// https://github.com/mannanarama/CS-256

#include "stdafx.h"

#include <SFML/Graphics.hpp>

#include <time.h>

#include <math.h>

using namespace sf;

//global variables

int num\_vertBox = 30, num\_horzBox = 30; //number of boxes to display

int size = 16; //number of pixels per image (Square, 16x16)

int w = size \* num\_horzBox; //background number of pixels in width

int h = size \* num\_vertBox; //background number of pixels in height

float delay;

//variables to control the direction and length of each snake

int direction;

int direction2;

int snake\_length = 4; // Length of first snake

int snake\_length2 = 4; // Length of second snake

//Maximum size of the snakes (100) and position of the snakes

struct Snake

{

int x, y;

}s[100];

struct Snake2

{

int a, b;

}s2[100];

//to hold the position of the fruit

struct Fruit

{

int x, y;

}food;

void Tick()

{

//Move remaining pieces of the first snake, s[1] - s[99]

for (int i = snake\_length; i > 0; --i)

{

s[i].x = s[i - 1].x;

s[i].y = s[i - 1].y;

}

//Move remaining pieces of the second snake

for (int i = snake\_length2; i > 0; --i)

{

s2[i].a = s2[i - 1].a;

s2[i].b = s2[i - 1].b;

}

//Head of snake depends on direction of user s[0], for snake 1

//User Up

if (direction == 3)

s[0].y -= 1;

//User Down

if (direction == 0)

s[0].y += 1;

//User Left

if (direction == 1)

s[0].x -= 1;

//User Right

if (direction == 2)

s[0].x += 1;

//Head of snake depends on direction of user s2[0], for snake 2

//User Up

if (direction2 == 3)

s2[0].b -= 1;

//User Down

if (direction2 == 0)

s2[0].b += 1;

//User Left

if (direction2 == 1)

s2[0].a -= 1;

//User Right

if (direction2 == 2)

s2[0].a += 1;

//If Snake 1 eats food it should grow

if ((s[0].x == food.x) && (s[0].y == food.y))

{

snake\_length++;

delay -= 0.005; // if snake eats, increases speed

//Randomly place food somewhere else

food.x = rand() % num\_horzBox;

food.y = rand() % num\_vertBox;

}

// If Snake 2 eats food it shoud grow

if ((s2[0].a == food.x) && (s2[0].b == food.y))

{

snake\_length2++;

delay -= 0.005; // if snake eats, increases speed

//Randomly place food somewhere else

food.x = rand() % num\_horzBox;

food.y = rand() % num\_vertBox;

}

//Boundary Checking for snake 1, screen loop back on other side

if (s[0].x > num\_horzBox - 1) //this needs to be -1 because otherwise an empty row and column will outside the drawn screen

s[0].x = 0;

if (s[0].x < 0)

s[0].x = num\_horzBox;

if (s[0].y > num\_vertBox - 1)

s[0].y = 0;

if (s[0].y < 0)

s[0].y = num\_vertBox;

// Boundary Checking for snake 2, screen loop back on other side

if (s2[0].a > num\_horzBox - 1)

s2[0].a = 0;

if (s2[0].a < 0)

s2[0].a = num\_horzBox;

if (s2[0].b > num\_vertBox - 1)

s2[0].b = 0;

if (s2[0].b < 0)

s2[0].b = num\_vertBox;

//Check if snake 1 went over itself

for (int i = 1; i < snake\_length; i++)

{

//Cut snake in half from place eaten

if (s[0].x == s[i].x && s[0].y == s[i].y)

{

snake\_length = i;

delay = .1; // if snakes goes over itself, goes back to original speed

}

}

// Check if snake 2 went over itself

for (int i = 1; i < snake\_length2; i++)

{

//Cut snake in half from place eaten

if (s2[0].a == s2[i].a && s2[0].b == s2[i].b)

{

snake\_length2 = i;

delay = .1; // if snakes goes over itself, goes back to original speed

}

}

}

//main loop

int main()

{

srand(time(0));

//create the window, named Snake Game

RenderWindow window(VideoMode(w, h), "Snake Game!");

//textures

Texture t1, t2, t3, t4;

t1.loadFromFile("images/white.png"); //the background

t2.loadFromFile("images/red.png"); //the color of Snake 1

t3.loadFromFile("images/green.png"); //the food

t4.loadFromFile("images/red.png"); //the color of Snake 2

//Sprite with physical dimensions

Sprite sprite1(t1);

Sprite sprite2(t2);

Sprite sprite3(t3);

Sprite sprite4(t4);

//starting location of food

food.x = 10;

food.y = 10;

Clock clock;

float timer = 0;

delay = .1; // change speed

//main game loop, until the user closes or exits the window

while (window.isOpen())

{

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

clock.restart();

timer += time;

//Allow us to check when a user does something

Event e;

//check when window is closed

while (window.pollEvent(e))

{

if (e.type == Event::Closed)

{

window.close();

}

}

//Control input from user for snake 1 (uses the arrow keys)

if (Keyboard::isKeyPressed(Keyboard::Up)) direction = 3;

if (Keyboard::isKeyPressed(Keyboard::Down)) direction = 0;

if (Keyboard::isKeyPressed(Keyboard::Left)) direction = 1;

if (Keyboard::isKeyPressed(Keyboard::Right)) direction = 2;

//Control input from user for snake 2 (uses WASD)

if (Keyboard::isKeyPressed(Keyboard::W)) direction2 = 3;

if (Keyboard::isKeyPressed(Keyboard::S)) direction2 = 0;

if (Keyboard::isKeyPressed(Keyboard::A)) direction2 = 1;

if (Keyboard::isKeyPressed(Keyboard::D)) direction2 = 2;

if (timer > delay)

{

timer = 0;

Tick();

}

//Draw

window.clear();

//Draw the background uses sprite1

for (int i = 0; i < num\_horzBox; i++)

{

for (int j = 0; j < num\_vertBox; j++)

{

sprite1.setPosition(i\*size, j\*size);

window.draw(sprite1);

}

}

//Draw snake 1 using sprite 2

for (int i = 0; i < snake\_length; i++)

{

sprite2.setPosition(s[i].x\*size, s[i].y\*size);

window.draw(sprite2);

}

//Draw the food using sprite 3

sprite3.setPosition(food.x\*size, food.y\*size);

window.draw(sprite3);

//Draw snake 2 using sprite 4

for (int i = 0; i < snake\_length2; i++)

{

sprite4.setPosition(s2[i].a\*(size), s2[i].b\*(size));

window.draw(sprite4);

}

window.display();

}

return 0;

}