

24-783 Lecture 01

- Course Overview
- Installing Visual Studio
- Installing XCode
- C++ and OpenGL programming Recap

Course Overview

- C++ Programming
 - Creating a project for Visual C++ and Xcode using Cmake
 - Managing source code with SubVersion
 - Writing, debugging and compiling a code in C++
- Engineering Computation
 - Selecting and/or designing an algorithm, and data structure for engineering applications
 - Understanding the new trends in computation
 - Visualizing with OpenGL and programmable shader
 - Writing re-usable and portable code
 - Coding for portable devices

Teaching Staff

Dr. Soji Yamakawa (Instructor)

Office: Hamerschlag Hall B127

Office Hour: Friday 15:00-16:00

E-Mail: soji@andrew.cmu.edu

Shreyans Kushwah (Graduate Course Assistant)

Office: TBA

Office Hour: TBA

E-Mail: skushwah@andrew.cmu.edu

Rahul Patil (Graduate Course Assistant)

Office: TBA

Office Hour: TBA

E-Mail: rpatil1@andrew.cmu.edu

Grading

A	95% and higher
A-	90-94.99%
B+	87-89.99%
B	84-86.99%
B-	80-83.99%
C+	77-79.99%
C	74-76.99%
C-	70-73.99%
D+	65-69.99%
D	60-64.99%

Final Score = 12% x 6 assignment + 28% Project

Project

- Last month of the semester.
- 4-5 person team
- Pick a topic from:
 - Reading, understanding, and implementing a research paper, or
 - Work on a current open-ended engineering & computational problem.

Course Overview

GOAL:

I want you to be able to ...

- write your own computational tools.
- understand what's in the blackbox.
- write a program that lasts long.

Writing your own computational tools.

- Many computational tools are available for download.
- What if available tools almost achieves your purpose, but not quite?
- Do you give up and wait?

Understanding what's in the blackbox. Make it a white box!

- Don't use an available program as a magic tool.
- Understand what's in the box.

Writing a program that lasts long.

- Ideally, write once, use everywhere, use forever.
- Minimize code rotting.

Current trend of programming – What's making it difficult?

- Too many programming languages:
<http://gizmodo.com/the-nsa-is-funding-a-project-to-roll-all-programming-la-1619295603>
- National Security Administration recognizes too many programming languages are a national security threat. (And the solution is adding one more programming language?)

Current trend of programming

- Many languages are developed for wrong purposes.
 - Gaining more market share.
 - Writing a research paper.
- Those languages have a short life because of the fragile language foundation. Or, the language specification needs to change rapidly. -> Your code stops running!
- A programming language should be designed for the only one purpose: Better programming environment.
- C/C++ were born exactly for the right purpose.
- Many of the short-living trendy languages are lesser copies of C/C++.

Portable and Cross-Platform programming is more important than ever!

- Once it was a Windows world. If your program ran on Windows, no problem. MacOSX, Linux, other UNIX-based OSES were too insignificant.
- That was in the past. Now we have iOS, Android, Mac OSX, Linux. Those OSES are no longer insignificant. Windows is trying hard to stay significant.
- You want to use your program on as many platforms as possible with minimum modification.

Portable and Cross-Platform programming is more important than ever!

- A program should be portable not just between platforms, but between different language versions.
- Poorly designed programming languages may make ad-hoc language-spec change in the newer versions (Objective-C, Python 2.x=>3.x).
- C/C++ are very well designed.
- C had been stable since ANSI standard was published.
- C++ has been expanding its boundaries, but still C++ source code from decades ago can be compiled, unless the source code is touching some undefined and platform-dependent features. (One very common problem was assuming int as 4-byte somewhere, and same size as a pointer somewhere else.)

How to write a portable program then?

- Know and STICK to the language specification.
- Avoid ambiguous and undefined features of the programming language.
- Minimize and isolate platform-dependent parts from the common part.
- Be prepared to write basic libraries by yourself. (Your external libraries may rot.)

GOAL:

I want you to be able to ...

