撰寫C++視窗程式

1. Install SFML 3.0.0 (選擇64位元)

<https://www.sfml-dev.org/download/sfml/3.0.0/>

1. 下載：[WinLibs UCRT 14.2.0 (64-bit)](https://github.com/brechtsanders/winlibs_mingw/releases/download/14.2.0posix-19.1.1-12.0.0-ucrt-r2/winlibs-x86_64-posix-seh-gcc-14.2.0-mingw-w64ucrt-12.0.0-r2.7z)
2. 下載：64-bit GCC 14.2.0 MinGW (SEH) (UCRT)
3. 兩個各自解壓縮，將兩個名字很長的資料夾放在C槽，資料夾中有所需要的C++編譯器和檔案。我的檔案解壓縮後最外層資料夾是兩個很長的名字：「winlibs-x86\_64-posix-seh-gcc-14.2.0-mingw-w64ucrt-12.0.0-r2」和「SFML-3.0.0-windows-gcc-14.2.0-mingw-64-bit」，如果你的最外層資料夾只有「mingw64」和「SFML-3.0.0」，那麼下面的路徑中要刪除「\winlibs-x86\_64-posix-seh-gcc-14.2.0-mingw-w64ucrt-12.0.0-r2」和「\SFML-3.0.0-windows-gcc-14.2.0-mingw-64-bit」，這是最常見的問題。
4. 設定系統環境變數Path：

C:\winlibs-x86\_64-posix-seh-gcc-14.2.0-mingw-w64ucrt-12.0.0-r2\mingw64\bin

C:\SFML-3.0.0-windows-gcc-14.2.0-mingw-64-bit\SFML-3.0.0\bin

設定步驟：

在桌面左下角「開始」按鈕上**右鍵** → 選「**系統**」

左側或下方點「**進階系統設定**」

在「系統內容」視窗中，點「**環境變數**」按鈕

在「系統變數」區塊，找到 Path 這個變數 → 選它 → 點「**編輯**」

你會看到很多現有的路徑，一行一個

要「新增」的話，按右邊「**新增**」，然後直接貼上你要加的資料夾路徑

修改好後，按「**確定**」儲存

所有視窗按「確定」關掉

1. 在Code::Blocks 20.03或25.03中建立一個新的專案Console Application (舊版16.01不能)
2. 在Settings > Compiler複製GNU GCC compiler產生一個新的Profile叫做WinLibs UCRT 14.2.0，在Toolchain executables頁籤選擇**C:\winlibs-x86\_64-posix-seh-gcc-14.2.0-mingw-w64ucrt-12.0.0-r2\mingw64**，在Compiler settings頁籤的Other compiler options加入：**-std=c++17**，關閉視窗。按下Set as default。
3. 在左邊Projects視窗在Project的名稱上按下右鍵，選擇Build Options，Selected compiler選WinLibs UCRT 14.2.0

在Search directories頁籤的Compiler頁框加入：

C:\SFML-3.0.0-windows-gcc-14.2.0-mingw-64-bit\SFML-3.0.0\include

在Search directories頁籤的Linker頁框加入：

C:\SFML-3.0.0-windows-gcc-14.2.0-mingw-64-bit\SFML-3.0.0\lib

在Linker Settings頁籤的Link libraries加入以下三行：

C:\SFML-3.0.0-windows-gcc-14.2.0-mingw-64-bit\SFML-3.0.0\lib\libsfml-graphics.a

C:\SFML-3.0.0-windows-gcc-14.2.0-mingw-64-bit\SFML-3.0.0\lib\libsfml-window.a

C:\SFML-3.0.0-windows-gcc-14.2.0-mingw-64-bit\SFML-3.0.0\lib\libsfml-system.a

關閉視窗

1. 複製以下程式並執行，應該看到一個視窗中有1個綠色的球：

#include <SFML/Graphics.hpp>

#include <optional>

int main()

{

// Create a window with resolution 800x600 and 32 bits per pixel.

sf::RenderWindow window(sf::VideoMode(sf::Vector2u(800, 600), 32), "SFML Test");

// Create a circle shape with a radius of 50 pixels.

sf::CircleShape circle(50.f);

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

circle.setPosition({375.f, 275.f});

// Main loop: run until the window is closed.

while (window.isOpen())

{

// Poll events using the new API that returns an std::optional<sf::Event>.

