

# Setting Environment

---

# In this lecture, you will learn

---

- GPU Server 사용법
- CUDA Install @ My PC
- OpenGL 설치

# GPU Server 사용법



# 배우게 될 것

---

- 서버 컴퓨터에 로그인하는 방법
- 서버 컴퓨터에 명령을 시키는 것(파일 생성, 디렉토리 탐색)
- 서버 컴퓨터에서 코딩하는 법
- 서버 컴퓨터에서 실행파일을 만드는 법
- 서버 컴퓨터에서 파일을 실행하는 법

# Access Server with SSH Using Putty

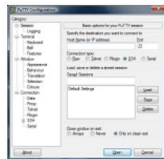
- PuTTY

- Server 컴퓨터와 SSH 프로토콜을 이용해 통신할 수 있도록 해주는 프로그램
- SSH 프로토콜
  - 네트워크 상의 다른 컴퓨터에 로그인하거나 원격 시스템에서 명령을 실행하고 다른 시스템으로 복사할 수 있도록 해 주는 프로토콜



# Putty Install

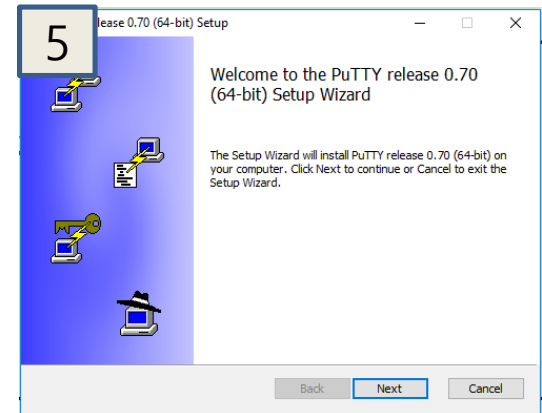
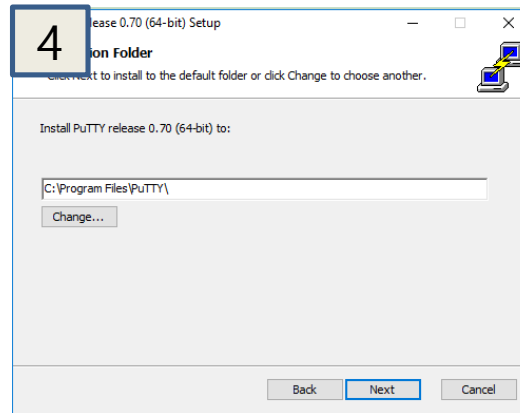
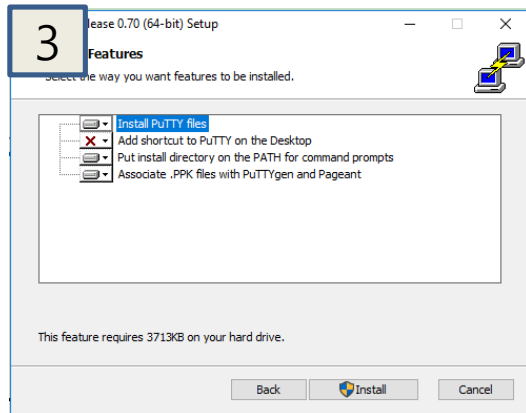
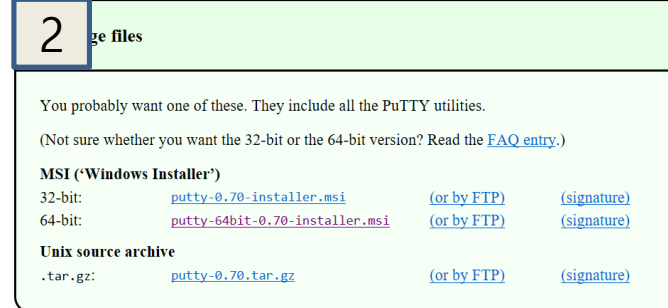
- PuTTY를 다운로드 <https://www.putty.org/>



## Download PuTTY

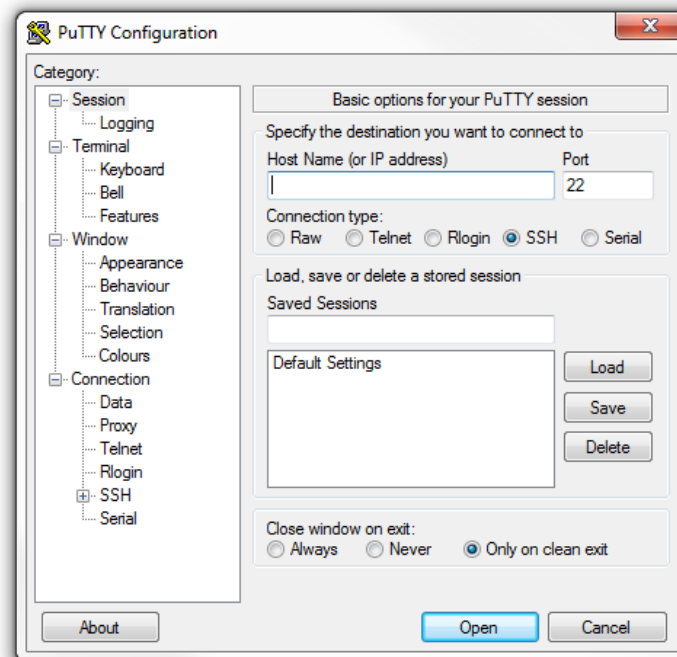
PuTTY is an SSH and telnet client software that is available with source code.

You can download PuTTY [here](#).



# GPU Server 접속 (Window)

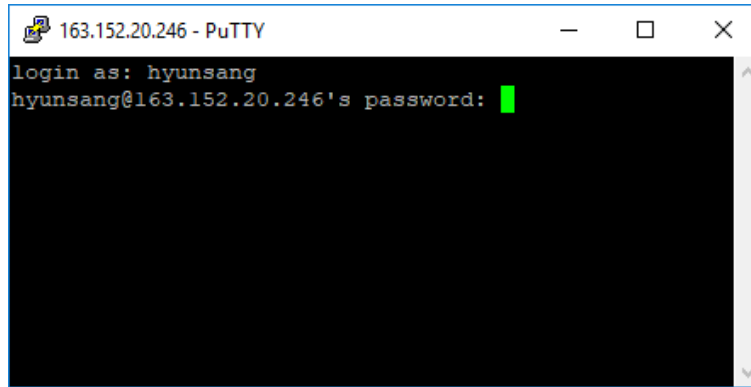
- 수업용 서버 ip(163.152.20.246)를 Host Name으로 입력



# GPU Server 접속 (Window) Login

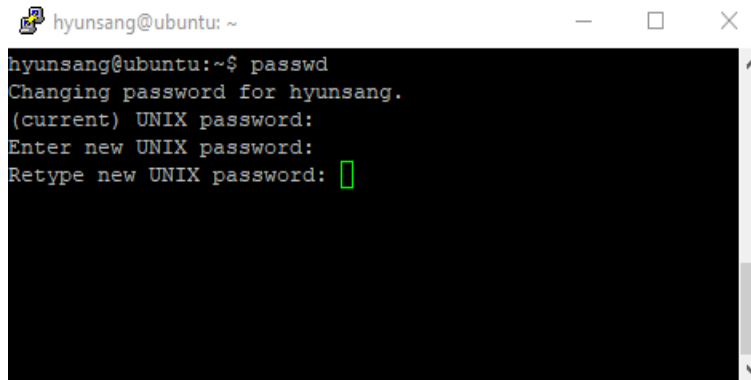
ID : id학번(ex: id2010190001)

Password: 0000 (초기값)



```
163.152.20.246 - PuTTY
login as: hyunsang
hyunsang@163.152.20.246's password: 
```

Password 변경: 명령어 passwd

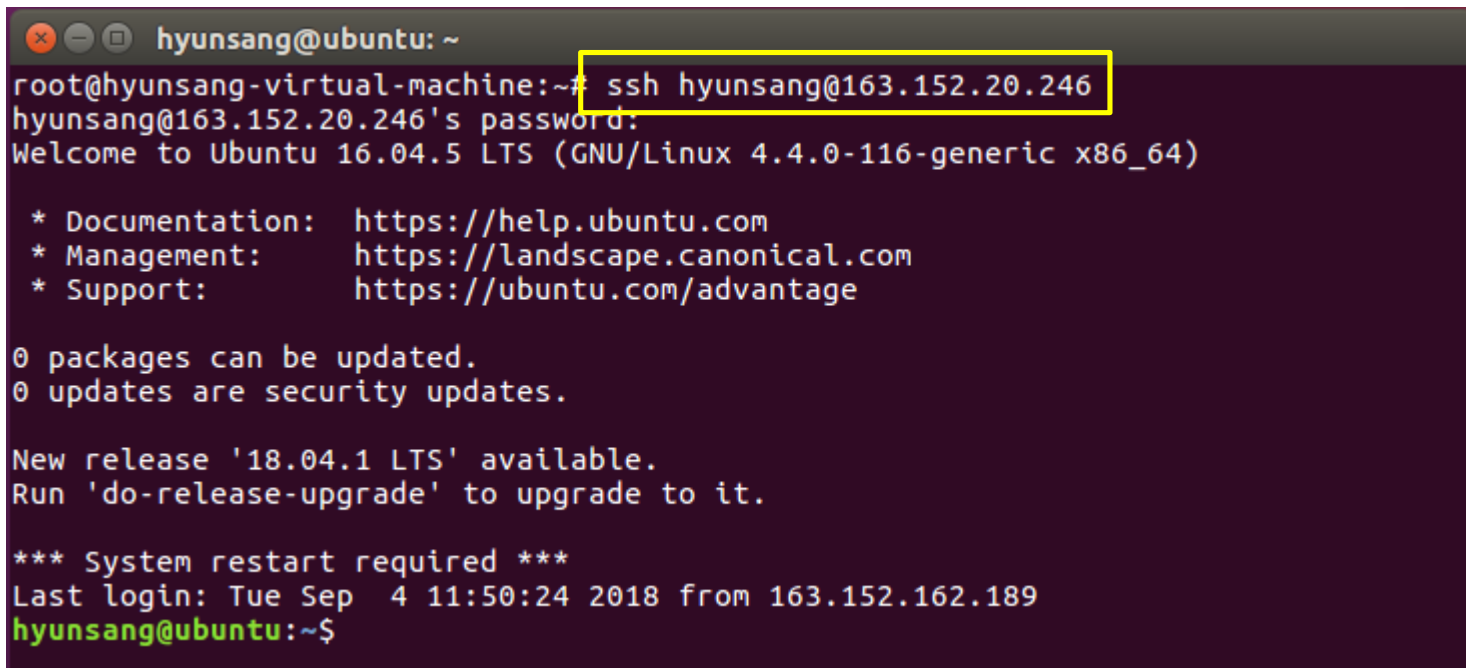


```
hyunsang@ubuntu: ~
hyunsang@ubuntu:~$ passwd
Changing password for hyunsang.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password: 
```



