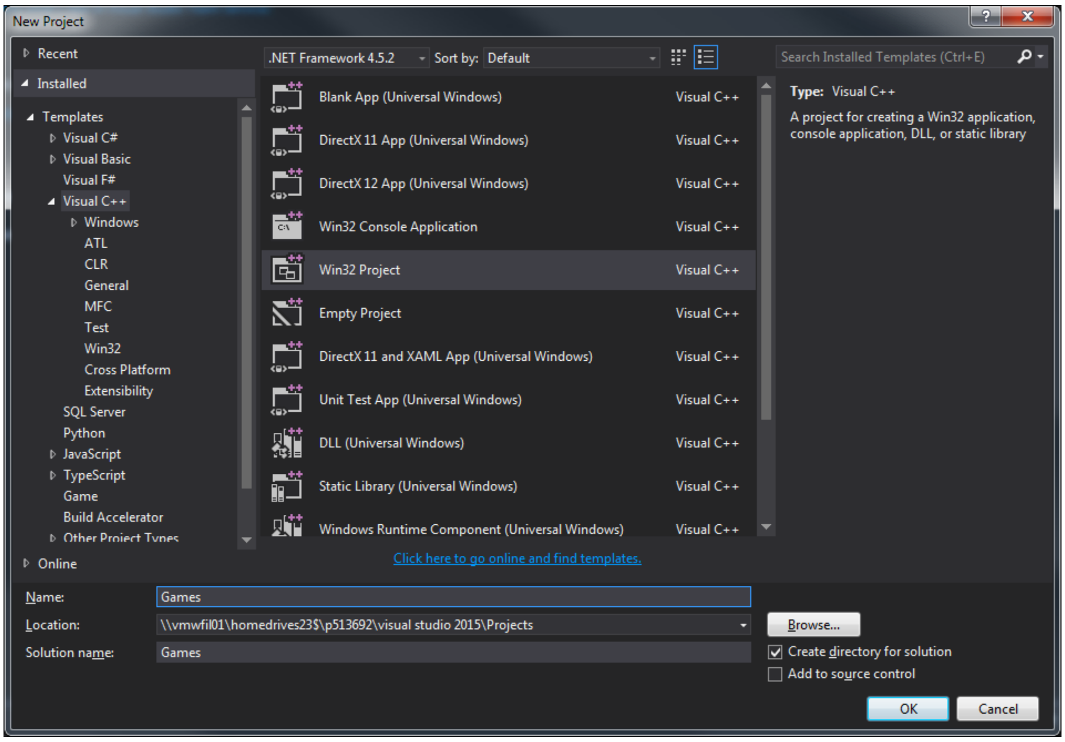
Lab 1 Game Loop

Setting up the project

We are going to create a new project from scratch, linking to the required libraries and rendering a few pieces of simple geometry. First, open Visual Studio 2015 and create a New project – new **Win32 Project**, name it (something sensible, here I name it Games).



New Project

Type: Visual C++

A project for creating a Win32 application, console application, DLL, or static library