- write your own computational tools.
- understand what's in the blackbox.
- write a program that lasts long.

Installing Visual Studio

- If you use Visual Studio for the assignments, you need to install Visual Studio 2013 or newer.
- There are multiple versions of Visual Studio 2013. Make sure to install Visual Studio 2013 Express for Desktop or Visual Studio 2013 Professional.
- I believe Visual Studio 2015 Community Edition includes everything.
- You can download Visual Studio 2013 from:
<https://www.cmu.edu/computing/software/all/dreamspark/>
- Select Visual Studio Professional 2013, and follow the instructions.

Installing XCode

- If you are using MacOSX, install XCode 5.x or newer.
- You can install XCode from AppStore, or from the following URL:

<http://developer.apple.com>

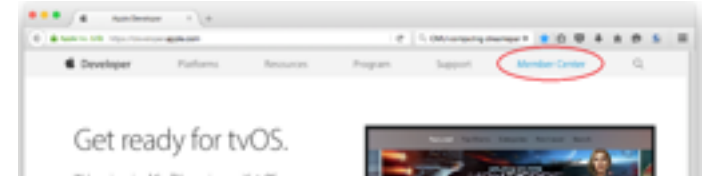
- Also install XCode command-line tools. Download the add-on from:

<http://developer.apple.com>

You will need to create an Apple developer ID for free. Unless you plan to distribute your program commercially, do not create a paid account.

Downloading XCode and Command-Line Tools

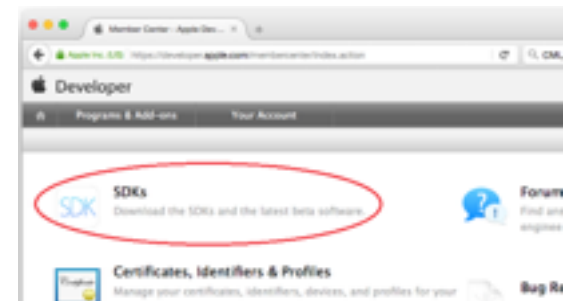
1. Open <https://developer.apple.com>, and click on Member Center



2. Log On with your Apple ID. If you haven't made your Apple ID as a developer ID, create one.

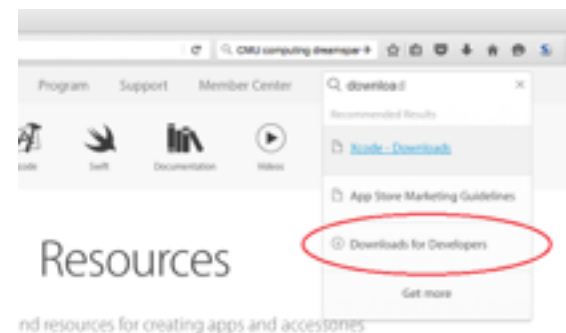


3. Click on SDKs.



4. Search for Downloads, and click on "Downloads for Developers"

5. Download XCode and XCode Command-Line Tools right for your OSX version.



In this course, you will also use CMake and SubVersion.
Installation instruction will be given in the next lecture.

C++ Programming Recap

- Minimum program
- Conditional statements (if and switch)
- Loops (for, while, do-while)
- Basic data types:
 - Integral types – int, short, char, and long long int. Can be signed or unsigned.
 - Floating-point types – float and double.
- Functions
- Call-by-value and call-by-reference
- Arrays
- Pointers and nullptr
- State Transition

