KNOCKOUT GAME

Knockout is a fighting game implemented in C++ with the aid of SFML for graphics and audio. The user will be controlling the mage player and the program will be handling the swordsman player. For both players the controls for movement are the up, down, right, left arrow keys. The attacks can be categorised as basic and ultimate and each player also has a defence mode. The controls for the above are W- Basic attack, A- Ultimate attack and D- Defence mode. The ultimate for the mage is a shooting blaster and the ultimate for the swordsman is a teleport strike. Hint: Run away from the swordsman when he is doing his ultimate attack as he will teleport to your position and the damage of the hit will be critical.

Weekly Sprints:

Week	Achieved as a group	Personal contribution
Week 1	• Separation of concerns by encapsulation. • Paper Prototype of the game. • Structure of the database. • Graphical assets for the game.	• Analysation of different functions needed for the game, and the type of each function. • Normalisation of the database.
Week 2	• Having the game screen setup with basic background graphical assets. • Code for player header file.	• Creating an SFML project template on Visual Studio with include libraries and dependencies. • Creating the game window with movements. • Code for the player header file.
Week 3	• Blaster Header file and collision	• Rotation of player using mouse.

	detection of shooting. • Rotating movement of player.	• Updated main code to include different states of game with the help of an enumerated class.
Week 4	• Automation of player movements. • Collision detection between both players. • Using a codio box server to access and connect the database.	• Updating player class to make sure player stays in the boundaries. • Automation of player movements. • Inputting database details on xampp. • Debugging errors in automation and collision.
Week 5	• Defence mode activation • Setting up database connection on visual studio using MySQL connector for C++.	• Added sound effects. • Refactoring of defence code and integrating it into the mail file and debugging errors that arose. • Debugging errors in linking dependencies for connecting the database.

Scope for further improvement:

- Using various textures to make the sprites move.
- Integrating a time limit on the defence and ultimate attacks.
- Adding a health bar that reduces in size as health decreases.
- Have a moving background.
- Using a HUD view when scrolling view is activated.

- Using a texture class to hold all textures and the assignments to their respective sprites.
- Health and speed can also be upgraded with increase in level.