Win32 Project — Visual C++

Name: Games

Location: \\vmwfil01\homedrives23$\p513692\visual studio 2015\Projects

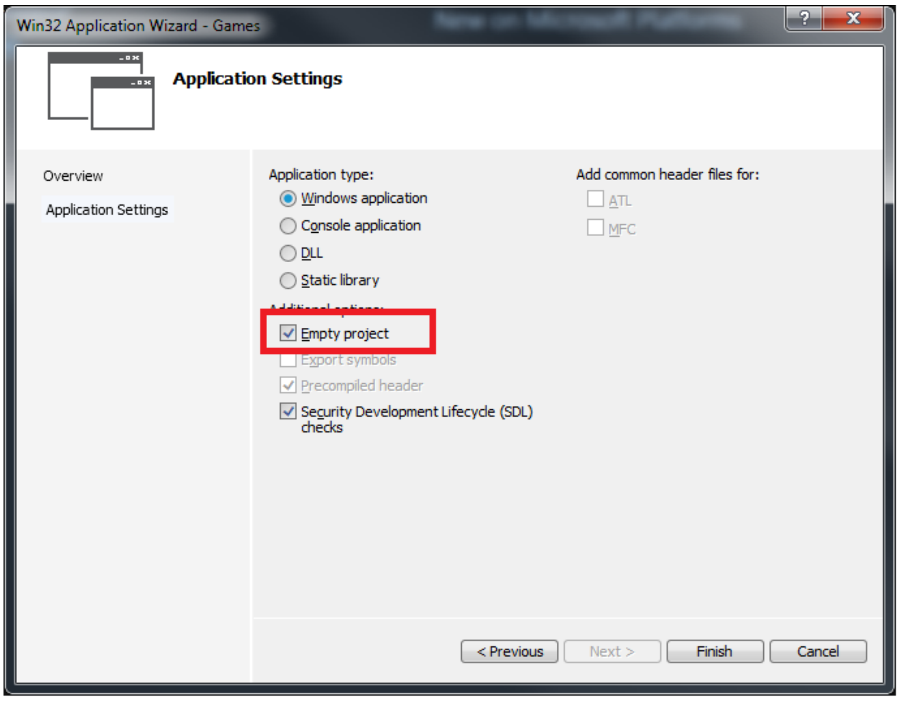
Solution name: Games

☑ Create directory for solution

☐ Add to source control

Recent
Installed
Templates
Visual C#
Visual Basic
Visual F#
Visual C++
Windows
ATL
CLR
General
MFC
Test
Win32
Cross Platform
Extensibility
SQL Server
Python
JavaScript
TypeScript
Game
Build Accelerator
Other Project Types
Online
.NET Framework 4.5.2
Sort by: Default
Search Installed Templates (Ctrl+E)
Blank App (Universal Windows)
DirectX 11 App (Universal Windows)
DirectX 12 App (Universal Windows)
Win32 Console Application
Win32 Project
Empty Project
DirectX 11 and XAML App (Universal Windows)
Unit Test App (Universal Windows)
DLL (Universal Windows)
Static Library (Universal Windows)
Windows Runtime Component (Universal Windows)
Click here to go online and find templates.
Browse...
OK
Cancel

When completing the project wizard make sure to set it as an empty project.



Win32 Application Wizard - Games

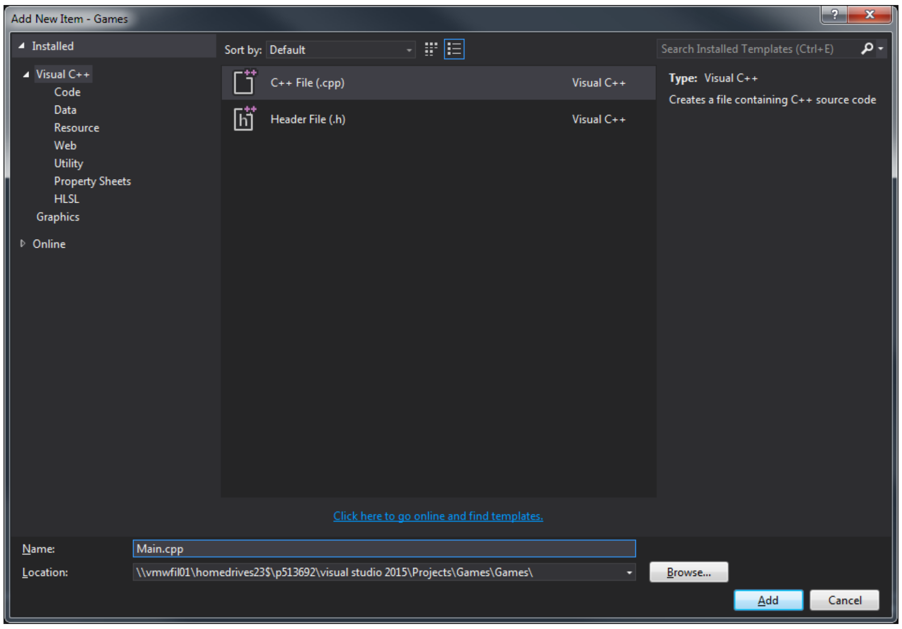
Application Settings

Application type:
- (●) Windows application
- ( ) Console application
- ( ) DLL
- ( ) Static library

