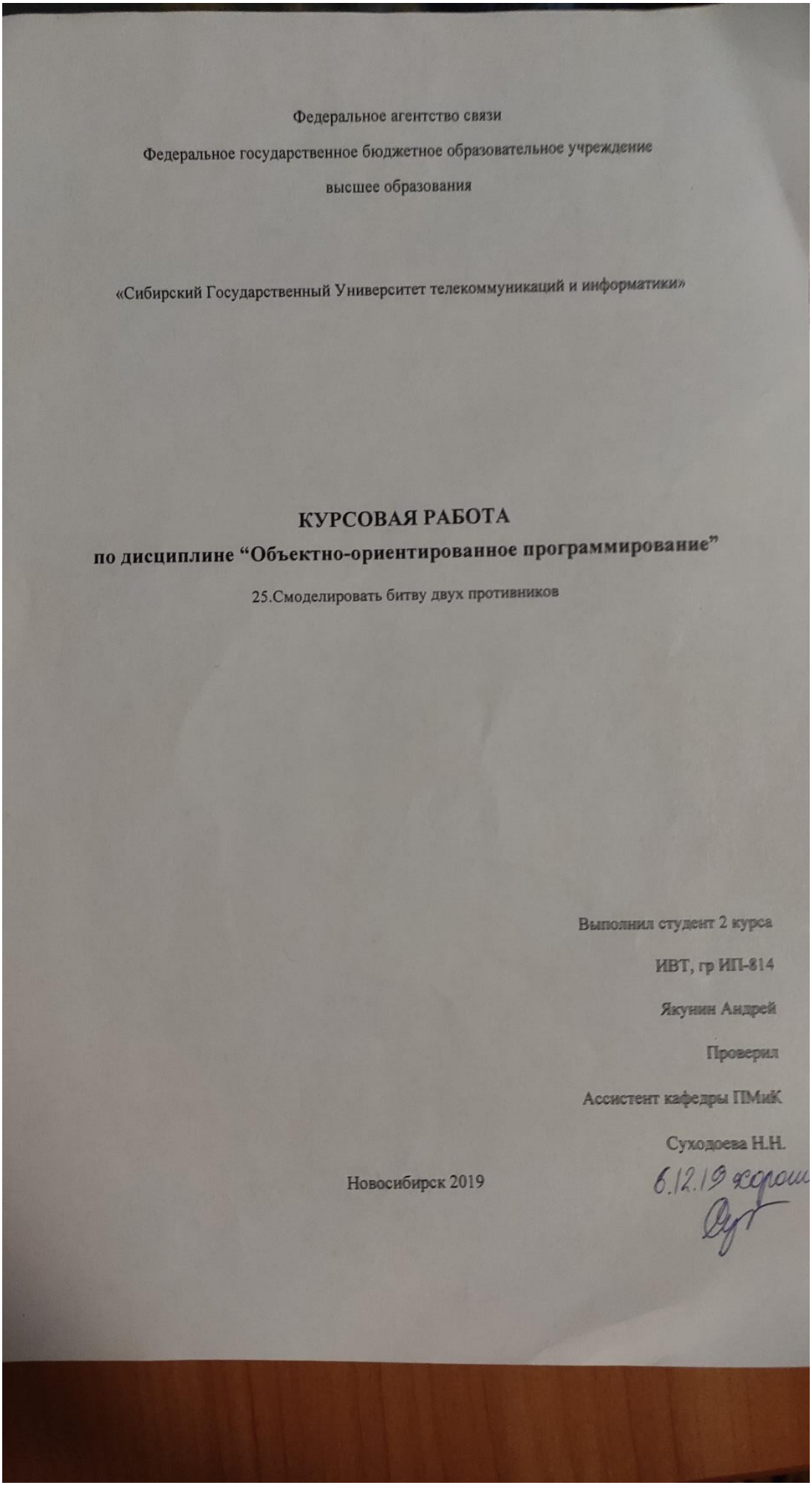


Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение

высшее образования

«Сибирский Государственный Университет телекоммуникаций и информатики»

КУРСОВАЯ РАБОТА

по дисциплине “Объектно-ориентированное программирование”

25.Смоделировать битву двух противников

Выполнил студент 2 курса

ИВТ, гр ИП-814

Якунин Андрей

Проверил

Ассистент кафедры ПМиК

Суходоева Н.Н.

Новосибирск 2019

6.12.19 хорош

1. Вариант курсовой работы №25

Написать программу, используя объектно-ориентированный подход, моделирующую сражение двух противников.