while (const std::optional eventOpt = window.pollEvent())

{

// Use the is<T>() helper function to check if the event is a Closed event.

if (eventOpt->is<sf::Event::Closed>())

{

window.close();

}

// You can also use getIf<T>() to get a pointer to the event subtype, e.g.:

// if (const auto\* keyPressed = eventOpt->getIf<sf::Event::KeyPressed>())

// {

// // process key pressed event

// }

}

window.clear(sf::Color::Black);

window.draw(circle);

window.display();

}

return 0;

}

一張含有 螢幕擷取畫面, 電腦 的圖片

AI 產生的內容可能不正確。

1. 複製以下程式並執行，應該看到一個視窗中有5個不同顏色的球(不會彼此碰撞)：

#include <SFML/Graphics.hpp>

#include <optional>

#include <vector>

#include <random>

#include <cstdint> // For std::uint8\_t

int main()

{

// Window dimensions and ball settings.

const unsigned int WINDOW\_WIDTH = 800;

const unsigned int WINDOW\_HEIGHT = 600;

const float BALL\_RADIUS = 25.f; // Using a smaller radius for variety.

const int NUM\_BALLS = 5;

// Create a window with resolution 800x600 and 32 bits per pixel.

sf::RenderWindow window(sf::VideoMode(sf::Vector2u(WINDOW\_WIDTH, WINDOW\_HEIGHT), 32), "SFML Test");

// Prepare random number generators.

std::random\_device rd;

std::mt19937 gen(rd());

// Ensure the ball starts fully inside the window.

std::uniform\_real\_distribution<float> posXDist(BALL\_RADIUS, WINDOW\_WIDTH - BALL\_RADIUS \* 2);

std::uniform\_real\_distribution<float> posYDist(BALL\_RADIUS, WINDOW\_HEIGHT - BALL\_RADIUS \* 2);

// Random velocity distribution (pixels per second).

std::uniform\_real\_distribution<float> velDist(-200.f, 200.f);

// Random color distribution.

std::uniform\_int\_distribution<int> colorDist(0, 255);

// Vectors to store balls and their velocities.

std::vector<sf::CircleShape> balls;

std::vector<sf::Vector2f> velocities;

// Create NUM\_BALLS balls with random properties.

for (int i = 0; i < NUM\_BALLS; ++i)

{

sf::CircleShape ball(BALL\_RADIUS);

ball.setFillColor(sf::Color(

static\_cast<std::uint8\_t>(colorDist(gen)),

static\_cast<std::uint8\_t>(colorDist(gen)),

static\_cast<std::uint8\_t>(colorDist(gen))

));

ball.setPosition({ posXDist(gen), posYDist(gen) });

balls.push\_back(ball);

// Set a random velocity.

velocities.push\_back({ velDist(gen), velDist(gen) });

}

sf::Clock clock; // Clock to measure elapsed time per frame.

// Main loop: run until the window is closed.

while (window.isOpen())

{

// Poll events using the new API that returns an std::optional<sf::Event>.

while (const std::optional eventOpt = window.pollEvent())

{

// Check if the event is a Closed event.

if (eventOpt->is<sf::Event::Closed>())

{

window.close();

}

}

// Determine the time elapsed since the last frame.

float dt = clock.restart().asSeconds();

// Update the position of each ball.

for (size\_t i = 0; i < balls.size(); ++i)

{

sf::Vector2f pos = balls[i].getPosition();

pos += velocities[i] \* dt;

// Bounce off the left or right boundaries.

if (pos.x < 0)

{

pos.x = 0;

velocities[i].x = -velocities[i].x;

}

else if (pos.x > WINDOW\_WIDTH - BALL\_RADIUS \* 2)

{

pos.x = WINDOW\_WIDTH - BALL\_RADIUS \* 2;

velocities[i].x = -velocities[i].x;

}

// Bounce off the top or bottom boundaries.

if (pos.y < 0)

{

pos.y = 0;

velocities[i].y = -velocities[i].y;

}

else if (pos.y > WINDOW\_HEIGHT - BALL\_RADIUS \* 2)

{

pos.y = WINDOW\_HEIGHT - BALL\_RADIUS \* 2;

velocities[i].y = -velocities[i].y;

}

balls[i].setPosition(pos);

}

// Draw the scene.