Additional options:
- ☑ Empty project
- ☐ Export symbols
- ☑ Precompiled header
- ☑ Security Development Lifecycle (SDL) checks

Add common header files for:
- ☐ ATL
- ☐ MFC

Next we need to add some code files and code. All the code below is based on that discussed in the lecture. First, Add new item: **main.cpp**



Type: Visual C++
Creates a file containing C++ source code

Name: Main.cpp
Location: \\vmwfil01\homedrives23$\p513692\visual studio 2015\Projects\Games\Games\

Add the following code:

```
#include "Game.h"

void main(int argc, char** argv[])
{

    sf::RenderWindow window(sf::VideoMode(800, 600), "Lab 1");

    Game game(&window);

    while (window.isOpen())
    {
        sf::Event event;
        while (window.pollEvent(event))
        {
            switch (event.type)
            {
                case sf::Event::Closed:
                window.close();
                break;
                case sf::Event::Resized:
                window.setView(sf::View(sf::FloatRect(0, 0,
                event.size.width, event.size.height)));
                    break;
                default:
                // don't handle other events
                break;
            }
        }
        game.handleInput();
        game.update();
        game.render();

    }
}
```

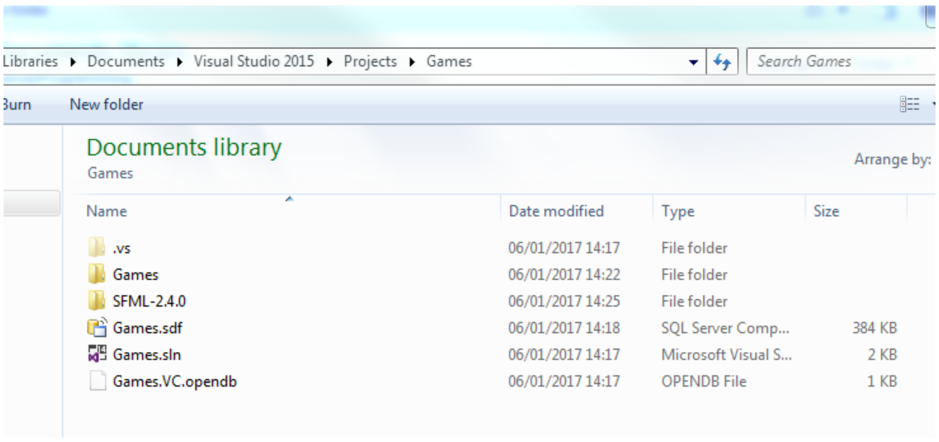
Add a new item: **Game.h** and add the following code:

```
#pragma once

#include <SFML/Graphics.hpp>

class Game {
public:
	Game(sf::RenderWindow* hwnd);
	~Game();

	void handleInput();
	void update();
	void render();

private:
	sf::RenderWindow* window;
	void beginDraw();
	void endDraw();


};
```

Add another new item; **Game.cpp** and add the following code:

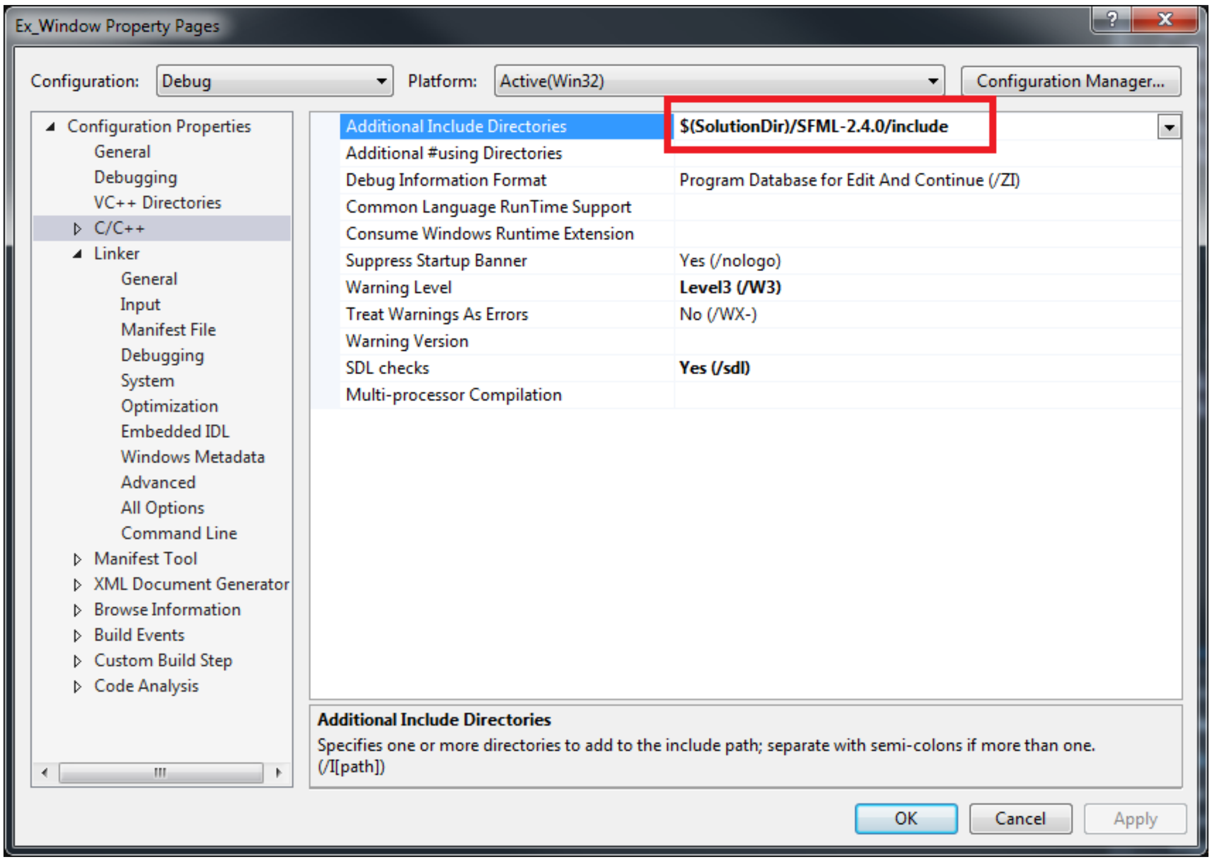
```
#include "Game.h"

Game::Game(sf::RenderWindow* hwnd)
{
	window = hwnd;
}

Game::~Game()
{
}

void Game::update()
{
}

void Game::handleInput()
{
}

void Game::render()
{
	beginDraw();

	endDraw();
}

void Game::beginDraw()
{
	window->clear(sf::Color::Black);
}

void Game::endDraw()
{
	window->display();
}
```

Now we need to setup links to the SFML library. Download a copy of SFML library from Blackboard (link on the module resources page). Extract the zip into the solution folder.


