FREE eBook

LEARNING

sfml

Free unaffiliated eBook created from
Stack Overflow contributors.

#sfml

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: sfml

It is an unofficial and free sfml ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official sfml.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with sfml

Remarks

SFML is multimedia

SFML provides a simple interface to the various components of your PC, to ease the development of games and multimedia applications. It is composed of five modules: system, window, graphics, audio and network.

Discover their features more in detail in the tutorials and the API documentation.

SFML is multi-platform

With SFML, your application can compile and run out of the box on the most common operating systems: Windows, Linux, Mac OS X and soon Android & iOS.

Pre-compiled SDKs for your favorite OS are available on the download page.

SFML is multi-language

SFML has official bindings for the C and .Net languages. And thanks to its active community, it is also available in many other languages such as Java, Ruby, Python, Go, and more.

Learn more about them on the bindings page.

Versions

Version	Release Date
1.0	2007-07-01
1.1	2007-09-18
1.2	2008-07-16
1.3	2008-06-22
1.4	2009-01-07
1.5	2009-06-04
1.6	2010-04-06

Version	Release Date
2.0	2013-04-29
2.1	2013-07-27
2.2	2014-12-17
2.3	2015-05-09
2.3.1	2015-07-11
2.3.2	2015-10-12
2.4.0	2016-08-10
2.4.1	2016-11-04
2.4.2	2017-02-08

Examples

Installation or Setup

- First, download a copy of SFML from the official page.

- Save it anywhere in your computer where it can be easily accessed.
- Open Codeblocks.
- Go to **Project->Build Options->LinkerSettings tab.**
- Click on the **Add** button and go to the bin folder of SFML and select all the files present there.
- Now go to the Search Directories tab and add the 'include' folder of SFML.
- In the same tab, click the sub-tab Linker Settings and add the 'bin' folder.

Basic SFML program

If everything whas been set up correctly, the following snippet will show a window titled "SFML works!" with a green circle:

```
#include <SFML/Graphics.hpp>

int main()
{
    sf::RenderWindow window(sf::VideoMode(200, 200), "SFML works!");
    sf::CircleShape shape(100.f);
    shape.setFillColor(sf::Color::Green);

    while (window.isOpen())
    {
        sf::Event event;
        while (window.pollEvent(event))
```

```
        {
            if (event.type == sf::Event::Closed)
                window.close();
        }

        window.clear();
        window.draw(shape);
        window.display();
    }

    return 0;
}
```

Installation - Linux

There are different approaches to the installation of SFML on Linux:

1. Install it directly from your distribution's package repository
2. Get the source code, build it and install it
3. Download the precompiled SDK and manually copy the files

Option 1 is the preferred one; if the version of SFML that you want to install is available in the official repository, then install it using your package manager. For example, on Debian you would do:

```
sudo apt-get install libsfml-dev
```

Option 2 requires more work: you need to ensure all of SFML's dependencies including their development headers are available, make sure CMake is installed, and manually execute some commands. This will result in a package which is tailored to your system. If you want to go this way, there's a dedicated tutorial on building SFML yourself.

Finally, option 3 is a good choice for quick installation if SFML is not available as an official package. Download the SDK from the download page, unpack it and copy the files to your preferred location: either a separate path in your personal folder (like /home/me/sfml), or a standard path (like /usr/local).

If you already had an older version of SFML installed, make sure that it won't conflict with the new version!

Installation - Windows

The most common way to install SFML on windows is to download the official SDK

You can then unpack the archive and use it in your environment of choice.

vcpkg

Although it's still heavily in development, if you use Visual studio 2017 or newer, you can also

install SFML via vcpkg which integrates with visual studio, greatly simplifying the installation process:

```
vcpkg install sfml
```

Insallation - macOS

First of all you need to download the SFML SDK. Then, in order to start developing SFML applications, you have to install the following items:

Header files and libraries

SFML is available either as dylibs or as frameworks. Only one type of binary is required although both can be installed simultaneously on the same system. We recommend using the frameworks.

Frameworks

- Copy the content of Frameworks to /Library/Frameworks.

dylib

- Copy the content of lib to /usr/local/lib and copy the content of include to /usr/local/include.

SFML dependencies

SFML depends on a few external libraries on Mac OS X. Copy the content of extlibs to /Library/Frameworks.

Xcode templates

If you use Xcode, installing the templates is strongly recommended. Copy the SFML directory from templates to /Library/Developer/Xcode/Templates (create the folders if they don't exist yet).

Hello World in a SFML Window

Let's write a small program which will open a window, and write "Hello World" on the screen.

```
#include <SFML\Graphics.hpp>
#include <cassert>

int main() {
    sf::RenderWindow sfmlWin(sf::VideoMode(600, 360), "Hello World SFML Window");
    sf::Font font;
    //You need to pass the font file location
    if (!font.loadFromFile(/*
```

```
                            Put the filename that identify the font file you want to
load*/"myfont.ttf")) {
        return -1;
    }
    sf::Text message("Hello, World !", font);

    while (sfmlWin.isOpen()) {

        sf::Event e;
        while (sfmlWin.pollEvent(e)) {

            switch (e.type) {
            case sf::Event::EventType::Closed:
                sfmlWin.close();
                break;
            }
        }

        sfmlWin.clear();
        sfmlWin.draw(message);
        sfmlWin.display();
    }
    return 0;
}
```

Let's explain what we did there.