# GPU Server 접속 (Linux)

- 명령어: `ssh serverID@Server IP address`



A terminal window titled 'hyunsang@ubuntu: ~' showing an SSH session. The user 'root@hyunsang-virtual-machine' enters the command 'ssh hyunsang@163.152.20.246', which is highlighted with a yellow box. The terminal then prompts for the password, displays the Ubuntu 16.04.5 LTS welcome message, and lists system updates and links for documentation, management, and support. The session ends with the prompt 'hyunsang@ubuntu:~\$'.

```
hyunsang@ubuntu: ~  
root@hyunsang-virtual-machine:~# ssh hyunsang@163.152.20.246  
hyunsang@163.152.20.246's password:  
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-116-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
0 packages can be updated.  
0 updates are security updates.  
  
New release '18.04.1 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
*** System restart required ***  
Last login: Tue Sep  4 11:50:24 2018 from 163.152.162.189  
hyunsang@ubuntu:~$
```

# Linux Commands

- **ls**
  - 현재 directory의 파일들을 검색
- **cd [디렉토리 경로]**
  - 다른 directory로 이동
- **cat [파일명]**
  - 파일 내용을 출력
- **rm [파일명]**
  - 파일을 삭제
- **rmdir [디렉토리명]**
  - 디렉토리 삭제

# Editor on Linux



- Putty를 이용하여 Linux에 연결된 상태에서 VIM을 이용하여 직접 코드를 생성하고 수정 가능.
- vim [파일명]
  - 파일이 없는 경우, 해당 파일명의 파일 생성.
  - 파일이 있는 경우, 해당 파일을 오픈

# vim

```
vim helloworld.cu
```

CUDA 파일 확장자명

- Command 창에서 vim 키워드를 이용하여 파일 생성

```
#include<iostream>

using namespace std;

int main()
{
    cout<<"hello world!"<<endl;
    return 0;
}
```

- i 키를 눌러 Insert Mode로 진입
- 소스 코드 입력
- ESC 키를 눌러 Command Mode로 진입
- :wq를 입력하여 파일을 저장하고 나가기

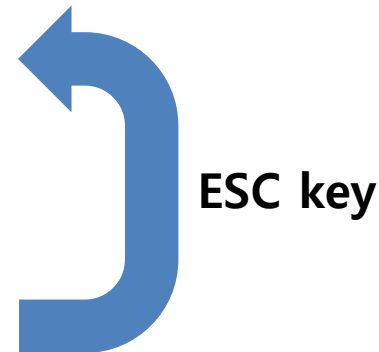
# vim basic commands

## 1. 저장 및 종료

명령어	설명
<code>:w</code>	저장
<code>:w file.txt</code>	file.txt 파일로 저장
<code>:w &gt;&gt; file.txt</code>	file.txt 파일에 덧붙여서 저장
<code>:q</code>	vim 종료
<code>ZZ</code>	저장 후 종료
<code>:wq</code>	저장 후 종료
<code>:e file.txt</code>	file.txt 불러옴

## 2. 입력모드 전환

명령어	설명
<code>a</code>	커서 위치 다음칸부터 입력
<code>A</code>	커서 행의 맨 마지막부터 입력
<code>i</code>	커서의 위치에 입력
<code>I</code>	커서행의 맨 앞에서부터 입력
<code>o</code>	커서의 다음행에 입력
<code>O</code>	커서의 이전 행에 입력
<code>s</code>	커서 위치의 한글자를 지우고 입력 <sup>64</sup>
<code>cc</code>	커서위치의 한 행을 지우고 입력



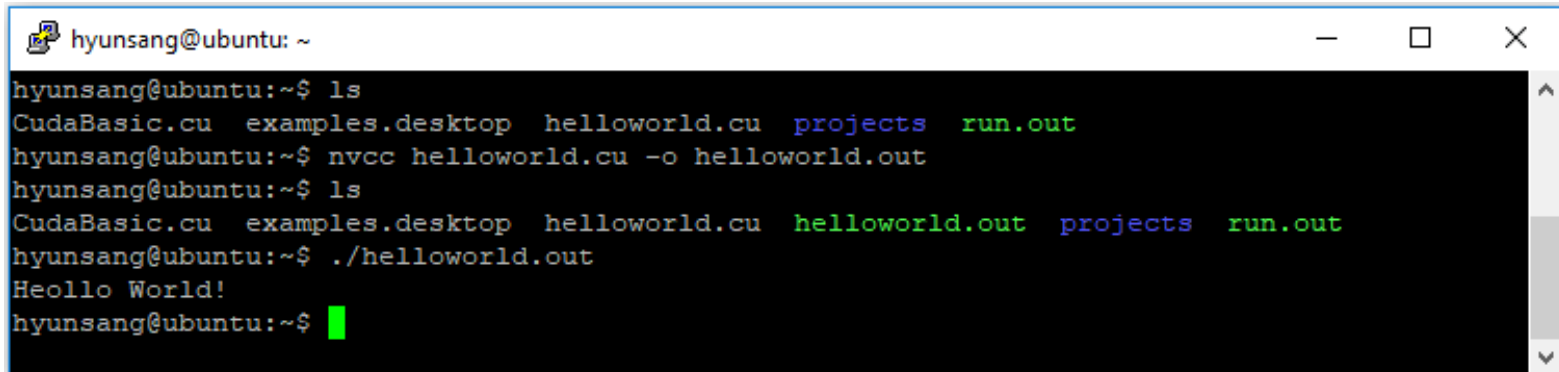
# Compile & Run

- **Compile**

- `nvcc [파일명.cu] -o [실행파일명]`
  - `nvcc`: CUDA 컴파일러

- **Run**

- `./[실행파일명]`
  - 실행파일의 확장자는 임의로 지정가능



```
hyunsang@ubuntu: ~  
hyunsang@ubuntu:~$ ls  
CudaBasic.cu  examples.desktop  helloworld.cu  projects  run.out  
hyunsang@ubuntu:~$ nvcc helloworld.cu -o helloworld.out  
hyunsang@ubuntu:~$ ls  
CudaBasic.cu  examples.desktop  helloworld.cu  helloworld.out  projects  run.out  
hyunsang@ubuntu:~$ ./helloworld.out  
Heollo World!  
hyunsang@ubuntu:~$
```

# File Transfer (PC <-> Server)



- 개인 PC에서 작성한 코드를 GPU Server로 전송해야하는 경우, Notepad++ 의 플러그인 기능인 nppFTP를 이용하여 전송가능.
- 본 강의에선 Notepad++의 nppFTP를 소개.

# Notepad++ 다운로드

- <https://notepad-plus-plus.org/>






# Notepad++ Plugin Manager 다운로드

- <https://github.com/bruderstein/nppPluginManager/releases>
- PluginManager\_v1.4.9\_UNI.zip 다운
- C:\Program Files (x86)\Notepad++ 경로에 plugins 폴더와 updater 폴더를 복사


## v1.4.9


 bruderstein released this on Apr 16 2017

### ▼ Assets 4

 [PluginManager\\_v1.4.9\\_UNI.zip](#) 342 KB

 [PluginManager\\_v1.4.9\\_x64.zip](#) 379 KB


 [Source code \(zip\)](#)

 [Source code \(tar.gz\)](#)

- [#61](#) always validate plugin list for faster update of the devlist
- [#62](#) own url for x64 plugin list in preparation to solve feature request [#7](#)

# nppFTP 다운로드

- <https://sourceforge.net/projects/nppftp/>
- 압축 해제 후 C:\Program Files (x86)\Notepad++\plugins 폴더에 NppFTP.dll 파일을 복사



## NppFTP

Brought to you by: [ashish\\_kulz](#), [harrybharry](#)

★★★★★


53 Reviews

Downloads:

2,032 This Week

Last Update:

2015-05-05

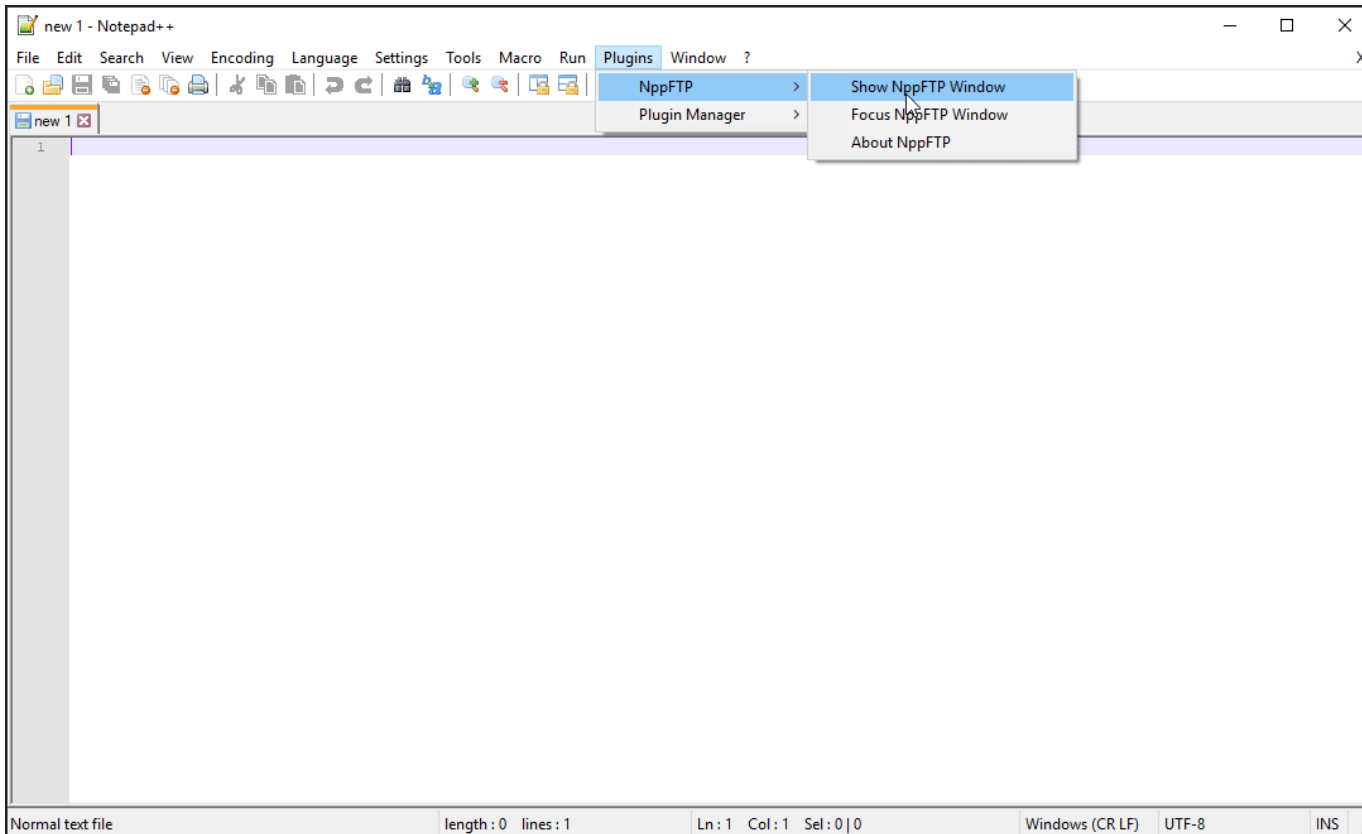
 **Download**

Get Updates

Share This

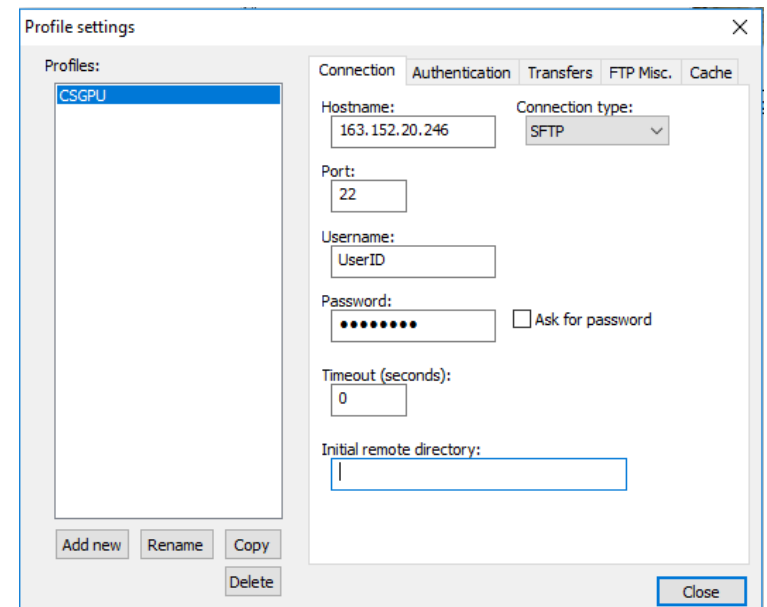
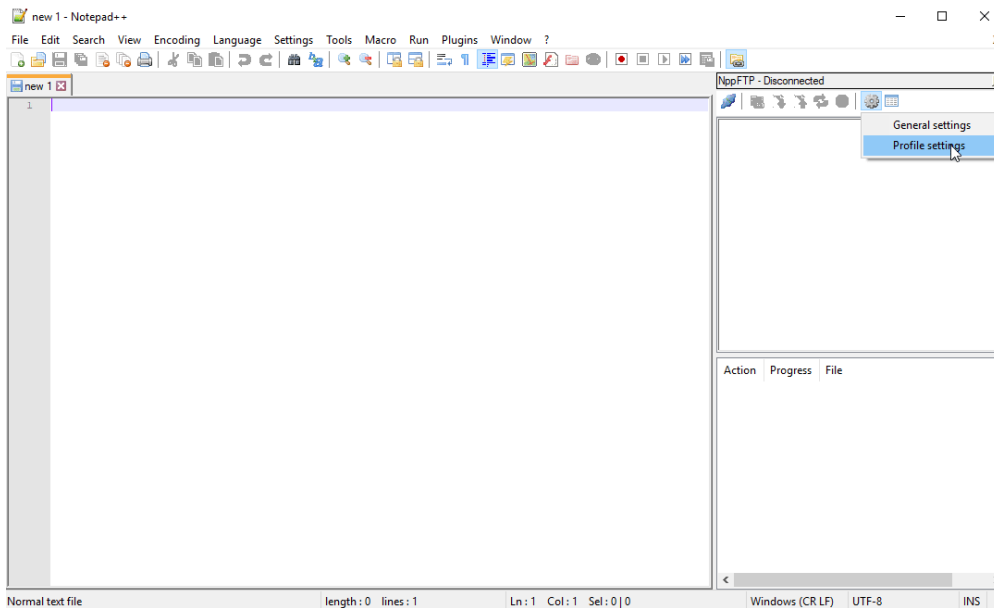
# FTP@Notepad++

- 메뉴표시줄 [Plugins] - [NppFTP] – Show NppFTP Window



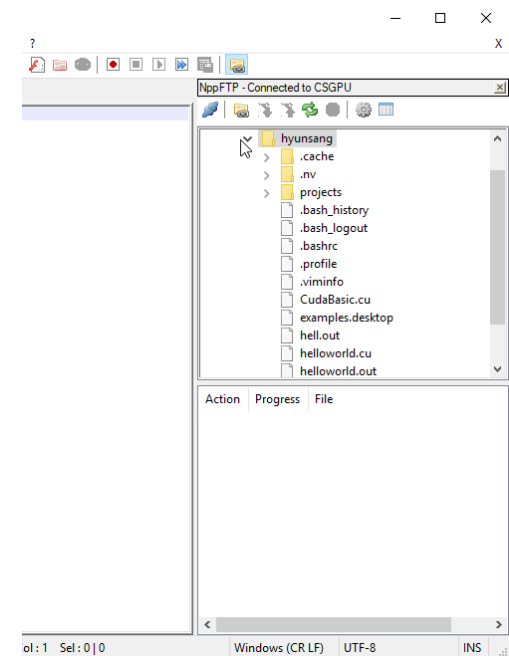
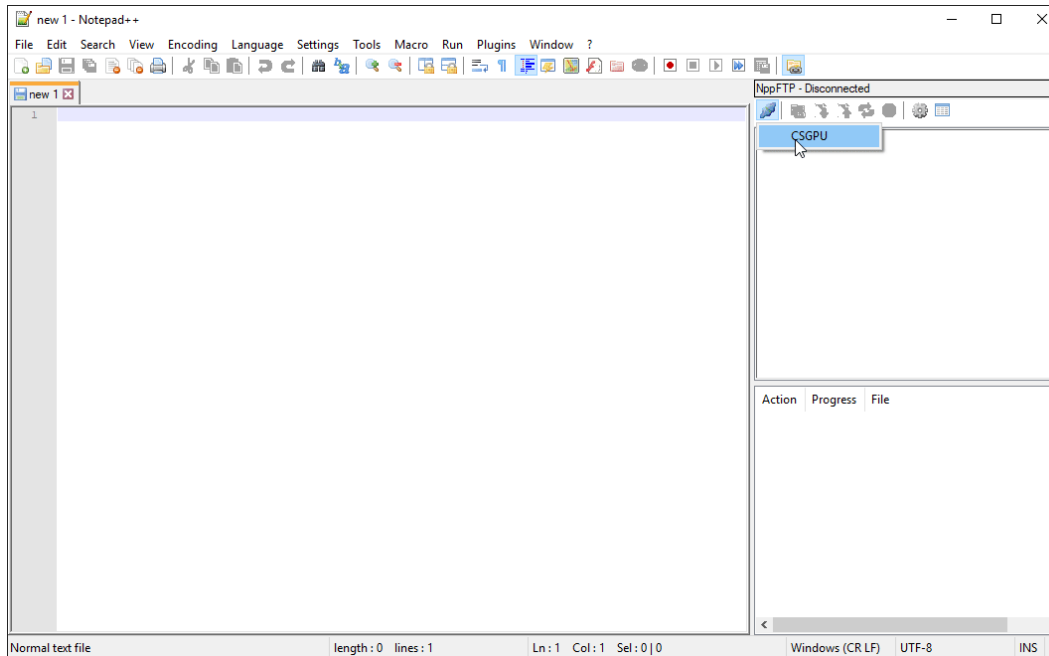
# nppFTP Setting

- NppFTP Window 에서 Profile settings 클릭
- [Add new] 단추 클릭하여 서버생성
- Hostname, Port, Username, Password를 입력



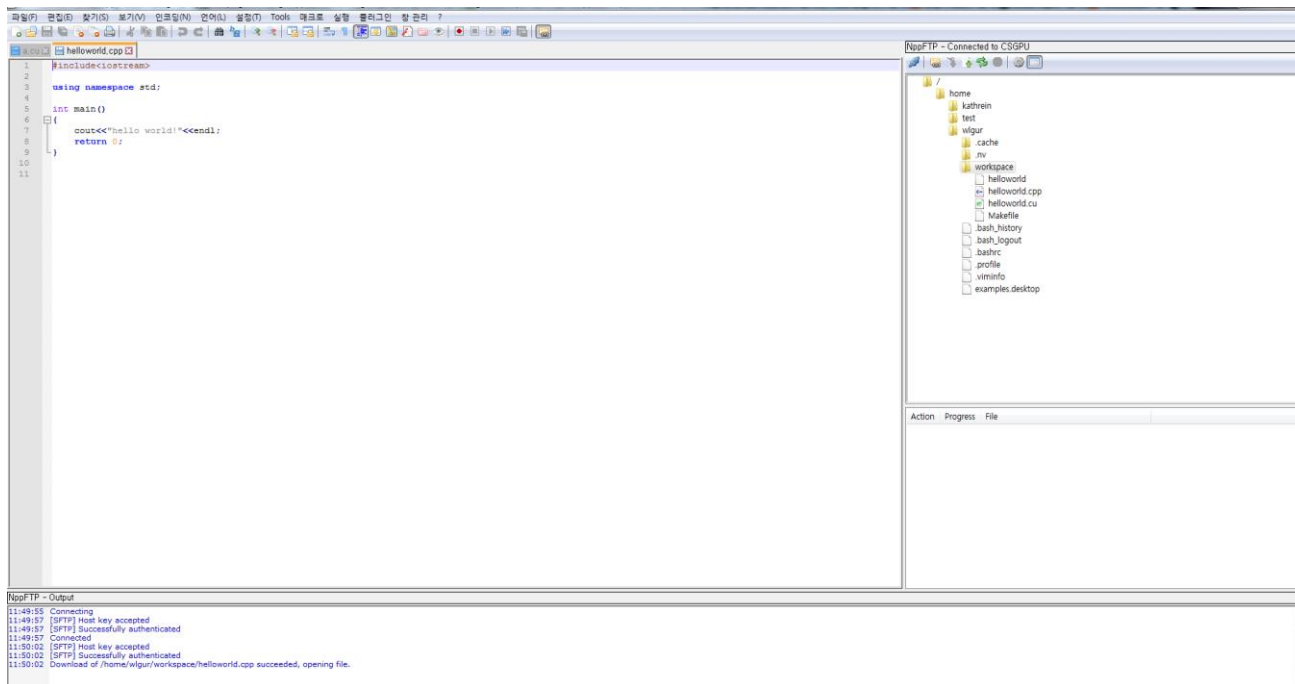
# Connect GPU Server with notepad++

- Profile Setting에 CONNECT button 누르면
- GPU 서버 내 폴더 확인 가능



# File Management with Notepad++

- 프로그램 파일 작성,
- 저장하면 자동으로 Server의 디렉토리 내에 프로그램 파일 저장됨.

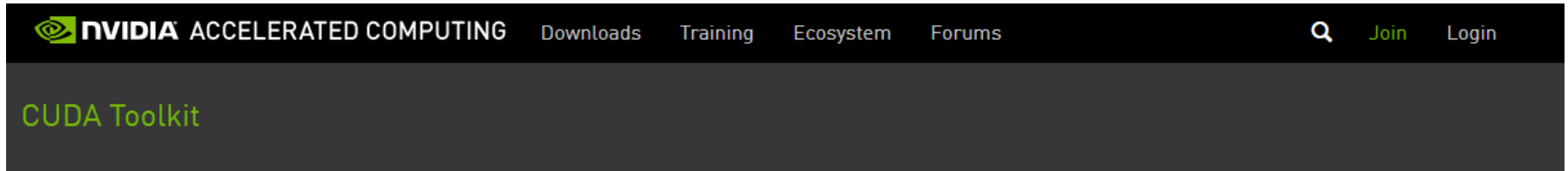


# CUDA install @ My PC



# CUDA-toolkit Download

- <https://developer.nvidia.com/cuda-toolkit>



[Home](#) > [ComputeWorks](#) > [CUDA Toolkit](#)

## Develop, Optimize and Deploy GPU-accelerated Apps

The NVIDIA® CUDA® Toolkit provides a development environment for creating high performance GPU-accelerated applications. With the CUDA Toolkit, you can develop, optimize and deploy your applications on GPU-accelerated embedded systems, desktop workstations, enterprise data centers, cloud-based platforms and HPC supercomputers. The toolkit includes GPU-accelerated libraries, debugging and optimization tools, a C/C++ compiler and a runtime library to deploy your application.