Libraries ▸ Documents ▸ Visual Studio 2015 ▸ Projects ▸ Games
Search Games
Burn
New folder
Documents library
Games
Arrange by:

Name	Date modified	Type	Size
.vs	06/01/2017 14:17	File folder	
Games	06/01/2017 14:22	File folder	
SFML-2.4.0	06/01/2017 14:25	File folder	
Games.sdf	06/01/2017 14:18	SQL Server Comp...	384 KB
Games.sln	06/01/2017 14:17	Microsoft Visual S...	2 KB
Games.VC.opendb	06/01/2017 14:17	OPENDB File	1 KB

Now we need to link Visual Studio with the SFML library, right-click on the project (in visual studio) and got to *properties*. Add an additional include directory: **$(SolutionDir)/SFML-2.4.0/include**

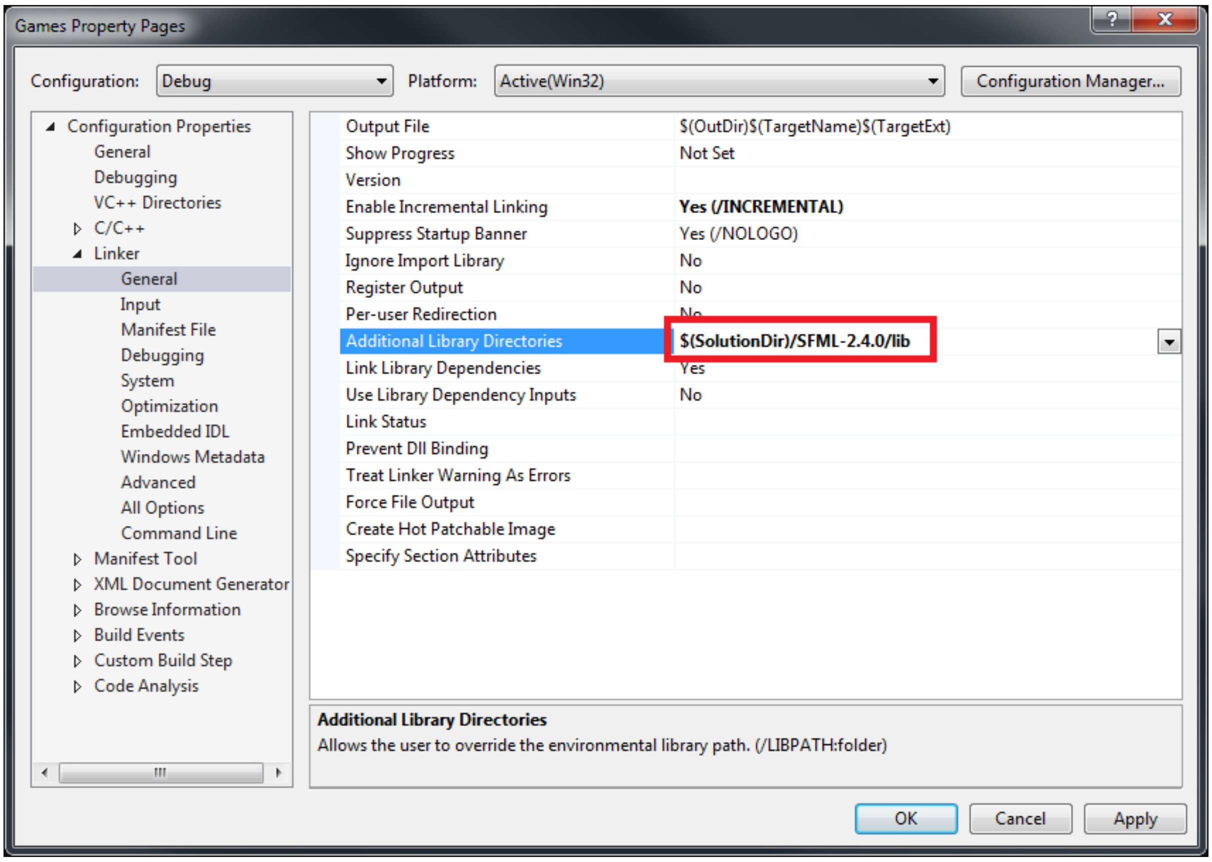


Configuration: Debug Platform: Active(Win32)