First, we created a `sf::Font` object. We need this object to store the font data that we will use to display the text. After that, we called the `loadFromFile` method, used to load the font in the memory. We should note that SFML don't know about your system fonts, so you need to provide a filename, not a font name

After that, we created a `sf::Text` object. We call a 3 parameter constructor taking :

- The string you want to display
- The font the object will use
- The character size in pixel, which we did not pass here, so il will be set to the default value : 30

Since the `sf::Text` object is ready, we just need to draw it in the main sfml loop, by calling the draw method on the `sfmlWin` render window object that we created before

Read Getting started with sfml online: https://riptutorial.com/sfml/topic/4181/getting-started-with-sfml

Chapter 2: Compile SFML for Android on Windows

Examples

1. Get the Tools

This are the tools you need to build SFML for Android on a Windows Machine

- CMake
- Git
- Android SDK
- Android NDK
- Apache Ant
- MinGW (msys basic)
- Java jre
- Java jdk
- Android USB Driver (Download: http://adbdriver.com/)

Make sure you've installed all tools (Tools -> Android SDK Tools / Platform-tools / Build-tools) in the Android SDK Manager.

If you have installed Visual Studio 2015 you might got some tools from above alerady. If so here are the default directories Visual Studio will put them:

- Android NDK: `C:\ProgramData\Microsoft\AndroidNDK (or AndroidNDK64)`
- Android SDK: `C:\Program Files (x86)\Android\android-sdk`
- Apache Ant: `C:\Program Files (x86)\Microsoft Visual Studio 14.0\Apps`
- Java SE jdk: `C:\Program Files (x86)\Java`
- Git: `C:\Program Files\Git`

2. Adjust your enviroment variables

Add following Paths to the PATH-Enviromentvariable

- `[Path to CMake]\bin`
- `[Path to Git]\bin`
- `[Path to SDK]\tools`
- `[Path to SDK]\platform-tools`
- `[Path to NDK]`
- `[Path to ANT]\bin`
- `[Path to MinGW]\bin`
- `[Path to MinGW]\msys\1.0\bin`
- `[Path to Java jre]\bin`
- `[Path to Java jdk]\bin`

Make sure you use backslashes(`\`) and seperate the paths with semicolons (`;`)!

Add two new enviroment variables

Name: `ANDROID_NDK`
Value: `[Path/to/NDK]`
(e.g. `C:/Android/NDK`)
Make sure you use forwardslashes(`/`)!

Name: `JAVA_HOME`
Value: `[PATH\to\jdk]`
(e.g. `C:\Program Files (x86)\Java\jdk1.7.0_55`)
Make sure you use backslashes(`\`)!

3. Compiling SFML

Clone the SFML Repository from Github.

Enter following comands in a cmd window:

```
git clone https://github.com/SFML/SFML.git SFML
```

If you already downloaded SFML before you can just use the existing one.

Create some folders for the build-files

```
cd SFML
mkdir build && cd build
mkdir armeabi-v7a && cd armeabi-v7a
```

Generate MSYS Makefiles for armeabi-v7a with cmake

```
cmake -DANDROID_ABI=armeabi-v7a -
DCMAKE_TOOLCHAIN_FILE=../../cmake/toolchains/android.toolchain.cmake ../.. -G "MSYS Makefiles"
```

You can exchange `armeabi-v7a` with other architectures as you like.

Compile SFML from the generated makefiles and install it to `$(NDK)/sources` folder.

For this action you probably need administrator privileges. (Run cmd.exe as admin)

```
make && make install
```

You can use `make install` for multiple architectures. It all uses one sfml tag in the `$(NDK)/sources` folder.

4. Build the SFML Android Sample

You can find the Android Sample in `[SFML_ROOT]\examples\android`
You can copy it to leave the SFML repository in it's original state. Open cmd.exe in the sample location.
To get a list of all available Android build targets:

```
android list target
```

Run Update Project for the Sample:

```
android update project --path [Path/to/Android/Sample] --target [targetID]
```

e.g.

```
android update project --path "" --target android-19
```

For path we can use `""` because we are running cmd in the sample path already.

To compile use this command:

```
ndk-build
```

Create the debug (or release) apk:

```
ant debug
ant release
```

Or use this command to directly install it on a device:

```
ant debug install
```

Read Compile SFML for Android on Windows online:
https://riptutorial.com/sfml/topic/4876/compile-sfml-for-android-on-windows

Chapter 3: Window Basics

Remarks

You have to use a `sf::RenderWindow` rather than a `sf::Window`, if you plan on drawing primitives provided by SFML, such as `sf::RectangleShape` or `sf::Sprite`.

Examples

Creating an OpenGL window

Windows in SFML are represented by one of two classes:

- `sf::Window` is a generic window provided by the operating system including an OpenGL render context.
- `sf::RenderWindow` is a specialized version of `sf::Window` that also acts as a `sf::RenderTarget`, allowing SFML's primitives to be rendered to it.

The basic usage is the same in both cases.

```
#include <SFML/Window.hpp>

int main(int argc, char *argv) {
    // Create and initialize a window object
    sf::Window window(sf::VideoMode(640, 480), "My SFML Window");

    // Repeat this as long as the window is open
    while (window.isOpen()) {
        // Handle window events ("event loop")
        sf::Event event;
        while (window.pollEvent(event)) {
            switch(event.type) {
                case sf::Event::Closed: // User tries to close the window
                    window.close(); // Actually close the window
                    break;
            }
        }

        // Render logic would be placed here

        // Swap buffers and update the window
        window.display();
    }
    return 0;
}
```

Read Window Basics online: https://riptutorial.com/sfml/topic/4808/window-basics

Credits

S. No	Chapters	Contributors
1	Getting started with sfml	Community, darkpsychic, Emile Bergeron, Gambit, HatsuPointerKun, Jonny Paton
2	Compile SFML for Android on Windows	Bruno Zell
3	Window Basics	Mario