Постановка задачи:

Игрокам предоставляется поле танкового сражения, каждый игрок управляет одним из танков. Победителем считается игрок, нанесший урон оппоненту. При попадании в танк оппонента, его так перестаёт двигаться и теряет возможность вести ответный огонь.

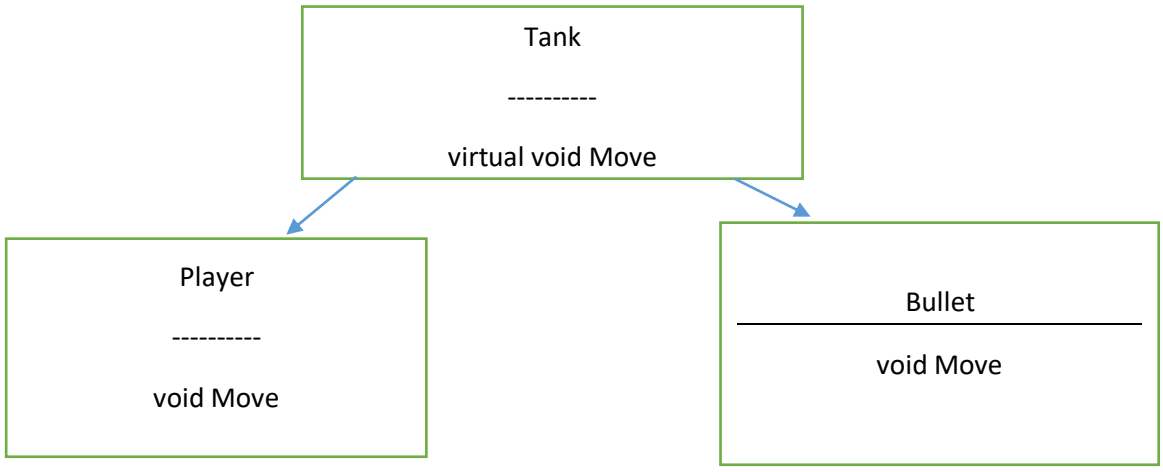
Реализация программного кода:

Для реализации Программы была использована свободная кроссплатформенная мультимедийная библиотека SFML.

Описание алгоритма основной программы

В основной программе идет объявление основных объектов и задаются настройки для окна, также объявляются необходимые для работы программы переменные. Далее в цикле, который работает пока не будет закрыто окно, в данном цикле проверяется нажатие клавиш двух игроков, движение пуль, отрисовка всех объектов и также проверяется коллизия пуль и танков.

Описание иерархии и методом класса



Технологии ООП использованные в работе

Инкапсуляция, наследование, полиморфизм, конструкторы, перегрузка контрукторов, списки инициализации, классы, виртуальные функции, массивы указателей на объекты.

Main:

В главном модуле были объявленны объекты **Player**и **Bullet**, затем был создан массив класса **Bullet**, в котором хранятся адреса объектов класса **Bullet**. Также загружается фоновое изображение. В основном цикле **while(window.isOpen())**, описана основная логика игры, проверяется нажатие клавиш двух игроков, проверяется коллизия танков и пуль и так же идет взаимодействие с внешними классами **Bullet**и**Player**.

Tank.h:

Абстрактный класс имеющий чистую виртуальную функцию и дающий классам наследникам общие переменные.

Player.h:

Класс Наследник Tank.h который отвечает за движение танков Игрока и Противника. Имеющий перегрузку конструкторов на основе которых создаются танк Игрока и Противника.

Player.cpp:

В данном модуле реализована работа функции Move отвечающей за перемещение танков по игровому полю.

Bullet.h:

Класс Наследник Tank.h который отвечает за движение пуль Игрока и Противника. Имеющий конструктор на основе которого создаются снаряды, наносящие урон Игроку и Противнику.

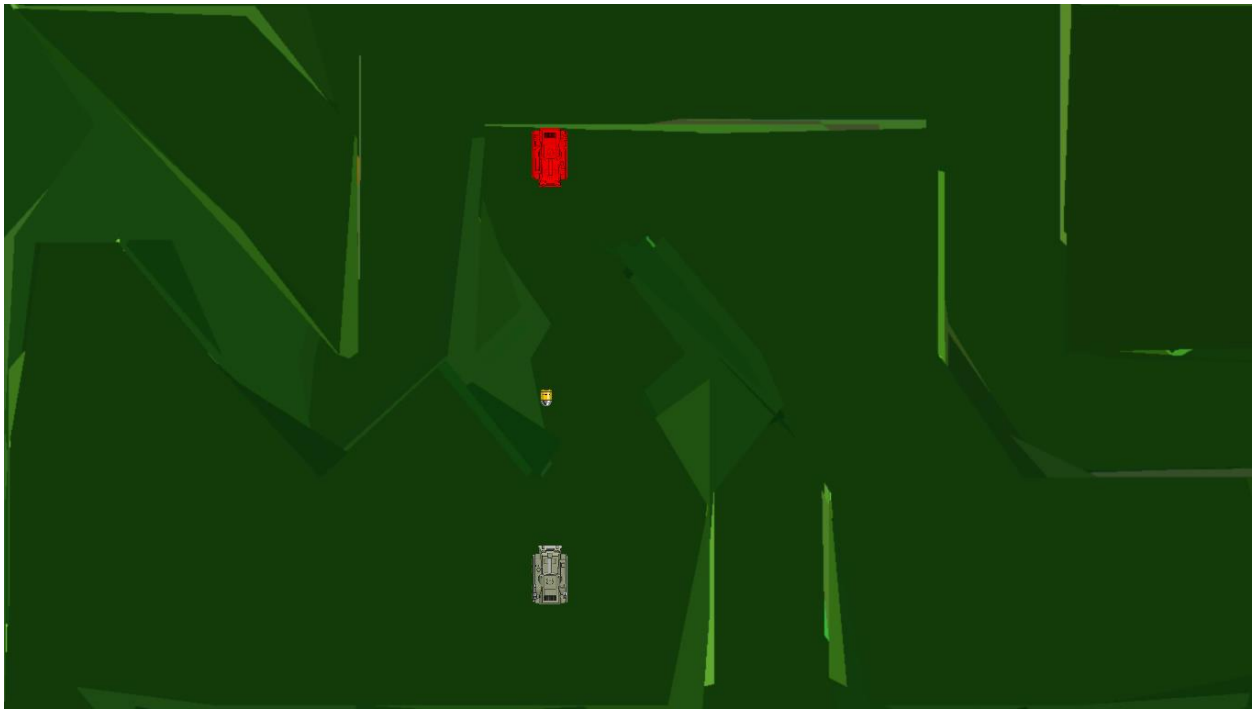
Bullet.cpp:

В данном модуле реализована работа функции Move отвечающей за перемещение танков по игровому полю.

Как можно заметить в модулях Player.cpp и Bullet.cpp используется полиморфизм, позволяющий именовать функции, а также переменные одинаковыми названиями.

Реализована Инкапсуляция, благодаря которой каждая Важная часть кода (например: Классы, Функции реализованные относительно классов) находятся в различных частях проекта и позволяют упрощенно воспринимать и вносить изменения в рабочий код программы.

Скриншоты работы программы



Заключение

В написанной нами курсовой работе нами использовались различные методы ООП, например списки инициализации в модуле Player.h. Так же был реализован полиморфизм для именования функций и переменных одинаковыми именами. Реализована Инкапсуляция, благодаря которой каждая Важная часть кода (например: Классы, Функции реализованные относительно классов) находятся в различных частях проекта и позволяют упрощенно воспринимать код и скрывать от пользователей часть кода в которую необходимо вносить изменения.Так же был сделан абстрактный класс являющийся отцовским для удобного наследования и объявления переменных. Была реализована Чистая виртуальная функция наследуемая двумя различными Кдочерними классами. Использование методов ООП облегчило задачу создания курсовой работы, помогло создать более устойчивый и удобный для чтения и изменения код.

Листинг

Main:

```
#include <iostream>
#include <SFML/Graphics.hpp>
#include <ctime>
#include "Tank.h"
#include "Player.h"
#include "Bullet.h"

using namespace std;
using namespace sf;

floatx,y;
int T = 0, T_enemy = 0;
intbullet_dir, bullet_enemy_dir, last , last_enemy, i = 0, e = 0, q = 5;

int main()
{
	RenderWindowwindow(sf::VideoMode(1366, 768), "TANCHIKI_simple",
Style::Fullscreen);
	Clock clock;
	srand(time(0));

	float time;
	boolT_on = false, T_on_enemy = false;
	Bullet *BULLETS[9];
```

```
Bullet obj_bullet_0("bullet.png", 78, 128, 0.15, 0.15);
        BULLETS[0] = &obj_bullet_0;
Bullet obj_bullet_1("bullet.png", 78, 128, 0.15, 0.15);
        BULLETS[1] = &obj_bullet_1;
Bullet obj_bullet_2("bullet.png", 78, 128, 0.15, 0.15);
        BULLETS[2] = &obj_bullet_2;
Bullet obj_bullet_3("bullet.png", 78, 128, 0.15, 0.15);
        BULLETS[3] = &obj_bullet_3;
Bullet obj_bullet_4("bullet.png", 78, 128, 0.15, 0.15);
        BULLETS[4] = &obj_bullet_4;

Bullet obj_bullet_enemy_0("bullet_enemy.png", 78, 128, 0.15, 0.15);
        BULLETS[5] = &obj_bullet_enemy_0;
Bullet obj_bullet_enemy_1("bullet_enemy.png", 78, 128, 0.15, 0.15);
        BULLETS[6] = &obj_bullet_enemy_1;
Bullet obj_bullet_enemy_2("bullet_enemy.png", 78, 128, 0.15, 0.15);
        BULLETS[7] = &obj_bullet_enemy_2;
Bullet obj_bullet_enemy_3("bullet_enemy.png", 78, 128, 0.15, 0.15);
        BULLETS[8] = &obj_bullet_enemy_3;
Bullet obj_bullet_enemy_4("bullet_enemy.png", 78, 128, 0.15, 0.15);
        BULLETS[9] = &obj_bullet_enemy_4;
Player obj_enemy("character.png", 300, 200, 78, 128, 0.5, 0.5);
Player obj_player("character_enemy.png");

Image map_image;
map_image.loadFromFile("image/map/pesok.jpg");
Texture map;
map.loadFromImage(map_image);
Sprite s_map;
s_map.setTexture(map);
```

```
        window.setVerticalSyncEnabled(true);
        time = clock.getElapsedTime().asMilliseconds();
        clock.restart();

while (window.isOpen())
   {
                time = time / 800;
                Event event;
while (window.pollEvent(event))
      {
if (event.type == Event::Closed)
window.close();
if (event.type == sf::Event::KeyPressed)
                if (event.key.code == sf::Keyboard::Escape)
                window.close();
        }

//////////////////////////////////////////Enemy - upravlenie
if (Keyboard::isKeyPressed(Keyboard::Left) and obj_enemy.life == true)
                {
                        obj_enemy.dir = 0;
                        obj_enemy.Move();
                        bullet_enemy_dir = 0;
                }
                else if (Keyboard::isKeyPressed(Keyboard::Right) and obj_enemy.life
== true)
                {
                        obj_enemy.dir = 1;
                        obj_enemy.Move();
```

```
                bullet_enemy_dir = 1;
            }
            else if (Keyboard::isKeyPressed(Keyboard::Up) and obj_enemy.life
== true)
            {
                obj_enemy.dir = 2;
                obj_enemy.Move();
                bullet_enemy_dir = 2;
            }
            else if (Keyboard::isKeyPressed(Keyboard::Down) and
obj_enemy.life == true)
            {
                obj_enemy.dir = 3;
                obj_enemy.Move();
                bullet_enemy_dir = 3;
            }
//////////////////////////////////////Enemy - upravlenie
//////////////////////////////////////Player - upravlenie
            if (Keyboard::isKeyPressed(Keyboard::A) and obj_player.life == true)
            {
                obj_player.dir = 0;
                obj_player.Move();
                bullet_dir = 0;
            }
            else if (Keyboard::isKeyPressed(Keyboard::D) and obj_player.life ==
true)
            {
                obj_player.dir = 1;
                obj_player.Move();
                bullet_dir = 1;
```

```
            }
            else if (Keyboard::isKeyPressed(Keyboard::W) and obj_player.life ==
true)
            {
                  obj_player.dir = 2;
                  obj_player.Move();
                  bullet_dir = 2;
            }
            else if (Keyboard::isKeyPressed(Keyboard::S) and obj_player.life ==
true)
            {
                  obj_player.dir = 3;
                  obj_player.Move();
                  bullet_dir = 3;
            }
//////////////////////////////////////////Player - upravlenie
//////////////////////////////////////////BULLET
            if (Keyboard::isKeyPressed(Keyboard::Space))
            {
                  BULLETS[e]->dir = bullet_dir;
                  if (last == 0 and T == 0 and BULLETS[e]->shoot == false and
obj_player.life == true)
                  {
                        x = obj_player.sprite.getPosition().x + 10;
                        y = obj_player.sprite.getPosition().y + 14;
                        BULLETS[e]->sprite.setPosition(x,y);
                        BULLETS[e]->shoot == true;
                        e++;
                        T = 0;
                        T_on = true;
```

```
                }
                if (last == 1 and T == 0 and BULLETS[e]->shoot == false and
obj_player.life == true)
                {
                        x = obj_player.sprite.getPosition().x + 30;
                        y = obj_player.sprite.getPosition().y + 14;
                        BULLETS[e]->sprite.setPosition(x,y);
                        BULLETS[e]->shoot == true;
                        e++;
                        T = 0;
                        T_on = true;
        }
        if (last == 2 and T == 0 and BULLETS[e]->shoot == false and
obj_player.life == true)
        {
                x = obj_player.sprite.getPosition().x + 14;
                        y = obj_player.sprite.getPosition().y + 10;
                        BULLETS[e]->sprite.setPosition(x,y);
                        BULLETS[e]->shoot == true;
                        e++;
                        T = 0;
                        T_on = true;
                }
                if (last == 3 and T == 0 and BULLETS[e]->shoot == false and
obj_player.life == true)
        {
                x = obj_player.sprite.getPosition().x + 14;
                        y = obj_player.sprite.getPosition().y + 30;
                        BULLETS[e]->sprite.setPosition(x,y);
                        BULLETS[e]->shoot == true;
```

```
                e++;
                T = 0;
                T_on = true;
            }
            if (e > 4) e = 0;
        }
////////////////////////////////////////BULLET
////////////////////////////////////////BULLET_enemy
        if (Keyboard::isKeyPressed(Keyboard::Numpad0))
        {
            BULLETS[q]->dir = bullet_enemy_dir;
            if (last_enemy == 0 and T_enemy == 0 and BULLETS[e]->shoot == false and obj_enemy.life == true)
            {
                x = obj_enemy.sprite.getPosition().x + 10;
                y = obj_enemy.sprite.getPosition().y + 14;
                BULLETS[q]->sprite.setPosition(x,y);
                BULLETS[q]->shoot == true;
                T_enemy = 0;
                T_on_enemy = true;
                q++;
            }
            if (last_enemy == 1 and T_enemy == 0 and BULLETS[e]->shoot == false and obj_enemy.life == true)
            {
                x = obj_enemy.sprite.getPosition().x + 30;
                y = obj_enemy.sprite.getPosition().y + 14;
                BULLETS[q]->sprite.setPosition(x,y);
                BULLETS[q]->shoot == true;
                T_enemy = 0;
```

```
                    T_on_enemy = true;
                    q++;
            }
            if (last_enemy == 2 and T_enemy == 0 and BULLETS[e]->shoot ==
false and obj_enemy.life == true)
            {
                x = obj_enemy.sprite.getPosition().x + 14;
                    y = obj_enemy.sprite.getPosition().y + 10;
                    BULLETS[q]->sprite.setPosition(x,y);
                    BULLETS[q]->shoot == true;
                    T_enemy = 0;
                    T_on_enemy = true;
                    q++;
                }
                if (last_enemy == 3 and T_enemy == 0 and BULLETS[e]-
>shoot == false and obj_enemy.life == true)
            {
                x = obj_enemy.sprite.getPosition().x + 14;
                    y = obj_enemy.sprite.getPosition().y + 30;
                    BULLETS[q]->sprite.setPosition(x,y);
                    BULLETS[q]->shoot == true;
                    T_enemy = 0;
                    T_on_enemy = true;
                    q++;
                }
                if (q > 9) q = 5;
            }
//////////////////////////////////////////BULLET_enemy

        for (int w = 0; w < 9; w++) BULLETS[w]->Move();
```

```
if (T_on_enemy == true)
{
	T_enemy++;
	if (T_enemy> 30)
	{
		T_enemy = 0;
		T_on_enemy = false;
	}
}

if (T_on == true)
{
	T++;
	if (T > 30)
	{
		T = 0;
		T_on = false;
	}
}

for (inten = 0; en< 4; en++) if (BULLETS[en]->sprite.getGlobalBounds().intersects(obj_enemy.sprite.getGlobalBounds()))
obj_enemy.life = false;
for (inten = 5; en< 9; en++) if (BULLETS[en]->sprite.getGlobalBounds().intersects(obj_player.sprite.getGlobalBounds()))
obj_player.life = false;

if (obj_player.life == false) obj_player.sprite.setColor(Color::Red);
if (obj_enemy.life == false) obj_enemy.sprite.setColor(Color::Red);
```

```
        if (Keyboard::isKeyPressed(Keyboard::R))
        {
                obj_player.life = true;
                obj_player.sprite.setColor(Color::White);
                obj_enemy.sprite.setColor(Color::White);
                obj_enemy.life = true;
        }

        window.clear();
        window.draw(s_map);
        window.draw(obj_player.sprite);
        window.draw(obj_enemy.sprite);
        for (int o = 0; o < 9; o++) window.draw(BULLETS[o]->sprite);
        window.display();
        }
        return 0;
}
```

Tank.h

```
#ifndef TANK_H
#define TANK_H
#include <iostream>
#include <SFML/Graphics.hpp>
#include<ctime>

using namespace std;
using namespace sf;

class Tank
{
```

```
        public:
                float w, h, x, y, speed, w_scale, h_scale;
                intdir;
                bool life = true;
                bool shoot = false;
                String File;
                Image image;
                Texture texture;
                Sprite sprite;
                virtualvoidMove()=0;
};

#endif
```

Player.h

```
#ifndef PLAYER_H
#define PLAYER_H
#include "Tank.h"

class Player : public Tank
{
        public:
                float w, h, x, y, speed, w_scale, h_scale;
                Player(String F, float X, float Y, float W, float H, float W_SCALE,
float H_SCALE);
                Player(String F) : w(0.5), h(0.5)
                {
                        File = F;
                        image.loadFromFile("image/character/" + File);
                        texture.loadFromImage(image);
                        sprite.setTexture(texture);
```

```
                sprite.setPosition(600, 300);
                sprite.setTexture(texture);
                sprite.setTextureRect(IntRect(21, 194, 78, 128));
                sprite.setScale(w, h);
            }
            voidMove();
        protected:
};

#endif
```

Player.cpp

```
#include "Player.h"

Player::Player(String F, float X, float Y, float W, float H, float W_SCALE, float H_SCALE)
{
    w_scale = W_SCALE;
    h_scale = H_SCALE;
    w = W;
    h = H;
    x = X;
    y = Y;
    File = F;
    image.loadFromFile("image/character/" + File);
    texture.loadFromImage(image);
    sprite.setPosition(x, y);
    sprite.setTexture(texture);
    sprite.setTextureRect(IntRect(21, 194, w, h));
    sprite.setScale(w_scale, h_scale);
}
```

```
void Player::Move()
{
	if (dir == 0)
	{
		sprite.move(-5,0);
		sprite.setTextureRect(IntRect(138,255,128,78));
	}

	if (dir == 1)
	{
		sprite.move(5,0);
		sprite.setTextureRect(IntRect(12,58,128,78));
	}

	if ( dir == 2)
	{
		sprite.move(0,-5);
		sprite.setTextureRect(IntRect(21,194,78,128));
	}

	if (dir == 3)
	{
		sprite.move(0,5);
		sprite.setTextureRect(IntRect(188,32,78,128));
	}
}
```

Bullet.h

```
#ifndef BULLET_H
#define BULLET_H
```

```
#include "Tank.h"

class Bullet : public Tank
{
	public:
		Bullet(String F, float W, float H, float W_SCALE, float H_SCALE)
		{
			w_scale = W_SCALE;
			h_scale = H_SCALE;
			w = W;
			h = H;
			File = F;
			image.loadFromFile("image/character/" + File);
			texture.loadFromImage(image);
			sprite.setTextureRect(IntRect(12,58,128,78));
			sprite.setTexture(texture);
			sprite.setScale(w_scale, h_scale);
		}
		voidMove();
};

#endif
```

Bullet.cpp

```
#include "Bullet.h"

void Bullet::Move()
{
	if (dir == 0)
	{
		sprite.move(-10,0);
```

```
            sprite.setTextureRect(IntRect(138,255,128,78));
        }

        if (dir == 1)
        {
            sprite.move(10,0);
            sprite.setTextureRect(IntRect(12,58,128,78));
        }

        if (dir == 2)
        {
            sprite.move(0,-10);
            sprite.setTextureRect(IntRect(21,194,78,128));
        }

        if (dir == 3)
        {
            sprite.move(0,10);
            sprite.setTextureRect(IntRect(188,32,78,128));
        }
}
```