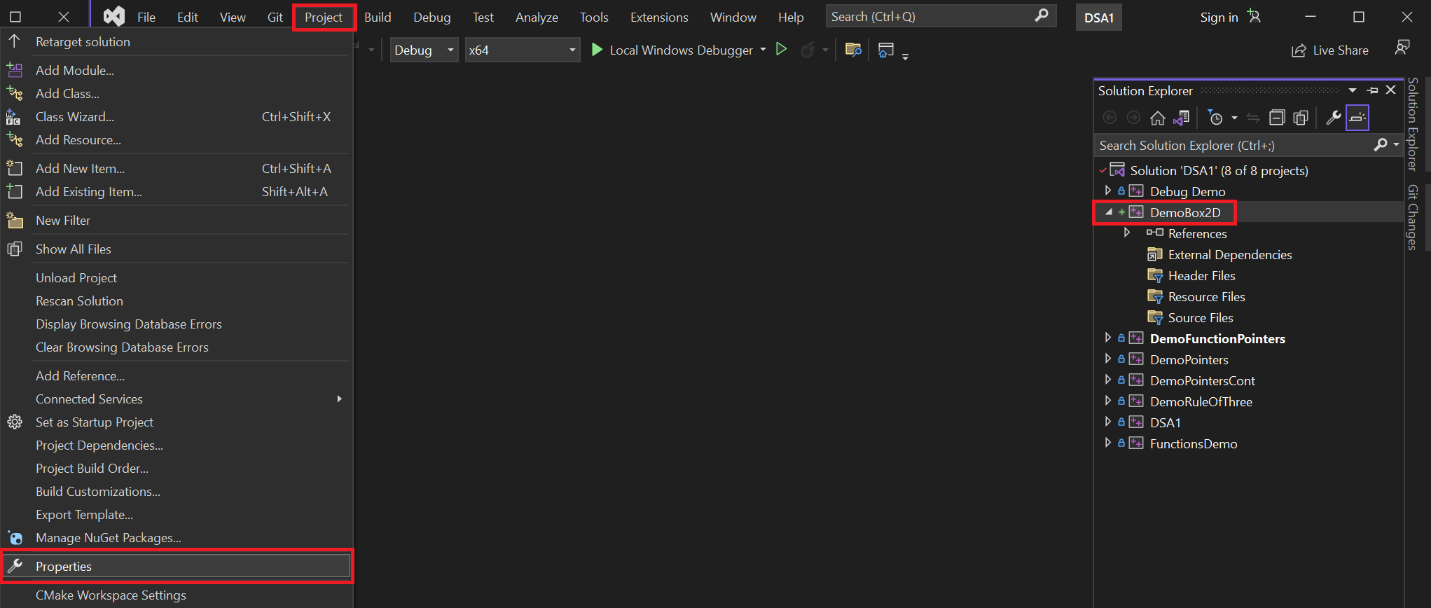
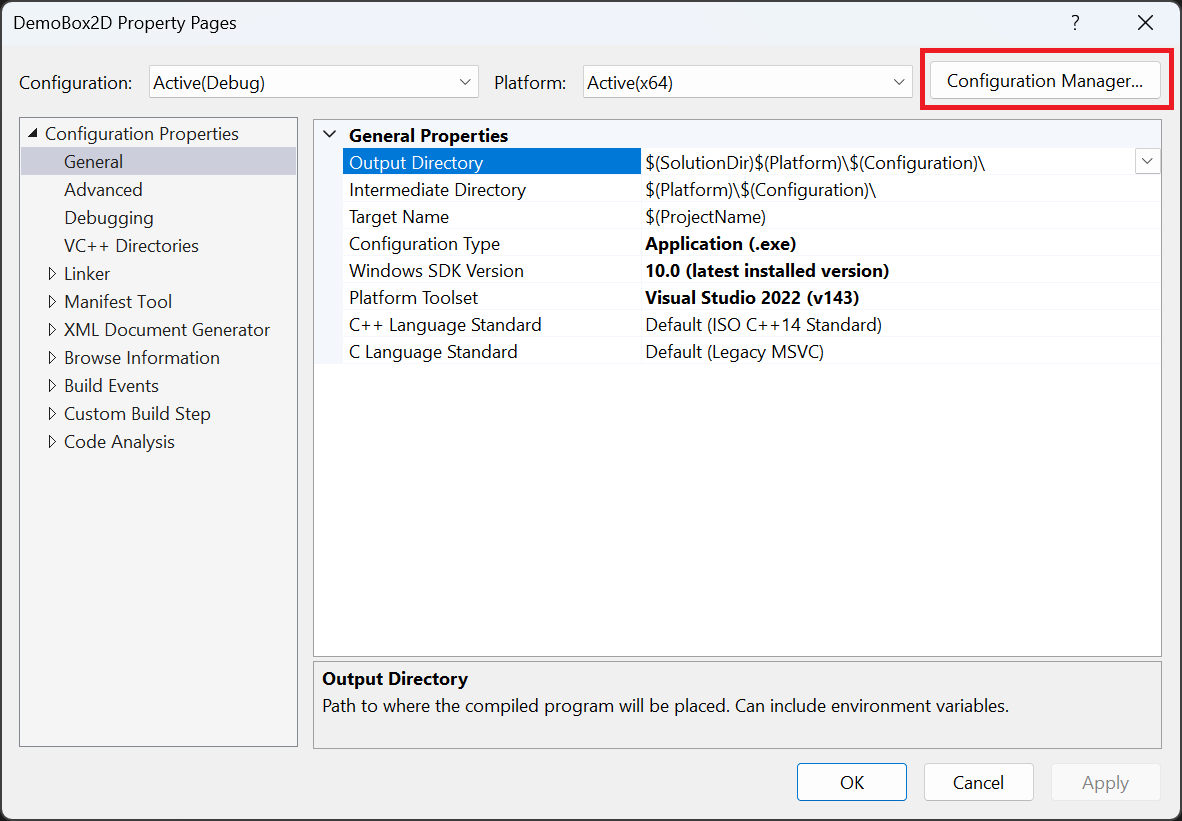
Create a new “Z\_DELETE” folder and a new “include” folder at the same level as your .sln file.

At the same level as your .sln file (solution directory) unzip the lib.zip file and inside of or your include folder unzip the Box2D.zip folder and your SFML.zip folder

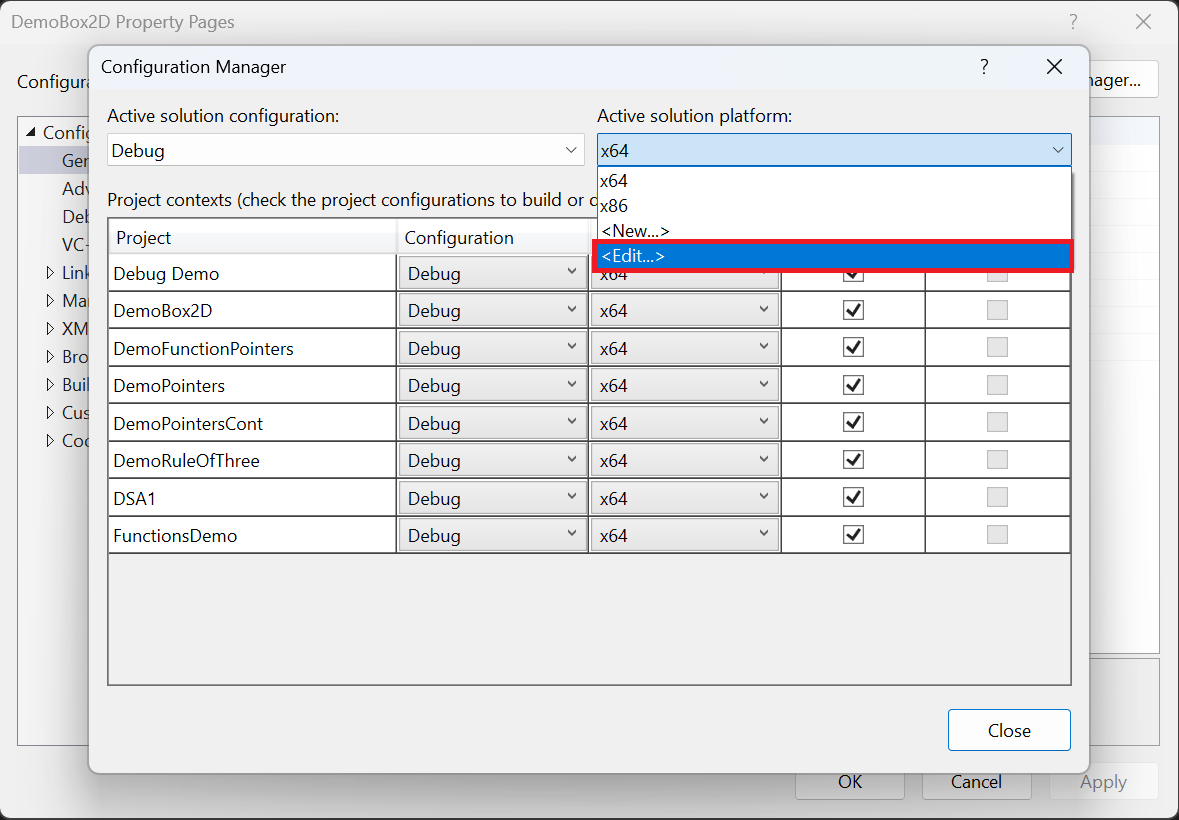
Select the project you want to apply the configuration to and under Project/Properties…



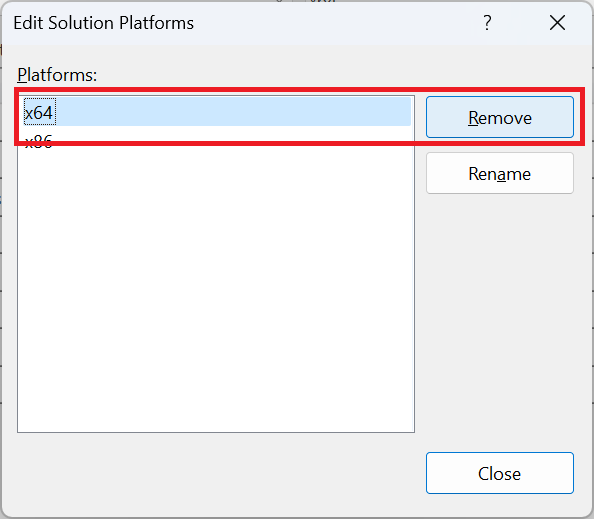
Click on configuration Manager



Edit



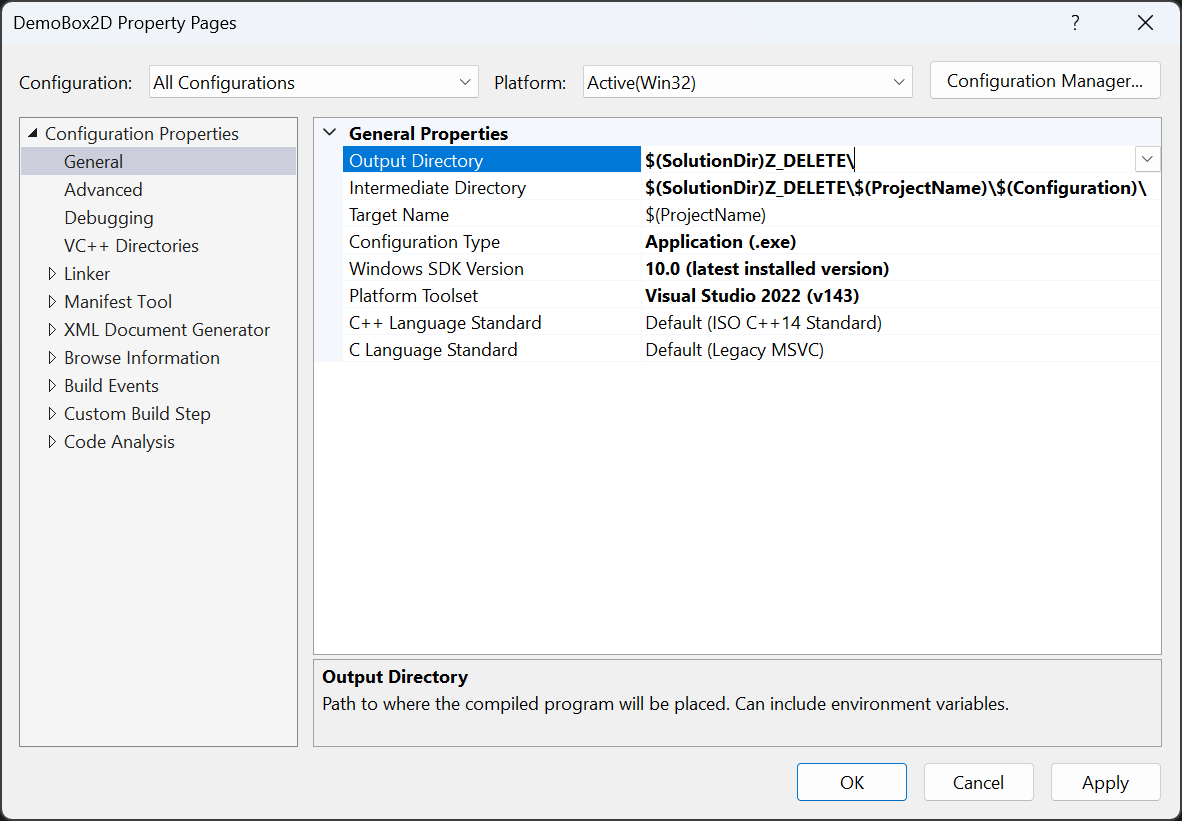
And remove all configurations for x64 versions (the libraries are compiled for x86 only)



Accept and close until you reach the property page again.