Additional Include Directories	**$(SolutionDir)/SFML-2.4.0/include**
Additional #using Directories	
Debug Information Format	Program Database for Edit And Continue (/ZI)
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
Warning Version	
SDL checks	**Yes (/sdl)**
Multi-processor Compilation	

Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])

Under Linker>>General, add and Additional library directory: **$(SolutionDir)/SFML-2.4.0/lib**

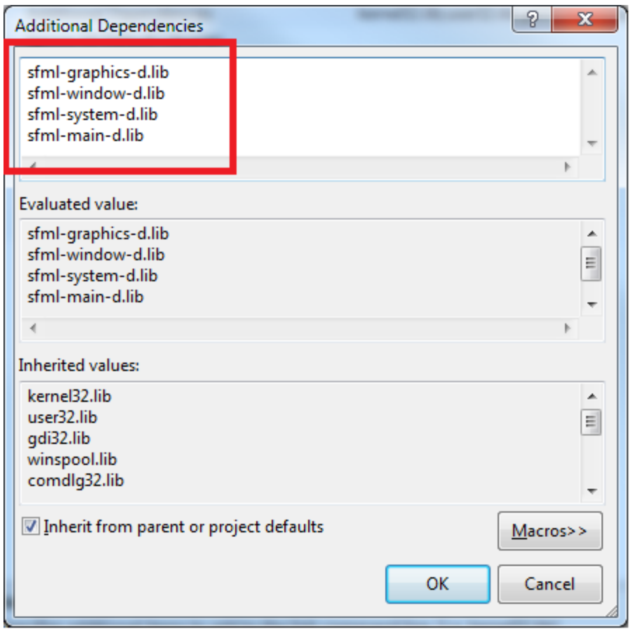


Configuration: Debug
Platform: Active(Win32)

Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	**Yes (/INCREMENTAL)**
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	**$(SolutionDir)/SFML-2.4.0/lib**
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	
Create Hot Patchable Image	
Specify Section Attributes	

Additional Library Directories
Allows the user to override the environmental library path. (/LIBPATH:folder)

Under Linker>>Input we need to add a few additional dependencies:
sfml-graphics-d.lib
sfml-window-d.lib
sfml-system-d.lib
sfml-main-d.lib



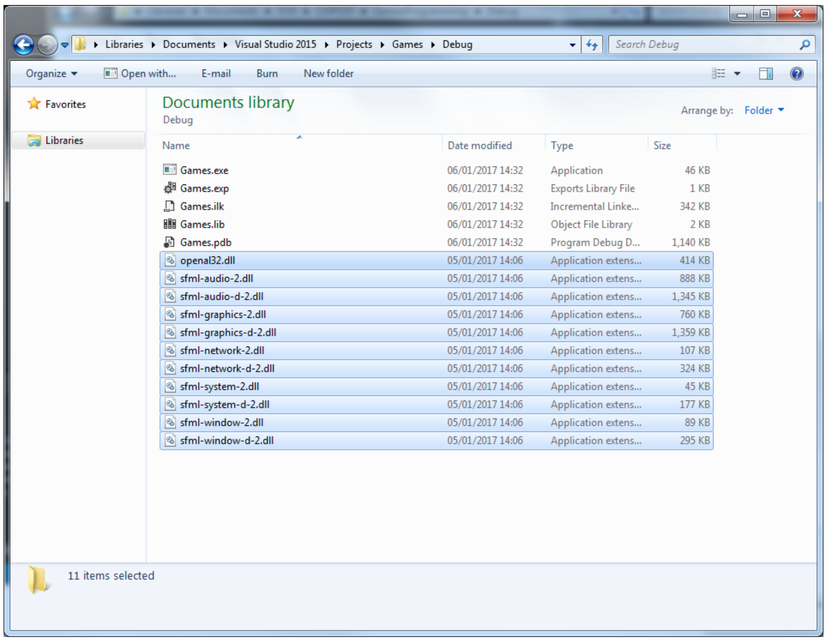
sfml-graphics-d.lib
sfml-window-d.lib
sfml-system-d.lib
sfml-main-d.lib

Evaluated value:
sfml-graphics-d.lib
sfml-window-d.lib
sfml-system-d.lib
sfml-main-d.lib

Inherited values:
kernel32.lib
user32.lib
gdi32.lib
winspool.lib
comdlg32.lib

[✓] Inherit from parent or project defaults

Compile the project. Copy all the files from the SFML-2.4.0/bin into the Debug folder in the solution directory.