OpenGL 1.1

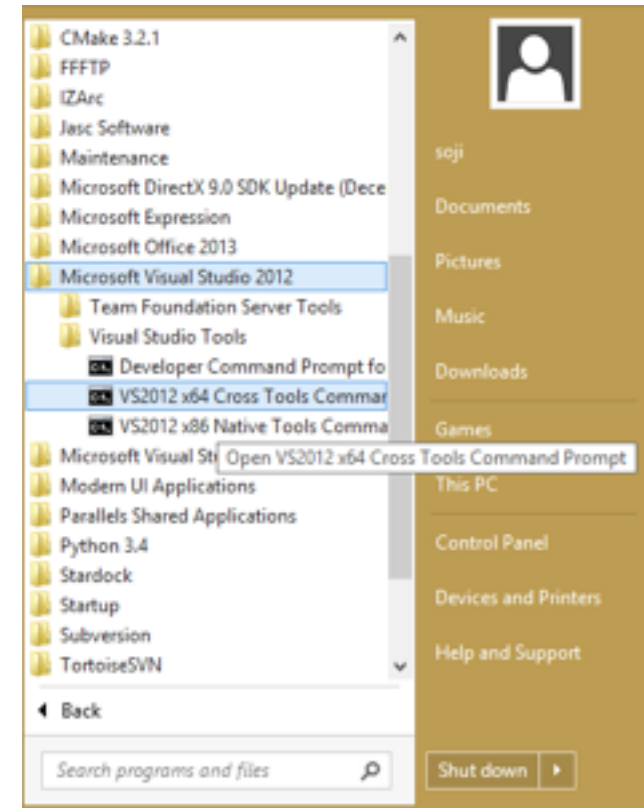
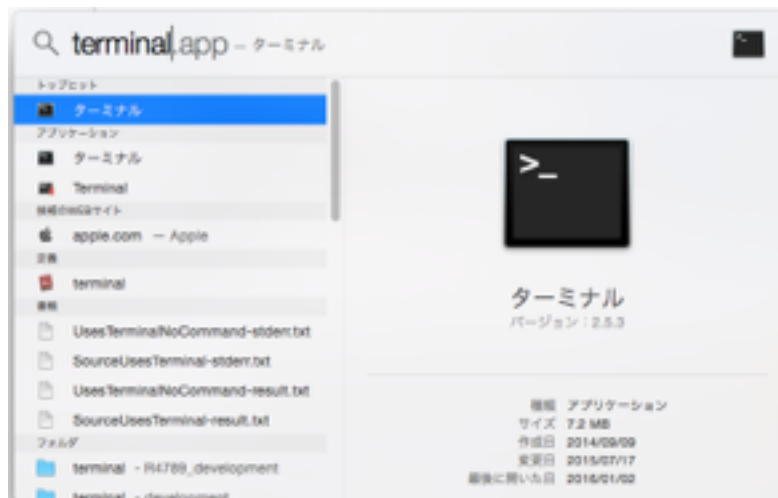
- OpenGL: Industry-standard and cross-platform 2D/3D graphics API.
- OpenGL 1.1 characterized by the fixed-function pipeline and the immediate mode
glBegin, glVertex, glEnd.
- Newer version (2.x and later) can utilized the programmable shader and the vertex arrays. (Will be covered in this class.)

Interactive program

- OpenGL does not have a standard window manager.
- Need to use a separate library for opening a window and taking input.
- External libraries such as:
 - Win32 API / MFC (Windows specific)
 - Cocoa (Mac OSX specific)
 - GLX (Specific to other Unix-based platforms)
 - GLUT (Cross-platform but too inflexible. Also no support for portable devices.)
 - Qt (Cross-platform but unnecessarily huge.)
- 24-780 used FsSimpleWindow framework.
- In this course, we will introduce FsLazyWindow framework, which covers iOS in addition to MacOSX, Linux, and Windows.

Compiling from command

- In Windows
Visual Studio x64 Command Prompt
- In MacOSX
Start Terminal.app from SpotLight.



Benefit

- You don't have to create a project every time you write a new program. (Especially useful when you are writing a very simple test program.)
- It is so quick!
- Can make it a batch process.

In Windows

- Start Visual Studio Command Prompt.
- Let's say you have
 - fssimplewindow.cpp
 - fssimplewindow.h
 - ysglfontdata.c
 - ysglfontdata.h
 - main.cpp

under:

(Your User Profile Directory)\eng_comp\Lecture\01\bounce

Type the following commands:

VS2012 x64 Cross Tools Command Prompt

Active code page: 65001

C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC>pushd %USERPROFILE%

D:\Users\soji>cd eng_comp\Lecture\01\bounce

D:\Users\soji\eng_comp\Lecture\01\bounce>cl main.cpp fssimplewindow.cpp ysglfontdata.c

Microsoft (R) C/C++ Optimizing Compiler Version 17.00.61030 for x64

Copyright (C) Microsoft Corporation. All rights reserved.

main.cpp

fssimplewindow.cpp

Generating Code...

