Week 1

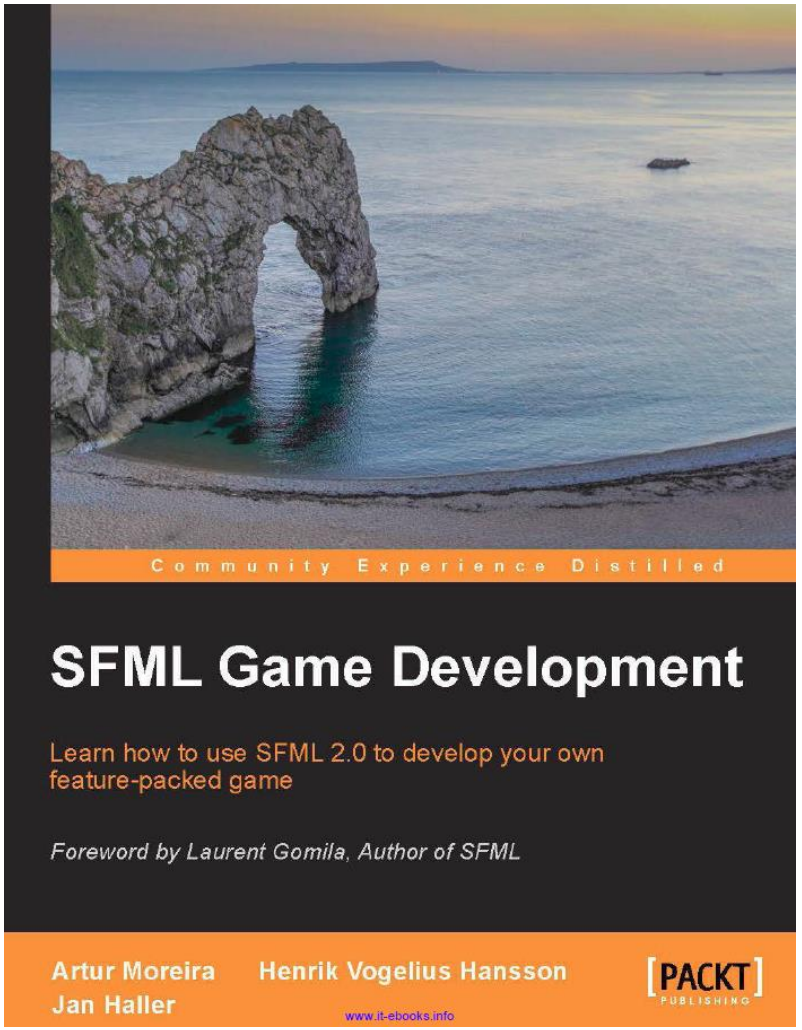
Introduction to SFML

Instructor: Hooman Salamat

2018-01-23

Textbook

- SFML Game Development
- Google it
- https://www.packtpub.com/game-development/sfml-game-development
- Source code on packtpub



Objectives

- Be introduced to SFML or Simple and Fast Multimedia Library, which is a C++ framework
- Learn how to download and install SFML
- Explore an example and see the format of an SFML program
- Examine the Game class of an SFML program

SFML Tutorials

- Before we begin, here is a link to the main SFML tutorial site:
  - http://www.sfml-dev.org/tutorials/2.4/
- Here you can also learn how to setup SFML for your version of Visual Studio – which we will go through in detail this week
- http://sfml-hooman.blogspot.ca/2017/12/setting-up-sfml-242-in-visual-studio.html

SFML and Visual Studio

- Here is a direct link to the Visual Studio page:
  - http://www.sfml-dev.org/tutorials/2.4/start-vc.php

Visual Studio Tips!

- If you choose to link the dynamic libraries, i.e.: *sfml-graphics.lib*, *sfml-window.lib* and *sfml-system.lib*, for **Release** or… *sfml-graphics-d.lib*, *sfml-window-d.lib* and *sfml-system-d.lib* for **Debug**…
  - **Do NOT add SFML_STATIC to the Preprocessor section**
  - **Remember to copy and paste the appropriate DLLs from bin to the same folder as your new .exe!**
- You can use main() instead of WinMain() even after choosing a Windows Application by including the appropriate sfml-main.lib or sfml-main-d.lib in the Linker->Input

Intro to SFML

- SFML is a library which adds multimedia content to your programs built in C++
- Five modules:
  - System
  - Window
  - Graphics
  - Audio
  - Network

- We'll start the course off by working with the first three for a few weeks

System Module

- The system is the core module
  - All other modules are built upon it
- It provides vector classes (2D and 3D), clocks, threads, Unicode strings and other things

- To use in your program:
  - Include sfml-system.lib in your Linker->Input
  - Or sfml-system-d.lib for Debug configuration

Window Module

- This module allows you to create application windows as well as collecting user input, such as mouse movement or key presses
  - You've seen Windows Application in Visual Studio before, but thus far your programs have been exclusively Console Applications
- To use in your program:
  - Include sfml-window.lib in your Linker->Input
  - Or sfml-window-d.lib for Debug configuration

Graphics Module

- The Graphics module allows you to include all functionality related to 2D rendering
  - Using images, texts, shapes and colors
- To use in your program:
  - Include sfml-graphics.lib in your Linker->Input
  - Or sfml-graphics-d.lib for Debug configuration

Audio Module

- The Audio module is, of course, provided so that you can add sounds to your game
  - Covers sound effects and music tracks
- To use in your program:
  - Include sfml-audio.lib in your Linker->Input
  - Or sfml-audio-d.lib for Debug configuration

Network Module

- Yes! SFML has a Network module that will allow you to setup multiplayer games
  - Includes everything you need to communicate over a LAN or the Internet using protocols such as HTTP and FTP
- And yes, we will be covering that in this course!
- To use in your program:
  - Include sfml-network.lib in your Linker->Input
  - Or sfml-network-d.lib for Debug configuration

SFML "Hello World"

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

int main()
{
    sf::RenderWindow window(sf::VideoMode(200, 200), "Hello World!");
    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;
}
```

Tips for Good Coding

- By this point, you should know how to code efficiently and use object-oriented features
- But let's reiterate some good concepts:

- Modularity
  - Keep your code separated into small pieces that perform a particular function
    - Separated into headers and implementation files
    - This will allow you to reuse that code easily, not only in the current program but other programs as well

Tips for Good Coding (cont'd.)

- Abstraction
  - Encapsulate functionality into classes and functions
  - This will prevent code duplications
  - Functions go way back to first term
- Consistency
  - Choose your coding style and stick to it so that it can be read easily and is more professional
  - Usually, this refers to how you use whitespace
  - Also how you use body braces, i.e.: { }

Abstraction into Practice

- To get you more familiar with SFML, we're going to take a minimal example on the next slide and convert the code into a class
- Through this, you should be able to see how we can break down the functionality into pieces and demonstrate how those pieces work together
- So let's get started!

Minimal Example

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

int main()
{
    sf::RenderWindow window(sf::VideoMode(640, 480), "SFML Application");
    sf::CircleShape shape;
    shape.setRadius(40.f);
    shape.setPosition(100.f, 100.f);
    shape.setFillColor(sf::Color::Cyan);
    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();
    }
}
```

Game Class

```
class Game
{
   public:
      Game();
      void run();
   private:
      void processEvents();
      void update();
      void render();
   private:
      sf::RenderWindow mWindow;
      sf::CircleShape mPlayer;
};

int main()
{
   Game game;
   game.run();
}
```

Game Implementation

```
Game::Game()
: mWindow(sf::VideoMode(640, 480), "SFML Application"), mPlayer()
{
   mPlayer.setRadius(40.f);
   mPlayer.setPosition(100.f, 100.f);
   mPlayer.setFillColor(sf::Color::Cyan);
}

void Game::run()
{
   while (mWindow.isOpen())
   {
       processEvents();
       update();
       render();
   }
}
```

Game Implementation (cont'd.)

```
void Game::processEvents()
{
   sf::Event event;
   while (mWindow.pollEvent(event))
   {
      if (event.type == sf::Event::Closed)
            mWindow.close();
   }
}

void Game::update()
{
}
```

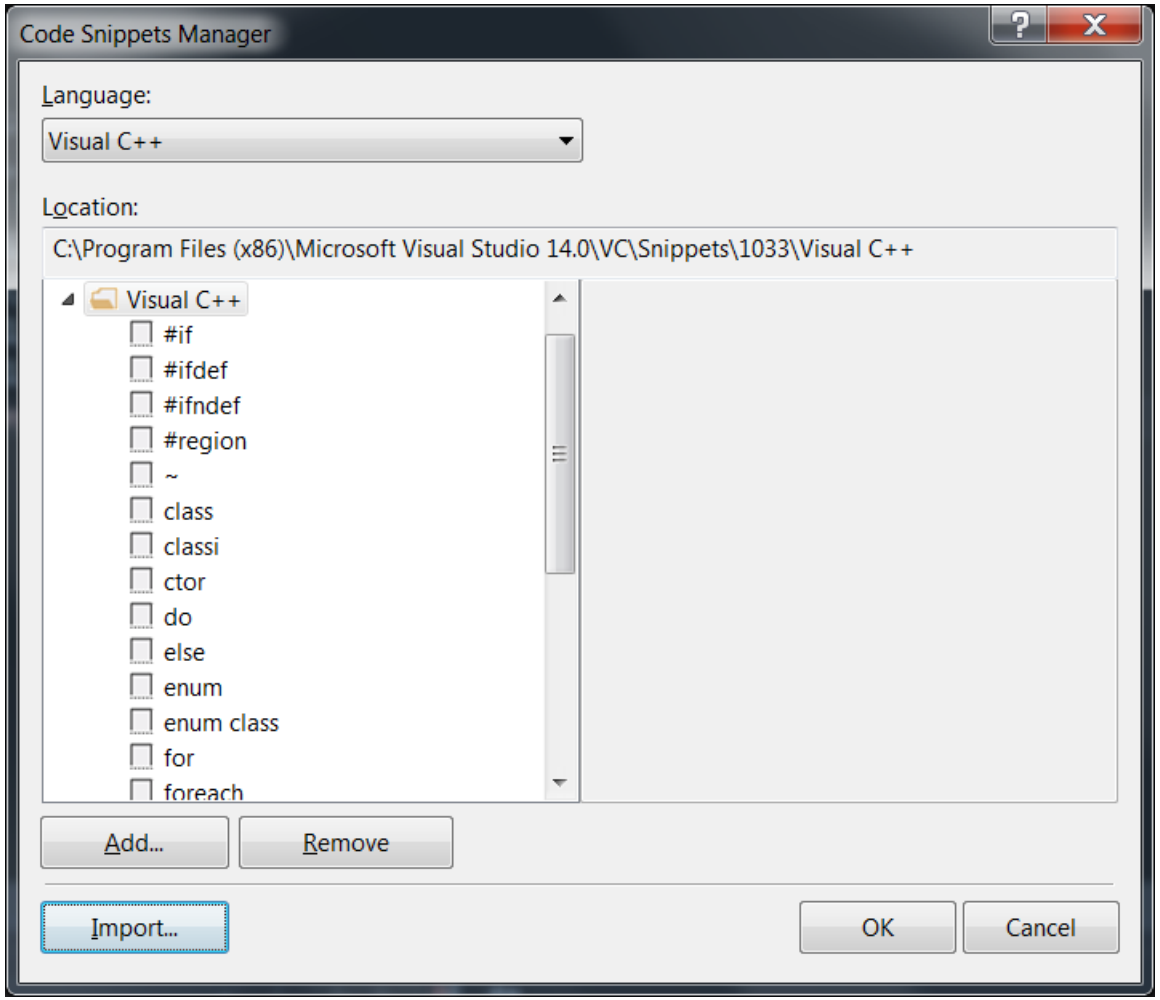
Appendix A – Code Snippet

- Templates you use to insert code
  - Speed up entry of common code constructs
- Named with a short alias
- Activate by either typing the alias, pressing the **Tab** key twice, or using IntelliSense menu to insert the code snippet
  - Pressing **Ctrl+K+X** activates the code snippet list
- Pressing **Tab** twice after selecting the item alias inserts the code snippet text

Code Snippet

- Code snippets are XML files
- Can create your own code snippets
- File needs to end with .snippet filename extension
- You can see all the code snippets that are currently installed, plus their location on disk, by clicking **Tools/Code Snippets Manager**. Snippets are displayed by language.
- You can add and remove snippet directories with the **Add** and **Remove** buttons in the **Code Snippets Manager** dialog. To add individual code snippets, use the **Import** button.

Code Snippet Manager



Code Snippet Template

```
<?xml version="1.0" encoding="utf-8"?>
<CodeSnippets

xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title></Title>
    </Header>
    <Snippet>
      <Code Language="">
        <![CDATA[]]>
      </Code>
    </Snippet>
  </CodeSnippet>
</CodeSnippets>
```

Code Snippets

- All the existing snippets for VS2015 are under C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\Snippets\1033\Visual C++\
- C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Xml\1033\Snippets

Mycpp.snippet

```
<?xml version="1.0" encoding="utf-8" ?>
<CodeSnippets  xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
		<CodeSnippet Format="1.0.0">
			<Header>
				<Title>test1</Title>
				<Shortcut>test1</Shortcut>
				<Description>Code snippet for 'test1'</Description>
				<Author>Hooman Salamat</Author>
			</Header>
			<Snippet>
				<Code Language="cpp"><![CDATA[Console.WriteLine("Hello, World!")]]>
				</Code>
			</Snippet>
		</CodeSnippet>
		<CodeSnippet Format="1.0.0">
			<Header>
				<Title>test2</Title>
				<Shortcut>test2</Shortcut>
				<Description>Code snippet for 'test2'</Description>
				<Author>Hooman Salamat</Author>
			</Header>
			<Snippet>
				<Code Language="cpp"><![CDATA[Console.WriteLine("Bye, World!")]]>
				</Code>
			</Snippet>
		</CodeSnippet>
</CodeSnippets>
```