window.clear(sf::Color::Black);

for (const auto& ball : balls)

{

window.draw(ball);

}

window.display();

}

return 0;

}

一張含有 螢幕擷取畫面, 鮮豔 的圖片

AI 產生的內容可能不正確。

1. 複製以下程式並執行，應該看到一個視窗中有5個不同顏色的球(彼此碰撞)：

#include <SFML/Graphics.hpp>

#include <optional>

#include <vector>

#include <random>

#include <cstdint>

#include <cmath> // for std::sqrt, std::pow

int main()

{

const unsigned int WINDOW\_WIDTH = 800;

const unsigned int WINDOW\_HEIGHT = 600;

const float BALL\_RADIUS = 25.f;

const int NUM\_BALLS = 5;

sf::RenderWindow window(sf::VideoMode(sf::Vector2u(WINDOW\_WIDTH, WINDOW\_HEIGHT), 32), "SFML Ball Collision");

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_real\_distribution<float> posXDist(BALL\_RADIUS, WINDOW\_WIDTH - BALL\_RADIUS \* 2);

std::uniform\_real\_distribution<float> posYDist(BALL\_RADIUS, WINDOW\_HEIGHT - BALL\_RADIUS \* 2);

std::uniform\_real\_distribution<float> velDist(-200.f, 200.f);

std::uniform\_int\_distribution<int> colorDist(0, 255);

std::vector<sf::CircleShape> balls;

std::vector<sf::Vector2f> velocities;

for (int i = 0; i < NUM\_BALLS; ++i)

{

sf::CircleShape ball(BALL\_RADIUS);

ball.setFillColor(sf::Color(

static\_cast<std::uint8\_t>(colorDist(gen)),

static\_cast<std::uint8\_t>(colorDist(gen)),

static\_cast<std::uint8\_t>(colorDist(gen))

));

ball.setPosition({ posXDist(gen), posYDist(gen) });

balls.push\_back(ball);

velocities.push\_back({ velDist(gen), velDist(gen) });

}

sf::Clock clock;

while (window.isOpen())

{

while (const std::optional eventOpt = window.pollEvent())

{

if (eventOpt->is<sf::Event::Closed>())

{

window.close();

}

}

float dt = clock.restart().asSeconds();

// 🛠️ 更新小球位置

for (size\_t i = 0; i < balls.size(); ++i)

{

sf::Vector2f pos = balls[i].getPosition();

pos += velocities[i] \* dt;

// 碰牆反彈

if (pos.x < 0)

{

pos.x = 0;

velocities[i].x = -velocities[i].x;

}

else if (pos.x > WINDOW\_WIDTH - BALL\_RADIUS \* 2)

{

pos.x = WINDOW\_WIDTH - BALL\_RADIUS \* 2;

velocities[i].x = -velocities[i].x;

}

if (pos.y < 0)

{

pos.y = 0;

velocities[i].y = -velocities[i].y;

}

else if (pos.y > WINDOW\_HEIGHT - BALL\_RADIUS \* 2)

{

pos.y = WINDOW\_HEIGHT - BALL\_RADIUS \* 2;

velocities[i].y = -velocities[i].y;

}

balls[i].setPosition(pos);

}

// 🛠️ 新增：小球之間互相碰撞

for (size\_t i = 0; i < balls.size(); ++i)

{

for (size\_t j = i + 1; j < balls.size(); ++j)

{

sf::Vector2f pos1 = balls[i].getPosition();

sf::Vector2f pos2 = balls[j].getPosition();

// 計算兩球中心的距離

float dx = (pos1.x + BALL\_RADIUS) - (pos2.x + BALL\_RADIUS);

float dy = (pos1.y + BALL\_RADIUS) - (pos2.y + BALL\_RADIUS);

float distance = std::sqrt(dx \* dx + dy \* dy);

// 如果兩球重疊（距離小於兩個半徑之和）

if (distance < BALL\_RADIUS \* 2)

{

// 交換兩球的速度（簡單處理）

std::swap(velocities[i], velocities[j]);

}

}

}

// 🎨 畫畫

window.clear(sf::Color::Black);

for (const auto& ball : balls)

{

window.draw(ball);

}

window.display();

}

return 0;

}

一張含有 螢幕擷取畫面, 電腦, 鮮豔, 作業系統 的圖片

AI 產生的內容可能不正確。