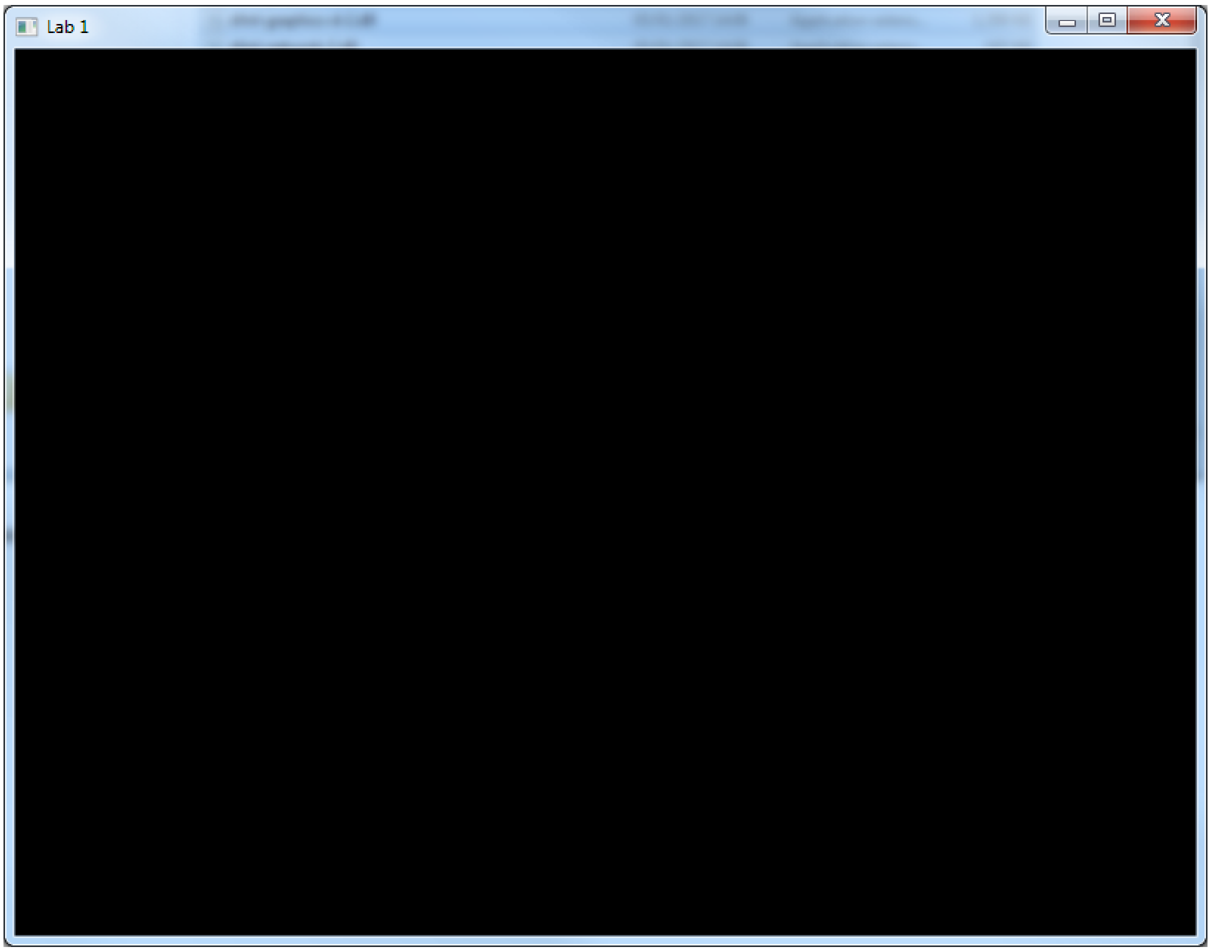
Libraries ▸ Documents ▸ Visual Studio 2015 ▸ Projects ▸ Games ▸ Debug