GPU-accelerated CUDA libraries enable drop-in acceleration across multiple domains such as linear algebra, image and video processing, deep learning and graph analytics. For developing custom algorithms, you can use available integrations with commonly used languages and numerical packages as well as well-published development APIs. Your CUDA applications can be deployed across all NVIDIA GPU families available on premise and on GPU instances in the cloud. Using built-in capabilities for distributing computations across multi-GPU configurations, scientists and researchers can develop applications that scale from single GPU workstations to cloud installations with thousands of GPUs.

To get started, browse through online getting started resources, optimization guides, illustrative examples and collaborate with the rapidly growing developer community.

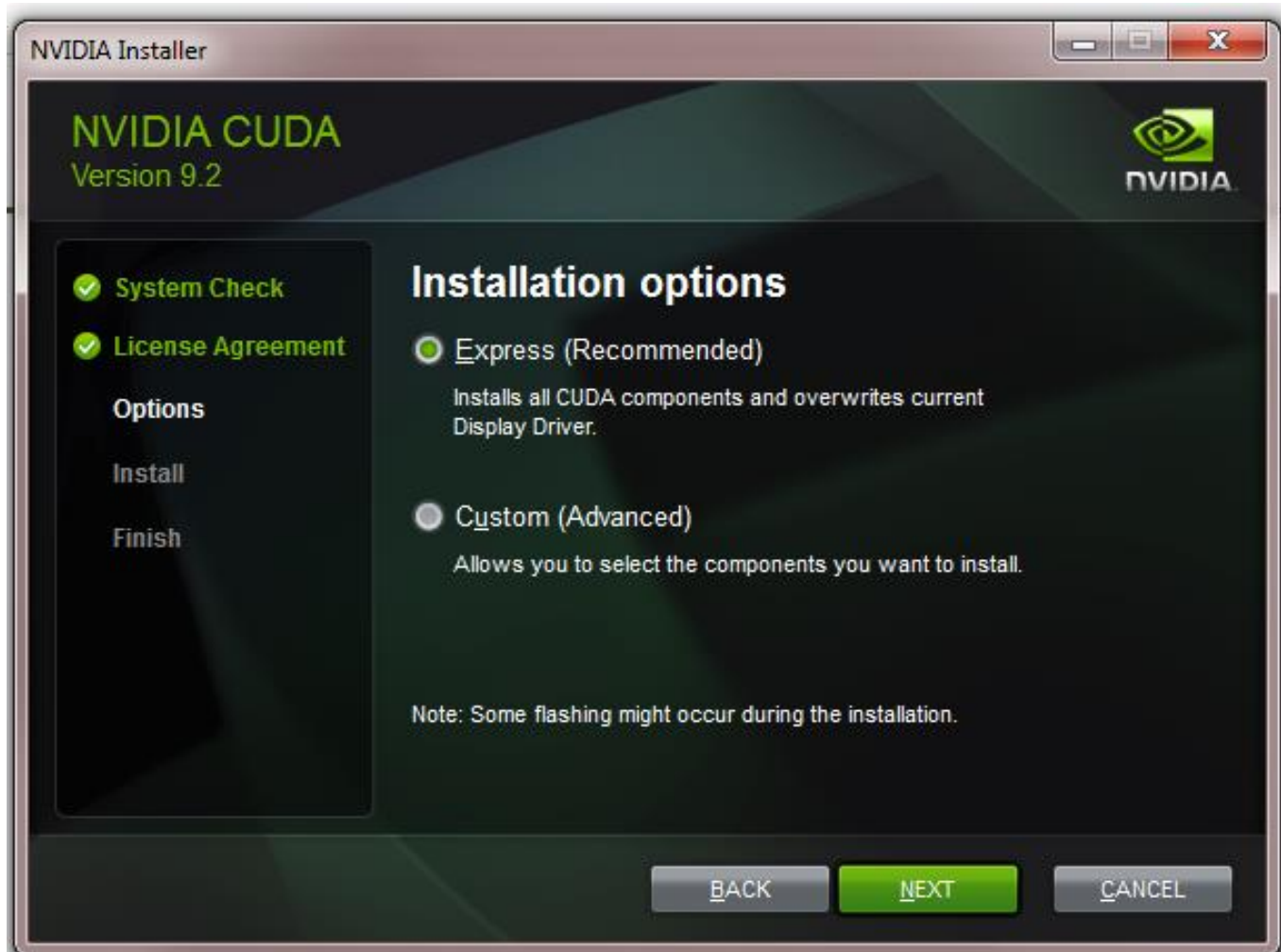
[Download Now >](#)



# Run and follow NVIDIA Installer



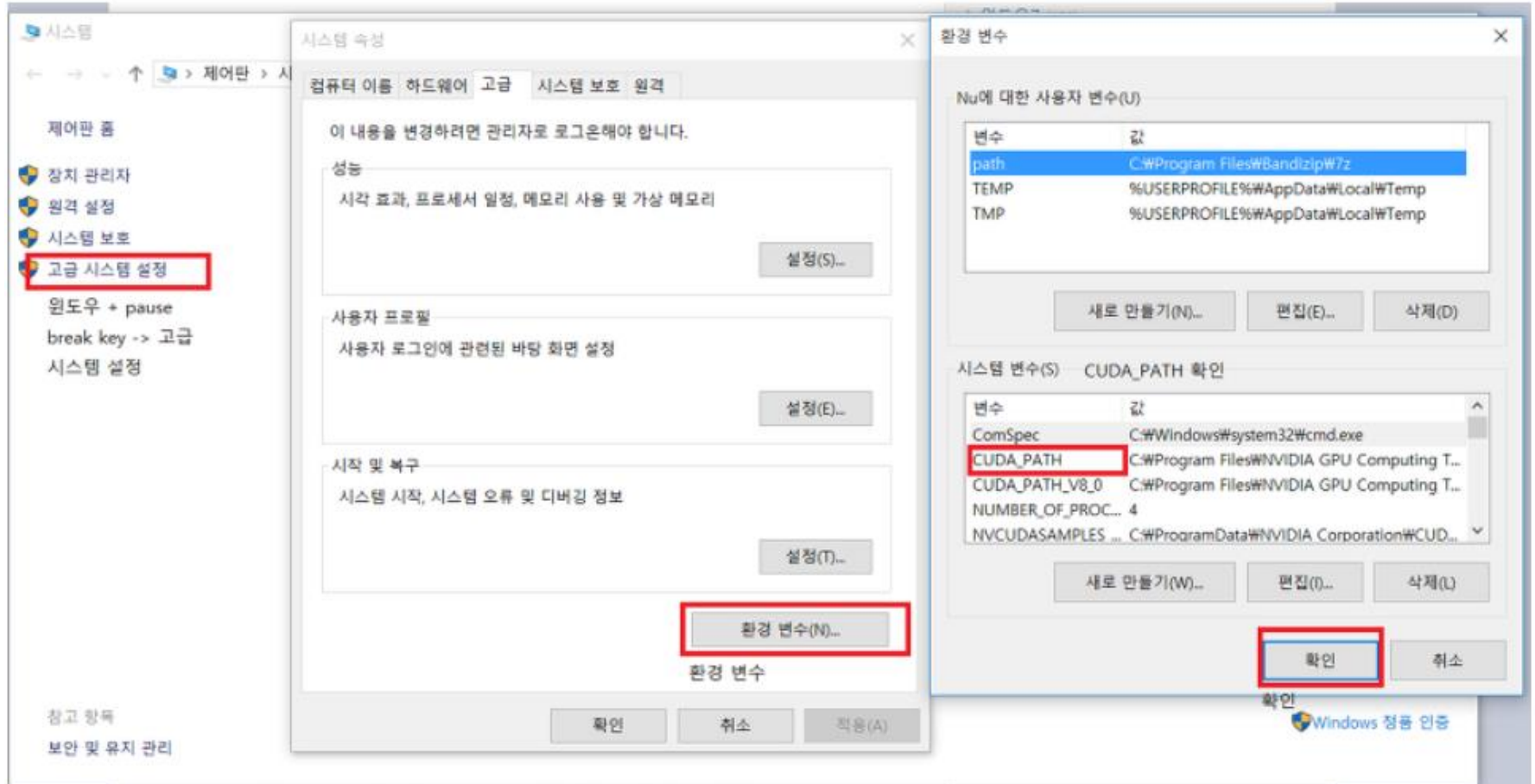
# Run and follow NVIDIA Installer



# Run and follow NVIDIA Installer



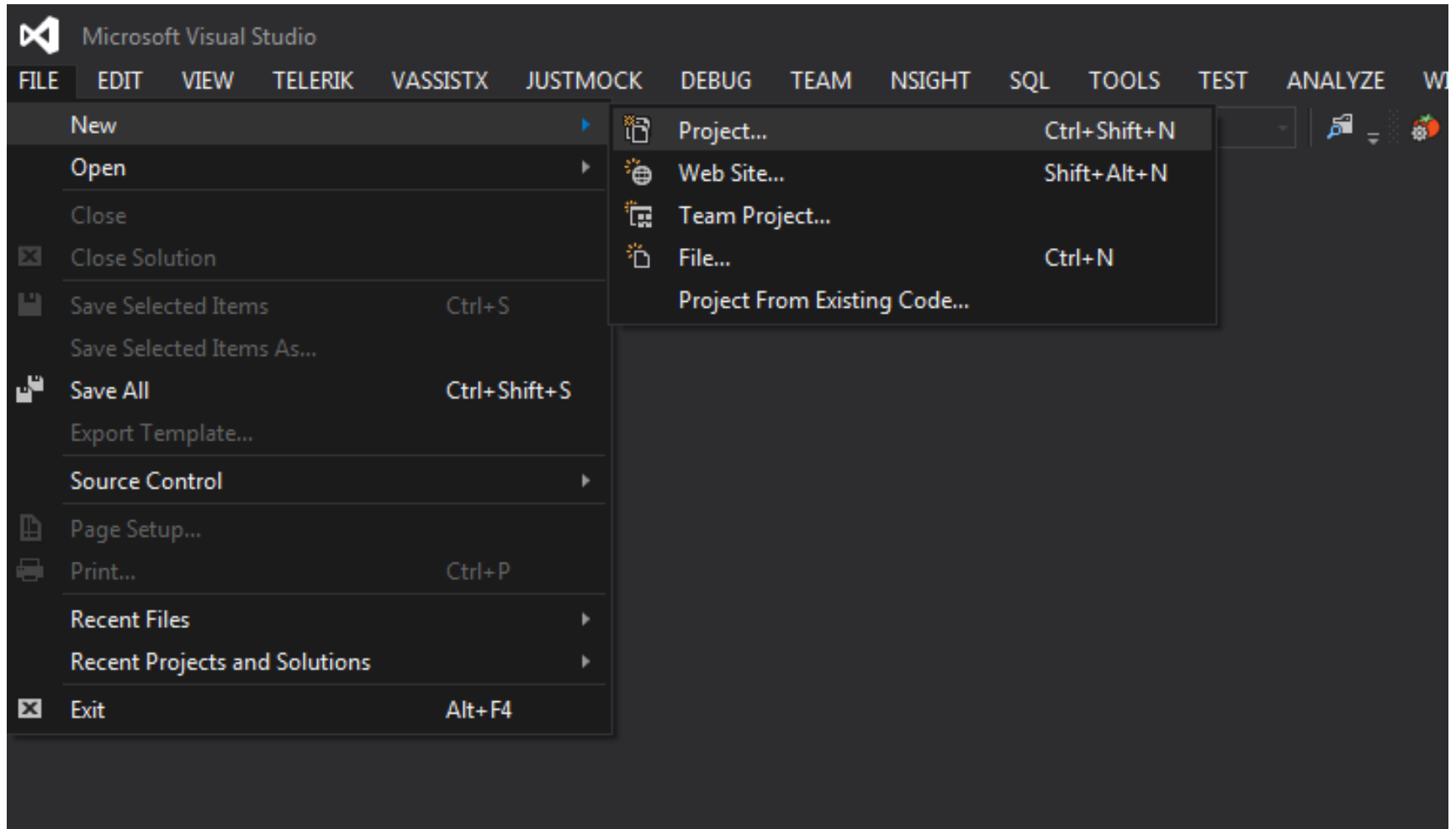
# Environment variable settings



CUDA 를 설치하면 환경 변수가 자동으로 등록된다.  
(여러 버전을 중복해서 설치할 경우 최근에 설치한 버전이 CUDA\_PATH 로 등록되며 각각의 버전은 CUDA\_PATH\_V8\_0 와 같은 형태로 등록된다.)

