#include "stdafx.h"

#include <SFML/Graphics.hpp>

#include <iostream>

#include <cstdlib>

#include <random>

#include <ctime>

#include <windows.h>

using namespace sf;

public void Start()

{

}

int main()

{

const int namberEnemy = 10;

const float timeDistanceCreate = 400000;

int heal = 3;

int score = 0;

bool end = true;

String healHave = ("осталось жизней");

std::printf("Heal have: ");

std::cout << heal << std::endl;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

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

sf::Image enemyimage; //создаем объект Image (изображение)

enemyimage.loadFromFile("image/enemy.png");//загружаем в него файл ЗАГРУЗКА ТЕКСТУРЫ КУРИЦЫ

sf::Texture enemytexture;//создаем объект Texture (текстура)

enemytexture.loadFromImage(enemyimage);//передаем в него объект Image (изображения)

sf::Sprite enemysprite;//создаем объект Sprite(спрайт)

enemysprite.setTexture(enemytexture);//передаём в него объект Texture (текстуры) ////////

int sizeImageX = enemyimage.getSize().x;

int sizeImageY = enemyimage.getSize().y;

///////

struct enemy //Создаем структуру

{

Sprite sprite;

double speed;

int coordinate1;

int coordinate2;

bool flagStartSprite;

bool flagIsReal;

};

///////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

enemy enemys[namberEnemy]; ////////// массив структур

for (int i = 0; i < namberEnemy; i++)

{

enemys[i].sprite = enemysprite;

enemys[i].flagIsReal = false;

}

// Главный цикл приложения

Clock clock;

float timeCreate = timeDistanceCreate;

while (window.isOpen())

{

float time = clock.getElapsedTime().asMicroseconds();

clock.restart();

timeCreate -= time;

// Обрабатываем события в цикле

sf::Event event;

while (window.pollEvent(event))

{

// Кроме обычного способа наше окно будет закрываться по нажатию на Escape

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

(event.type == sf::Event::KeyPressed && event.key.code == sf::Keyboard::Escape))

window.close();

}

if (timeCreate <= 0)

{

for (int i = 0; i < namberEnemy; i++)

{

if (!(enemys[i].flagIsReal))

{

float coor1, coor2;// /////////

std::random\_device rd; // non-deterministic generator ВЫЧИСЛЕНИЕ КООРДИНАТЫ ПОЯВЛЕНИЕ

std::mt19937 gen(rd()); // to seed mersenne twister.

std::uniform\_int\_distribution<> dist(0, 32700); //

coor2 = dist(gen) % (window.getSize().y - enemyimage.getSize().y);

if (dist(gen) % 2) { coor1 = window.getSize().x - enemyimage.getSize().x; }

else { coor1 = 0; }

enemys[i].coordinate1 = coor1;

enemys[i].coordinate2 = coor2;

enemys[i].speed = 0.0001 \* (dist(gen) % 6 + 4);

enemys[i].flagIsReal = true;

enemys[i].flagStartSprite = true;

break;

}

}

timeCreate = timeDistanceCreate;

}

window.clear();

for (int i = 0; i < namberEnemy; i++)

{

if (heal > 0)

{

if (enemys[i].flagIsReal)

{

if (enemys[i].flagStartSprite)

{

enemys[i].sprite.setPosition(enemys[i].coordinate1, enemys[i].coordinate2);

enemys[i].flagStartSprite = false;

}

////////////////////////////////////////////////////////////////////////////// ДВИЖЕНИЕ КУРИЦЫ ПО ЭКРАНУ ///////////////////////

if (enemys[i].coordinate1 > 0) { (enemys[i].sprite).move(-enemys[i].speed\*time, 0.0); (enemys[i].sprite).setTextureRect(IntRect(0, 0, sizeImageX, sizeImageY)); } ////////

if (enemys[i].coordinate1 == 0) { (enemys[i].sprite).move(enemys[i].speed\*time, 0.0); (enemys[i].sprite).setTextureRect(IntRect(sizeImageX, 0, -sizeImageX, sizeImageY)); } ////////

if (enemys[i].sprite.getPosition().x <= 0 || enemys[i].sprite.getPosition().x >= window.getSize().x)

{

enemys[i].flagIsReal = false;

heal = heal - 1;

std::printf("Heal have: ");

std::cout << heal << std::endl;

continue;

}

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

{

int mousX = Mouse::getPosition(window).x;

int mousY = Mouse::getPosition(window).y;

if ((((mousX > enemys[i].sprite.getPosition().x) && (mousX < enemys[i].sprite.getPosition().x + enemyimage.getSize().x))

&& ((mousY > enemys[i].sprite.getPosition().y) && (mousY < enemys[i].sprite.getPosition().y + enemyimage.getSize().y))))

{

enemys[i].flagIsReal = false;

score++;

}

}

if (enemys[i].flagIsReal) { window.draw(enemys[i].sprite); }

}

}

else if (end)

{

std::printf("Score: ");

std::cout << score << std::endl;

end = false;

}

}

window.display();

}

std::cout << score << std::endl;

return 0;

}