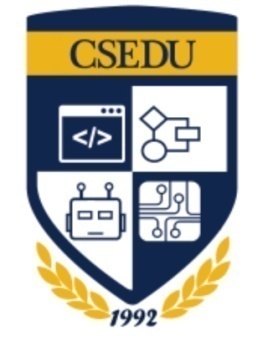
****

**University of Dhaka**

**Department of Computer Science and Engineering**

**CSE – 1211**

**Programming Fundamentals** **Lab Project**

**AIRSTrIKE DEFENSE**

Submitted by-

**1.Sakib Hasan [Roll: AE – 63]**

**2.Tanjid Hasan Tonmoy [Roll: FH – 09]**

Submitted to –

Md.Shiplu Hawladar

Lecturer, CSE,DU

Submission Date: 24 November, 2016

|  |  |  |
| --- | --- | --- |
|  | **Table of Contents** |  |
|  |  |  |
| **No.** | **Contents** | **Pages** |
|  |  |  |
| 1. | Introduction | 03 |
|  |  |  |
| 2. | Project Name | 03 |
|  |  |  |
| 3. | Game Objective | 03 |
|  |  |  |
| 4. | Project Outline | 03 |
|  |  |  |
| 5. | Main Features | 04 |
|  |  |  |
| 6. | Special Features | 05 |
|  |  |  |
| 7. | Additional Features | 05 |
|  |  |  |
| 8. | Graphical Enhancements and Coding | 05 |
|  |  |  |
| 9. | Coding Challenges | 05 |
|  |  |  |
| 10. | Graphical Interface | 06 |
|  |  |  |
| 11. | Main Game Interface | 07 |
|  |  |  |
| 12. | Project Layout and Coding Algorithm | 09 |
|  |  |  |
| 13. | Source Code | 14 |
|  |  |  |
| 14. | Conclusion | 31 |
|  |  |  |
| 15. | References |  |
|  |  |  |

**Introduction:**

The project was aimed at making the students familiar with the use of programming language in a practical manner using a graphics library. We used SFML(Simple and Fast Multimedia Library) for graphics and multimedia to create a game which fun, challenging and use our understanding of programming concepts.

**Project name:**

AirStrike Defense

**Project introduction:**

The project is two-dimensional shooting game with multiple levels and varying difficulty. The enemy attacks from the air and the player must defend his own helicopter or tank while attacking the enemy at the same time.

**Game objective:**

The main purpose of the project is to provide the player with a challenging but entertaining experience. Special attention has been paid to the design and structure of the game so that the player is always intrigued while playing the game.

The game requires full attention of the player and we tried to make the experience as smooth as possible. This project was an opportunity to implement the ideas we have using the C and C++ programming language. We tried to make sure the player has a fun time playing the game.

In this game, the player must shoot enemy helicopters and dodge the bullets from the enemy at the same time. There are different ammunition and vehicles both for the player and enemy in each level which makes the gameplay more interesting. The game aims at using our C programming knowledge and implementing it to create something fun, attractive and interactive.

**Project outline:**

The project is an exquisite shooting game with multiple levels of difficulties. There are three levels which share a common theme the player must defend himself and kill the enemy. But the level has a distinct feel with different capabilities of the player and the enemy.

In the first level, the player shoots helicopter in the air from an advanced artillery tank. Though this the easiest level, it is difficult to keep concentration and win. The enemy moves in a straight line in this level, but as the player progresses to level two, the enemy uses a curved path making it more difficult to shoot correctly. The player also gains ability with a helicopter that can go up but it simulates gravity (comes down when UP key is not pressed) which increases the difficulty further.

There is a bonus third level where there is a different enemy which has multiple lives, if the player can defeat the final enemy the player wins. Each level has a different look and feel and keeps the player engaged. We used several animations to make the visual experience better.

**Main features:**

1. **Flicker-free smooth game play:**

One of the main reasons we chose the SFML library was the better-looking graphics it can render compared to BGI. SFML uses double buffering by default making the game flicker-free and smooth. The game looks pleasing to the eye compared to headache inducing flickering that hampers the overall experience.

**(ii) Challenging and fun:**

The game is challenging and fun at the same time with increasing difficulties.

**(iii) Multiple Levels:**

The game has multiple levels including a bonus level each with different enemies and difficult challenges.

**(iv) Sound Effects:**

This is another feature that makes this game different. We used the audio module of the SFML library and used Music for larger files and Sound buffer for smaller music files. This feature helps this game to be more interesting for the player.

**(v) Movement of the Enemy:**

In the first level, the enemy moves in a straight line horizontally, but as the player progresses to level two, the enemy uses a curved path making it more difficult to shoot correctly. We used trigonometric functions to achieve the motion. The Final enemy(“Boss”) also has a similar motion but is faster. The movement makes shooting difficult and changes the directions of the missiles shot by the enemy.

**(vi) Movement of the Player:**

In the first level the player gets an artillery tank which is replaced with a helicopter in the second level. The helicopter has ability with a helicopter to go up but falls down (similar to gravity) which increases the difficulty further.

g.

**(vi) Lives and Scores:**

The player has 3 lives by default and loses life when it gets shot but gets bonus lives when it reaches boss level. The player scores 10 point per successful shot in first level, 20 points for the second one and 30 points for the third one.

**Additional features:**

1. **Welcome Screen:**

The user is greeted with a welcome screen which lasts a few seconds and a background music is played. However, the player can skip at any time by pressing “ENTER” to go to the main menu.

1. **Main Menu:**
   1. **START:** By clicking this tab, a page with the following three optionsappear-

* Level 1
* Level 2
* Level 3

1. **HELP:**

This tab includes the information of the game controls and other additional features.

c. **ABOUT:**

This includes additional information about the game and credits.

d. **EXIT:**

User can simply exit the game by clicking on this tab.

**Other features:**

**1.** **Use of basic keys:**

The users can use some basic keys to carry out simple operations. For example, the users are allowed to ENTER button to return to main menu.

**5.** **Exit option:**

The users can exit the game by simply clicking on the exit option displayed in the Main Menu tab or by pressing ESC at any time.