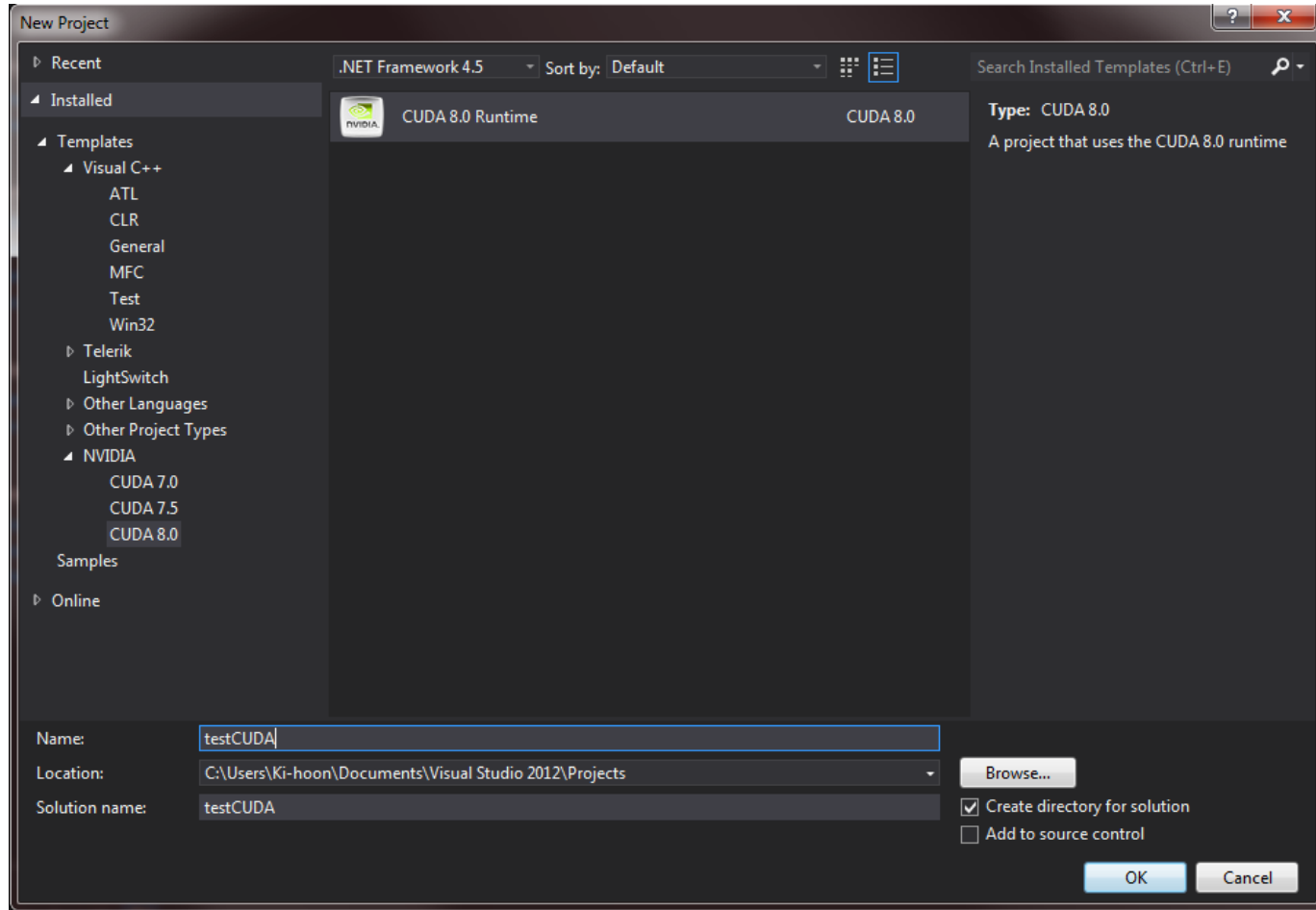
# Make CUDA Project on Visual Studio

- File -> New -> Project



# Make CUDA Project on Visual Studio

- Installed->Templates->NVIDIA-> CUDA X.X (Latest version is 9.2)



# Make CUDA Project on Visual Studio

testCUDA - Microsoft Visual Studio

FILE EDIT VIEW TOLERIK VASSISTX JUSTMOCK PROJECT BUILD DEBUG TEAM NSIGHT SQL TOOLS TEST ANALYZE WINDOW HELP

Local Windows Debugger - Auto - Debug - Win32

kernel.cu

int main()

(Global Scope)

main()

```
1 #include "cuda_runtime.h"
2 #include "device_launch_parameters.h"
3 #include <stdio.h>
4
5 cudaError_t addWithCuda(int *c, const int *a, const int *b, unsigned int size);
6
7 _global__ void addKernel(int *c, const int *a, const int *b)
8 {
9     int i = threadIdx.x;
10    c[i] = a[i] + b[i];
11 }
12
13 int main()
14 {
15     const int arraySize = 50;
16     const int a[arraySize] = { 10, 20, 30, 40, 50 };
17     const int b[arraySize] = { 0 };
18
19     // Add vectors in parallel.
20     cudaError_t cudaStatus = addWithCuda(c, a, b, arraySize);
21     if (cudaStatus != cudaSuccess) {
22         fprintf(stderr, "addWithCuda failed!");
23         return 1;
24     }
25
26     printf("1,2,3,4,5 + 10,20,30,40,50 = (%d,%d,%d,%d,%d)\n",
27           c[0], c[1], c[2], c[3], c[4]);
28
29     // Optional: Use cudaDeviceReset() to free all CUDA resources from the application.
30     cudaStatus = cudaDeviceReset();
31     if (cudaStatus != cudaSuccess) {
32         fprintf(stderr, "cudaDeviceReset failed!");
33         return 1;
34     }
35
36     return 0;
37 }
```

100 %

Output

Show output from:

Output | Find Symbol Results

Solution Explorer

Search Solution Explorer (Ctrl+):

Solution 'testCUDA' (1 project)

- testCUDA
- External Dependencies
- kernel.cu

Team Explorer

Class View

main

C++

(Name)	main
File	c:\users\kdi-hoon\docum
FullName	main
IsInjected	False
IsInline	False
IsOverloaded	False
IsSealed	False
IsTemplate	False
TypeString	int

Creating project 'testCUDA'... project creation successful.

Ln 20 Col 30 Ch 30 INS

## Create CUDA Project with Sample CODE

## You can begin the CUDA programming

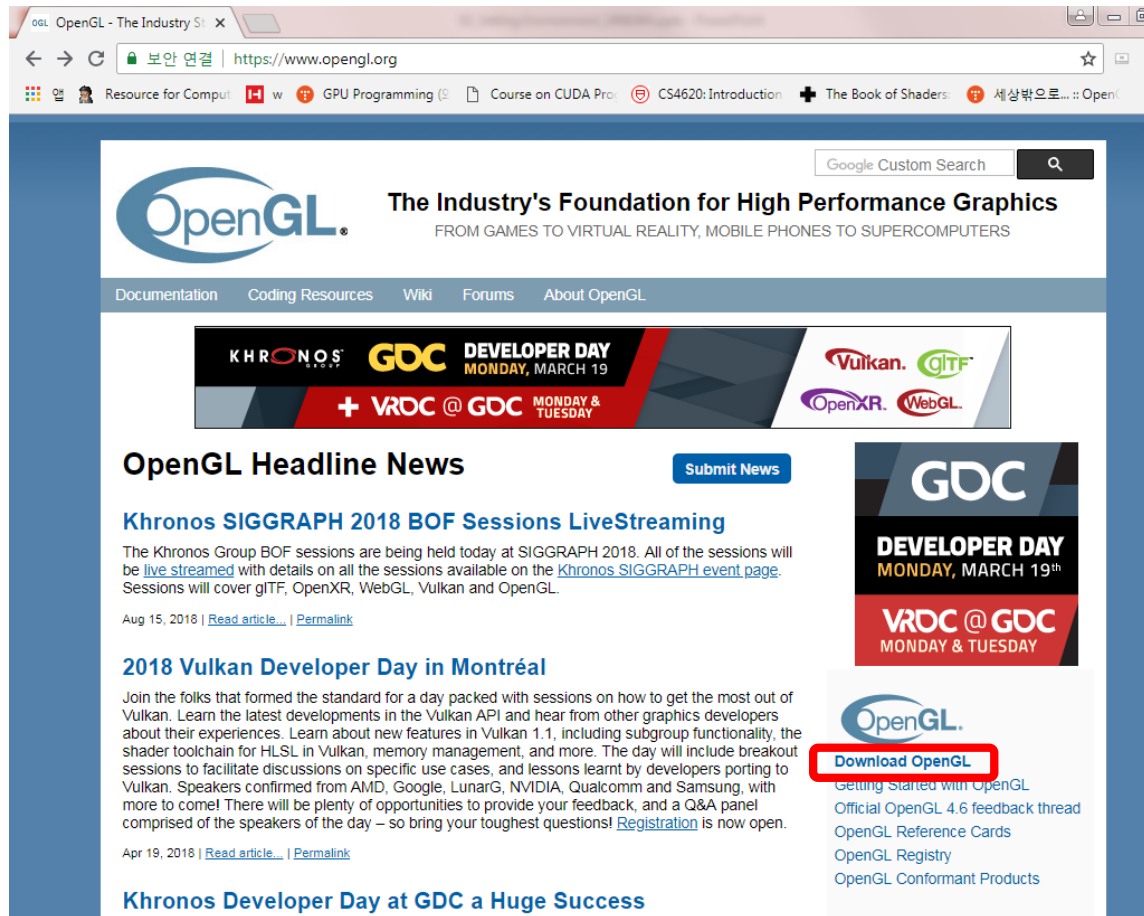
# OpenGL Setting





# OpenGL Download

- <https://www.opengl.org/>



# OpenGL Necessary Files

- Latest OpenGL version: 4.6
- Required files
  - Header File
    - gl.h, glu.h, **glut.h**
      - To the 'include' directory
  - Import Library
    - opengl32.lib, glu32.lib, **glut32.lib**
      - To the 'lib' directory
  - Dynamic Library
    - opengl32.dll, glu32.dll, glut.dll, **glut32.dll**
      - C:\WINDOWS\system32
      - C:\WINDOWS\SysWow64

# Glut files download

<https://www.opengl.org/resources/libraries/glut/>

---


## Basic GLUT Information/Downloads


- [The GLUT 3.7 page with all info and all downloads](#)
- [GLUT 3.7 Source Code Download for Win32](#)
- [GLUT Sample Programs w/ Source Code](#)
- [GLUT 3 Specification \(PostScript\) \(HTML\)](#)
- [Frequently Asked GLUT 3.7 Questions](#)


---


## Other GLUT Information/Downloads


- [Installing GLUT on a Windows machine](#)
- [Pre-compiled binaries for Solaris on X86 GLUT 3.7](#)
- [Pre-compiled binaries for Solaris on X86 64-bit GLUT 3.7](#)
- [Pre-compiled binaries for Solaris on SPARC GLUT 3.7](#)
- [Pre-compiled binaries for Solaris on SPARC 64-bit GLUT 3.7](#)
- [Pre-compiled Win32 for Intel GLUT 3.7 DLLs for Windows 95 & NT](#)
- [Pre-compiled Win32 for Alpha GLUT 3.6 DLLs for Windows NT](#)
- [GlutMaster: C++ wrapper classes for GLUT programming](#)

 glut.dll

 glut.h

 glut.lib

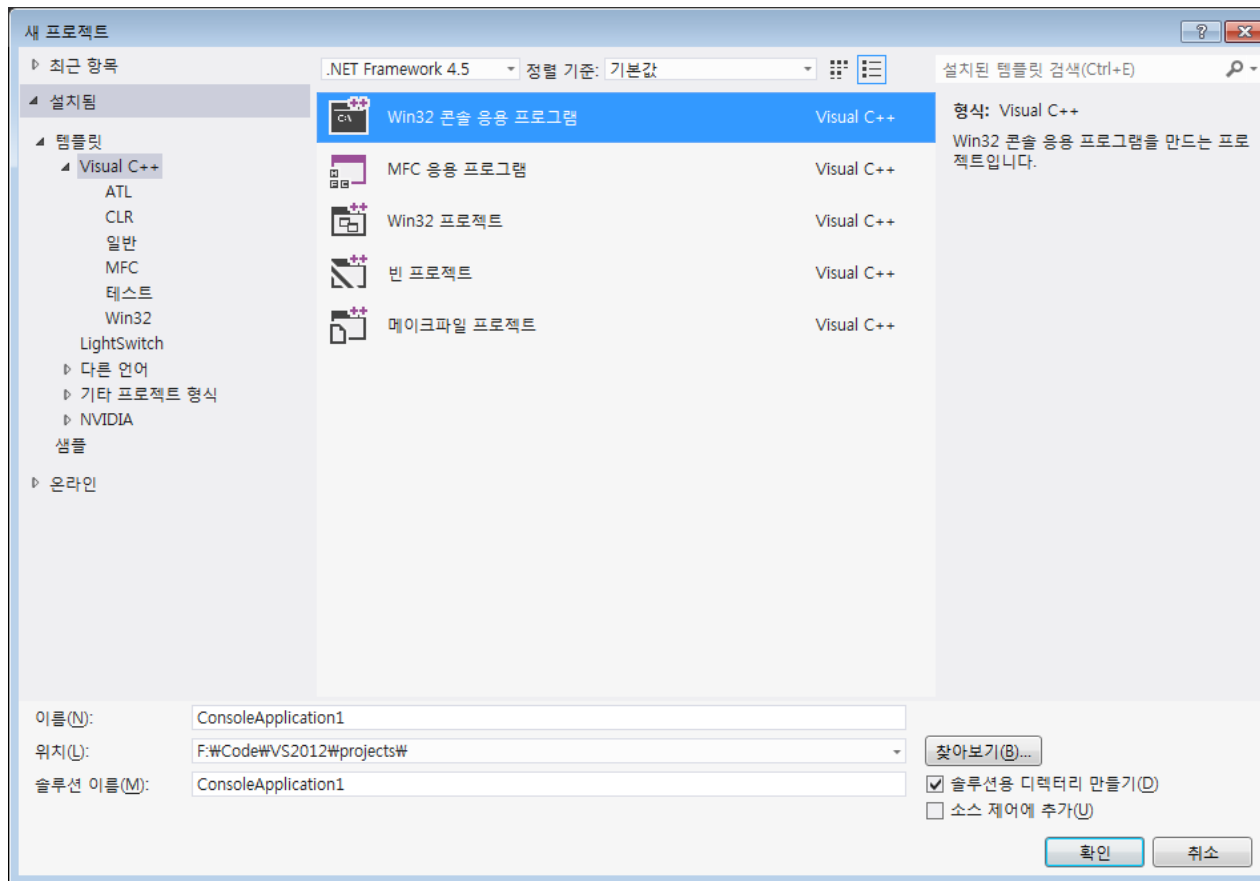
 glut32.dll

 glut32.lib

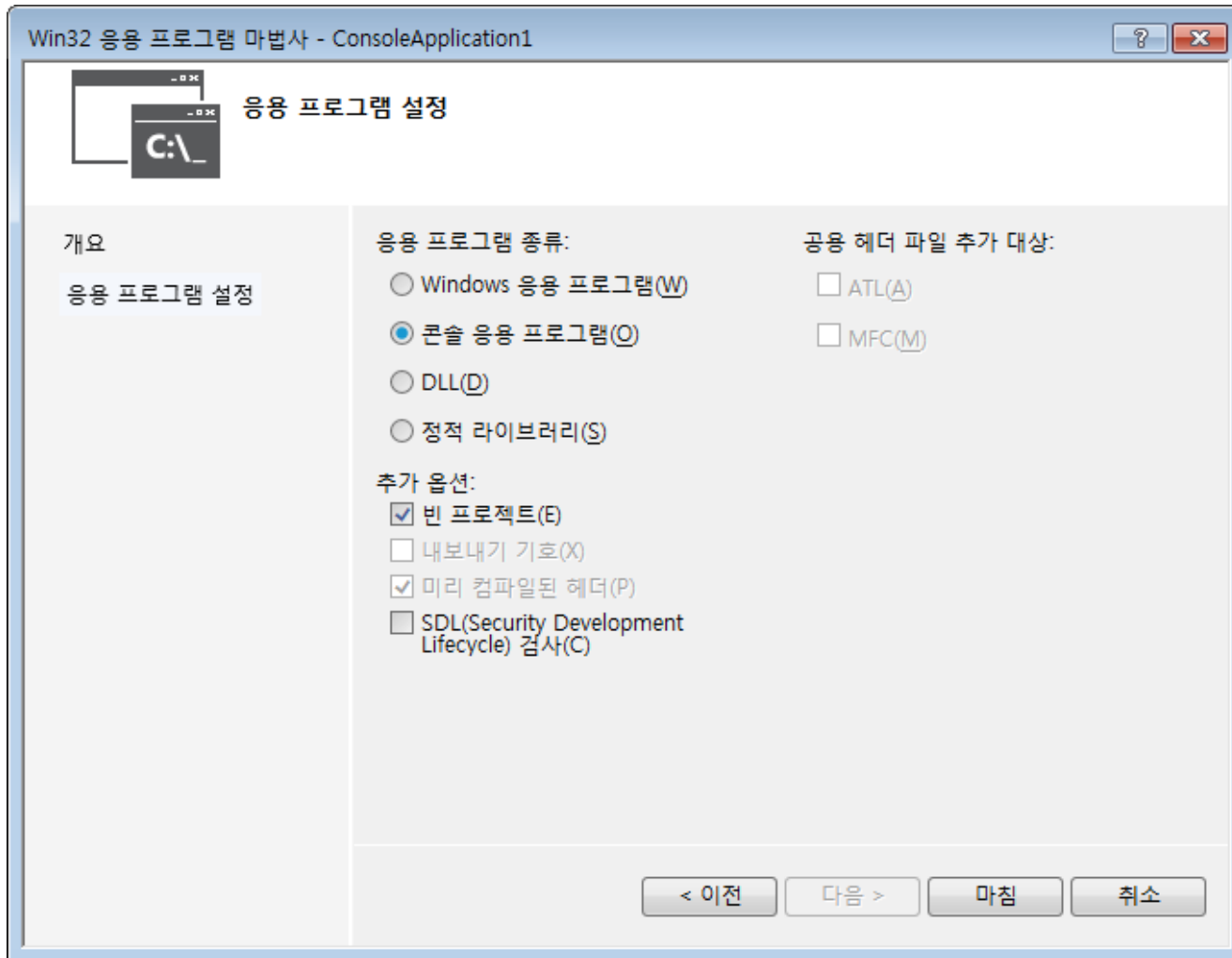
Other files except for glut files may be installed via existing graphics drivers

# Project Creation[Visual Studio 2012] (1/5)

- [File] → [New] (Ctrl+Shift+N)

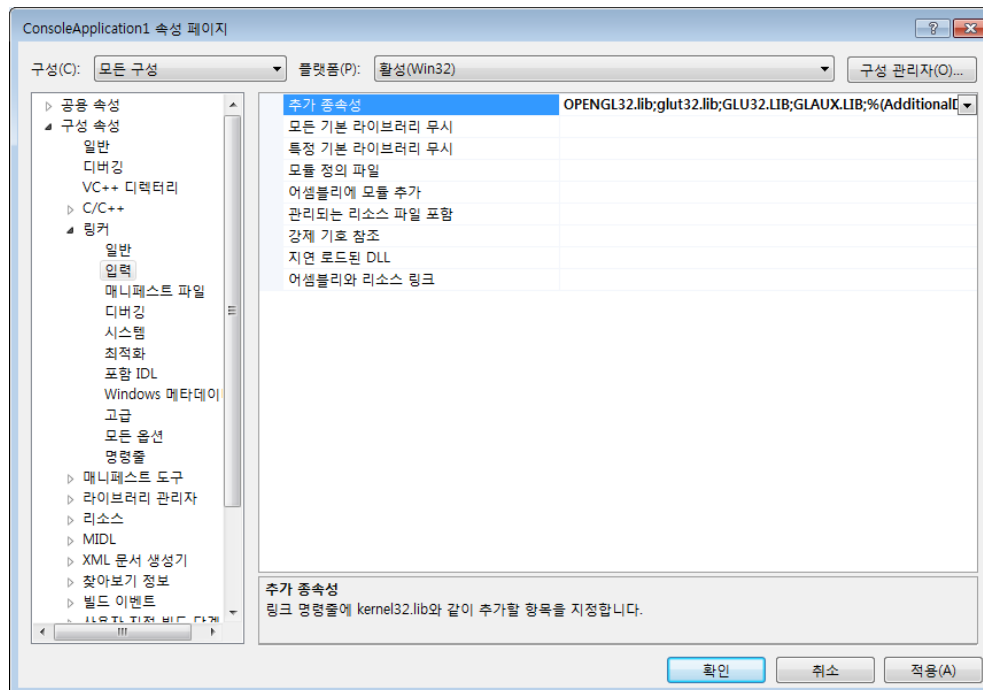


# Project Creation[Visual Studio 2012] (2/5)



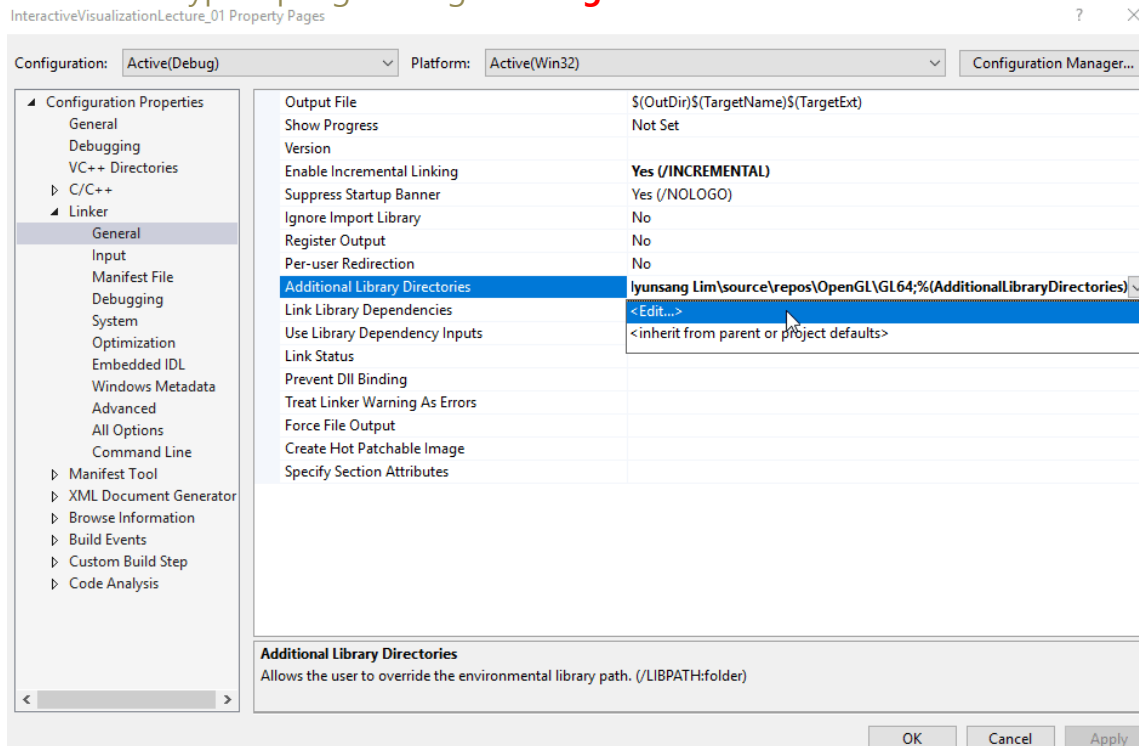
# Project Creation[Visual Studio 2012] (3/5)

- [Project] → [Properties] (Alt+F7)
  - [Configuration Properties] → [Linker] → [Input]
  - [Additional Dependencies]
    - Type 'opengl32.lib glu32.lib **glut32.lib**'



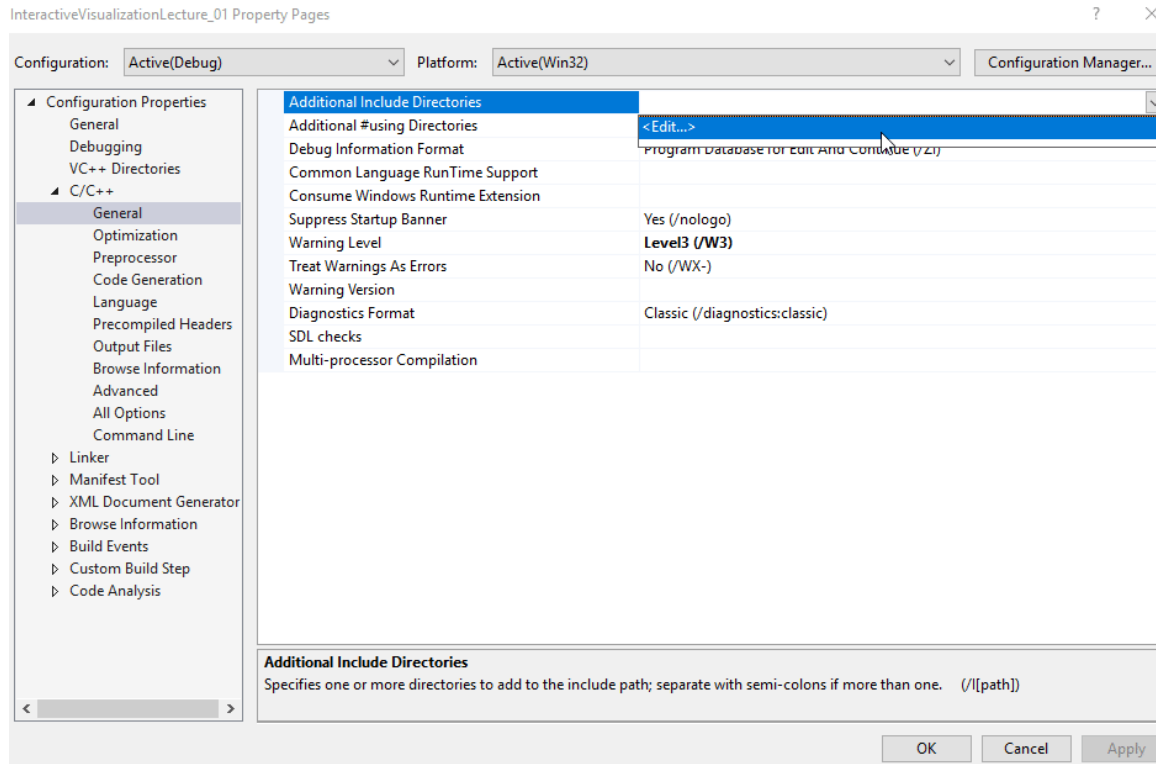
# Project Creation[Visual Studio 2012] (4/5)

- **How to link library files in an additional path other than the default folder**
- [Project] → [Properties]
  - [Configuration Properties] → [Linker] → [Input]
  - [Additional Library Directories]
    - Type 'opengl32.lib glu32.lib **glut32.lib**'



# Project Creation[Visual Studio 2012] (5/5)

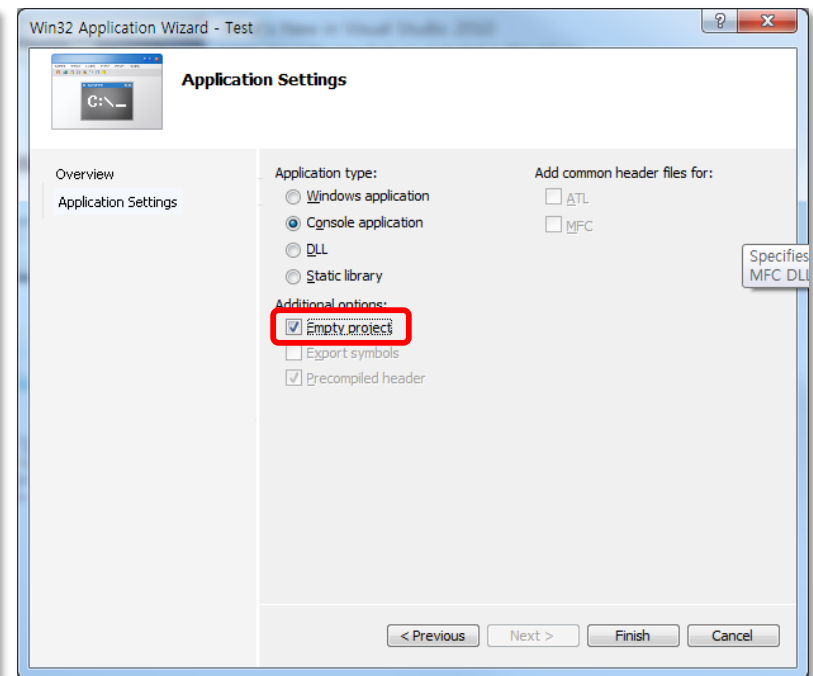
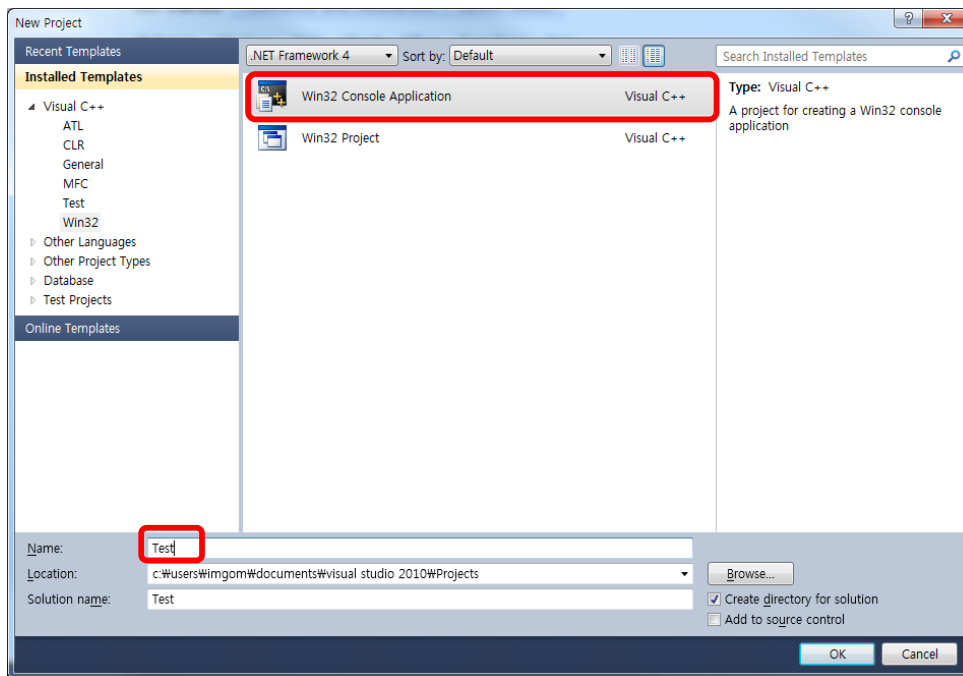
- **How to link header files in an additional path other than the default folder**
- **[Project] → [Properties]**
  - [Configuration Properties] → [C/C++] → [General]
  - [Additional Include Directories]





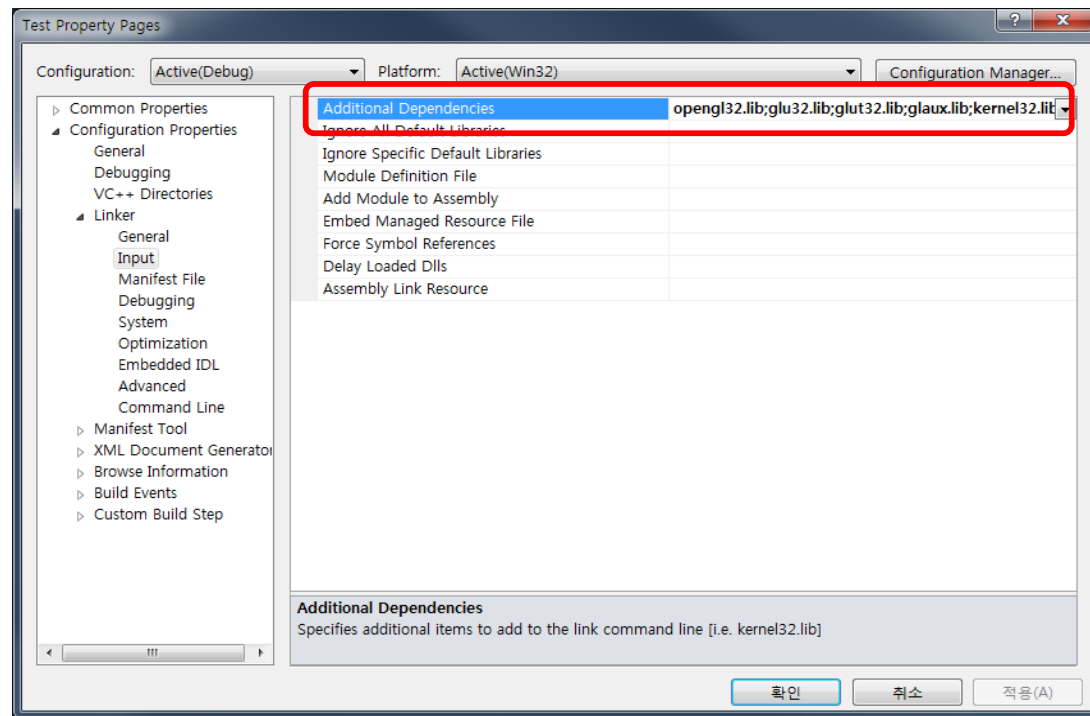
# Project Creation[Visual Studio 2010] (1/2)

- [File] → [New] → [Project] (Ctrl+Shift+N)
  - [Visual C++ ] → [Win32] → [Win32 Console App]
  - Enter the project name
  - [Empty project]



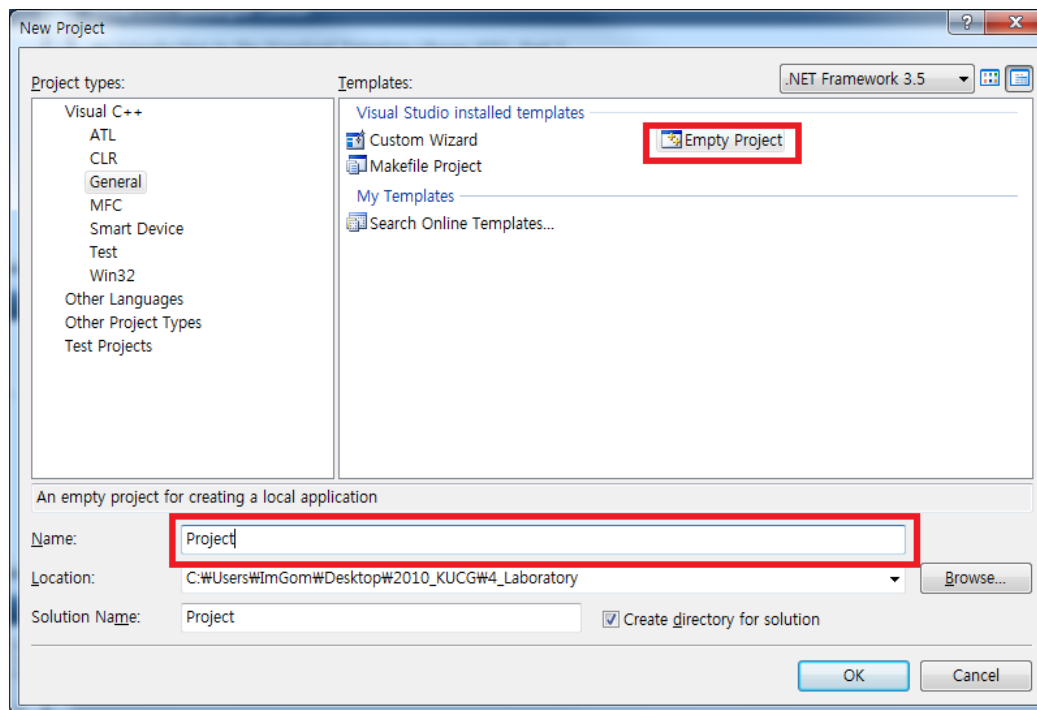
# Project Creation[Visual Studio 2010] (2/2)

- [Project] → [Properties] (Alt+F7)
  - [Configuration Properties] → [Linker] → [Input]
  - [Additional Dependencies]
    - Type 'opengl32.lib glu32.lib **glut32.lib**'



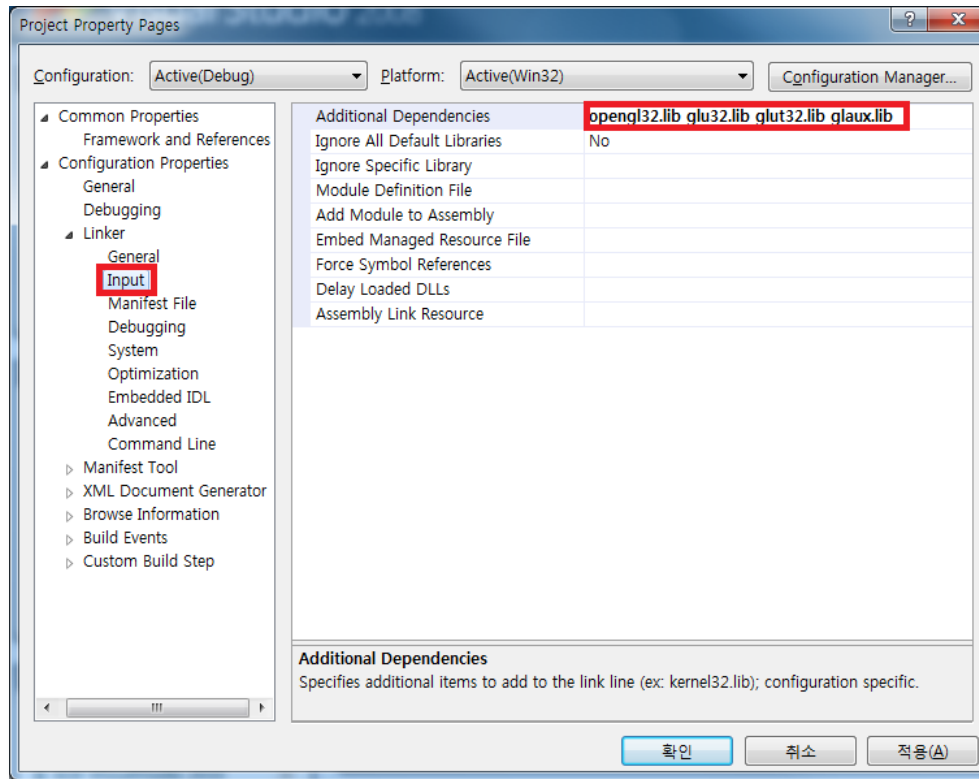
# Project Creation[Visual Studio 2008] (1/2)

- [File] → [New] (Ctrl+Shift+N)
  - [Project types] → [General]
  - [Templates] → [Empty Project]
  - Enter the project file name



# Project Creation[Visual Studio 2008] (2/2)

- [Project] → [Settings...] (Alt+F7)
  - [Link] → [Input] → [Additional Dependencies]
    - Type 'opengl32.lib glu32.lib **glut32.lib**'



# GLEW

- **GLEW?**
  - To use GLSL functions, GLEW( The OpenGL Extension Wrangler Library) must be installed

<http://glew.sourceforge.net/index.html>

Latest Release: 2.1.0



Download  
Usage  
Building  
Installation  
Source Generation  
Change Log

GitHub  
Issues  
Pull Requests  
Authors  
Licensing

SourceForge Page

Last Update: 07-31-17



## The OpenGL Extension Wrangler Library

The OpenGL Extension Wrangler Library (GLEW) is a cross-platform open-source C/C++ extension loading library. GLEW provides efficient run-time mechanisms for determining which OpenGL extensions are supported on the target platform. OpenGL core and extension functionality is exposed in a single header file. GLEW has been tested on a variety of operating systems, including Windows, Linux, Mac OS X, FreeBSD, Irix, and Solaris.

### Downloads

GLEW is distributed as source and precompiled binaries.  
The latest release is 2.1.0[07-31-17]:

Source [ZIP](#) | [TGZ](#)  
Binaries [Windows 32-bit and 64-bit](#)

An up-to-date copy is also available using git:

- [github](#)  
`git clone https://github.com/nigels-com/glew.git glew`

### Supported Extensions

The latest release contains support for OpenGL 4.6, compatibility and forward-compatible contexts and the following extensions:

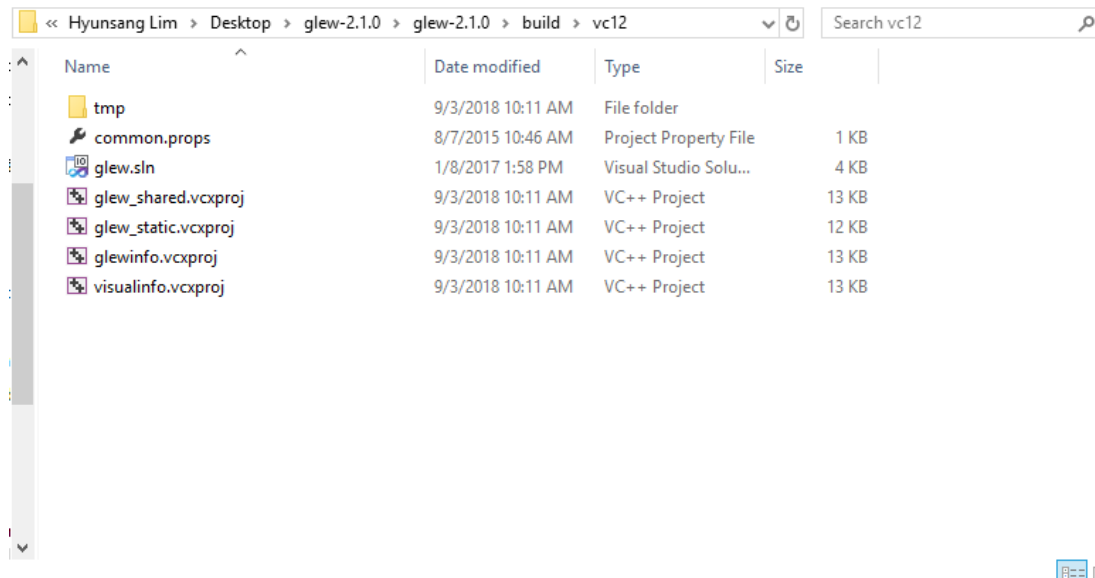
- OpenGL extensions
- WGL extensions
- GLX extensions

### News

- [07-31-17] GLEW 2.1.0 adds support for OpenGL 4.6, new extensions and minor bug fixes
- [07-24-16] GLEW 2.0.0 adds support for forward-compatible contexts, adds new extensions, OSMesa and EGL support, MX discontinued and minor bug fixes
- [08-10-15] GLEW 1.13.0 adds support for new extensions, fixes minor bugs
- [26-01-15] GLEW 1.12.0 fixes minor bugs and adds new extensions
- [08-11-14] GLEW 1.11.0 adds support for OpenGL 4.5, new extensions
- [07-22-13] GLEW 1.10.0 adds support for OpenGL 4.4, new extensions

# GLEW Building

- Download .zip file and extract it.
- Open [glew-2.1.0] - [build] – [vc12] – [glew.sln] and Build it.



# GLEW Installation

- **Header files are in [glew-2.1.0]-[include]-[GL]**
  - Copy header files to your own Include folder
- **Lib files are in [glew-2.1.0] - [lib] – [Debug] – [Win32]**
  - Copy library files to your own Include folder
- **DLL file is in [glew-2.1.0] - [bin] – [Debug] – [Win32]**
  - Copy dll file to C:\WINDOWS\system32 and C:\WINDOWS\SysWow64