#include <SFML/Graphics.hpp>

#include <cstdlib>

#include <ctime>

#include <iostream>

using namespace std;

const int SCREEN\_WIDTH = 640;

const int SCREEN\_HEIGHT = 480;

const int BIRD\_RADIUS = 15;

const int PIPE\_WIDTH = 40;

const int INITIAL\_PIPE\_GAP = 150;

const int GRAVITY = 1;

const int JUMP\_HEIGHT = 15;

const int MAX\_LEVELS = 9;

int birdY, birdX;

int pipeX, pipeY;

int pipeGap;

int pipeSpeed;

int score;

bool isGameOver;

bool wingUp;

bool exitGame;

bool isPaused;

sf::Texture backgroundTexture;

sf::Sprite backgroundSprite;

// Pipe movement variables

int pipeDirection = 1; // Direction of pipe movement (1 for down, -1 for up)

// Function to load background image

void loadBackgroundImage() {

if (!backgroundTexture.loadFromFile("images/background.png")) {

cerr << "Error loading background image!" << endl;

exitGame = true;

}

else {

backgroundSprite.setTexture(backgroundTexture);

}

}

// Function to draw the background with dynamic scaling

void drawBackground(sf::RenderWindow& window) {

sf::Vector2u windowSize = window.getSize();

sf::Vector2f scale(

static\_cast<float>(windowSize.x) / backgroundTexture.getSize().x,

static\_cast<float>(windowSize.y) / backgroundTexture.getSize().y

);

backgroundSprite.setScale(scale);

window.draw(backgroundSprite);

}

// Initialize the game state

void initializeGame() {

birdY = SCREEN\_HEIGHT / 2;

birdX = SCREEN\_WIDTH / 6;

pipeX = SCREEN\_WIDTH;

pipeY = rand() % (SCREEN\_HEIGHT - INITIAL\_PIPE\_GAP);

pipeGap = INITIAL\_PIPE\_GAP;

pipeSpeed = 2;

isGameOver = false;

score = 0;

wingUp = true;

}

// Draw the bird

void drawBird(sf::RenderWindow& window) {

sf::CircleShape bird(BIRD\_RADIUS);

bird.setPosition(birdX - BIRD\_RADIUS, birdY - BIRD\_RADIUS);

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

window.draw(bird);

// Wings

sf::CircleShape wing(7, 4);

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

if (wingUp) {

wing.setPosition(birdX - 17, birdY - 14);

window.draw(wing);

wing.setPosition(birdX + 3, birdY - 14);

window.draw(wing);

}

else {

wing.setPosition(birdX - 17, birdY + 6);

window.draw(wing);

wing.setPosition(birdX + 3, birdY + 6);

window.draw(wing);

}

wingUp = !wingUp;

}

// Draw the pipes

void drawPipes(sf::RenderWindow& window) {

sf::RectangleShape upperPipe(sf::Vector2f(PIPE\_WIDTH, pipeY));

upperPipe.setPosition(pipeX, 0);

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

window.draw(upperPipe);

sf::RectangleShape lowerPipe(sf::Vector2f(PIPE\_WIDTH, SCREEN\_HEIGHT - (pipeY + pipeGap)));

lowerPipe.setPosition(pipeX, pipeY + pipeGap);

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

window.draw(lowerPipe);

}

// Move the pipes

void movePipes() {

pipeX -= 3;

// Move the pipe in the vertical direction

pipeY += pipeDirection \* pipeSpeed;

// Toggle pipe direction if needed

if (pipeY < 50 || pipeY + pipeGap > SCREEN\_HEIGHT - 50) {

pipeDirection \*= -1; // Change direction

}

// If the pipe goes off the screen, reset its position

if (pipeX + PIPE\_WIDTH < 0) {

pipeX = SCREEN\_WIDTH;

pipeY = 50 + rand() % (SCREEN\_HEIGHT - INITIAL\_PIPE\_GAP - 100); // Randomize the gap

score++;

}

}

// Collision detection

void checkCollision() {

if (birdY - BIRD\_RADIUS < 0 || birdY + BIRD\_RADIUS > SCREEN\_HEIGHT) {

isGameOver = true;

return;

}

if (birdX + BIRD\_RADIUS > pipeX && birdX - BIRD\_RADIUS < pipeX + PIPE\_WIDTH) {

if (birdY - BIRD\_RADIUS < pipeY || birdY + BIRD\_RADIUS > pipeY + pipeGap) {

isGameOver = true;

}

}

}

// Display the score

void displayScore(sf::RenderWindow& window) {

sf::Font font;

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

cerr << "Error loading font!" << endl;

exitGame = true;

}

sf::Text scoreText;

scoreText.setFont(font);

scoreText.setString("Score: " + to\_string(score));

scoreText.setCharacterSize(24);

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

scoreText.setPosition(10, 10);

window.draw(scoreText);

}

// Display the "Game Over" screen

void displayGameOver(sf::RenderWindow& window) {

window.clear();

drawBackground(window);

sf::Font font;

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

cerr << "Error loading font!" << endl;

exitGame = true;

}

sf::Text gameOverText;

gameOverText.setFont(font);

gameOverText.setString("GAME OVER");

gameOverText.setCharacterSize(48);

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

gameOverText.setPosition(SCREEN\_WIDTH / 2 - 120, SCREEN\_HEIGHT / 2 - 50);

sf::Text scoreText;

scoreText.setFont(font);

scoreText.setString("Final Score: " + to\_string(score));

scoreText.setCharacterSize(32);

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

scoreText.setPosition(SCREEN\_WIDTH / 2 - 120, SCREEN\_HEIGHT / 2);

window.draw(gameOverText);

window.draw(scoreText);

window.display();

sf::Event event;

while (window.waitEvent(event)) {

if (event.type == sf::Event::KeyPressed) {

break;

}

}

}

// Function to create buttons and check for clicks

void createButton(sf::RenderWindow& window, const string& label, const sf::Vector2f& position, const sf::Color& color, sf::RectangleShape& buttonShape, sf::Text& buttonText) {

sf::Font font;

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

cerr << "Error loading font!" << endl;

exitGame = true;

}

buttonShape.setSize(sf::Vector2f(200, 50));

buttonShape.setPosition(position);

buttonShape.setFillColor(color);

buttonText.setFont(font);

buttonText.setString(label);

buttonText.setCharacterSize(24);

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

buttonText.setPosition(position.x + (buttonShape.getSize().x - buttonText.getLocalBounds().width) / 2,

position.y + (buttonShape.getSize().y - buttonText.getLocalBounds().height) / 2);

window.draw(buttonShape);

window.draw(buttonText);

}

// Check if the mouse click is inside a button

bool isButtonClicked(sf::RectangleShape& button, sf::Event& event) {

if (event.type == sf::Event::MouseButtonPressed) {

if (event.mouseButton.button == sf::Mouse::Left) {

sf::FloatRect bounds = button.getGlobalBounds();

if (bounds.contains(event.mouseButton.x, event.mouseButton.y)) {

return true;

}

}

}

return false;

}

// Show the welcome screen with colorful "Flappy Bird"

void showWelcomeScreen(sf::RenderWindow& window) {

window.clear();

drawBackground(window);

sf::Font font;

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

cerr << "Error loading font!" << endl;

exitGame = true;

}

// Create the "Flappy Bird" text with colorful effects

sf::Text titleText;

titleText.setFont(font);

titleText.setString("Flappy Bird");

titleText.setCharacterSize(80);

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

titleText.setPosition(SCREEN\_WIDTH / 2 - 180, SCREEN\_HEIGHT / 5);

window.draw(titleText);

// Create the main menu buttons

sf::RectangleShape playButton, scoreboardButton, exitButton;

sf::Text playButtonText, scoreboardButtonText, exitButtonText;

createButton(window, "Play", { SCREEN\_WIDTH / 2 - 100, SCREEN\_HEIGHT / 2 }, sf::Color::Green, playButton, playButtonText);

createButton(window, "Scoreboard", { SCREEN\_WIDTH / 2 - 100, SCREEN\_HEIGHT / 2 + 70 }, sf::Color::Yellow, scoreboardButton, scoreboardButtonText);

createButton(window, "Exit", { SCREEN\_WIDTH / 2 - 100, SCREEN\_HEIGHT / 2 + 140 }, sf::Color::Red, exitButton, exitButtonText);

window.display();

// Wait for mouse click to start the game or exit

sf::Event event;

while (window.waitEvent(event)) {

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

window.close();

exitGame = true;

}

if (isButtonClicked(playButton, event)) {

// Start the game when "Play" button is clicked

return;

}

if (isButtonClicked(exitButton, event)) {

// Exit the game when "Exit" button is clicked

window.close();

exitGame = true;

return;

}

}

}

// Show the pause menu with options to resume or go to the main menu

void showPauseMenu(sf::RenderWindow& window) {

window.clear();

drawBackground(window);

sf::Font font;

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

cerr << "Error loading font!" << endl;

exitGame = true;

}

// Create the "Paused" text

sf::Text pausedText;

pausedText.setFont(font);

pausedText.setString("Paused");

pausedText.setCharacterSize(48);

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

pausedText.setPosition(SCREEN\_WIDTH / 2 - 100, SCREEN\_HEIGHT / 3);

window.draw(pausedText);

// Create the pause menu buttons

sf::RectangleShape resumeButton, mainMenuButton;

sf::Text resumeButtonText, mainMenuButtonText;

createButton(window, "Resume", { SCREEN\_WIDTH / 2 - 100, SCREEN\_HEIGHT / 2 }, sf::Color::Green, resumeButton, resumeButtonText);

createButton(window, "Main Menu", { SCREEN\_WIDTH / 2 - 100, SCREEN\_HEIGHT / 2 + 70 }, sf::Color::Yellow, mainMenuButton, mainMenuButtonText);

window.display();

// Wait for mouse click or key press

sf::Event event;

while (window.waitEvent(event)) {

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

window.close();

exitGame = true;

}

if (isButtonClicked(resumeButton, event)) {

// Resume the game when "Resume" button is clicked

isPaused = false;

return;

}

if (isButtonClicked(mainMenuButton, event)) {

// Go to main menu when "Main Menu" button is clicked

isPaused = false; // To stop the pause menu

initializeGame(); // Reset the game state to the initial setup

showWelcomeScreen(window); // Show the main menu again

return; // Ensure no other actions happen after this

}

}

}

// Main game loop

int main() {

sf::RenderWindow window(sf::VideoMode(SCREEN\_WIDTH, SCREEN\_HEIGHT), "Flappy Bird");

window.setFramerateLimit(60);

srand(static\_cast<unsigned int>(time(0)));

exitGame = false;

// Load background image

loadBackgroundImage();

while (!exitGame) {

showWelcomeScreen(window); // Show the welcome screen

// Start the game after pressing any key

if (!exitGame) {

initializeGame();

sf::Clock clock;

while (!isGameOver && !exitGame && window.isOpen()) {

sf::Event event;

while (window.pollEvent(event)) {

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

window.close();

exitGame = true;

}

if (event.type == sf::Event::KeyPressed) {

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

isPaused = true;

}

else if (event.key.code == sf::Keyboard::Space) {

birdY -= JUMP\_HEIGHT;

}

else if (event.key.code == sf::Keyboard::Up) {

if (birdY - BIRD\_RADIUS > 0) { // Prevent the bird from moving off the top

birdY -= JUMP\_HEIGHT;

}

}

else if (event.key.code == sf::Keyboard::Down) {

if (birdY + BIRD\_RADIUS < SCREEN\_HEIGHT) { // Prevent the bird from moving off the bottom

birdY += JUMP\_HEIGHT;

}

}

}

}

if (isPaused) {

showPauseMenu(window); // Show the pause menu

continue; // Skip the rest of the game logic while paused

}

birdY += GRAVITY; // Apply gravity

movePipes();

checkCollision();

// Draw everything

window.clear();

drawBackground(window);

drawBird(window);

drawPipes(window);

displayScore(window);

// Display the window

window.display();

// Check if game over

if (isGameOver) {

displayGameOver(window);

}

}

}

}

return 0;

}