**Coding Challenges:**

As SFML is object oriented library it was hard for us to learn and code using SFML and C++. Moreover this library is still under development. That’s why we had to face some problems while coding.

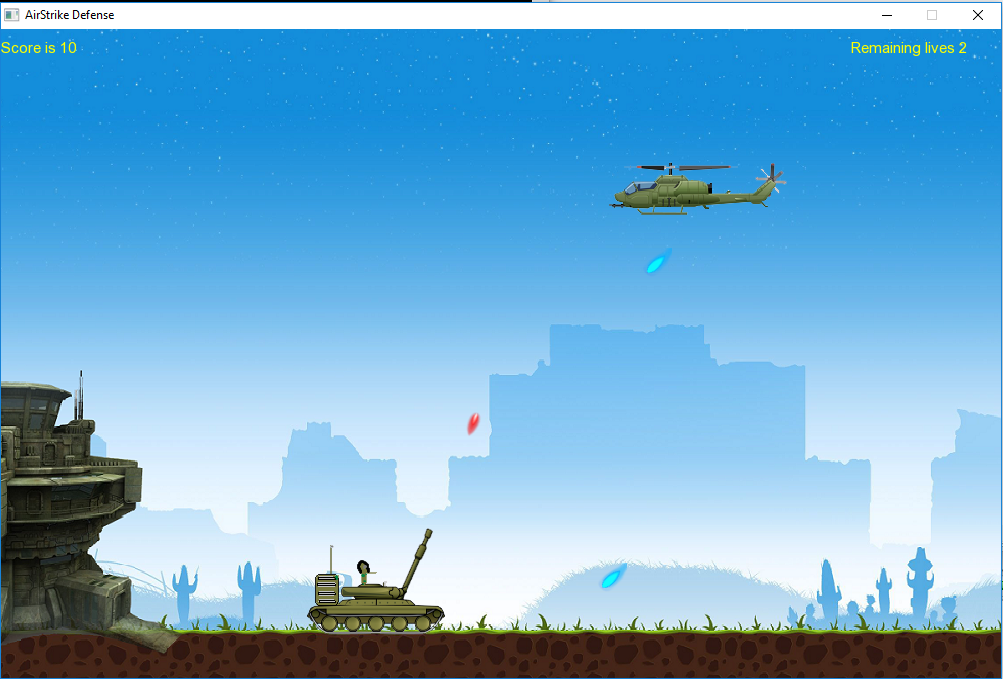
**Graphical Enhancement:**

We used Sprite Sheet to show animation of characters and making visual effects.

**Graphical Interface:**



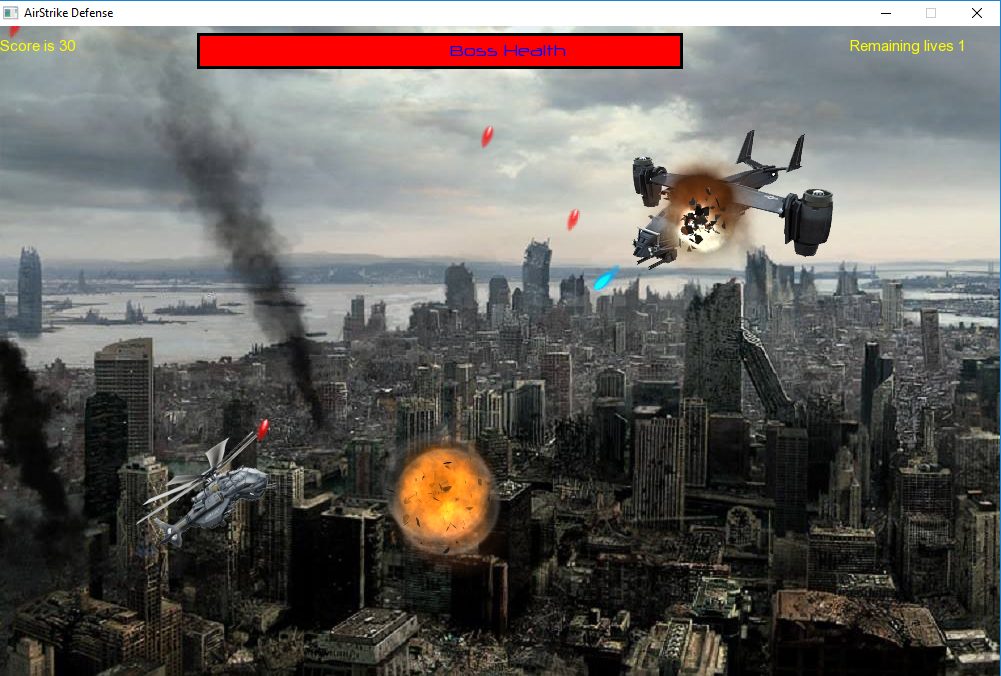
Level 1



Level 2



Level 3



Game over



**Functions Used:**

**Sf::Window**

|  |  |
| --- | --- |
|  | [Window](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Window.php#a5359122166b4dc492c3d25caf08ccfc4) () |
|  | Default constructor. |
|  | |
|  | [Window](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Window.php#a1bee771baecbae6d357871929dc042a2) ([VideoMode](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1VideoMode.php) mode, const [String](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1String.php) &title, Uint32 style=Style::Default, const [ContextSettings](http://www.sfml-dev.org/documentation/2.4.1/structsf_1_1ContextSettings.php)&settings=[ContextSettings](http://www.sfml-dev.org/documentation/2.4.1/structsf_1_1ContextSettings.php)()) |
|  | Construct a new window. |
|  | |
|  | [Window](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Window.php#a6d60912633bff9d33cf3ade4e0201de4) (WindowHandle handle, const [ContextSettings](http://www.sfml-dev.org/documentation/2.4.1/structsf_1_1ContextSettings.php) &settings=[ContextSettings](http://www.sfml-dev.org/documentation/2.4.1/structsf_1_1ContextSettings.php)()) |
|  | Construct the window from an existing control. |
| bool | [isOpen](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Window.php#a5aa9c2b2b0e51d3423c2b66c80253337) () const |
|  | Tell whether or not the window is open. |
|  | |
|  | |
| bool | [pollEvent](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Window.php#a338e996585faf82e93069858e3b531b7) ([Event](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Event.php) &event) |
|  |  |
|  | Pop the event on top of the event queue, if any, and return it. |
| void | [setSize](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Window.php#ab94ea32f22d15c0df11588e319de2546) (const [Vector2u](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Vector2.php) &size) |
|  | Change the size of the rendering region of the window. |
|  | |
| void | [setFramerateLimit](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Window.php#af4322d315baf93405bf0d5087ad5e784) (unsigned int limit) |
|  | Limit the framerate to a maximum fixed frequency. |
| void | [display](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Window.php#adabf839cb103ac96cfc82f781638772a) () |
|  | Display on screen what has been rendered to the window so far. |

**sf::videomode**

|  |
| --- |
| [VideoMode](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1VideoMode.php#a04c9417e5c304510bef5f6aeb03f6ce1)() |
|  | Default constructor. |

Sf::Text

|  |  |
| --- | --- |
|  | [Text](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Text.php#aff7cab6a92e5948c9d1481cb2d87eb84) () |
|  | Default constructor. |
|  | |
|  | |
| void | [setString](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Text.php#a7d3b3359f286fd9503d1ced25b7b6c33) (const [String](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1String.php) &string) |
|  | Set the text's string. |
|  | |
| void | [setFont](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Text.php#a2927805d1ae92d57f15034ea34756b81) (const [Font](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Font.php) &font) |
|  | Set the text's font. |
|  | |
| void | [setCharacterSize](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Text.php#ae96f835fc1bff858f8a23c5b01eaaf7e) (unsigned int size) |
|  | Set the character size. |
|  | |
|  |  |
|  |  |
|  | |
| void | [setFillColor](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Text.php#ab7bb3babac5a6da1802b2c3e1a3e6dcc) (const [Color](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Color.php) &color) |
|  | Set the fill color of the text. |
|  | |
| void | [setOutlineColor](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Text.php#aa19ec69c3b894e963602a6804ca68fe4) (const [Color](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Color.php) &color) |
|  | Set the outline color of the text. |
|  | |
| void | [setOutlineThickness](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Text.php#ab0e6be3b40124557bf53737fe6a6ce77) (float thickness) |
|  | Set the thickness of the text's outline. |
|  | |
| void | [setPosition](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#a4dbfb1a7c80688b0b4c477d706550208) (float x, float y) |
|  | set the position of the object |
|  | |
| void | [setPosition](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#af1a42209ce2b5d3f07b00f917bcd8015) (const [Vector2f](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Vector2.php) &position) |
|  | set the position of the object |
|  | |
| void | [setRotation](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#a32baf2bf1a74699b03bf8c95030a38ed) (float angle) |
|  | |
| void | [setScale](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#aaec50b46b3f41b054763304d1e727471) (float factorX, float factorY) |
|  | set the scale factors of the object |
|  | |
|  | |
| void | [move](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#a86b461d6a941ad390c2ad8b6a4a20391) (float offsetX, float offsetY) |
|  | Move the object by a given offset. |
|  | |
| void | [move](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#ab9ca691522f6ddc1a40406849b87c469) (const [Vector2f](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Vector2.php) &offset) |
|  | Move the object by a given offset. |
|  | |
| void | [rotate](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#af8a5ffddc0d93f238fee3bf8efe1ebda) (float angle) |
|  | Rotate the object. |
|  | |

**Sf::Texture**

|  |  |
| --- | --- |
|  | [Texture](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Texture.php#a3e04674853b8533bf981db3173e3a4a7) () |
|  | Default constructor. |
| bool | [loadFromFile](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Texture.php#a8e1b56eabfe33e2e0e1cb03712c7fcc7) (const std::string &filename, const [IntRect](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Rect.php) &area=[IntRect](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Rect.php)()) |
|  | Load the texture from a file on disk. |
| void | [setSmooth](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Texture.php#a0c3bd6825b9a99714f10d44179d74324) (bool smooth) |
|  | Enable or disable the smooth filter. |

**Sf::Sprite**

|  |  |
| --- | --- |
|  | [Sprite](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sprite.php#a92559fbca895a96758abf5eabab96984) () |
|  | Default constructor. |
|  | |
|  | [Sprite](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sprite.php#a2a9fca374d7abf084bb1c143a879ff4a) (const [Texture](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Texture.php) &texture) |
|  | Construct the sprite from a source texture. |
|  | |
| void | [setTexture](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sprite.php#a3729c88d88ac38c19317c18e87242560) (const [Texture](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Texture.php) &texture, bool resetRect=false) |
|  | Change the source texture of the sprite. |
|  | |
| void | [setTextureRect](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sprite.php#a3fefec419a4e6a90c0fd54c793d82ec2) (const [IntRect](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Rect.php) &rectangle) |
|  | Set the sub-rectangle of the texture that the sprite will display. |
|  | |
|  | |
| [FloatRect](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Rect.php) | [getLocalBounds](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sprite.php#a69557a8369bc2e26dd2e2eb2c50f5c90) () const |
|  | Get the local bounding rectangle of the entity. |
|  | |
| [FloatRect](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Rect.php) | [getGlobalBounds](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sprite.php#a203d2d8087bfdca2ebc3c0485cdb7409) () const |
|  | Get the global bounding rectangle of the entity. |
|  | |
| void | [setPosition](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#a4dbfb1a7c80688b0b4c477d706550208) (float x, float y) |
|  | set the position of the object |
|  | |
| void | [setPosition](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#af1a42209ce2b5d3f07b00f917bcd8015) (const [Vector2f](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Vector2.php) &position) |
|  | set the position of the object |
|  | |
|  | |
| void | [setScale](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#aaec50b46b3f41b054763304d1e727471) (float factorX, float factorY) |
|  | set the scale factors of the object |
|  | |
|  | |
|  |  |
| void | [move](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#a86b461d6a941ad390c2ad8b6a4a20391) (float offsetX, float offsetY) |
|  | Move the object by a given offset. |
|  | |
| void | [move](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Transformable.php#ab9ca691522f6ddc1a40406849b87c469) (const [Vector2f](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Vector2.php) &offset) |
|  | Move the object by a given offset. |

**sf::SoundBuffer**

|  |  |
| --- | --- |
|  | [SoundBuffer](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1SoundBuffer.php#a0cabfbfe19b831bf7d5c9592d92ef233) () |
|  | Default constructor. |
|  | |
|  | [SoundBuffer](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1SoundBuffer.php#aaf000fc741ff27015907e8588263f4a6) (const [SoundBuffer](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1SoundBuffer.php) &copy) |
|  | Copy constructor. |
| bool | [loadFromFile](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1SoundBuffer.php#a2be6a8025c97eb622a7dff6cf2594394) (const std::string &filename) |
|  | Load the sound buffer from a file. |

**sf::Sound**

|  |  |
| --- | --- |
|  | [Sound](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sound.php#a36ab74beaaa953d9879c933ddd246282) () |
|  | Default constructor. |
|  | |
|  | |
| void | [play](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sound.php#a2953ffe632536e72e696fd880ced2532) () |
|  | Start or resume playing the sound. |
|  | |
| void | [pause](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sound.php#a5eeb25815bfa8cdc4a6cc000b7b19ad5) () |
|  | Pause the sound. |
|  | |
| void | [stop](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sound.php#aa9c91c34f7c6d344d5ee9b997511f754) () |
|  | stop playing the sound |
|  | |
| void | [setBuffer](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Sound.php#a8b395e9713d0efa48a18628c8ec1972e) (const [SoundBuffer](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1SoundBuffer.php) &buffer) |
|  | Set the source buffer containing the audio data to play. |

**Sf::Time**

|  |  |
| --- | --- |
|  | [Time](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Time.php#acba0cfbc49e3a09a22a8e079eb67a05c) () |
|  | Default constructor. |
|  | |
| float | [asSeconds](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Time.php#a7538140d095e48da9d7eee015dd455a9) () const |
|  | Return the time value as a number of seconds. |
|  | |
| Int32 | [asMilliseconds](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Time.php#a85e6deb41fa71896508ce0f64059a6ae) () const |
|  | Return the time value as a number of milliseconds. |

Mouse

|  |  |
| --- | --- |
| static bool | [isButtonPressed](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Mouse.php#ab647159eb88e369a0332a9c5a7ba6687) ([Button](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Mouse.php#a4fb128be433f9aafe66bc0c605daaa90) button) |
|  | Check if a mouse button is pressed. |
|  | |
| static [Vector2i](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Vector2.php) | [getPosition](http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Mouse.php#ac368680f797b7f6e4f50b5b7928c1387) () |

**The source code:**

#include<SFML/Graphics.hpp>

#include<SFML/Audio.hpp>

#include<iostream>

#include<sstream>

#include<cmath>

int main()

{

sf::Vector2i screen(1000, 650);

sf::RenderWindow window(sf::VideoMode(screen.x, screen.y), "AirStrike Defense", sf::Style::Close | sf::Style::Titlebar); //not allowing to resize the window

window.setFramerateLimit(40);

int ii, temp, shoot = 0, score = 0, lives = 3, over\_reason = 2;

bool draw\_explo = false;

bool draw\_explo2 = false;

//variable for handling scenes

int scene = 0; //initialized to 0 so the first scene starts

//textures for scene 1

sf::Texture t6, t7, t8;

t6.loadFromFile("bg.jpg");

t7.loadFromFile("bg11.png");

t8.loadFromFile("explo.png");

//sprites for scene 1

sf::Sprite background1, start\_scr, explo;

start\_scr.setTexture(t7);

background1.setTexture(t6);

explo.setTexture(t8);

t6.setSmooth(true);

t7.setSmooth(true);

sf::IntRect r2(0, 0, 1000, 650);

start\_scr.setTextureRect(r2);

float start\_hold = 12;

//sizing the texture of background in scene 1

sf::IntRect r1(100, 100, 1000, 650);

background1.setTextureRect(r1);

//variables for explosion animaion

int ex\_l = 192, ex\_t = 0, ex\_w = 192, ex\_h = 192, driver = 0; //l for left r right, h height, w width

int ex\_l2 = 192, ex\_t2 = 0, ex\_w2 = 192, ex\_h2 = 192, driver2 = 0;

//variables for mouse input;

sf::Vector2i localpos;

//loading the fonts

sf::Font neuro, pdark, arial;

pdark.loadFromFile("pdark.ttf");

if (!arial.loadFromFile("arial.ttf"))

return -7;

if (!neuro.loadFromFile("neuro.ttf"))

return EXIT\_FAILURE;

//designing the title (used in scene 0 and 1)

sf::Text title;

title.setFont(pdark);

title.setPosition(50, 440);

title.setCharacterSize(80);

title.setFillColor(sf::Color::White);

title.setString("Airstrike Defence");

sf::Time elapsed;

sf::Clock cclock;

cclock.restart();

//loading and playing music for first few seconds

sf::Music music;

if (!music.openFromFile("music1.ogg"))

return -1; // error

if (scene == 0) music.play();

//sound for shooting

sf::SoundBuffer firebuffer;

firebuffer.loadFromFile("fire.wav");

sf::Sound fire\_sound;

fire\_sound.setBuffer(firebuffer);

//sound for explosion

sf::SoundBuffer explobuffer;

explobuffer.loadFromFile("fire2.wav");

sf::Sound explo\_sound;

explo\_sound.setBuffer(explobuffer);

sf::Texture t2, t3, t4, t5, texp, bosst, lv2, heliv2, bg3, lvls; ////textures for scene 2(game scene)