Compiling...

ysglfontdata.c

Generating Code...

Microsoft (R) Incremental Linker Version 11.00.61030.0

Copyright (C) Microsoft Corporation. All rights reserved.

/out:main.exe

main.obj

fssimplewindow.obj

ysglfontdata.obj

LINK : warning LNK4031: no subsystem specified; CONSOLE assumed

D:\Users\soji\eng_comp\Lecture\01\bounce>main.exe

- `pushd`

Changes the current working directory. Can quickly come back with `popd`. Also can change across the current drives.

- `cd`

Changes the current working directory. If you want to change the current drive with this command, add `/D` option.

- `cl`

Compile with Visual C++.

In MacOSX

- A little bit of additional steps.
 - .cpp files and .c/.m files need to be compiled separately.
 - The executables of a graphical program need to be stored in a specific directory structure called Bundle.
 - program.app
 - Contents
 - Resources
 - MacOS
 - (executable file)
 - Data files required for the program must be stored under the Resources directory.

- Let's say you have source files and header files under:
 ~/eng_comp/Lecture/01/bounce
 (~ means your user directory in the Unix environment.)
- First you compile .c and .m files and create .o files.
- Then compile .cpp file and link together with .o files.

Type:

```
bounce — bash — 130x60

[~] % cd ~/eng_comp/Lecture/01/bounce
[~/eng_comp/Lecture/01/bounce] % clang -c fssimplewindowobjc.m
fssimplewindowobjc.m:1114:12: warning: 'loadNibNamed:owner:' is deprecated: first deprecated in OS X 10.8
    [NSBundle loadNibNamed:@"MainMenu" owner:NSApp];
               ^
/System/Library/Frameworks/AppKit.framework/Headers/NSNibLoading.h:27:1: note: 'loadNibNamed:owner:' has been explicitly marked
deprecated here
+ (BOOL)loadNibNamed:(NSString *)nibName owner:(id)owner NS_DEPRECATED_MAC(10_0, 10_8); // Deprecated in Mac OS X 10.8
^
fssimplewindowobjc.m:1118:22: warning: sending 'YsMacDelegate *' to parameter of incompatible type 'id<NSFileManagerDelegate>'
    [NSApp setDelegate: ysDelegate];
                   ^~~~~~
/System/Library/Frameworks/Foundation.framework/Headers/NSFileManager.h:109:47: note: passing argument to parameter 'delegate'
here
@property (assign) id <NSFileManagerDelegate> delegate NS_AVAILABLE(10_5, 2_0);
^
fssimplewindowobjc.m:1380:13: warning: 'convertScreenToBase:' is deprecated: first deprecated in OS X 10.7 - Use
-convertRectFromScreen: instead [-Wdeprecated-declarations]
    loc=[ysWnd convertScreenToBase:loc];
           ^
/System/Library/Frameworks/AppKit.framework/Headers/NSWindow.h:662:1: note: 'convertScreenToBase:' has been explicitly marked
deprecated here
- (NSPoint)convertScreenToBase:(NSPoint)aPoint NS_DEPRECATED_MAC(10_0, 10_7, "Use -convertRectFromScreen: instead");
^
fssimplewindowobjc.m:1381:14: warning: 'convertPointFromBase:' is deprecated: first deprecated in OS X 10.7
    [loc=[ysView convertPointFromBase:loc];
           ^
/System/Library/Frameworks/AppKit.framework/Headers/NSView.h:248:1: note: 'convertPointFromBase:' has been explicitly marked
deprecated here
- (NSPoint)convertPointFromBase:(NSPoint)aPoint NS_DEPRECATED_MAC(10_5, 10_7);
^
4 warnings generated.
[~/eng_comp/Lecture/01/bounce] % clang -c ysglfontdata.c
[~/eng_comp/Lecture/01/bounce] % mkdir -p program.app/Contents/MacOS
[~/eng_comp/Lecture/01/bounce] % clang main.cpp fssimplewindowcpp.cpp -std=c++11 -lstdc++ fssimplewindowobjc.o ysglfontdata.o -o p
rogram.app/Contents/MacOS/main -framework Cocoa -framework OpenGL
[~/eng_comp/Lecture/01/bounce] % program.app/Contents/MacOS/main
```