Documents library
Debug

Name	Date modified	Type	Size
Games.exe	06/01/2017 14:32	Application	46 KB
Games.exp	06/01/2017 14:32	Exports Library File	1 KB
Games.ilk	06/01/2017 14:32	Incremental Linke...	342 KB
Games.lib	06/01/2017 14:32	Object File Library	2 KB
Games.pdb	06/01/2017 14:32	Program Debug D...	1,140 KB
openal32.dll	05/01/2017 14:06	Application extens...	414 KB
sfml-audio-2.dll	05/01/2017 14:06	Application extens...	888 KB
sfml-audio-d-2.dll	05/01/2017 14:06	Application extens...	1,345 KB
sfml-graphics-2.dll	05/01/2017 14:06	Application extens...	760 KB
sfml-graphics-d-2.dll	05/01/2017 14:06	Application extens...	1,359 KB
sfml-network-2.dll	05/01/2017 14:06	Application extens...	107 KB
sfml-network-d-2.dll	05/01/2017 14:06	Application extens...	324 KB
sfml-system-2.dll	05/01/2017 14:06	Application extens...	45 KB
sfml-system-d-2.dll	05/01/2017 14:06	Application extens...	177 KB
sfml-window-2.dll	05/01/2017 14:06	Application extens...	89 KB
sfml-window-d-2.dll	05/01/2017 14:06	Application extens...	295 KB

11 items selected

Now run the project, the result should be a window with a black background.



Rendering basic geometry

To render a rectangle shape, first declare a new variable in the game.h:

```
sf::RectangleShape rect;
```

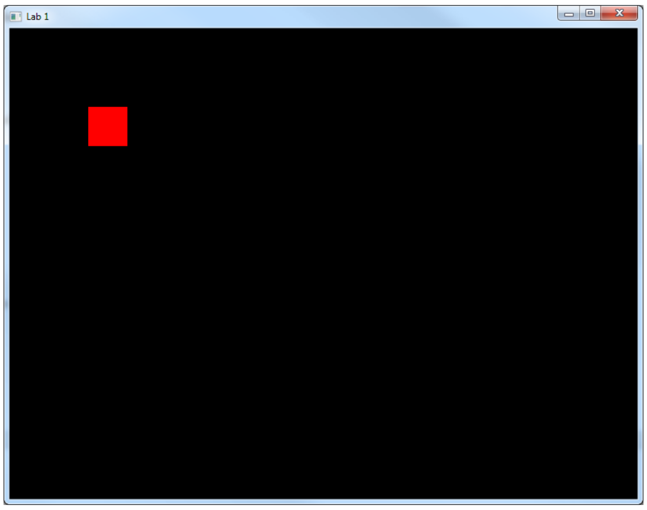
In Game.cpp, in the Game object constructor, we need to initialise the **rect** object, using the following code:

```
rect.setSize(sf::Vector2f(50, 50));
rect.setPosition(100, 100);
rect.setFillColor(sf::Color::Red);
```

Finally, you need to render the **rect** object by calling the draw function in the render function, for example:

```
void Game::render()
{
	beginDraw();

	window->draw(rect);

	endDraw();
}
```

Run the application and it should look like this:



Tasks

1. Add a circle to the level. The circle should be blue with a red outline, with a radius of 4 and should be positioned at the centre of the window.
2. Add a second rectangle; this rectangle should be positioned at the bottom-right corner of the window and should be green.

3. Draw three rectangles on top of each other with different colours to create the following shape: