Hi, all

As most members of our team are familiar with Windows and Mac, so I will introduce configuration in Windows as well as in Mac(I did not have Mac so I could not try whether it works well.)

The basic tools we will use is:

code editing tools: Used to edit our source code.

Any text edit tools, but I recommend use programming developers since they can highlight class and functions.

Compile tools: Used to compile source code and generate an executable file.

Visual Studio for Windows, cmake for any platform(Windows, Linux, Mac). Before you want to install cmake in windows make sure that you have installed Visual Studio or other compilers. Since we have already install the VS in our system so we can just use VS to compile the source code in our laptop and use cmake in our lab computers.

Commit tools: Used to control code version, share resource.

Git and SVN. We choose Git.

Library: SFML, we may need other library later.

Our step will be: **Compile tools -----> Library ------->Commit tools**

Visual Studio on Windows (Thanks to Brandon):

Instructions on how to get Visual Studio 2013 on Windows:

-Go to the dreamspark website (DreamSpark) and create a student account (all you need is an active .edu address)
-click the software catalog link in the top menu bar
-Click Visual Studio Professional with Update 3
-Under Select Product Version pick 32 bit(depends on your machine. I am the 64-bit laptop),
-Click Download and follow the prompts. (you'll be asked to download a secure download manager)
-After following the prompts you should have an .iso file saved on your computer.

Skip the following step if you have Windows 8
-Download Virtual CloneDrive from Virtual CloneDrive
-For .iso file you can use demon tools to install it.
http://www.daemon-tools.cc/downloads

Continue from here:
-If you have Windows 8 click on the .iso file and select mount from the menu bar / If you don't have Windows 8 right click on the .iso file and select mount
-There should be a setup.exe file inside of the mounted drive, run it and after ~30 minutes Visual Studio 2013 will be installed on your computer.

To compile SFML on Windows go to this website and follow the instructions SFML Windows Binaries + Template

Cmake configuration(Windows and Mac, not required if you have VS)

-Download cmake from the address
http://www.cmake.org/cmake/resources/software.html

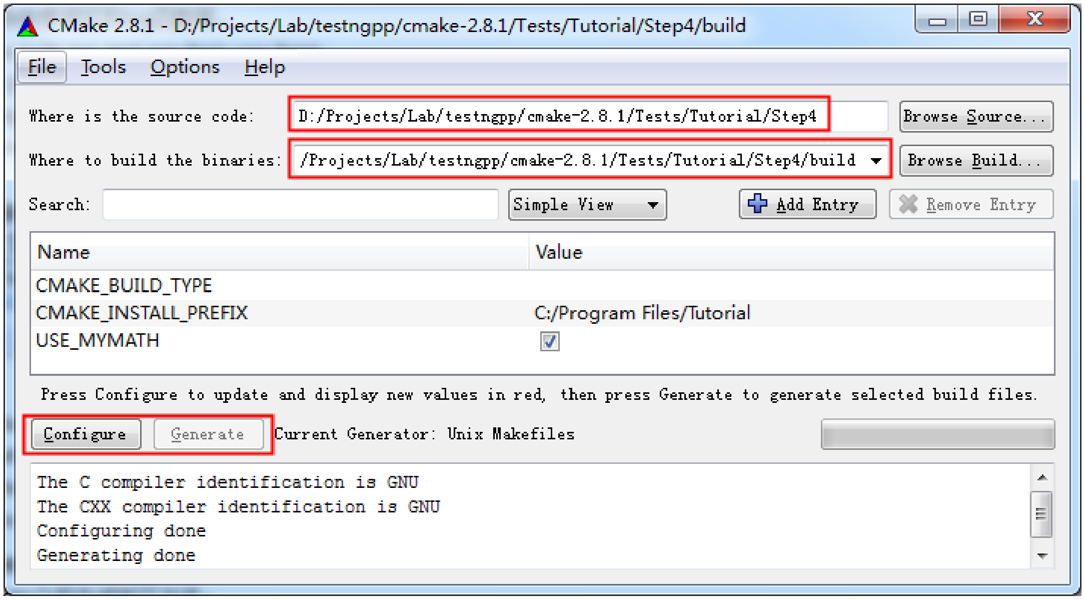
-You can use **binary distributions** and **source distributions**(With source distribution we need to build the source distributions)

-To build the **source distributions**, unpack themwith zip or tar and follow the instructions in Readme.txt at the top of source tree.

-To build the **binary distributions** you can just install with the file.

For example, the GUI for cmake will be as:

cmake-2.8.1-win32-x86\bin\cmake-gui.exe



CMake 2.8.1 - D:/Projects/Lab/testngpp/cmake-2.8.1/Tests/Tutorial/Step4/build

Where is the source code: D:/Projects/Lab/testngpp/cmake-2.8.1/Tests/Tutorial/Step4

Where to build the binaries: /Projects/Lab/testngpp/cmake-2.8.1/Tests/Tutorial/Step4/build

Search:

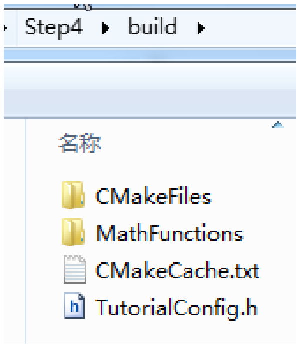
Name	Value
CMAKE_BUILD_TYPE	
CMAKE_INSTALL_PREFIX	C:/Program Files/Tutorial
USE_MYMATH	☑

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles

```
The C compiler identification is GNU
The CXX compiler identification is GNU
Configuring done
Generating done
```

after executing configure:

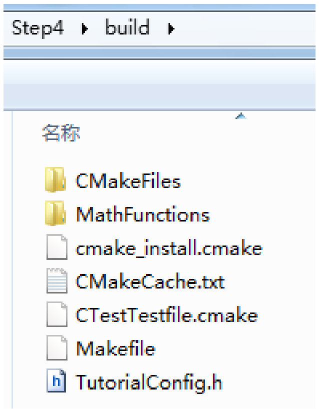


Step4 ▸ build ▸

名称

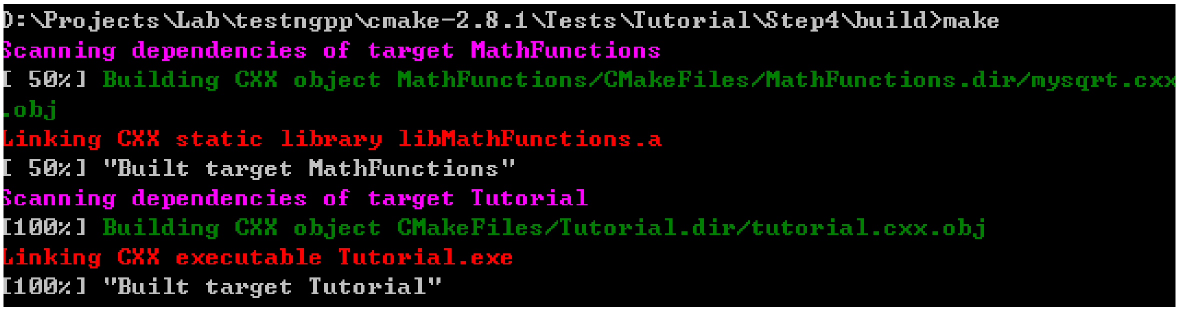
CMakeFiles
MathFunctions
CMakeCache.txt
TutorialConfig.h

execute generate, you can get Makefile:



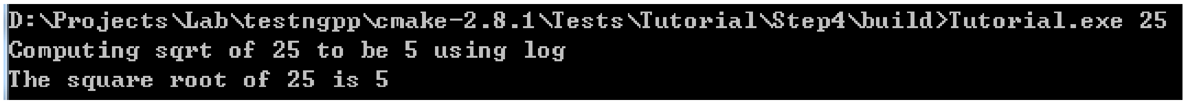
```
CMakeFiles
MathFunctions
cmake_install.cmake
CMakeCache.txt
CTestTestfile.cmake
Makefile
TutorialConfig.h
```

compile with make command:



```
D:\Projects\Lab\testngpp\cmake-2.8.1\Tests\Tutorial\Step4\build>make
Scanning dependencies of target MathFunctions
[ 50%] Building CXX object MathFunctions/CMakeFiles/MathFunctions.dir/mysqrt.cxx
.obj
Linking CXX static library libMathFunctions.a
[ 50%] "Built target MathFunctions"
Scanning dependencies of target Tutorial
[100%] Building CXX object CMakeFiles/Tutorial.dir/tutorial.cxx.obj
Linking CXX executable Tutorial.exe
[100%] "Built target Tutorial"
```

Then you will generate Tutorial.exe, execute Tutorial.exe 25, you will see:



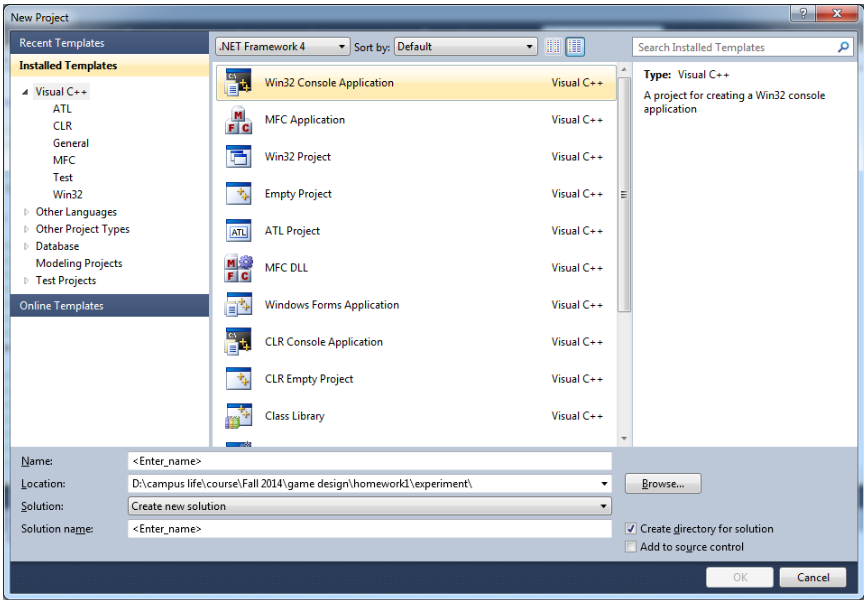
```
D:\Projects\Lab\testngpp\cmake-2.8.1\Tests\Tutorial\Step4\build>Tutorial.exe 25
Computing sqrt of 25 to be 5 using log
The square root of 25 is 5
```

-There are something addition you can learn from cmake Tutorial

http://www.cmake.org/cmake/help/cmake_tutorial.html

SFML configuration(For visual Studio)

-After installing the Visual Studio, you could build a windows console application project



Type: Visual C++

A project for creating a Win32 console application

Name: <Enter_name>

Location: D:\campus life\course\Fall 2014\game design\homework1\experiment\

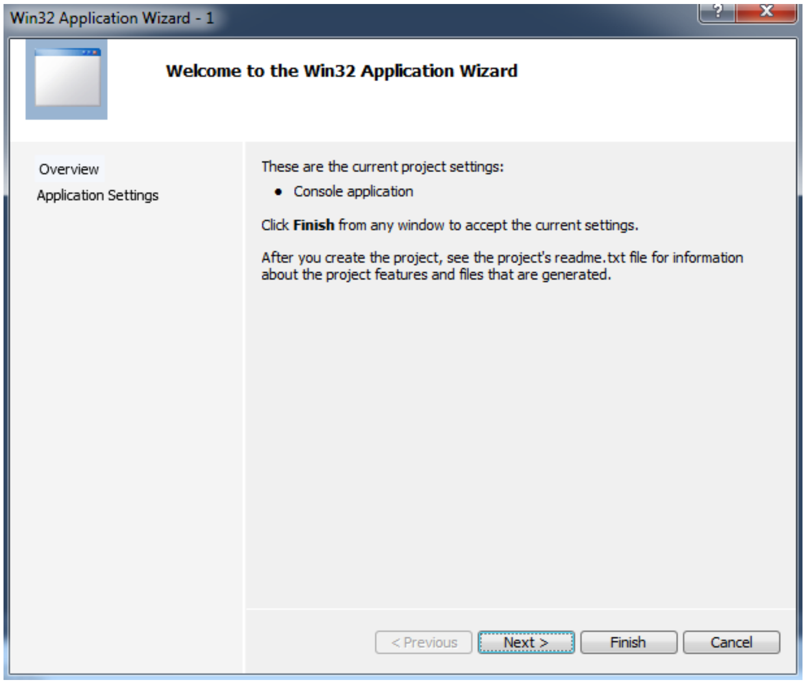
Solution: Create new solution

Solution name: <Enter_name>

☑ Create directory for solution

☐ Add to source control

Input your project name,



Welcome to the Win32 Application Wizard

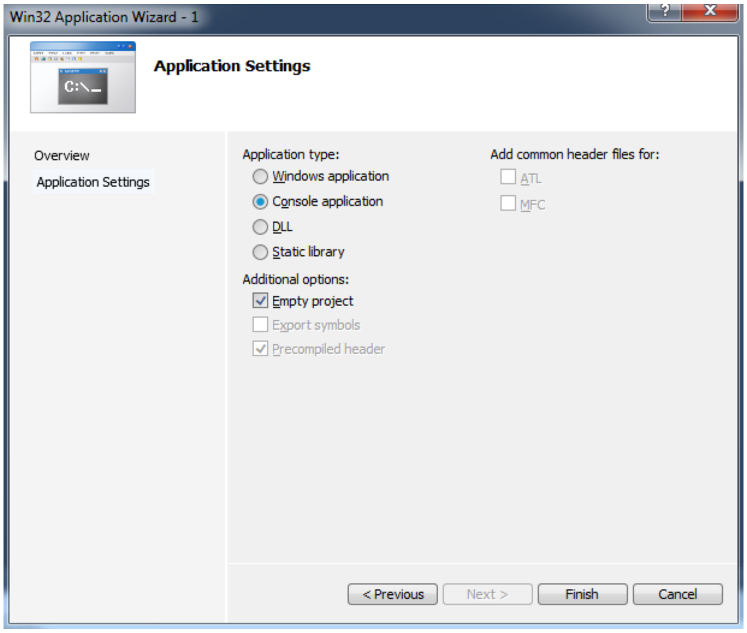
These are the current project settings:

- Console application

Click **Finish** from any window to accept the current settings.

After you create the project, see the project's readme.txt file for information about the project features and files that are generated.

-->Next



Application Settings

Application type:

○ Windows application

● Console application

○ DLL

○ Static library

Additional options:

☑ Empty project

□ Export symbols

☑ Precompiled header

Add common header files for:

□ ATL

□ MFC

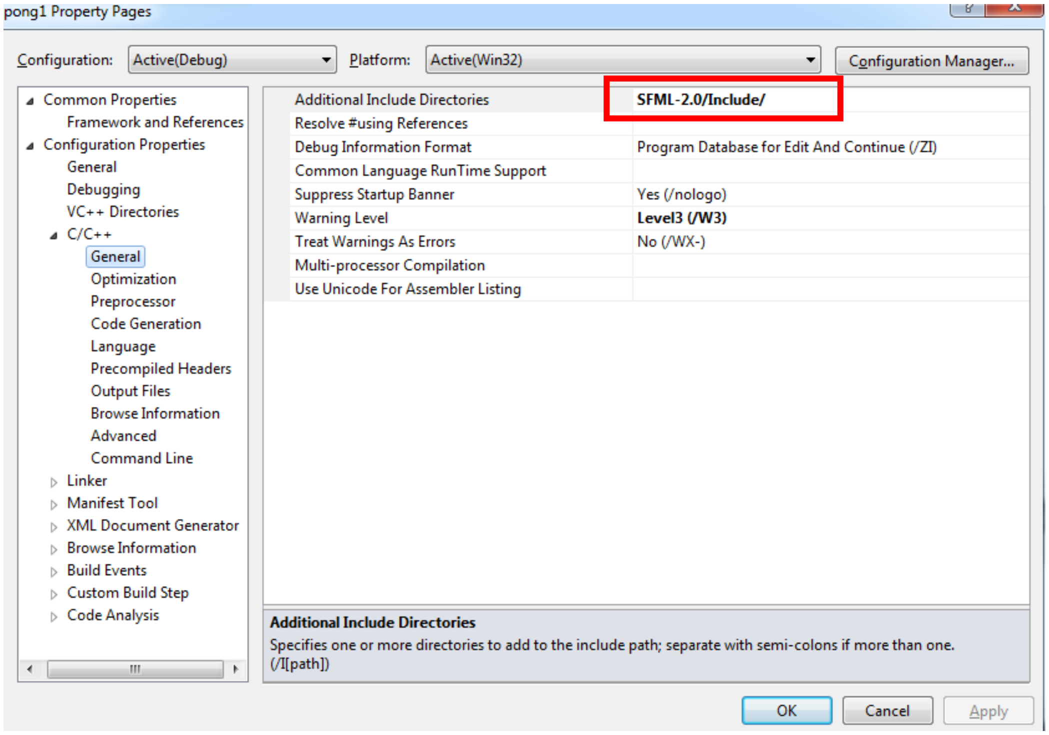
Build an empty project.

Simply use the original file provided by Prof. Peers.

Now a project has been built.

-configure the environment from project properties (three step):

1-add your SFML include path in your c/c++ general property.



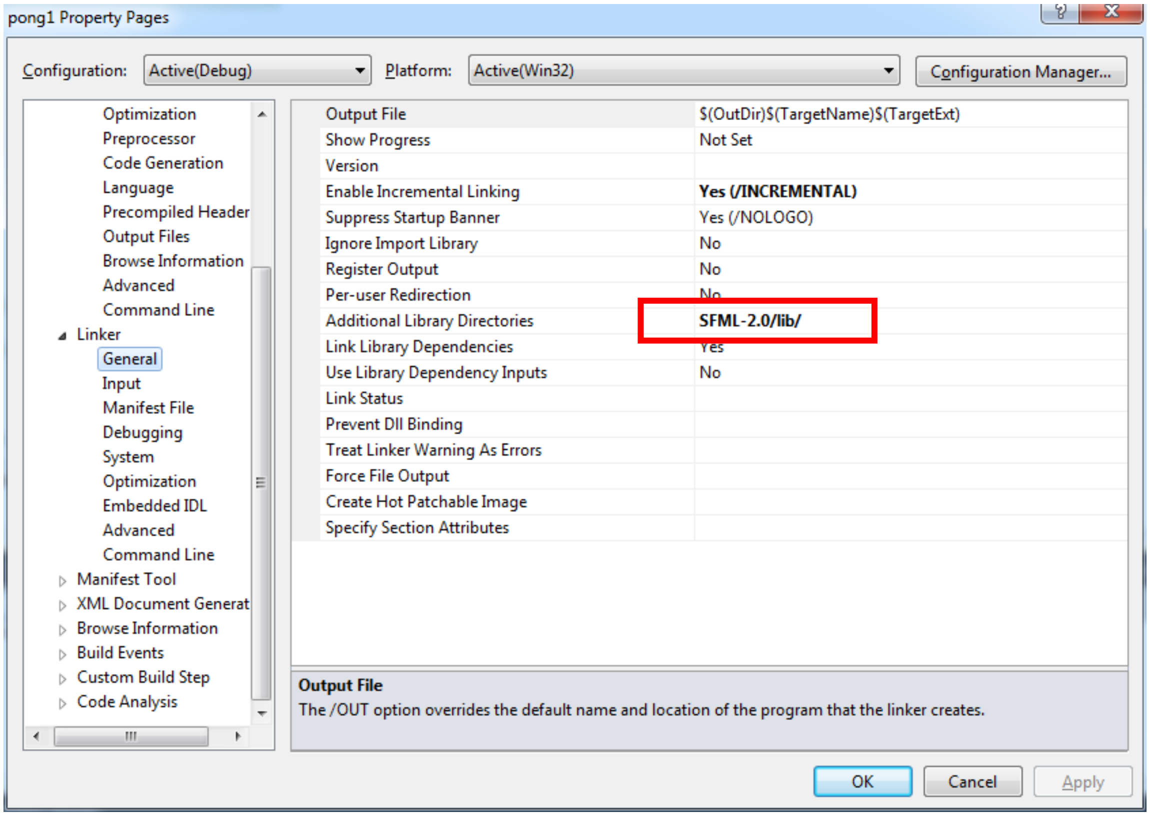
Configuration: Active(Debug) Platform: Active(Win32)

Additional Include Directories	**SFML-2.0/Include/**
Resolve #using References	
Debug Information Format	Program Database for Edit And Continue (/ZI)
Common Language RunTime Support	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
Multi-processor Compilation	
Use Unicode For Assembler Listing	

Additional Include Directories

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

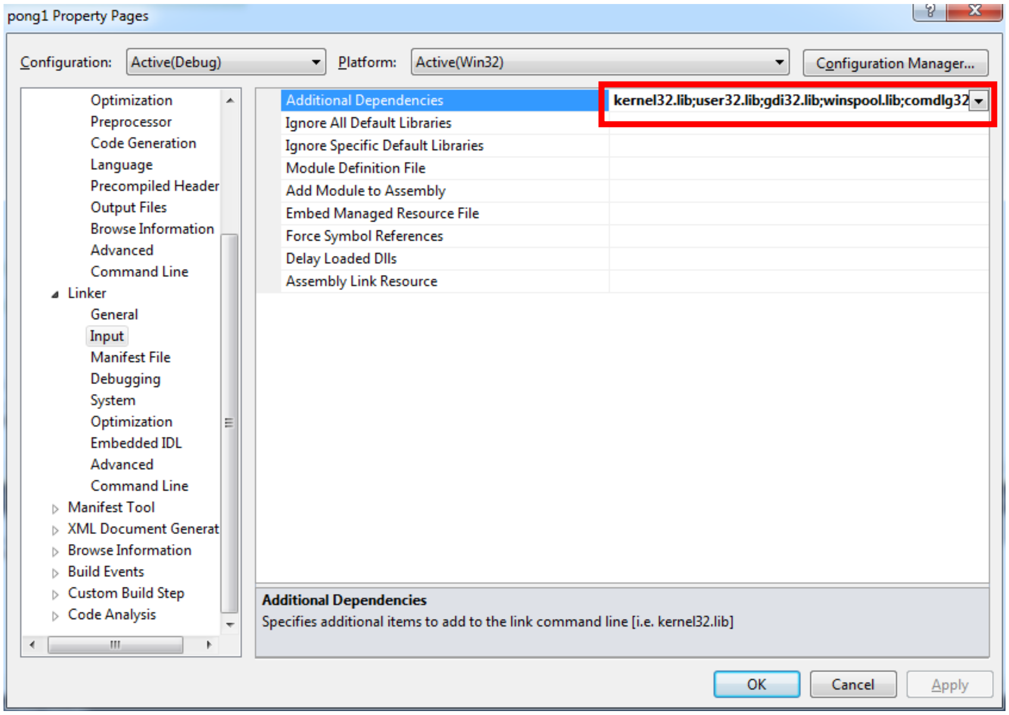
2-add SFML lib directory path in Linker general property



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	**SFML-2.0/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	

Output File
The /OUT option overrides the default name and location of the program that the linker creates.

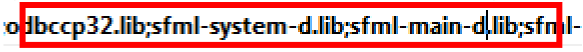
3-add "sfml-system-d.lib;sfml-main-d.lib;sfml-graphics-d.lib;sfml-audio-d.lib;sfml-network-d.lib;sfml-window-d.lib" in the Linker Input properties.



Additional Dependencies	**kernel32.lib;user32.lib;gdi32.lib;winspool.lib;comdlg32**
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link command line [i.e. kernel32.lib]

Copy the red line at the end of the libs.(In my computer, it will be like this. Use ";" to separate different lib.)



odbccp32.lib;sfml-system-d.lib;sfml-main-d.lib;sfml-

Then you can run your program now.

Git command

-Register an account from github.

https://github.com/

-Install git also have two choice, one is source distribution, the other is binary distribution.

Linux

For linux **source distribution**, we could use "yum" command in Fedora and "apt-get" command in Debian to install. Since we need curl, zlib, openssl, expat, libconv library so we could use command to install:

```
$ yum install curl-devel expat-devel gettext-devel \
  openssl-devel zlib-devel

$ apt-get install libcurl4-gnutls-dev libexpat1-dev gettext \
  libz-dev libssl-dev
```

Then download source code from Git website:

```
http://git-scm.com/download
```

Compile and install:

```
$ tar -zxf git-1.7.2.2.tar.gz
$ cd git-1.7.2.2
$ make prefix=/usr/local all
$ sudo make prefix=/usr/local install
```

For **binary distribution,** use command directly:

```
$ yum install git-core
```

```
$ apt-get install git
```

Windows

```
Install Git in Windows is quite easy. Download .exe file from
http://msysgit.github.com/ and install it.
```

Mac

In Mac there are two path:

1-Download the file as the GUI method from http://code.google.com/p/git-osx-installer/

Then double click the .pkg file you can install it.

Except command terminal, you can download the OpenInGitGui in the same website: http://code.google.com/p/git-osx-installer/

2-Use macPorts (`http://www.macports.org`), then use command:

```
$ sudo port install git-core +svn +doc +bash_completion +gitweb
```

or use homebrew (`https://github.com/mxcl/homebrew`) with command:

```
brew install git
```

After installation, you can use **git config** to configure your environment such as user information, text eidtor, difference analysis tools and check the information

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

```
$ git config --global core.editor emacs
```

```
$ git config --list
user.name=Scott Chacon
user.email=schacon@gmail.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

The other git command we may need will be on the list. Just try to understand them then you can use git to do simple commitment. Good luck, guys.

```
git status

git pull

git push

git commit -a

git checkout

git brantch

git log

git clone
```