Make sure to change the configuration to **All Configurations**

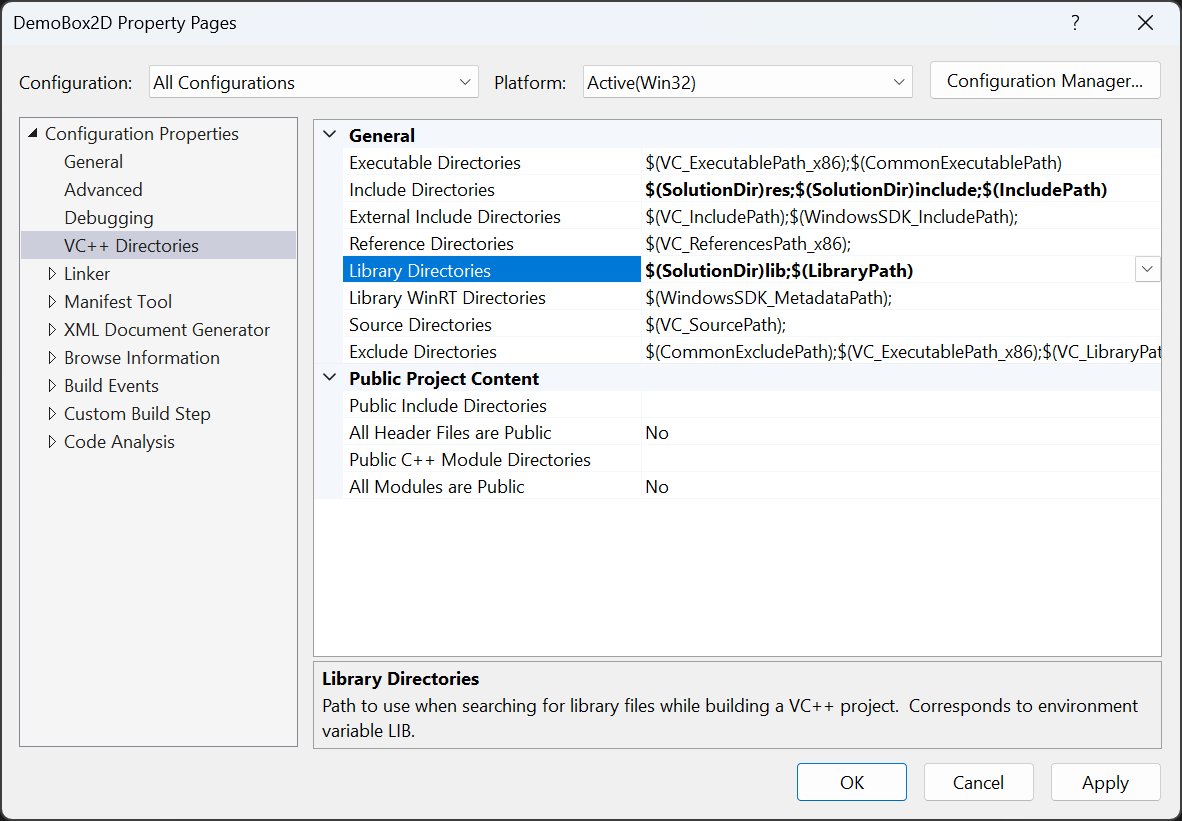
Under General modify the Output Directory and the Intermediate Directory



$(SolutionDir)Z\_DELETE\

$(SolutionDir)Z\_DELETE\$(ProjectName)\$(Configuration)\

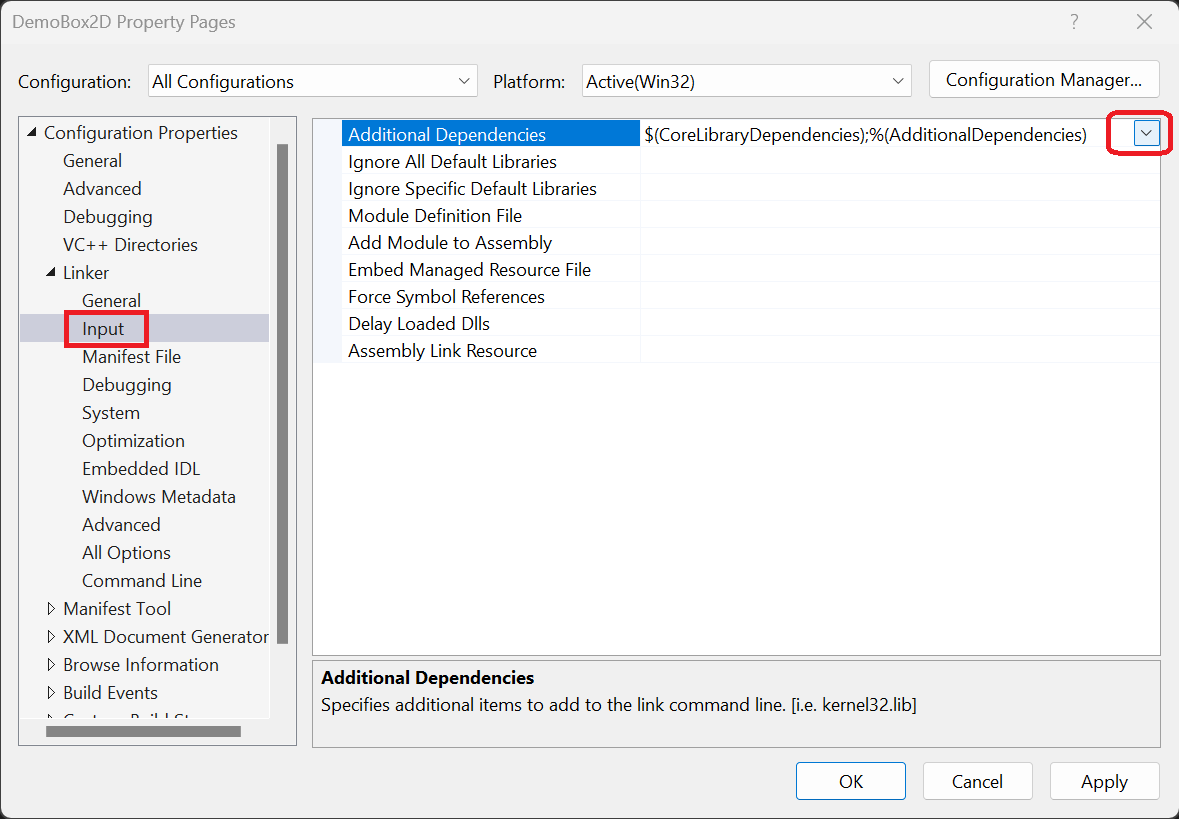
Under VC++ Directores make sure you add the following Include and Library directores:

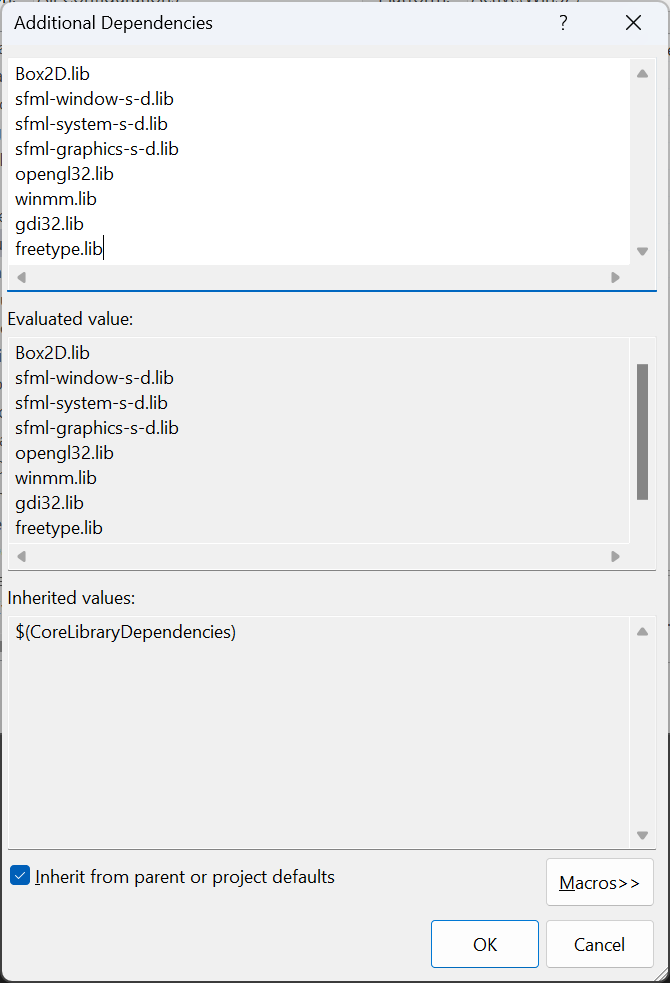


$(SolutionDir)res;$(SolutionDir)include;$(IncludePath)

$(SolutionDir)lib;$(LibraryPath)

Under Linker / Input make sure you add the Additional Dependencies by clicking on the dropdown and edit.



Box2D.lib

sfml-window-s-d.lib

sfml-system-s-d.lib

sfml-graphics-s-d.lib

opengl32.lib

winmm.lib

gdi32.lib

freetype.lib

Add a Main.h file and fill it with:

#pragma once

#include <iostream>

#include "Resource.h"

#define SFML\_STATIC

#include <SFML/Window.hpp>

#include <SFML/Graphics.hpp>

#include <Box2D/Box2D.h>

Add a Main.cpp and fill it with:

#include "Main.h"

/\*

\* This project in meant to show you the functionality of the Box2D library

\*/

int main(void)

{

std::cout << "This project in meant to show you the functionality of the Box2D library" << std::endl;

b2Vec2 gravity(0.0, -9.8); //directional vector for gravity

b2World world(gravity); //Space of simulation

//Ground of the world

b2BodyDef groundBodyDef; //floor

sf::Vector2f floorPos(20.0f, 0.0f); //Position of the floor

sf::Vector2f floorSize(10.0f, 0.5f); //Size of the floor

groundBodyDef.position.Set(floorPos.x, floorPos.y); //Where in the world is located

b2Body\* groundBody = world.CreateBody(&groundBodyDef); //physics body for the ground

b2PolygonShape groundBox;//Shape for this object

groundBox.SetAsBox(floorSize.x / 2.0f, floorSize.y / 2.0f); //Half-width of the ground

groundBody->CreateFixture(&groundBox, 0.0f);//fixed into the world density affects how the object reacts

//Box in the world

b2BodyDef boxDef; //object

sf::Vector2f boxPos(20.0f, 20.0f); //Position of the floor

sf::Vector2f boxSize(1.0f, 1.0f); //Size of the floor

boxDef.type = b2\_dynamicBody; //will it move?

boxDef.position.Set(boxPos.x, boxPos.y); //position in the world

b2Body\* boxBody = world.CreateBody(&boxDef); //physics Body of the box

b2PolygonShape dynamicBox;//Shape of this object

dynamicBox.SetAsBox(boxSize.x / 2.0f, boxSize.y / 2.0f); //Half-width of the box

b2FixtureDef fixtureDef; //fixture of the box

fixtureDef.shape = &dynamicBox; //this is a movable object

fixtureDef.friction = 0.3f; //friction will not really do anything for this application but this is an example

boxBody->CreateFixture(&fixtureDef);

b2Vec2 position; //position of the falling box

sf::Clock deltaClock; //Clock to keep track of our frames

sf::Time deltaTime; //time between frames

// create the window

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

// run the program as long as the window is open

while (window.isOpen())

{

// check all the window's events that were triggered since the last iteration of the loop

sf::Event event;

while (window.pollEvent(event))

{

// "close requested" event: we close the window

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

window.close();

}

deltaTime = deltaClock.getElapsedTime();//check the time between frames

deltaClock.restart(); //restart the clock to measure for the next frame

world.Step(deltaTime.asSeconds(), 6, 2); //Simulate in real time

position = boxBody->GetPosition(); //Get the position of the falling box

std::cout << "(" << position.x << ", " << position.y << ")" << std::endl;

// clear the window with black color

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

//Scaler for displaying purposes

float fScaler = 20.0f;

// ground display

sf::RectangleShape floor(floorSize \* fScaler);

floor.setFillColor(sf::Color(250, 0, 0));

floor.setPosition(floorPos \* fScaler);

floor.setOutlineThickness(1);

floor.setOutlineColor(sf::Color(255, 255, 255));

window.draw(floor);

// falling box display

sf::RectangleShape box(boxSize \* fScaler);

box.setFillColor(sf::Color(0, 0, 255));

box.setPosition(sf::Vector2f(position.x, position.y) \* fScaler);

box.setOutlineThickness(1);

box.setOutlineColor(sf::Color(255, 255, 255));

window.draw(box);

// end the current frame

window.display();

}

return 0;

}

If you did everything right you will see two windows our usual command prompt and a new SFML window like these:

