University of Dhaka

Department of Computer Science and Engineering

CSE – 1211

LAB PROJECT: Zombie Crush

Submitted to:

Hasnain Heickal
Lecturer

Md. Shiplu Hawlader
Lecturer

Submitted by:

Mashrur Rashik, Roll: 29
Mahmudur Rahman, Roll: 89

Date-of-Submission: November 24, 2016

Table of Contents

Introduction:

The objective of this project was to stimulate a graphics based game project based on our excitement of knowing about zombies. We wanted to construct this game as a means of fun, side by side learning the methods of a C++ graphics library. The graphical interface we used was SFML, or formally known as simple fast multimedia library. We tried to develop a game which would be simultaneously interesting and challenging for the player.

Project Name:

Zombie Crush

Overview:

The game is made up of a combination of sprite used to show different characters of the game. Here a number of zombies will appear and the player must crush as much zombie as possible. The more the zombies a player gets to crush, the more is the score. The level of the game is determined by the number of zombies appearing and the speed of the zombie. But if a player misses a sum total of five zombies then the game ends.

Resources:

The entire program was coded in C++.

The graphical interface used is Simple fast multimedia library.

We used the website http://www.sfml-dev.org/tutorials/2.4/ as a means of SFML resource.

Besides we gathered a number of sprites for displaying the objects.

Components of our program:

This was one of the most interesting project for nus so far and also the largest code we ever wrote. Now, we will discuss the step by step sequence of our game.

The beginnings:

1. Menu: The menu consists of a number of windows, which added a large number of options for the user as, new game, Scoreboard, Credits and Exit. This made the game much for interactive and interesting for us.

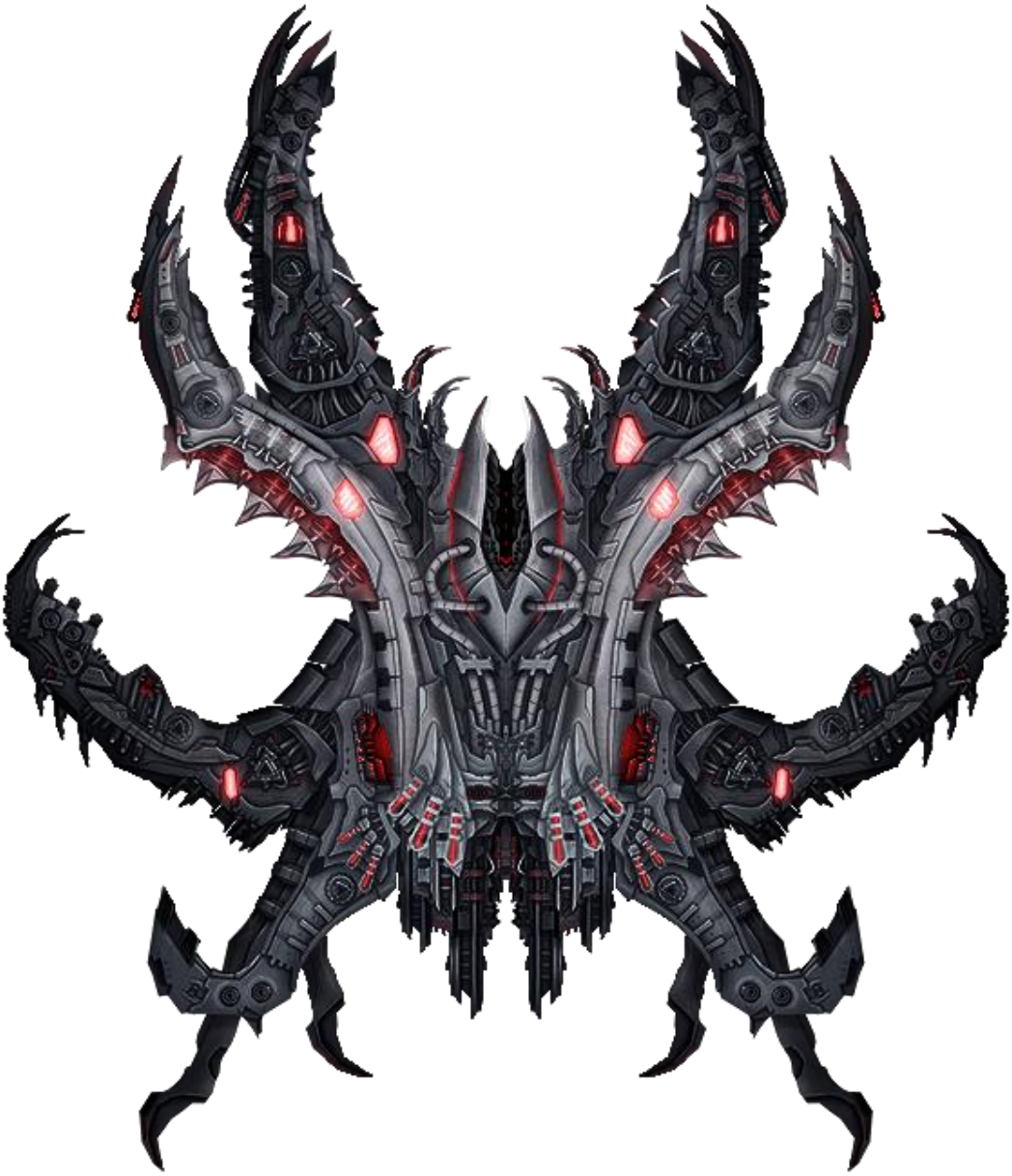
2. Player Name: Here we added an option to input the player name and on the basis of this we updated our scoreboard, to make the game more challenging and interesting.

3. Exit Menu: Here we also added an exit menu so that the user gets an option to either return to the main game menu or simply exit.

Overview:

1) Background: Here the background consist of a

straight road with flames on both the sides. Besides there is a life bar on the right side of the screen, which helps the player to be aware of the number of zombies he can miss.



2) The car: The car is a simple alien car which has been loaded in the form of the sprite. The car can be moved in all directions with the help of the arrow keys.

3) The zombies: The zombies appear in an arbitrary order based on some position set in an array. The number of zombies and their speed of appearance vary on the basis of the game level.

User defined functions:

Here we added no user defined functions.

SFML built-in functions:

1. renderwindow
2. window.draw()
3. rectangle
4. loadfromfile
5. Setcolor()
6. Setscale()
7. vector2f()
8. ifstream
9. ofstream

Main Features:

1. Multi-direction car movement. Here there is an option for moving the car in all four directions with the help of arrow keys.

2. Random zombie appearance. Here the zombies will appear randomly on the basis of some positions set in the array.

3. Zombie appearance order. The zombies will appear randomly and their number varies on the basis of level.

4. Increase in life. The life of the player decreases as they miss zombies and with a satisfactory distance there is an option for raising life. The life is shown on the right side with the help of a bar.

5. Rise in difficulty. The difficulty of the game changes with the level, by increasing the car speed and also the number of zombies.

6. Scoreboard. The game also includes a scoreboard which stores the progress of the player and compares it with other players.

Additional Features:

1. Menu: The menu helps the user to switch between windows.

2. Tutorial: A tutorial window has been added at the beginning of the game play which gives the user a clear instruction for playing the game.

Coding Challenge:

we tried hard to make the game as smooth as possible. The main problem we had was setting the zombie position, and later was successful with the help of array.

We also worked hard to set the sound buffer, as playing sound with each collision is quite challenging as the sound plays multiple times, and we had to play it once.

A bigger challenge was to set the level change, as with the change in level the complexity of the game increases.

Algorithm and approach:

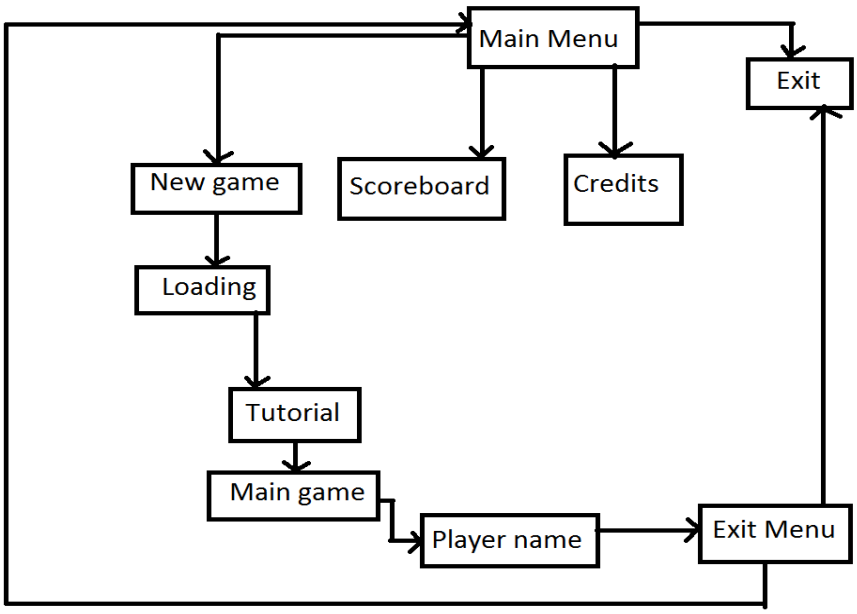
1. Position:
   The positioning of the zombies was done with the help of an array. The car position was changed with the help of instruction from the arrow keys.
   The car was launched from a particular position and changed on the basis of user input. The flames on the sides was changed by simply changing the y-coordinate.

The position of the car is set in a manner to trap the player. This game has been designed in a manner to make the game interesting and challenging for the player.

2. Movement:

   The movement of the car is done with the help of user input. As the player presses one of the arrow key, the car moves in any of the four directions. However, the flame sprite on both sides were moved using modular algorithm. Here we did not use draw and erase technique as it would make the game slower and less efficient

Flowchart:



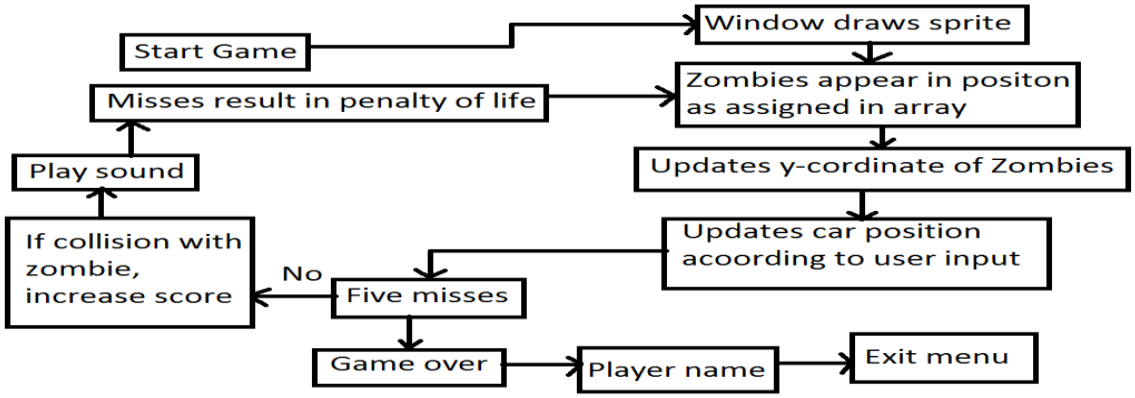


Fig: Flow chart of main game

3. Constructive conditions:

We handled a lot of conditions in the game play which included things like the sound play and for raising score. Here we also applied modular algorithm for changing the flame sprite position and also for changing the zombie position.

4. Collision detection:

For detecting the collision between the car and a zombie we got the global position of the zombie and the car and checked whether they intersect or not, and it is similar to that of two

rectangle collision. This collision detection was in-fact the main challenge in making this game.

Here for checking car collision and for getting the global position of the car we used,

sprite_1.getGlobalbounds().intersects(pic.getGlobalBounds())

Discussion and conclusion:

At first we had high expectation from our game project and started off using BGI, but our requirements were quite high compared to advantages of BGI. So, we decided to to shift to SFML and this was one of the greatest decision made. SFML had a variety of options opened for us and this helped us in making the game in accordance to our requirement.

Source code:

```
#include <SFML/Audio.hpp>

#include <SFML/Window.hpp>

#include <SFML/Graphics.hpp>

#include <iostream>

#include<sstream>

#include<bits/stdc++.h>

using namespace std;

sf::Clock delay;

int key_delay1=200,key_name_delay=150;


struct Point{

int x;

int y;

};


int main(){
```

```
Point en1,en2;

bool start=true,start1=true;

int score=0,x=300,y=400,y1=450,y2=240,y3=30,life=400,ic=10,brick1_y=0,brick2_y=-
500,brick3_y=-1000;

        sf::RenderWindow window(sf::VideoMode(800,600), "Zombie Crush");


sf::Texture
mummy,zm_n1,mouse1,blood,brick,maze,car,f11,f12,f13,f21,f22,f23,zombie1,heart,end1,entry
,entry1,entry2,enemy1,enemy2,enemy3,enemy4,w1,w2;

sf::Sprite
brick_t1,pic,pic1,f_p11,f_p12,f_p13,f_p21,f_p22,f_p23,zombie_t1,heart_t1,end_t1,entry_t1,en
try1_t1,entry2_t1,enemy11,enemy21,enemy31,enemy41;

sf::Sprite mummy1,brick1_t1,brick2_t1,w11,w21,blood1,mouse11,zm_n;


sf::SoundBuffer dead,intro;

sf::Sound sound;



int zm=0,zy1=-5,zy2=-25,zy3=-145,zy4=-85,miss_count=0,zx1,zx2,zx3,zx4;

int
zom[31]={240,466,500,353,450,389,444,500,384,230,498,210,433,500,324,359,390,499,460,31
0,434,383,456,399,400,501,415,245,359,500,290};


// sounds

// dead zombie

if (!dead.loadFromFile("dead2.wav"))
```

```
printf("ERR LOAD IMG\n");

// Intro

sf::Music music,music1,music_back,music_back1,sel;

if (!music.openFromFile("intro1.wav"))

printf("ERR LOAD IMG\n");

if (!music1.openFromFile("intro.wav"))

printf("ERR LOAD IMG\n");

if (!sel.openFromFile("click.wav"))

printf("ERR LOAD IMG\n");


// Background

if (!music_back.openFromFile("back1.wav"))

printf("ERR LOAD IMG\n");

if (!music_back1.openFromFile("back2.wav"))

printf("ERR LOAD IMG\n");


// Mouse Pointer

if(!mouse1.loadFromFile("mouse.png"))

printf("ERR LOAD IMG\n");

mouse11.setTexture(mouse1);

mouse11.setScale(sf::Vector2f(0.21f,0.21f));


// Intro Zombie

if(!zm_n1.loadFromFile("zombienew.png"))
```

```
printf("ERR LOAD IMG\n");
zm_n.setTexture(zm_n1);
zm_n.setScale(sf::Vector2f(0.7f,0.7f));
zm_n.setPosition(sf::Vector2f(10,200));



// Zombie Blood
if(!blood.loadFromFile("bl.png"))
printf("ERR LOAD IMG\n");
blood1.setTexture(blood);
blood1.setScale(sf::Vector2f(0.2f,0.2f));

// side Walls
if(!w1.loadFromFile("blood.jpg"))
printf("ERR LOAD IMG\n");
w1.setSmooth(true);
w11.setTexture(w1);
w11.setScale(sf::Vector2f(0.82f,1.5f));
w11.setPosition(sf::Vector2f(-50,0));

if(!w2.loadFromFile("blood1.jpg"))
printf("ERR LOAD IMG\n");
w2.setSmooth(true);
```

```
w21.setTexture(w2);
w21.setScale(sf::Vector2f(0.82f,1.5f));
w21.setPosition(sf::Vector2f(590,0));


// Zombies
if(!enemy1.loadFromFile("green1.png"))
printf("ERR LOAD IMG\n");
enemy1.setSmooth(true);
enemy11.setTexture(enemy1);
enemy11.setScale(sf::Vector2f(1.3f,1.3f));


if(!enemy2.loadFromFile("green1.png"))
printf("ERR LOAD IMG\n");
enemy2.setSmooth(true);
enemy21.setTexture(enemy2);
enemy21.setScale(sf::Vector2f(1.3f,1.3f));


if(!enemy3.loadFromFile("green2.png"))
printf("ERR LOAD IMG\n");
enemy3.setSmooth(true);
enemy31.setTexture(enemy3);
```

```
enemy31.setScale(sf::Vector2f(1.3f,1.3f));

if(!enemy4.loadFromFile("green2.png"))
printf("ERR LOAD IMG\n");
enemy4.setSmooth(true);
enemy41.setTexture(enemy4);
enemy41.setScale(sf::Vector2f(1.3f,1.3f));


// Starting Menu
if(!entry.loadFromFile("back3.jpg"))
printf("ERR LOAD IMG\n");
entry_t1.setTexture(entry);
entry_t1.setScale(sf::Vector2f(0.43f,0.6f));
entry_t1.setPosition(sf::Vector2f(0,0));
int zinc=0;
if(!entry2.loadFromFile("z.jpg"))
printf("ERR LOAD IMG\n");
entry2_t1.setTexture(entry2);
entry2_t1.setScale(sf::Vector2f(1.0f,1.0f));
entry2_t1.setPosition(sf::Vector2f(zinc,160));

if(!entry1.loadFromFile("loading.png"))
printf("ERR LOAD IMG\n");
```

```
entry1_t1.setTexture(entry1);
entry1_t1.setScale(sf::Vector2f(1.0f,1.3f));
entry1_t1.setPosition(sf::Vector2f(0,0));

sf::Text text_horror;
sf::Font font_h,font_h1,font_h2;
if (!font_h.loadFromFile("youmurdererbb_reg.ttf"))
printf("ERR LOAD IMG\n");
text_horror.setFont(font_h);
if (!font_h1.loadFromFile("bloodlust.ttf"))
printf("ERR LOAD IMG\n");
if (!font_h2.loadFromFile("SnackerComic_PerosnalUseOnly.ttf"))
printf("ERR LOAD IMG\n");


// life meter
sf::RectangleShape rectangle(sf::Vector2f(40,life));
rectangle.setPosition(680,150);
rectangle.setFillColor(sf::Color::Green);


// Kills
string temp;
sf::Text text,tx;
```

```
sf::Font font;
if (!font.loadFromFile("Blox2.ttf"))
printf("ERR LOAD IMG\n");
text.setFont(font);
text.setCharacterSize(20);
text.setColor(sf::Color::Green);
text.setStyle(sf::Text::Bold);

tx.setFont(font);
tx.setCharacterSize(20);
tx.setColor(sf::Color::Red);
tx.setStyle(sf::Text::Regular);


// ENDING
if(!end1.loadFromFile("black_b.jpg"))
printf("ERR LOAD IMG\n");
end_t1.setTexture(end1);
end_t1.setScale(sf::Vector2f(1.0f,1.0f));
end_t1.setPosition(sf::Vector2f(0,0));


// Background Zombie
if(!zombie1.loadFromFile("zombie2.png"))
```

```
printf("ERR LOAD IMG\n");
zombie_t1.setTexture(zombie1);
zombie_t1.setScale(sf::Vector2f(0.53f,0.6f));
zombie_t1.setPosition(sf::Vector2f(-9,270));


// Life Meter
if(!heart.loadFromFile("lives.png"))
printf("ERR LOAD IMG\n");
heart.setSmooth(true);
heart_t1.setTexture(heart);
heart_t1.setScale(sf::Vector2f(0.15f,0.15f));
heart_t1.setPosition(sf::Vector2f(646,33));


// Flames Background
if(!f11.loadFromFile("f11.png"))
printf("ERR LOAD IMG\n");
f_p11.setTexture(f11);
f_p11.setScale(sf::Vector2f(0.5f,0.5f));
f_p12.setTexture(f11);
f_p12.setScale(sf::Vector2f(0.5f,0.5f));
f_p13.setTexture(f11);
f_p13.setScale(sf::Vector2f(0.5f,0.5f));
```

```
if(!f21.loadFromFile("f21.png"))
printf("ERR LOAD IMG\n");
f_p21.setTexture(f21);
f_p21.setScale(sf::Vector2f(0.5f,0.5f));
f_p22.setTexture(f21);
f_p22.setScale(sf::Vector2f(0.5f,0.5f));
f_p23.setTexture(f21);
f_p23.setScale(sf::Vector2f(0.5f,0.5f));


// Background
if(!maze.loadFromFile("back2.jpg"))
printf("ERR LOAD IMG\n");

pic1.setTexture(maze);
pic1.setScale(sf::Vector2f(3.2f, 3.0f));
pic1.setPosition(sf::Vector2f(0,0));


// car
if(!car.loadFromFile("alien2.png"))
printf("ERR LOAD IMG\n");
```

```
pic.setTexture(car);
pic.setScale(sf::Vector2f(0.15f, 0.15f));



string str;
int c=0,pos1=0;
long long int distance=0,level=1;
bool finished=true,credits=false,score_high=false,start_game=false;
bool c1=true;

window.clear();
delay.restart();
while(finished){
c1=true;

// Starting Menu
while(window.isOpen() && finished && !start_game && !score_high && !credits){
sf::Event event;
                while(window.pollEvent(event)){
switch(event.type){
case sf::Event::Closed:
window.close();
break;
}
```

```
}

window.draw(entry_t1);
window.draw(zm_n);

str={"Zombie Crush"};
text_horror.setString(str);
text_horror.setFont(font_h);
text_horror.setCharacterSize(150);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);
text_horror.setPosition(107,10);

window.draw(text_horror);


text_horror.setFont(font_h1);
text_horror.setCharacterSize(60);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);


str={"NEW GAME"};
```

```
text_horror.setString(str);
text_horror.setPosition(310,200);
if(pos1==0){
text_horror.setColor(sf::Color::White);
text_horror.setPosition(280,200);
text_horror.setCharacterSize(90);
}

window.draw(text_horror);
text_horror.setCharacterSize(60);
text_horror.setColor(sf::Color::Red);



str={"SCORE BOARD"};
text_horror.setString(str);
text_horror.setPosition(295,300);
if(pos1==1){
text_horror.setColor(sf::Color::White);
text_horror.setPosition(260,285);
text_horror.setCharacterSize(90);
}

window.draw(text_horror);
text_horror.setCharacterSize(60);
```

```
text_horror.setColor(sf::Color::Red);



str={"CREDITS"};
text_horror.setString(str);
text_horror.setPosition(330,400);
if(pos1==2){
text_horror.setColor(sf::Color::White);
text_horror.setPosition(307,380);
text_horror.setCharacterSize(90);
}

window.draw(text_horror);
text_horror.setCharacterSize(60);
text_horror.setColor(sf::Color::Red);



str={"EXIT"};
text_horror.setString(str);
text_horror.setPosition(360,500);
if(pos1==3){
text_horror.setColor(sf::Color::White);
text_horror.setPosition(344,480);
```

```
text_horror.setCharacterSize(90);
}

window.draw(text_horror);
text_horror.setCharacterSize(60);
text_horror.setColor(sf::Color::Red);



if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up) &&
delay.getElapsedTime().asMilliseconds()>key_name_delay){
sel.play();
pos1=pos1-1;
if(pos1<0) pos1=0;
delay.restart();
}
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down) &&
delay.getElapsedTime().asMilliseconds()>key_name_delay){
sel.play();
pos1=(pos1+1);
if(pos1>3) pos1=3;
delay.restart();
}
```

```
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Return) &&
delay.getElapsedTime().asMilliseconds()>key_delay1){


sel.play();

if(pos1==0){

start_game=true;

delay.restart();

break;

}

else if(pos1==1){

score_high=true;

break;

}

else if(pos1==2){

credits=true;

break;

}

else if(pos1==3){

finished=false;

break;

}



}
```

```
text_horror.setFont(font_h);

text_horror.setCharacterSize(25);

text_horror.setColor(sf::Color::Green);

text_horror.setStyle(sf::Text::Regular);


str={"Zero_639"};

text_horror.setString(str);

text_horror.setPosition(720,565);


window.draw(text_horror);



if(c1) {music.play();c1=false;}

sf::sleep(sf::milliseconds(6));



window.display();


}



music.stop();
```

```
// Bragging a little :v
while(window.isOpen() && finished && !start_game && !score_high && credits){
sf::Event event;
                    while(window.pollEvent(event)){
switch(event.type){
case sf::Event::Closed:
window.close();
break;
}
}

window.draw(entry_t1);



text_horror.setFont(font_h1);
text_horror.setCharacterSize(160);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);

str={"DESIGN & CODE"};
text_horror.setString(str);
text_horror.setPosition(120,6);
```

```
window.draw(text_horror);



text_horror.setFont(font_h1);
text_horror.setCharacterSize(60);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);

str={"MASHRUR RASHIK ( ROLL: 29 )"};
text_horror.setString(str);
text_horror.setPosition(170,200);

window.draw(text_horror);

str={"MD.MAHMUDUR RAHMAN ( ROLL: 89 )"};
text_horror.setString(str);
text_horror.setPosition(130,300);

window.draw(text_horror);



text_horror.setFont(font_h1);
text_horror.setCharacterSize(50);
text_horror.setColor(sf::Color::Red);
```

```
text_horror.setStyle(sf::Text::Regular);

str={"Press <-- To Return"};
text_horror.setString(str);
text_horror.setPosition(250,430);

window.draw(text_horror);



text_horror.setFont(font_h);
text_horror.setCharacterSize(25);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);

str={"Zero_639"};
text_horror.setString(str);
text_horror.setPosition(720,565);

window.draw(text_horror);

if (sf::Keyboard::isKeyPressed(sf::Keyboard::BackSpace)){
sel.play();
window.clear();
credits=false;
```

```
break;
}



window.display();

}

bool file1=true;

// High Scores
while(window.isOpen() && finished && !start_game && score_high && !credits){
sf::Event event;
                while(window.pollEvent(event)){
switch(event.type){
case sf::Event::Closed:
window.close();
break;
}
}

window.draw(entry_t1);
```

```
text_horror.setFont(font_h1);
text_horror.setCharacterSize(140);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);

str={"SCORE BOARD"};
text_horror.setString(str);
text_horror.setPosition(180,6);

window.draw(text_horror);


text_horror.setFont(font_h2);
text_horror.setCharacterSize(70);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);

str={"PLAYER"};
text_horror.setString(str);
text_horror.setPosition(30,166);
window.draw(text_horror);
```

```
text_horror.setFont(font_h2);
text_horror.setCharacterSize(70);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);

str={"KILLS"};
text_horror.setString(str);
text_horror.setPosition(330,166);
window.draw(text_horror);


text_horror.setFont(font_h2);
text_horror.setCharacterSize(70);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);

str={"DISTANCE"};
text_horror.setString(str);
text_horror.setPosition(555,166);
window.draw(text_horror);

text_horror.setFont(font_h);
text_horror.setCharacterSize(25);
text_horror.setColor(sf::Color::Green);
```

```
text_horror.setStyle(sf::Text::Regular);

str={"Zero_639"};
text_horror.setString(str);
text_horror.setPosition(720,565);

window.draw(text_horror);



std::ifstream file("high1.txt");
std::string str_high;
int yx=250;

// Player name
while (std::getline(file,str_high)){
if(str_high.size()==0) continue;
text_horror.setFont(font_h1);
text_horror.setCharacterSize(40);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);
text_horror.setString(str_high);
text_horror.setPosition(30,yx);
yx=yx+40;
window.draw(text_horror);
```

```
}


std::ifstream file2("high2.txt");


yx=250;


while (std::getline(file2, str_high)){
if(str_high.size()==0) continue;
text_horror.setFont(font_h1);
text_horror.setCharacterSize(40);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);

text_horror.setString(str_high);
text_horror.setPosition(370,yx);
yx+=40;
window.draw(text_horror);
}
```

```
std::ifstream file3("high3.txt");



yx=250;



while (std::getline(file3, str_high)){
if(str_high.size()==0) continue;
text_horror.setFont(font_h1);
text_horror.setCharacterSize(40);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);

text_horror.setString(str_high);
text_horror.setPosition(590,yx);
yx+=40;
window.draw(text_horror);
}



text_horror.setFont(font_h1);
text_horror.setCharacterSize(60);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);
```

```
temp="<--";
text_horror.setString(temp);
text_horror.setPosition(700,7);
window.draw(text_horror);



if (sf::Keyboard::isKeyPressed(sf::Keyboard::BackSpace)){
sel.play();
window.clear();
score_high=false;
break;
}




window.display();


}




ic=0;


// Loading Menu
```

```
while(window.isOpen() && finished && start_game && !score_high && !credits && 0){
sf::Event event;
                while(window.pollEvent(event)){
switch(event.type){
case sf::Event::Closed:
window.close();
break;
}
}

window.draw(entry1_t1);

sf::RectangleShape rec(sf::Vector2f(ic,50));
rec.setPosition(150,400);
rec.setFillColor(sf::Color::Green);
window.draw(rec);
ic+=5;

sf::sleep(sf::milliseconds(20));
if (ic>450){
music1.stop();
window.clear();
break;
}
```

```
window.display();


}



// Tutorial
while (window.isOpen() && finished && start_game && !score_high && !credits){
                sf::Event event;
                while(window.pollEvent(event)){
switch(event.type){
case sf::Event::Closed:
window.close();
break;
}
}

window.draw(entry_t1);


text_horror.setFont(font_h1);
text_horror.setCharacterSize(120);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);
```

```
temp="TUTORIAL";
text_horror.setString(temp);
text_horror.setPosition(265,20);

window.draw(text_horror);



text_horror.setFont(font_h1);
text_horror.setCharacterSize(30);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);
temp="Use the arrow keys to control your car";
text_horror.setString(temp);
text_horror.setPosition(200,200);

window.draw(text_horror);

text_horror.setFont(font_h1);
text_horror.setCharacterSize(30);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);
temp="The score increases as you eat more and more zombies";
text_horror.setString(temp);
text_horror.setPosition(150,260);
```

```
window.draw(text_horror);


text_horror.setFont(font_h1);
text_horror.setCharacterSize(30);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);
temp="Make sure you don't miss more than five zombies as it will result in game over!!!";
text_horror.setString(temp);
text_horror.setPosition(30,320);

window.draw(text_horror);



text_horror.setFont(font_h1);
text_horror.setCharacterSize(60);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);
temp="PRESS ENTER TO CONTINUE";
text_horror.setString(temp);
text_horror.setPosition(200,440);

window.draw(text_horror);
```

```
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Return) &&
delay.getElapsedTime().asMilliseconds()>key_delay1){

sel.play();

window.clear();

delay.restart();

break;

}


window.display();

}



// Initialization

sf::sleep(sf::milliseconds(16));

miss_count=0;

zm=0;

c=0;

distance=0;

level=1;

zy1=-5;

zy4=-75;

zy3=-145;
```

```
zy2=-215;
score=0;
long long int z_n1=0,z_n4=0,z_n3=0,z_n2=0;
bool f1=false,f2=false,f3=false,f4=false;
int speed=3;

if(start_game) music1.play();
rectangle.setFillColor(sf::Color::Green);

// Main Game
        while (window.isOpen() && finished && start_game && !score_high && !credits){
                sf::Event event;
                while(window.pollEvent(event)){
switch(event.type){
case sf::Event::Closed:
window.close();
break;
}
}

// delay for Zombie
c++;
```

```
if(distance%20000==0 && distance) music_back.play();
else if(distance%10000==0 && distance) music_back1.play();

// Level
if(distance>=0&&distance<10000) {level=1; speed=3;}
else if(distance>=10000&&distance<50000) {level=2; speed=2;}
else {level=3; speed=1;}



// Flames Position
f_p11.setPosition(sf::Vector2f(148,y1));
f_p12.setPosition(sf::Vector2f(148,y2));
f_p13.setPosition(sf::Vector2f(148,y3));
f_p21.setPosition(sf::Vector2f(580,y1));
f_p22.setPosition(sf::Vector2f(580,y2));
f_p23.setPosition(sf::Vector2f(580,y3));



// car position
pic.setPosition(sf::Vector2f(x,y));
```

```
// draws background
window.draw(pic1);

// Side Walls
window.draw(w11);
window.draw(w21);



// Decoration
window.draw(zombie_t1);

// Life

window.draw(heart_t1);
window.draw(rectangle);

if(miss_count>=3) rectangle.setFillColor(sf::Color::Red);
else rectangle.setFillColor(sf::Color::Green);
rectangle.setSize(sf::Vector2f(50,life));
rectangle.setPosition(680,150+(miss_count*80));
life=400-(80*miss_count);

// Zombies
enemy11.setPosition(sf::Vector2f(zom[zx1],zy1));
```

```
if(level>=2) enemy31.setPosition(sf::Vector2f(zom[zx3],zy3));
enemy41.setPosition(sf::Vector2f(zom[zx4],zy4));
if(distance>=30000) enemy21.setPosition(sf::Vector2f(zom[zx2],zy2));

zm=(zm+1)%31;

if(!f1)
window.draw(enemy11);
if(!f2)
window.draw(enemy41);
if(!f3 && level>=2)
window.draw(enemy31);
if(!f4 && distance>=30000) window.draw(enemy21);

// Zombie Movement
if(c%speed==0){
zy1++;
if(level>=2)
zy3++;
zy4++;
if(distance>=30000) zy2++;
if(life>0) distance++;

if(zy1>529&&zy1<600&&!f1||zy4>529&&zy4<600&&!f2){
```

```
miss_count++;
}
if(zy3>529&&zy3<600&&!f3&&level>=2) miss_count++;
if(zy2>529&&zy2<600&&!f4&&distance>=30000) miss_count++;

c=0;
}

if(zy1>529){
z_n1=0;
f1=false;
zx1=zm;
zy1=-5;
}
if(zy4>529){
z_n4=0;
f2=false;
zx4=(zm+1)%31;
zy4=-75;
}
if(zy3>529 && level>=2){
z_n3=0;
f3=false;
zx3=(zm+2)%31;
```

```
zy3=-145;
}
if(zy2>529 && distance>=30000){
z_n2=0;
f4=false;
zx2=(zm+3)%31;
zy2=-215;
}

// TM
text_horror.setFont(font_h);
text_horror.setCharacterSize(25);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);

str={"Zero_639"};
text_horror.setString(str);
text_horror.setPosition(720,565);

window.draw(text_horror);


// Score
text.setString("KILLS ");
```

```
text.setPosition(15,20);
window.draw(text);
temp=std::to_string(score);
tx.setString(temp);
tx.setPosition(78,20);
window.draw(tx);

text.setString("DISTANCE ");
text.setPosition(5,50);
window.draw(text);
temp=std::to_string(distance);
tx.setString(temp);
tx.setPosition(100,50);
window.draw(tx);


tx.setCharacterSize(40);
tx.setColor(sf::Color::Red);
tx.setStyle(sf::Text::Regular);
text.setString("LEVEL ");
text.setPosition(40,150);
window.draw(text);
temp=std::to_string(level);
tx.setString(temp);
```

```
tx.setPosition(50,180);
window.draw(tx);

tx.setCharacterSize(20);
tx.setColor(sf::Color::Red);
tx.setStyle(sf::Text::Regular);



// THE END
if(life<=0){
// clear all
sound.stop();
music1.stop();
window.clear();
break;
}


// Draws Flames
window.draw(f_p11);
window.draw(f_p12);
window.draw(f_p13);
window.draw(f_p21);
```

```
window.draw(f_p22);

window.draw(f_p23);


// Level Change

if (sf::Keyboard::isKeyPressed(sf::Keyboard::D) &&
delay.getElapsedTime().asMilliseconds()>key_delay1){

level++;

if(level==1) distance=0;

else if(level==2) distance=10001;

else distance=50001;


if(level>3) level=3;

delay.restart();

}

if (sf::Keyboard::isKeyPressed(sf::Keyboard::A) &&
delay.getElapsedTime().asMilliseconds()>key_delay1){

level--;

if(level==1) distance=0;

else if(level==2) distance=10001;

else distance=50001;


if(level<1) level=1;

delay.restart();

}
```

```
// draws car
window.draw(pic);

// steering

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left)) x-=1;
else if(sf::Keyboard::isKeyPressed(sf::Keyboard::Right)) x+=1;
else if(sf::Keyboard::isKeyPressed(sf::Keyboard::Up)) y-=1;
else if(sf::Keyboard::isKeyPressed(sf::Keyboard::Down)) y+=1;
if(x<213) x=213;
if(x>501) x=501;
if(y<7) y=7;
if(y>429) y=429;



// Scores
if(enemy11.getGlobalBounds().intersects(pic.getGlobalBounds())){
f1=true;
z_n1++;
if(z_n1==1){
sound.setBuffer(dead);
sound.play();
```

```
}
if(z_n1==1)
score+=1;
}

if(enemy41.getGlobalBounds().intersects(pic.getGlobalBounds())){
f2=true;
z_n4++;
if(z_n4==1){
sound.setBuffer(dead);
sound.play();
}
if(z_n4==1)
score+=1;
}

if(enemy31.getGlobalBounds().intersects(pic.getGlobalBounds()) && level>=2){
f3=true;
z_n3++;
if(z_n3==1){
sound.setBuffer(dead);
sound.play();
}
if(z_n3==1)
```

```
score+=1;
}

if(enemy21.getGlobalBounds().intersects(pic.getGlobalBounds()) && distance>=30000){
f4=true;
z_n2++;
if(z_n2==1){
sound.setBuffer(dead);
sound.play();
}
if(z_n2==1)
score+=1;
}


if(f1 && life>0){
blood1.setPosition(sf::Vector2f(zom[zx1],zy1));
window.draw(blood1);
}
if(f2 && life>0){
blood1.setPosition(sf::Vector2f(zom[zx4],zy4));
window.draw(blood1);
}
if(f3 && life>0 && level>=2){
```

```
blood1.setPosition(sf::Vector2f(zom[zx3],zy3));
window.draw(blood1);
}
if(f4 && life>0 && distance>=30000){
blood1.setPosition(sf::Vector2f(zom[zx2],zy2));
window.draw(blood1);
}

if(distance%10000==0){
miss_count--;
if(miss_count<0) miss_count=0;
}

// Flames Moving
y1=(y1+2)%529;
y2=(y2+2)%529;
y3=(y3+2)%529;

                window.display();

        }

        sound.stop();
        music_back.stop();
```

```
        music_back1.stop();


pos1=0;


string name;


// PLayer Name
while (window.isOpen() && finished && start_game && !score_high && !credits){
sf::Event event;


                while(window.pollEvent(event)){
switch(event.type){


if (event.type == sf::Event::Closed)
window.close();




}
}


// Entering Name
if(name.size())
```

```
if (sf::Keyboard::isKeyPressed(sf::Keyboard::BackSpace) &&
delay.getElapsedTime().asMilliseconds()>key_name_delay){

name.pop_back();

delay.restart();

}



if (event.type == sf::Event::TextEntered &&
delay.getElapsedTime().asMilliseconds()>key_name_delay){

if (event.text.unicode<128 && name.size()<=15){

name.push_back((char)event.text.unicode);

delay.restart();

}

}



window.draw(end_t1);


// TM

text_horror.setFont(font_h);

text_horror.setCharacterSize(25);

text_horror.setColor(sf::Color::Green);

text_horror.setStyle(sf::Text::Regular);
```

```
str={"Zero_639"};
text_horror.setString(str);
text_horror.setPosition(720,565);

window.draw(text_horror);



text_horror.setFont(font_h1);
text_horror.setCharacterSize(90);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);

str={"ENTER YOUR NAME"};
text_horror.setString(str);
text_horror.setPosition(200,160);
window.draw(text_horror);



sf::RectangleShape r(sf::Vector2f(580,80));
r.setPosition(100,300);
r.setFillColor(sf::Color::Black);
window.draw(r);
```

```
text_horror.setFont(font_h1);

text_horror.setCharacterSize(90);

text_horror.setColor(sf::Color::White);

text_horror.setStyle(sf::Text::Regular);

text_horror.setPosition(200,280);

text_horror.setString(name);


window.draw(text_horror);


if (sf::Keyboard::isKeyPressed(sf::Keyboard::Return) &&
delay.getElapsedTime().asMilliseconds()>key_delay1){


ofstream myfile;

myfile.open ("high1.txt",std::ios::out|std::ios::app);

myfile<<name<<std::endl;

myfile.close();


myfile.open ("high2.txt",std::ios::out|std::ios::app);

temp=to_string(score);

myfile<<temp<<std::endl;

myfile.close();


myfile.open ("high3.txt",std::ios::out|std::ios::app);
```

```
temp=to_string(distance);
myfile<<temp<<std::endl;
myfile.close();
delay.restart();
break;
}



window.display();
}




        // THE END
        while(window.isOpen() && finished && start_game && !score_high && !credits){
sf::Event event;

                while(window.pollEvent(event)){
switch(event.type){
if (event.type == sf::Event::Closed)
window.close();
break;
}
}
```

```
window.draw(end_t1);

// TM
text_horror.setFont(font_h);
text_horror.setCharacterSize(25);
text_horror.setColor(sf::Color::Green);
text_horror.setStyle(sf::Text::Regular);

str={"Zero_639"};
text_horror.setString(str);
text_horror.setPosition(720,565);

window.draw(text_horror);



text_horror.setFont(font_h1);
text_horror.setCharacterSize(200);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);
str={"GAME OVER"};
text_horror.setString(str);
text_horror.setPosition(130,20);
window.draw(text_horror);
```

```
// Results
text.setColor(sf::Color::Green);
text.setString("KILLS ");
text.setPosition(260,270);
window.draw(text);
temp=std::to_string(score);
tx.setString(temp);
tx.setPosition(460,270);
window.draw(tx);

text.setString("DISTANCE ");
text.setPosition(260,300);
window.draw(text);
temp=std::to_string(distance);
tx.setString(temp);
tx.setPosition(460,300);
window.draw(tx);

text.setString("PRESS ENTER TO SELECT");
text.setPosition(270,550);
window.draw(text);
```

```
text_horror.setFont(font_h1);
text_horror.setCharacterSize(50);
text_horror.setColor(sf::Color::Red);
text_horror.setStyle(sf::Text::Regular);

str={"MAIN MENU"};
text_horror.setString(str);
text_horror.setPosition(310,360);
if(pos1==0){
text_horror.setPosition(270,360);
text_horror.setCharacterSize(90);
text_horror.setColor(sf::Color::White);
}
window.draw(text_horror);

text_horror.setCharacterSize(50);
text_horror.setColor(sf::Color::Red);


str={"EXIT"};
text_horror.setString(str);
text_horror.setPosition(360,450);
```

```
if(pos1==1){
text_horror.setPosition(340,410);
text_horror.setCharacterSize(90);
text_horror.setColor(sf::Color::White);
}

window.draw(text_horror);

text_horror.setCharacterSize(50);
text_horror.setColor(sf::Color::Red);

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up) &&
delay.getElapsedTime().asMilliseconds()>key_name_delay){
sel.play();
pos1--;
if(pos1<0) pos1=0;
delay.restart();
}
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down) &&
delay.getElapsedTime().asMilliseconds()>key_name_delay){
sel.play();
pos1++;
if(pos1>1) pos1=1;
```

```
delay.restart();

}



if (sf::Keyboard::isKeyPressed(sf::Keyboard::Return) &&
delay.getElapsedTime().asMilliseconds()>key_delay1){

sel.play();


if(pos1==0){

start_game=false;

delay.restart();

break;

}

else{

finished=false;

delay.restart();

break;

}


}



window.display();
```

```
}

sf::sleep(sf::milliseconds(5));

window.clear();

}



return 0;



}
```