N button, the game will exit. You can use text methods of SFML for writing text to screen (https://www.sfml-dev.org/tutorials/2.5/graphics-text.php). A font file that you can use will be provided with the project.

3.1 Classes

You are expected to use inheritance in this part of the project. You will design a base **Object** class with at least the **init** and the **paint** methods and **window**, **sprite**, **texture** and **pos** attributes. All other classes except **Game** and **BulletList** (explained below) will inherit from this class. You can override the methods of this class if you need to.

In addition, the following new classes need to be implemented:

1. **Bullet**

The bullet class

Private Attributes:

```
float speed                //Bullet speed (you can pick any speed unit you wish)
float angle;               //Bullet travel angle
Bullet* next;              //Pointer to next bullet in a linked list
```

Public Methods:

`void move(void):`
Moves the bullet in its travel direction (angle)

`void setSpeed(float speed):`
Sets the bullet speed
Parameters:

- `speed` `//Bullet speed`

2. **BulletList**

The bullet list class

Private Attributes:

```
Bullet* list;              //Pointer to a linked list of bullets
```

Public Methods:

`BulletList(sf::RenderWindow* window):`
Constructor for the BulletList
Parameters:

- `Window` `//SFML window object`

`void add(Coord pos, int state):`
Adds a new bullet to the list
Parameters:

- `pos` `//Position to instantiate a bullet at`
- `state` `//State of the player when the bullet was fired`

`void update(void):`
Moves every bullet in the list

`void checkCollision(Player* players, Barrel* barrels, Sandbag* sandbags, int np, int nb, int ns):`
Checks whether a bullet in the list collided with other objects or with the edge of the screen. If there is a collision it should delete the bullet from the list and take appropriate action

Parameters:

- `players` `//Pointer to player objects`
- `barrels` `//Pointer to barrel objects`
- `sandbags` `//Pointer to sandbag objects`
- `np` `//Number of players`
- `nb` `//Number of barrels`
- `ns` `//Number of sandbags`

Returns an integer value indicating the collided object or 0 if no collusion. You are free to design this return value as you wish or you may not use the return value at all.

In addition, the following attributes and methods should be added to the other classes:

Barrel class:

```
int isVisible                    //Private attribute that determines whether barrel is visible
bool getVisible(void)            //Returns isVisible attribute of barrel
void setVisible(bool visible)    //Hides or shows the barrel (changes its visible attribute)
```

Game class:

```
BulletList *bullets              //Pointer to BulletList
sf::Text text;                   //Text object
sf::Font font;                   //Font object
```

Player class:

```
int score                        //Score of the player
void incrementScore(void)        //Increments score of the player
int getScore(void)               //Returns the current score of the player
```

4. Project Deliverables

Your submission should include a zip file containing a project report and the contents of your project folder.

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 individually. Put them into a compressed zip archive and submit the zip archive. Name your zip archive with your full name such as ahmet_bilir.zip
3. Submit your homework zip file as an attachment to Ninova.

Below is the template for the project report.

Introduction

Briefly describe the project goals here.

Implementation

Here, you should describe the new classes you implemented and the modifications you made to the existing classes You should also describe how you implemented the bullet collision

Discussion

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