//loading the textures

t2.loadFromFile("bg22.png");

t3.loadFromFile("helis2.png");

t4.loadFromFile("sbullet2.png");

t5.loadFromFile("tank12.png");

bg3.loadFromFile("bgl3.jpg");

texp.loadFromFile("explo.png");

bosst.loadFromFile("boss.png");

lvls.loadFromFile("levels.png");

sf::Sprite explo2, bglv2, helilv2, back3, boss, lvchoice;

explo2.setTexture(texp);

t2.setSmooth(true);

lv2.loadFromFile("bg2.png");

bglv2.setTexture(lv2);

heliv2.loadFromFile("heliv2.png");

helilv2.setTexture(heliv2);

boss.setTexture(bosst);

back3.setTexture(bg3);

lvchoice.setTexture(lvls);

int no\_of\_bullet = 49;

sf::Sprite background2, heli, bulet[49], tank;

int tankX = 50, tankY = 500, helilv2X, helilv2Y;

helilv2X = tankX;

helilv2Y = tankY;

background2.setTexture(t2);

heli.setTexture(t3);

tank.setTexture(t5);

tank.setPosition(tankX, tankY); //setting initial position of the tank

);//setting initial position of the heli lvl2

helilv2.setPosition(helilv2X, helilv2Y);

//setting bullet texture

for (ii = 0; ii<no\_of\_bullet; ii++)

{

bulet[ii].setTexture(t4);

}

int bullet\_no = 0;

//variables for helicopter1

sf::Vector2f tankpos, helipos, bosspos, helilv2pos;

int heliD = 0, heliX = 850, bossX = 850;

heli.setPosition(heliX, 100);

boss.setPosition(bossX, 100);

//textures, sprites and text for scene 3(game over)

sf::Texture t10;

t10.loadFromFile("game\_over.jpg");

sf::Sprite bg\_over;

bg\_over.setTexture(t10);

//textures, sprites and text for scene 4(HELP)

sf::Texture helpg;

helpg.loadFromFile("help.jpg");

sf::Sprite helpbg;

helpbg.setTexture(helpg);

//textures, sprites and text for scene 5(About)

sf::Texture aboutt;

aboutt.loadFromFile("about.jpg");

sf::Sprite about1;

about1.setTexture(aboutt);

sf::Text skip\_command, skip\_command2, reason1, reason2, Score, Life, High\_score, Level2, Hscr;

skip\_command.setFont(arial);

skip\_command.setPosition(10, 630);

skip\_command.setCharacterSize(15);

//skip\_command.setFillColor(sf::Color::White);

skip\_command.setString("Press ENTER to skip to the main menu");

Life.setFillColor(sf::Color::Yellow);

Score.setFillColor(sf::Color::Yellow);

High\_score.setFillColor(sf::Color::Red);

High\_score.setFont(arial);

High\_score.setPosition(330, 450);

High\_score.setCharacterSize(50);

Hscr.setFillColor(sf::Color::Green);

Hscr.setFont(arial);

Hscr.setPosition(250, 540);

Hscr.setCharacterSize(50);

skip\_command2.setFont(arial);

skip\_command2.setPosition(880, 630);

skip\_command2.setCharacterSize(10);

skip\_command2.setString("Press ESC to exit");

reason1.setFillColor(sf::Color::Black);

reason1.setFont(arial);

reason1.setPosition(300, 100);

reason1.setCharacterSize(60);

reason1.setString("Enemies win");

reason2.setFillColor(sf::Color::Black);

reason2.setFont(arial);

reason2.setPosition(200, 100);

reason2.setCharacterSize(60);

reason2.setString("No more bullets left");

Score.setFont(arial);

Score.setPosition(0, 10);

Score.setCharacterSize(15);

Life.setFont(arial);

Life.setPosition(850, 10);

Life.setCharacterSize(15);

//level change text

Level2.setFillColor(sf::Color::Red);

Level2.setFont(pdark);

Level2.setPosition(200, 550);

Level2.setCharacterSize(40);

//variable for rectangle shapes

float rectPosx, rectPosy;

rectPosx = 100;

rectPosy = 180;

//shapes and texts for scene 1

sf::RectangleShape rect1, rect2, rect3, rect4; //4 rectangular shape for the menus

rect1.setSize(sf::Vector2f(300, 50));

rect1.setOutlineColor(sf::Color::Black);

rect1.setOutlineThickness(5);

rect1.setFillColor(sf::Color::White);

rect1.setPosition(rectPosx, rectPosy);

sf::Text start("START", neuro);

start.setFillColor(sf::Color::Red);

start.setCharacterSize(45);

start.setPosition(rectPosx + 80, rectPosy);

rect2.setSize(sf::Vector2f(300, 50));

rect2.setOutlineColor(sf::Color::Black);

rect2.setOutlineThickness(5);

rect2.setFillColor(sf::Color::White);

rect2.setPosition(rectPosx, rectPosy + 70);

sf::Text help("HELP", neuro);

help.setFillColor(sf::Color::Red);

help.setCharacterSize(40);

help.setPosition(rectPosx + 80, rectPosy + 70);

rect3.setSize(sf::Vector2f(300, 50));

rect3.setOutlineColor(sf::Color::Black);

rect3.setOutlineThickness(5);

rect3.setPosition(rectPosx, rectPosy + 70 \* 2);

rect3.setFillColor(sf::Color::White);

sf::Text about("ABOUT", neuro);

about.setFillColor(sf::Color::Red);

about.setCharacterSize(40);

about.setPosition(rectPosx + 80, rectPosy + 140);

rect4.setSize(sf::Vector2f(300, 50));

rect4.setOutlineColor(sf::Color::Black);

rect4.setOutlineThickness(5);

rect4.setFillColor(sf::Color::White);

rect4.setPosition(rectPosx, rectPosy + 70 \* 3);

sf::Text exit("EXIT", neuro);

exit.setFillColor(sf::Color::Red);

exit.setCharacterSize(40);

exit.setPosition(rectPosx + 85, rectPosy + 210);

sf::RectangleShape blife;

//boss life

int blifex = 600;

blife.setSize(sf::Vector2f(blifex, 30));

blife.setOutlineColor(sf::Color::Black);

blife.setOutlineThickness(3);

blife.setFillColor(sf::Color::Red);

blife.setPosition(200, 10);

sf::Text lifeb("Boss Health", neuro);

lifeb.setFillColor(sf::Color::Blue);

lifeb.setCharacterSize(15);

lifeb.setPosition(450, 15);

//variable for collision handling

sf::Vector2f bulpos;

//experimental: Enemy bullets

sf::Texture ebl;

ebl.loadFromFile("sbullet3.png");

int jj;

sf::Sprite ebulet[15];

for (jj = 0; jj<15; jj++)

{

ebulet[jj].setTexture(ebl);

}

int ebullet\_no = 0;

//time handling each one is shot after a certain time

sf::Clock enemy\_b;

sf::Time lastb;

enemy\_b.restart();

//experiment end

//helicopter variables

float y\_shift, h\_angle = 0, pi;

pi = std::acos(-1);

//view (scrolling)

//adding gravity

int gravity = 1;

float jumpspeed = 12, velocity = 0;

int ground = tankY;

//changing speeds and directions when levels change

int level = 1;

sf::Clock lvll2;

sf::Time lvlchange;

bool draw\_change = true, draw\_change3 = true;

int lvlebulspeedx[3] = { -4,-9,-11 };

int lvlebulspeedy[3] = { 7,9,11 };

int lvlebultime[3] = { 1100,600, 600 };

int lvlcngscore = 30, lvlcngscore3 = 60;

//file ops (high score)

FILE \*file;

int scoreF;

int score\_flag;

while (window.isOpen())

{

sf::Event event;

while (window.pollEvent(event))

{

if (event.type == sf::Event::Closed)

window.close();

else if (sf::Event::KeyPressed)

{

if (event.key.code == sf::Keyboard::Escape)

{

window.close();

}

else if (scene == 2 && event.key.code == sf::Keyboard::Space)

{

{

shoot++;

}

}

}

}

if (shoot == 2)

{

// shoot

if (level == 1)

{

tankpos = tank.getPosition();

bulet[bullet\_no++].setPosition(tankpos.x + 105, tankpos.y - 12);

fire\_sound.play();

shoot = 0; ////bullet has been already shot, resetting the variable

}

if (level == 2 || level == 3)

{

helilv2pos = helilv2.getPosition();

bulet[bullet\_no++].setPosition(helilv2pos.x + 155, helilv2pos.y - 12);

fire\_sound.play();

shoot = 0; //bullet has been already shot, resetting the variable

}

}

elapsed = cclock.getElapsedTime();

//scene 0 will end after start\_hold seconds

if (scene == 0 && elapsed.asSeconds()>start\_hold) {

scene = 1;

}

if (scene == 0)

{

//control part

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Return))

{

scene = 1;

music.stop();

}

//drawing part

window.clear();

window.draw(start\_scr);

window.draw(title);

title.move(0, -1);

window.draw(skip\_command);

window.draw(skip\_command2);

window.display();

}

if (scene == 1)

{

int lvlcngscore = 30, lvlcngscore3 = 60;

blifex = 600;//bossdie=0;

//setting the title position

title.setPosition(50, 50);

score = 0; //resetting score

//resetting level

level = 1;

draw\_change = true;

//animating the explosion

driver++; //variable is used for animating the explosion

if (driver>64)

{

driver = 0;

}

explo.setPosition(500, 300);

sf::IntRect ex\_rect(ex\_l\*driver, ex\_t, ex\_w, ex\_h);

explo.setTextureRect(ex\_rect);

/\*control part \*/

//using the mouse input for selection using live input(not events)

if (sf::Mouse::isButtonPressed(sf::Mouse::Left))

{

localpos = sf::Mouse::getPosition(window);

//closing window when exit is clicked

if (localpos.x> rectPosx + 85 && localpos.x< rectPosx + 385 && localpos.y> rectPosy + 210 && localpos.y < rectPosy + 260)

{

window.close();

}

//selecting start

else if (localpos.x> rectPosx + 85 && localpos.x< rectPosx + 385 && localpos.y> rectPosy && localpos.y < rectPosy + 50)

{

scene = 10;

heli.setPosition(heliX, 100);

}

//selecting help

else if (localpos.x> rectPosx + 85 && localpos.x< rectPosx + 385 && localpos.y> rectPosy + 70 && localpos.y < rectPosy + 120)

{

//action

scene = 4;

}

else if (localpos.x> rectPosx + 85 && localpos.x< rectPosx + 385 && localpos.y> rectPosy + 140 && localpos.y < rectPosy + 190)

{

//action

scene = 5;

}

}

//setting back to regular

rect1.setFillColor(sf::Color::White);

rect2.setFillColor(sf::Color::White);

rect3.setFillColor(sf::Color::White);

rect4.setFillColor(sf::Color::White);

//highlighting when mouse pointer is on a particular point

localpos = sf::Mouse::getPosition();

rectPosy += 85;

rectPosx += 85;

if (localpos.x> rectPosx + 85 && localpos.x< rectPosx + 385 && localpos.y> rectPosy + 210 && localpos.y < rectPosy + 260)

{

rect4.setFillColor(sf::Color::Yellow);

std::cout << localpos.y << std::endl;

}

//selecting start

else if (localpos.x> rectPosx + 85 && localpos.x< rectPosx + 385 && localpos.y> rectPosy && localpos.y < rectPosy + 50)

{

rect1.setFillColor(sf::Color::Yellow);

}

else if (localpos.x> rectPosx + 85 && localpos.x< rectPosx + 385 && localpos.y> rectPosy + 70 && localpos.y < rectPosy + 120)

{

rect2.setFillColor(sf::Color::Yellow);

}

else if (localpos.x> rectPosx + 85 && localpos.x< rectPosx + 385 && localpos.y> rectPosy + 140 && localpos.y < rectPosy + 190)

{

rect3.setFillColor(sf::Color::Yellow);

}

rectPosy -= 85;

rectPosx -= 85;