C++ Code

<Github >
https://github.coventry.ac.uk/lingamr/Game-Project.git

```
/** Libraries needed are:
sfml-graphics-d.lib
sfml-window-d.lib
sfml-system-d.lib
sfml-network-d.lib
sfml-audio-d.lib **/

/** Please find the libraries on https://www.sfml-dev.org/download.php**/

/*********** Player Header File ***********/
#pragma once
#include <iostream>
#include <SFML/Graphics.hpp>

using namespace sf;
using namespace std;

class Swordsman
{

private:
        const float STARTING_SPEED = 600;
        const float STARTING_HEALTH = 100000;

        Vector2f s_Position;

        Sprite s_Sprite;

        Texture s_Texture;

        Vector2f s_Resolution;

        // to track the directions the player is currently moving in
        bool s_UpPressed;
        bool s_DownPressed;
```

```
        bool s_LeftPressed;
        bool s_RightPressed;

        int s_Health;
        int s_MaxHealth;

        // Time of the last hit
        Time s_LastHit;

        // Speed in pixels per second
        float s_Speed;

        float xVelocity = .2f;
        float yVelocity = .2f;

        int s_right = 0;
        int s_left = 0;

        int s_top = 0;
        int s_bottom = 0;

public:

        Swordsman(string filename);

        //to spawn the player at the firs position
        void spawn(Vector2f resolution, int x, int y);

        // track player hits
        bool hit(Time timeHit);

        //Time elapsed since last hit
        Time getLastHitTime();

        // Position of player with respect to the screen coordinates
        FloatRect getGlobalBound();

        //Set position of Player
        void setPosition(Vector2f position);

        // Origin of the player
        Vector2f getCenter();

        // Angle the player is facing
        float getRotation();

        //Getter function to copy the sprite
        Sprite getSprite();

        // The next four functions manipulate the boolean movement variables
        void moveLeft();

        void moveRight();
```

```
        void moveUp();

        void moveDown();

        // Stop the player moving in a specific direction
        void stopLeft();

        void stopRight();

        void stopUp();

        void stopDown();

        // Getter function for health
        int getHealth();

        //Setter function for health
        void setHealth(int num);

        // Update the player each frame
        void update(float elapsedTime, Vector2f resolution);


         // Update for a player with rotating characteristics
        void update(float elapsedTime, Vector2i mousePosition, Vector2f resolution);

         //function to implement teleport
        void ultimate(Swordsman& other);

       //four functions to get four boundaries of player
        void getRight();

        void getLeft();

        void getTop();

        void getBottom();

        //function to reset stats
        void resetPlayerStats();

};

Swordsman::Swordsman(string filename)
{
        s_Speed = STARTING_SPEED;
        s_Health = STARTING_HEALTH;
        s_MaxHealth = STARTING_HEALTH;

        // Associate a texture with the sprite
        // !!Watch this space!!
        s_Texture.loadFromFile(filename);
        s_Sprite.setTexture(s_Texture);
```

```
        s_right = s_Sprite.getPosition().x + s_Sprite.getGlobalBounds().width;
        s_left = s_Sprite.getPosition().x;

        s_top = s_Sprite.getPosition().y;
        s_bottom = s_Sprite.getPosition().y + s_Sprite.getGlobalBounds().height;


        // Set the origin of the sprite to the centre,
        // for smooth rotation
        s_Sprite.setOrigin(25, 25);
}

void Swordsman::spawn(Vector2f resolution, int x, int y)
{
        // Place the player in the middle of the arena
        s_Position.x = resolution.x / 2;
        s_Position.y = resolution.y / 2;

        // Strore the resolution for future use
        s_Resolution.x = resolution.x;
        s_Resolution.y = resolution.y;

        s_Sprite.setPosition(x, y);

}

Time Swordsman::getLastHitTime()
{
        return s_LastHit;
}

FloatRect Swordsman::getGlobalBound()
{
        return s_Sprite.getGlobalBounds();
}

void Swordsman::setPosition(Vector2f position)
{
        s_Sprite.setPosition(position);
}

Vector2f Swordsman::getCenter()
{
        return s_Position;
}

float Swordsman::getRotation()
{
        return s_Sprite.getRotation();
}

Sprite Swordsman::getSprite()
{
        return s_Sprite;
```

```
}


void Swordsman::moveLeft()
{
	s_LeftPressed = true;
}

void Swordsman::moveRight()
{
	s_RightPressed = true;
}

void Swordsman::moveUp()
{
	s_UpPressed = true;
}

void Swordsman::moveDown()
{
	s_DownPressed = true;
}

void Swordsman::stopLeft()
{
	s_LeftPressed = false;
}

void Swordsman::stopRight()
{
	s_RightPressed = false;
}

void Swordsman::stopUp()
{
	s_UpPressed = false;
}

void Swordsman::stopDown()
{
	s_DownPressed = false;
}

int Swordsman::getHealth()
{
	return s_Health;
}


void Swordsman::setHealth(int num)
{
	s_Health = s_Health - num;
}
```

```
void Swordsman::update(float elapsedTime, Vector2f resolution)
{

	if (s_UpPressed)
	{
		s_Position.y -= s_Speed * elapsedTime;
	}

	if (s_DownPressed)
	{
		s_Position.y += s_Speed * elapsedTime;
	}

	if (s_RightPressed)
	{
		s_Position.x += s_Speed * elapsedTime;
	}

	if (s_LeftPressed)
	{
		s_Position.x -= s_Speed * elapsedTime;
	}


	// Make sure the player does not go out of screen coordinates
	if (s_Position.x >= 1920 - 150 || s_Position.x <= 0)
	{
		s_Position.x = resolution.x / 2;
	}

	if (s_Position.y >= 1080 - 240 || s_Position.y <= 0)
	{
		s_Position.y = resolution.y / 2;
	}
	s_Sprite.setPosition(s_Position);
}
```

‘’ ’

The code given below has been written with the help of the tutorial:
https://www.linkedin.com/learning/c-plus-plus-game-programming-1/
and has been modified to this project

‘’ ’

```
void Swordsman::update(float elapsedTime, Vector2i mousePosition, Vector2f resolution)
{
	if (s_UpPressed)
	{
		s_Position.y -= s_Speed * elapsedTime;
	}
```

```
	if (s_DownPressed)
	{
		s_Position.y += s_Speed * elapsedTime;
	}

	if (s_RightPressed)
	{
		s_Position.x += s_Speed * elapsedTime;
	}

	if (s_LeftPressed)
	{
		s_Position.x -= s_Speed * elapsedTime;
	}

	// Modifications in making sure the player is in the boundary has been made
	// Make sure the player does not go out of screen coordinates
	if (s_Position.x >= 1920 - 150 || s_Position.x <= 0)
	{
		s_Position.x = s_Position.x / 2;
	}


	if (s_Position.y >= 1080 - 240 || s_Position.y <= 0)
	{
		s_Position.y = s_Position.y / 2;
	}
	s_Sprite.setPosition(s_Position);

	/*if (s_left = 0 || s_right > 1920)
	{
		s_Position.x = resolution.x / 2;
	}

	if (s_top = 0 || s_bottom > 1080)
	{
		s_Position.y = resolution.y / 2;
	}*/

	s_Sprite.setPosition(s_Position);

	// Calculate the angle the player is facing
	double angle = (atan2(mousePosition.y - s_Resolution.y / 2, mousePosition.x - 
s_Resolution.x / 2) * 180) / 3.141;

	s_Sprite.setRotation(angle);
}

bool Swordsman::hit(Time timeHit)
{
	if (timeHit.asMilliseconds() - s_LastHit.asMilliseconds() > 200)
	{
		s_LastHit = timeHit;
		return true;
```

```
        }
        else
        {
                return false;
        }

}
```

‘’ ’

This end the code which has been written with the help of the tutorial:
https://www.linkedin.com/learning/c-plus-plus-game-programming-1/
and has been modified to this project

‘’ ’

```
void Swordsman::ultimate(Swordsman& other)
{
        Vector2f p1 = getCenter();
        other.s_Position.x = p1.x + 100;
        other.s_Position.y = p1.y;
        other.s_Sprite.setPosition(other.s_Position);
}
```

```
void Swordsman::getRight()
{
        s_right= s_Sprite.getPosition().x + s_Sprite.getGlobalBounds().width;
}

void Swordsman::getLeft()
{
        s_left= s_Sprite.getPosition().x;
}

void Swordsman::getTop()
{
        s_top= s_Sprite.getPosition().y;
}

void Swordsman::getBottom()
{
        s_bottom=s_Sprite.getPosition().y + s_Sprite.getGlobalBounds().height;
}
void Swordsman::resetPlayerStats()
{
        s_Speed = STARTING_SPEED;
        s_Health = STARTING_HEALTH;
        s_MaxHealth = STARTING_HEALTH;
}
```

```
/********** Main CPP File Code ***********/


#include <iostream>
#include <sstream>
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>
#include "Player.h" //Player class to implement characteristics
#include "Blaster.h" //Blaster class to shoot blasters
#include <vector>

using namespace sf;
using namespace std;

int main()
{
	// Screen will be updated based on which of the states the game is in
	enum class State { PAUSED, GAME_OVER, PLAYING };

	// Starting with the GAME_OVER state
	State state = State::GAME_OVER;

	// Set the screen resolution by 1920x1080
	Vector2f resolution;
	resolution.x = 1920;
	resolution.y = 1080;

	// Create a video mode object
	VideoMode vm(1920, 1080);

	//Antialiasing smooths the textures and shapes. Depends on compatibility with
      graphics card
	ContextSettings settings;
	settings.antialiasingLevel = 8;

	// Create and open a window for the game
	RenderWindow window(vm, "Knockout", Style::Fullscreen, settings);

	// Creates a scrolling SFML View for the main action
	//View mainView(sf::FloatRect(0, 0, resolution.x, resolution.y));

	// Clock class variable to track game time
	Clock clock;

	// tracks time in which  the PLAYING state been active
	Time gameTimeTotal;


	// Where is the mouse in relation to world coordinates
	Vector2f mouseWorldPosition;

	// Where is the mouse in relation to screen coordinates
	Vector2i mouseScreenPosition;
```

```
// Creates a texture for the background
Texture textureBackground;

// Load a graphic into the texture
textureBackground.loadFromFile("graphics/background.png");

// Creates a sprite for the background
Sprite spriteBackground;

// Attachs the texture to the sprite
spriteBackground.setTexture(textureBackground);

// Set the spriteBackground to cover the screen
spriteBackground.setPosition(0, 0);

// Assiging and creating the bat sprite
Texture textureBat;
textureBat.loadFromFile("graphics/bats6.png");
Sprite spriteBat;
spriteBat.setTexture(textureBat);
spriteBat.setPosition(0, 800);

// To track bat movement
bool batActive = false;

// Set batspeed
float batSpeed = 0.0f;

// Texture for the clouds
Texture textureCloud;

// Load texture
textureCloud.loadFromFile("graphics/cloud2.png");
```