(Shorter version)

```
bounce — bash — 130x60


[~] % cd ~/eng_comp/Lecture/01/bounce/
[~/eng_comp/Lecture/01/bounce] % clang -c *.c *.m
fssimplewindowobjc.m:1114:12: warning: 'loadNibNamed:owner:' is deprecated: first deprecated in OS X 10.8
    [NSBundle loadNibNamed:@"MainMenu" owner:NSApp];
               ^
/System/Library/Frameworks/AppKit.framework/Headers/NSNibLoading.h:27:1: note: 'loadNibNamed:owner:' has been explicitly marked
deprecated here
+ (BOOL)loadNibNamed:(NSString *)nibName owner:(id)owner NS_DEPRECATED_MAC(10_0, 10_8); // Deprecated in Mac OS X 10.8
^
fssimplewindowobjc.m:1118:22: warning: sending 'YsMacDelegate *' to parameter of incompatible type 'id<NSFileManagerDelegate>'
    [NSApp setDelegate: ysDelegate];
                   ^
/System/Library/Frameworks/Foundation.framework/Headers/NSFileManager.h:109:47: note: passing argument to parameter 'delegate'
here
@property (assign) id <NSFileManagerDelegate> delegate NS_AVAILABLE(10_5, 2_0);
^
fssimplewindowobjc.m:1380:13: warning: 'convertScreenToBase:' is deprecated: first deprecated in OS X 10.7 - Use
    -convertRectFromScreen: instead [-Wdeprecated-declarations]
    loc=[ysWnd convertScreenToBase:loc];
               ^
/System/Library/Frameworks/AppKit.framework/Headers/NSWindow.h:662:1: note: 'convertScreenToBase:' has been explicitly marked
deprecated here
- (NSPoint)convertScreenToBase:(NSPoint)aPoint NS_DEPRECATED_MAC(10_0, 10_7, "Use -convertRectFromScreen: instead");
^
fssimplewindowobjc.m:1381:14: warning: 'convertPointFromBase:' is deprecated: first deprecated in OS X 10.7
    [ysView convertPointFromBase:loc];
               ^
/System/Library/Frameworks/AppKit.framework/Headers/NSView.h:248:1: note: 'convertPointFromBase:' has been explicitly marked
deprecated here
- (NSPoint)convertPointFromBase:(NSPoint)aPoint NS_DEPRECATED_MAC(10_5, 10_7);
^
4 warnings generated.
[~/eng_comp/Lecture/01/bounce] % mkdir -p program.app/Contents/MacOS
[~/eng_comp/Lecture/01/bounce] % clang *.cpp -std=c++11 -lstdc++ *.o -framework Cocoa -framework OpenGL -o program.app/Contents/Ma
cOS/main
[~/eng_comp/Lecture/01/bounce] % program.app/Contents/MacOS/main
```


Command-Line Arguments


- Another benefit of running from the command line.
- You can pass many information by command parameters.
(File names, options, etc.)

```
int main(int argc, char *argv[])
```

```
{  
}
```



Number of arguments
including the command
itself.



Arguments. The first one is
how the command is started.
(Exactly as you type.)

Command-line Arguments

- Example: Just print command arguments.

```
#include <stdio.h>
```

```
int main(int ac,char *av[])
```

```
{
```

```
    for(int i=0; i<ac; ++i)
```

```
    {
```

```
        printf("%d: %s\n",i,av[i]);
```

```
    }
```

```
    return 0;
```

```
}
```