//drawing part

window.clear();

window.draw(background1);

window.draw(title);

window.draw(rect1);

window.draw(rect2);

window.draw(rect3);

window.draw(rect4);

window.draw(start);

window.draw(help);

window.draw(about);

window.draw(exit);

window.draw(explo);

window.display();

}

if (scene == 2)

{

if (level == 1)

{

//handing live keyboard input

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left))

{

// left key is pressed: move our character

tank.move(-10, 0);

tankpos = tank.getPosition();

if (tankpos.x<tankX) //tank position fixing

{

tank.setPosition(tankX, tankY);

}

//std::cout<<tankpos.x <<std::endl;

}

else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right))

{

tank.move(8, 0);

}

}

//changing character

else if (level == 2 || level == 3)

{

//handing live keyboard input

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left))

{

// left key is pressed: move our character

helilv2.move(-10, 0);

helilv2pos = helilv2.getPosition();

if (helilv2pos.x<helilv2X) //tank position fixing

{

helilv2.setPosition(helilv2X, helilv2Y);

}

//std::cout<<helilv2pos.x <<std::endl;

}

else if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right))

{

helilv2.move(8, 0);

}

}

helilv2pos = helilv2.getPosition();

if (helilv2pos.x>1100)

{

helilv2.setPosition(tankX, tankY);

}

if (score == lvlcngscore && draw\_change)

{

lvll2.restart();

lvlchange = lvll2.getElapsedTime();

Level2.setString("You Reached Level 2");

Level2.setPosition(200, 550);

heli.setPosition(heliX, 100);

while (lvlchange.asMilliseconds()<2000)

{

window.clear();

window.draw(background2);

window.draw(Level2);

Level2.move(0, -4);

window.display();

lvlchange = lvll2.getElapsedTime();

}

level = 2;

draw\_change = false;

}

if (score>lvlcngscore && score < lvlcngscore3)

{

draw\_change = true;

level = 2;

}

//condition for entering level 3

if (score == lvlcngscore3 && draw\_change3)

{

lvll2.restart();

lvlchange = lvll2.getElapsedTime();

Level2.setPosition(200, 550);

Level2.setString("You reached Level 3");

boss.setPosition(bossX, 100);

blifex = 600;

while (lvlchange.asMilliseconds()<2000)

{

window.clear();

window.draw(back3);

window.draw(Level2);

Level2.move(0, -4);

window.display();

lvlchange = lvll2.getElapsedTime();

}

level = 3;

draw\_change3 = false;

}

if (score>lvlcngscore3)

{

draw\_change3 = true;

level = 3;

}

if (level == 2 || level == 3)

{

//adding gravity

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up))

{

velocity = -1 \* jumpspeed;

}

//control with gravity

helilv2.move(0, velocity);

if (helilv2.getPosition().y < ground)

{

velocity += gravity;

}

else

{

helilv2.setPosition(helilv2.getPosition().x, helilv2Y);

velocity = 0;

}

}

//game over conditions

if (lives<1)

{

bullet\_no = 0;

lives = 3;

scene = 3;

over\_reason = 10;

}

if (bullet\_no>49)

{

bullet\_no = 0;

lives = 3;

scene = 3;

over\_reason = 2;

}

helipos = heli.getPosition();

//resetting position when helicopter is out of the window

if (helipos.x<0 || helipos.x>1300)

{

heli.setPosition(heliX, 100);

lives--;

}

bosspos = boss.getPosition();

//resetting position when boss is out of the window

if (bosspos.x<0 || bosspos.x>1300)

{

boss.setPosition(bossX, 100);

lives--;

}

if (level == 2 || level == 3)

{

//helicopter movement

y\_shift = std::sin(1.2\*h\_angle\*pi / 180);

/\*angle loop \*/

if (h\_angle>360)

{

h\_angle = 0;

}

h\_angle++;

//std::cout<<y\_shift <<std::endl;

}

else y\_shift = 0;

if (level<3) heli.move(-3, y\_shift);

else boss.move(-3, y\_shift);

//collision detection for bullet and enemy(bounding box)

temp = bullet\_no - 1;

while (temp >= 0)

{

if (level<3){ if(bulet[temp].getGlobalBounds().intersects(heli.getGlobalBounds()))

{

//std::cout << "The sprite have collided" << std::endl;

helipos = heli.getPosition();

explo.setPosition(helipos);

heli.setPosition(heliX + 170, 100);

bulet[temp].setPosition(-1, -5);

score += 10 \* level;

explo\_sound.play();

draw\_explo = true;

driver = 5;

}

}

else {

if(bulet[temp].getGlobalBounds().intersects(boss.getGlobalBounds()))

{

//std::cout << "The sprite have collided" << std::endl;

blifex -= 120;

blife.setSize(sf::Vector2f(blifex, 30));

score += 10 \* level;

bosspos = boss.getPosition();

explo.setPosition(bosspos);

// boss.setPosition(bossX+170, 100);

bulet[temp].setPosition(-1, -5);

explo\_sound.play();

draw\_explo = true;

driver = 5;

if (blifex <0) {

over\_reason = 11;

scene = 3; reason2.setString(" You Win!!");

}

}

}

temp--;

}

if (level <3)

{

int etemp = 14;

//experimental

lastb = enemy\_b.getElapsedTime();

if (lastb.asMilliseconds()>lvlebultime[level - 1])

{

ebulet[ebullet\_no++].setPosition(helipos.x + 60, helipos.y + 50);

enemy\_b.restart();

}

if (ebullet\_no>14)

{

ebullet\_no = 0;

}

}

else if (level == 3)

{

int etemp = 14;

sf::Vector2f bosspos(boss.getPosition());

lastb = enemy\_b.getElapsedTime();

if (lastb.asMilliseconds()>lvlebultime[level - 1])

{

ebulet[ebullet\_no++].setPosition(bosspos.x + 60, bosspos.y + 50);

enemy\_b.restart();

}

if (ebullet\_no>14)

{

ebullet\_no = 0;

}

}

//enemy bullet collision with User sprite

int etemp = 14;

while (etemp >= 0)

{

if (level == 1)