‘’ ’

The code given below has been written with the help of the tutorial:
https://www.linkedin.com/learning/c-plus-plus-game-programming-1/
And
https://www.sfml-dev.org/documentation/2.5.1/

‘’ ’

```
const int NUM_CLOUDS = 6;
Sprite clouds[NUM_CLOUDS];
int cloudSpeeds[NUM_CLOUDS];
bool cloudsActive[NUM_CLOUDS];

for (int i = 0; i < NUM_CLOUDS; i++)
{
```

```
            clouds[i].setTexture(textureCloud);
            clouds[i].setPosition(-300, i * 150);
            cloudsActive[i] = false;
            cloudSpeeds[i] = 0;

        }

        // Health bar
        RectangleShape healthBar;
        float healthBarStartingWidth = 400;
        float healthBarHeight = 80;
        healthBar.setSize(Vector2f(healthBarStartingWidth, healthBarHeight));
        healthBar.setFillColor(Color::Red);
        healthBar.setPosition((420 / 2) - healthBarStartingWidth / 2, 120);

        float healthRemaining = 6.0f;
        float healthBarWidthPerSecond = healthBarStartingWidth / healthRemaining;

        //second health bar
        RectangleShape healthBar2;
        float healthBarStartingWidth2 = 400;
        float healthBarHeight2 = 80;
        healthBar2.setSize(Vector2f(healthBarStartingWidth2, healthBarHeight2));
        healthBar2.setFillColor(Color::Red);
        healthBar2.setPosition((1700) - healthBarStartingWidth2 / 2, 120);


        float healthRemaining2 = 6.0f;
        float healthBarWidthPerSecond2 = healthBarStartingWidth2 / healthRemaining2;
```

‘’ ’

This end the code written with the help of the tutorial:

https://www.linkedin.com/learning/c-plus-plus-game-programming-1/

And

https://www.sfml-dev.org/documentation/2.5.1/

‘’ ’

```
// Manipulating text on the screen
    Text pauseText;
    Text healthText;
    Text healthText2;
    Text gameoverText;

    Font font;
    font.loadFromFile("fonts/Montague.ttf");

    // Set the font to our message
    pauseText.setFont(font);
```

```
healthText.setFont(font);
healthText2.setFont(font);
gameoverText.setFont(font);


// Assign the actual message
pauseText.setString("Press Enter to start!");
healthText.setString("Health=0");
healthText2.setString("Health=0");


// Set size of text
pauseText.setCharacterSize(80);
healthText.setCharacterSize(60);
healthText2.setCharacterSize(60);
gameoverText.setCharacterSize(80);

// Set color of text
pauseText.setFillColor(Color::White);
healthText.setFillColor(Color::White);
healthText2.setFillColor(Color::White);

// Position the text to the center

FloatRect textRect = pauseText.getLocalBounds();

pauseText.setOrigin(textRect.left +
        textRect.width / 2.0f,
        textRect.top +
        textRect.height / 2.0f);

pauseText.setPosition(1920 / 2.0f, 1080 / 2.0f);

healthText.setPosition(20, 20);
healthText2.setPosition(1500, 20);

// Backgrounds for the text
RectangleShape rect1;
rect1.setFillColor(sf::Color(0, 0, 0, 150));
rect1.setSize(Vector2f(400, 80));
rect1.setPosition(0, 30);

RectangleShape rect2;
rect2.setFillColor(sf::Color(0, 0, 0, 150));
rect2.setSize(Vector2f(400, 80));
rect2.setPosition(1500, 30);

// Prepare the player
/*Texture texturePlayer;
texturePlayer.loadFromFile("graphics/player4.png");
Sprite spritePlayer;
spritePlayer.setTexture(texturePlayer);
spritePlayer.setPosition(1920 / 2, 600);
```

```
/*Texture texturePlayer2;
texturePlayer2.loadFromFile("graphics/player6.png");
Sprite spritePlayer2;
spritePlayer2.setTexture(texturePlayer2);
spritePlayer2.setPosition(1920 / 4, 600);*/

//Assigning Player texture was transferred to the player class constructor
// Create an instance of the Player class
Swordsman player("graphics/player6.png");
Swordsman player2("graphics/player4.png");
player.spawn(resolution, 1920 / 4, 400);
player2.spawn(resolution, 1920 / 2, 600);

//Variables to track attack type of players
bool base_attack = false;
bool defense = false;
bool ultimate = false;
bool base_attack2 = false;
bool defense2 = false;
bool ultimate2 = false;
int lastDrawn = 0;

SoundBuffer blasterBuffer;
blasterBuffer.loadFromFile("sound/blaster.wav");
Sound blaster;
blaster.setBuffer(blasterBuffer);

SoundBuffer swordBuffer;
swordBuffer.loadFromFile("sound/sword2.wav");
Sound sword;
sword.setBuffer(swordBuffer);

// The main game loop
while (window.isOpen())
{
        /************** Input and Event Handling *************/

        Event event;
        while (window.pollEvent(event))
        {
                if (event.type == Event::KeyPressed)
                {
                        // Pause a game while playing
                        if (event.key.code == Keyboard::Return &&
                                state == State::PLAYING)
                        {
                                state = State::PAUSED;
                        }

                        // Restart while paused
                        else if (event.key.code == Keyboard::Return &&
                                state == State::PAUSED)
                        {
                                state = State::PLAYING;
```

```
                        // Reset the clock so there isn't a frame jump
                        clock.restart();
                }

                // Start a new game while in GAME_OVER state
                else if (event.key.code == Keyboard::Return &&
                        state == State::GAME_OVER)
                {
                        state = State::PLAYING;
                }

                if (state == State::PLAYING)
                {
                }

        }
}

// Closing the window
if (Keyboard::isKeyPressed(Keyboard::Escape))
{
        window.close();
}

// Handle WASD and movement
if (state == State::PLAYING)
{

        // Handle the pressing and releasing of the WASD keys
        if (Keyboard::isKeyPressed(Keyboard::Up))
        {
                player.moveUp();
                circle.setPosition(20000, 20000);
        }
        else
        {
                player.stopUp();

        }

        if (Keyboard::isKeyPressed(Keyboard::Down))
        {
                player.moveDown();
                circle.setPosition(20000, 20000);
        }
        else
        {
                player.stopDown();
        }

        if (Keyboard::isKeyPressed(Keyboard::Left))
        {
                player.moveLeft();
                circle.setPosition(20000, 20000);
```

```
}
else
{
        player.stopLeft();
}

if (Keyboard::isKeyPressed(Keyboard::Right))
{
        player.moveRight();
        circle.setPosition(20000, 20000);
}
else
{
        player.stopRight();
}
if (Keyboard::isKeyPressed(Keyboard::W))
{
        base_attack = true;
        sword.play();
        circle.setPosition(20000, 20000);
}
if (Keyboard::isKeyPressed(Keyboard::A))
{
        ultimate = true;
        isFiring = true;
        blaster.play();
        circle.setPosition(20000, 20000);
}
if (Keyboard::isKeyPressed(Keyboard::D))
{
        defense = true;
        circle.setPosition(player.getCenter());
}
//Automation of player 2 ,ovements and attcaks
srand((int)time(0));
int r = (rand() % 5);
if (r == 1)
{
        player2.moveUp();
        circle2.setPosition(20000, 20000);
}
else
{
        player2.stopUp();
}
if (r == 2)
{
        player2.moveDown();
        circle2.setPosition(20000, 20000);
}
else
{
        player2.stopDown();
}
```

```
		if (r == 3)
		{
			player2.moveLeft();
			circle2.setPosition(20000, 20000);
		}
		else
		{
			player2.stopLeft();
		}
		if (r == 4)
		{
			player2.moveRight();
			circle2.setPosition(20000, 20000);
		}
		else
		{
			player2.stopRight();
		}
		srand((int)time(0));
		int i = (rand() % 4);
		if (i == 1)
		{
			base_attack2 = true;
			sword.play();
			circle2.setPosition(20000, 20000);
		}
		if (i == 2)
		{
			ultimate2 = true;
			player.ultimate(player2);
			circle2.setPosition(20000, 20000);
		}
		if (i == 3)
		{
			defense2 = true;
			circle2.setPosition(player2.getCenter());
		}
}

/***************** UPDATING THE FRAME *****************/
if (state == State::PLAYING)
{

		// Update the time
		Time dt = clock.restart();
		// Update the total game time
		gameTimeTotal += dt;
		// Caluclate dt in seconds
		float dtAsSeconds = dt.asSeconds();
		// Subtract from the amount of time remaining
		healthRemaining -= player.getHealth();
		healthRemaining2 -= player2.getHealth();
		// size up the time bar
```

```
        //healthBar.setSize(Vector2f(healthBarWidthPerSecond*healthRemaining,
healthBarHeight));

        //healthBar2.setSize(Vector2f(healthBarWidthPerSecond2*healthRemaining2,
healthBarHeight2));
                        healthBar.setSize(Vector2f(player.getHealth(),
healthBarHeight));
                        healthBar2.setSize(Vector2f(player2.getHealth(),
healthBarHeight2));

                        // Automate bat movements
                        if (!batActive)
                        {

                                // How fast is the bat
                                srand((int)time(0) * 10);
                                batSpeed = (rand() % 200) + 200;

                                // How high is the bat
                                srand((int)time(0) * 10);
                                float height = (rand() % 500) + 500;
                                spriteBat.setPosition(2000, height);
                                batActive = true;

                        }
                        else
                        {

                                spriteBat.setPosition(spriteBat.getPosition().x -
(batSpeed * dt.asSeconds()),spriteBat.getPosition().y);

                                // Make sure bat stays in the screen boundaries
                                if (spriteBat.getPosition().x < -100)
                                {
                                        batActive = false;
                                }
                        }
```