{

if (ebulet[etemp].getGlobalBounds().intersects(tank.getGlobalBounds()))

{

//std::cout << "The sprite have collided" << std::endl;

//helipos = heli.getPosition();

tankpos = tank.getPosition();

explo2.setPosition(tankpos);

explo\_sound.play();

tank.setPosition(1200, 900);

//tank.setPosition(tankX, tankY);

ebulet[etemp].setPosition(-2, -3);

lives--;

draw\_explo2 = true;

driver2 = 10;

}

}

else if (level == 2 || level == 3)

{

if (ebulet[etemp].getGlobalBounds().intersects(helilv2.getGlobalBounds()))

{

//std::cout <<\SFML-2.4.1\includet << "The //sprite have collided" << std::endl;

//helipos = heli.getPosition();

helilv2pos = helilv2.getPosition();

explo2.setPosition(helilv2pos);

explo\_sound.play();

helilv2.setPosition(1200, 900);

//tank.setPosition(tankX, tankY);

ebulet[etemp].setPosition(-2, -3);

lives--;

draw\_explo2 = true;

driver2 = 10;

}

}

etemp--;

}

//animating the explosion

if (draw\_explo)

{

driver++; //variable is used for animating the explosion

if (driver>45)

{

draw\_explo = false;

}

}

sf::IntRect ex\_rect(ex\_l\*driver, ex\_t, ex\_w, ex\_h);

explo.setTextureRect(ex\_rect);

if (draw\_explo2)

{

driver2++; //variable is used for animating the explosion

if (driver2>40)

{

draw\_explo2 = false;

if (level == 1)

{

tank.setPosition(tankX, tankY);

}

else if (level == 2)

{

helilv2.setPosition(helilv2X, helilv2Y);

}

}

}

sf::IntRect ex\_rect2(ex\_l2\*driver2, ex\_t2, ex\_w2, ex\_h2);

explo2.setTextureRect(ex\_rect2);

//std::cout << helipos.x <<" " << helipos.y <<std::endl;

//sprite animation for helicopter

heliD++;

if (heliD>2)

{

heliD = 0;

}

sf::IntRect pos2(205 \* heliD, 0, 205, 119);

heli.setTextureRect(pos2);

std::stringstream ss;

ss << score;

std::string scr = ss.str();

Score.setString("Score is " + scr);

std::stringstream sv;

sv << lives;

std::string str = sv.str();

Life.setString("Remaining lives " + str);

/\*Drawing part\*/

window.clear();

if (level == 1)

{

window.draw(background2);

window.draw(tank);

}

else if (level == 2)

{

window.draw(bglv2);

window.draw(helilv2);

}

else if (level == 3)

{

window.draw(back3);

window.draw(helilv2);

window.draw(blife);

window.draw(lifeb);

}

if (level<3) window.draw(heli);

else window.draw(boss);

//drawing bullet

temp = bullet\_no - 1;

while (temp >= 0)

{

window.draw(bulet[temp]);

bulet[temp--].move(3.5, -7.5);

}

//enemy bullet collition with player check

etemp = 14;

while (etemp >= 0)

{

window.draw(ebulet[etemp]);

ebulet[etemp--].move(lvlebulspeedx[level - 1], lvlebulspeedy[level - 1]);

}

//collision checking end

if (draw\_explo)

{

window.draw(explo);

}

if (draw\_explo2)

{

window.draw(explo2);

}

window.draw(Score);

window.draw(Life);

window.display();

}

if (scene == 3)

{

//control part

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Return))

{

scene = 1;

}

file = fopen("scores.txt", "r");

fscanf(file, "%d", &scoreF);

if (scoreF <= score)

{

fclose(file);

file = fopen("scores.txt", "w");

fprintf(file, "%d", score);

fclose(file);

}

else

{

std::stringstream ss;

ss << scoreF;

std::string scr = ss.str();

Hscr.setString("Current High Score: " + scr);

score\_flag = 10;

fclose(file);

}

std::stringstream ss;

ss << score;

std::string scr = ss.str();

High\_score.setString("SCORE: " + scr);

if (score\_flag != 10) Hscr.setString("Current High Score: " + scr);

//initialization

level = 1;

lvlcngscore = 50;

lvlcngscore3 = 50 + 20 \* 5;

//drawing part

window.clear();

window.draw(bg\_over);

window.draw(skip\_command);

window.draw(skip\_command2);

if (over\_reason == 10)

{

window.draw(reason1);

}

else

{

window.draw(reason2);

}

window.draw(High\_score);

window.draw(Hscr);

window.display();

}

if (scene == 4) //Help page

{

window.clear();

window.draw(helpbg);

window.display();

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Return))

{

scene = 1;

}

}

if (scene == 5) //about page

{

window.clear();

window.draw(about1);

window.display();

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Return))

{

scene = 1;

}

}

if (scene == 10) //about page

{

/\*control part \*/

//using the mouse input for selection using live input(not events)

if (sf::Mouse::isButtonPressed(sf::Mouse::Left))

{

localpos = sf::Mouse::getPosition(window);

//std::cout<<localpos.x <<" " <<localpos.y<<std::endl;

if (localpos.x> 534 && localpos.x< 725)

{

if (localpos.y> 173 && localpos.y < 286)

{

scene = 2;

level = 1;

}

else if (localpos.y> 305 && localpos.y < 419)

{

//level = 2;

scene = 2;

lvlcngscore = 0;

//draw\_change = false;

}

else if (localpos.y> 435 && localpos.y < 548)

{

level = 3;

scene = 2;

lvlcngscore3 = 0;

lvlcngscore = 3000;

}

}

}

window.clear();

window.draw(lvchoice);

window.display();

}

}

return 0;

}

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Overview and conclusion:**

The project was a very good way to encourage young students to develop their skills and make something useful with their knowledge. This project is a perfect example of all the exceptional things that can be done with some basic C/C++ programming.

This project will also encourage young minds who want to be a game developer in the future. The project promotes originality and creativity and the skill of working in a team.

The project also summarizes the basic tools of C/C++ language as they had to be implemented while making the game.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~