‘’ ’

The code given below has been written with the help of the tutorial:

https://www.linkedin.com/learning/c-plus-plus-game-programming-1/

And

https://www.sfml-dev.org/documentation/2.5.1/

‘’ ’

```
                        //Automate cloud movements
                        for (int i = 0; i < NUM_CLOUDS; i++)
                        {
```

```
                                if (!cloudsActive[i])
                                {

                                        srand((int)time(0) * i);
                                        cloudSpeeds[i] = (rand() % 200);

                                        srand((int)time(0) * i);
                                        float height = (rand() % 150);
                                        clouds[i].setPosition(-200, height);
                                        cloudsActive[i] = true;

                                }
                                else
                                {
                                        clouds[i].setPosition(clouds[i].getPosition().x
+(cloudSpeeds[i] * dt.asSeconds()),clouds[i].getPosition().y);

                                // Make sure the cloud stays in the screen boundaries
                                        if (clouds[i].getPosition().x > 1920)
                                        {
                                                cloudsActive[i] = false;
                                        }

                                }

                        }
                        // Where is the mouse pointer
                        mouseScreenPosition = Mouse::getPosition();

                        // Convert mouse position to world coordinates of mainView
                mouseWorldPosition =
window.mapPixelToCoords(Mouse::getPosition());//mainView//);
```

‘’ ’

This end the code written with the help of the tutorial:

https://www.linkedin.com/learning/c-plus-plus-game-programming-1/

And

https://www.sfml-dev.org/documentation/2.5.1/

‘’ ’

```
                        // Update the player
                        player.update(dtAsSeconds, resolution);
                        player2.update(dtAsSeconds, resolution);
                        //player.update(dtAsSeconds, Mouse::getPosition(), resolution);

                        // Make a note of the players new position
                        Vector2f playerPosition(player.getCenter());
```

```
                        // Make the view centre around the player (useful in scrolling
views)
                        //mainView.setCenter(player.getCenter());
                        lastDrawn++;
                }

                //Update score every 100 frames
                if (lastDrawn == 100) {
                        // Update the score text
                        stringstream ss;
                        stringstream ss2;
                        ss << "HEALTH=" << player.getHealth();
                        healthText.setString(ss.str());
                        ss2 << "HEALTH=" << player2.getHealth();
                        healthText2.setString(ss2.str());
                        lastDrawn = 0;
                }
                if ((player.getHealth()) <= 0 || (player2.getHealth()) <= 0)
                {
        // Pause the game
                        state = State::GAME_OVER;
                        // Change the message shown to the player
                        if ((player.getHealth()) <= 0)

                        {
                                stringstream ss3;
                                ss3 << "HEALTH=" << 0;
                                healthText.setString(ss3.str());
                                gameoverText.setString("Game Over!!");

                                //Reposition the text based on its new size
                                FloatRect textRect = gameoverText.getLocalBounds();
                                gameoverText.setOrigin(textRect.left +
                                        textRect.width / 2.0f,
                                        textRect.top +
                                        textRect.height / 2.0f);

                                gameoverText.setPosition(1920 / 2.0f, 1080 / 2.0f);
                        }
                        if ((player2.getHealth()) <= 0)
                        {
                                stringstream ss4;
                                ss4 << "HEALTH=" << 0;
                                healthText2.setString(ss4.str());
                                gameoverText.setString("You win!!");

                                //Reposition the text based on its new size
                                FloatRect textRect = gameoverText.getLocalBounds();
                                gameoverText.setOrigin(textRect.left +
                                        textRect.width / 2.0f,
                                        textRect.top +
                                        textRect.height / 2.0f);

                                gameoverText.setPosition(1920 / 2.0f, 1080 / 2.0f);
```

```
                }

        }
        /*************** Draw the scene based on the state of the game
***************/

        if (state == State::PLAYING)
        {
                window.clear();

                // set the mainView to be displayed in the window
                //window.setView(mainView);
                // Draw our game backgpround
                window.draw(spriteBackground);

                // Draw the clouds
                for (int i = 0; i < NUM_CLOUDS; i++)
                {
                        window.draw(clouds[i]);
                }

                // Draw backgrounds for the text
                window.draw(rect1);
                window.draw(rect2);


                // Draw the player
                window.draw(player.getSprite());
                window.draw(player2.getSprite());
                // Drawraw the bat
                window.draw(spriteBat);

                // Draw the health text
                window.draw(healthText);
                window.draw(healthText2);

                window.draw(circle);
                window.draw(circle2);

                // Draw the healthbar
                //window.draw(healthBar);
                //window.draw(healthBar2);

        // Draw our message
                window.draw(gameoverText);

        }

        if (state == State::PAUSED)
        {
                window.draw(spriteBackground);

                // Draw the clouds
                for (int i = 0; i < NUM_CLOUDS; i++)
```

```
                {
                        window.draw(clouds[i]);
                }

                // Draw backgrounds for the text
                window.draw(rect1);
                window.draw(rect2);

                window.draw(spriteBat);

                // Draw the health
                window.draw(healthText);
                window.draw(healthText2);
                // Draw our message
                window.draw(pauseText);
        }

        if (state == State::GAME_OVER)
        {
                window.draw(spriteBackground);

                // Draw the clouds
                for (int i = 0; i < NUM_CLOUDS; i++)
                {
                        window.draw(clouds[i]);
                }

                // Draw backgrounds for the text
                window.draw(rect1);
                window.draw(rect2);

                window.draw(spriteBat);

                // Draw the health
                window.draw(healthText);
                window.draw(healthText2);
                // Draw our message
                window.draw(gameoverText);
        }

        window.display();

    }// End the main game loop

    return 0;
}
```