# OpenGL Homework 1

CS 550000 Computer Graphics

March 8, 2017

CGV Lab, NTHUCS



# Outline

- Goal
- Demo
- Grading
- Submission
- Reminders

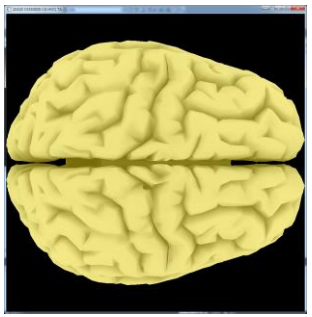
# Outline

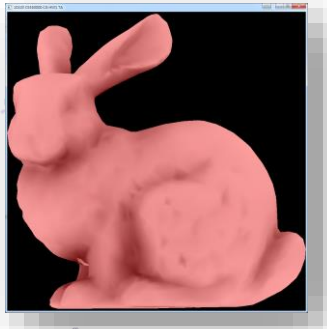
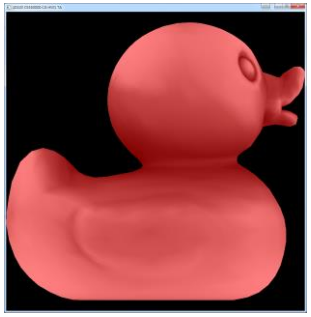
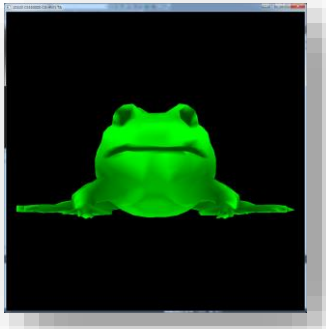
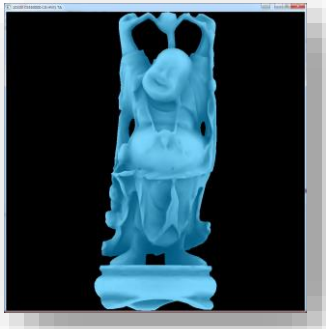
- Goal
- Demo
- Grading
- Submission
- Reminders

# Goal

- Render a 3D model properly on the screen.






# Goal

- Render a 3D model properly on the screen.
- Step by step
  1) Load model(s) from external *.obj file(s).
  2) Normalize the model into [-1, 1] and show it on the screen.
  3) Add color filtering function into your **shader**
  4) Submit your project and get full score.
- Reminders
  - You CANNOT use the existing transform API of OpenGL 1.X.
    e.g. glRotate, glTranslate, glScale
  - You **DO NEED** to modify **vertex/fragment** shaders.

# Outline

- Goal
- Demo
- Grading
- Submission
- Reminders


DEMO

# Outline

- Goal
- Demo
- Grading
- Submission
- Reminders

# Grading

Total score: 100

- Loading model (80 pt)

- Display 3D models properly.

  Vertex correctness : 10 pt

  Color correctness : 10 pt

  Normalize to [-1, +1] : 20 pt

  Wireframe/solid mode : 10 pt

- color filter function

  4 kinds of filter : 30 pt
  (normal, R only, G only, B only)

Keyboard/Mouse (10 pt)

- Utilize the callback function.

  h - to show help menu

  z/x - switch different models
  (**at least 3** models)

  w - switch solid/wireframe mode

  c - color filter function

- Report (10 pt)

- Express your work.

  How to operate your program

  How do you normalize models into [-1, 1]

  Implementation and problems you met

  Other efforts you have done

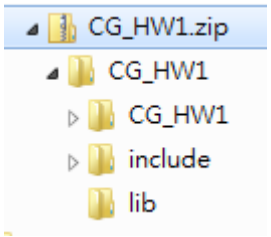
  Screenshots

# Outline

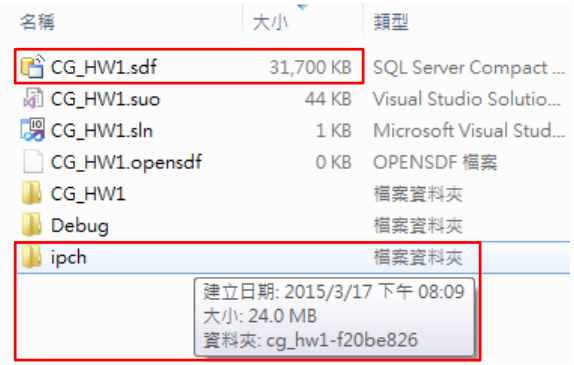
- Goal
- Demo
- Grading
- Submission
- Reminders

# Submission

- Submit your project to iLMS.
- Filename: HW1_XXXXXXXXX.zip
- Put both “lib” and “include” in your zip file




- *** Remove “ipch” folder and “.sdf” file. ***



# Outline

- Goal
- Demo
- Grading
- Submission
- Reminders

# Reminders

- Late submission is accepted, DO NOT give up!
- Ask and share information through iLMS

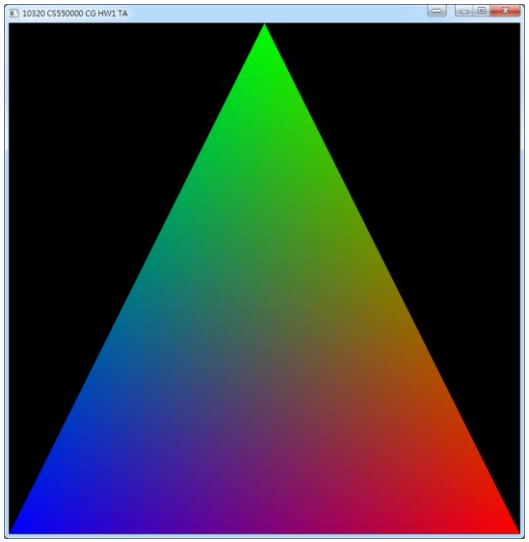
# Appendix

CS 550000 Computer Graphics
March 8, 2017
Department of Computer Science
National Tsing Hua University



# How to create an OpenGL project?

- Download the framework by TA from iLMS.
- Start programming with OpenGL.




Hello Triangle!

- If you want to create a project from scratch…

# How to create an OpenGL project?

1. Make everything ready…

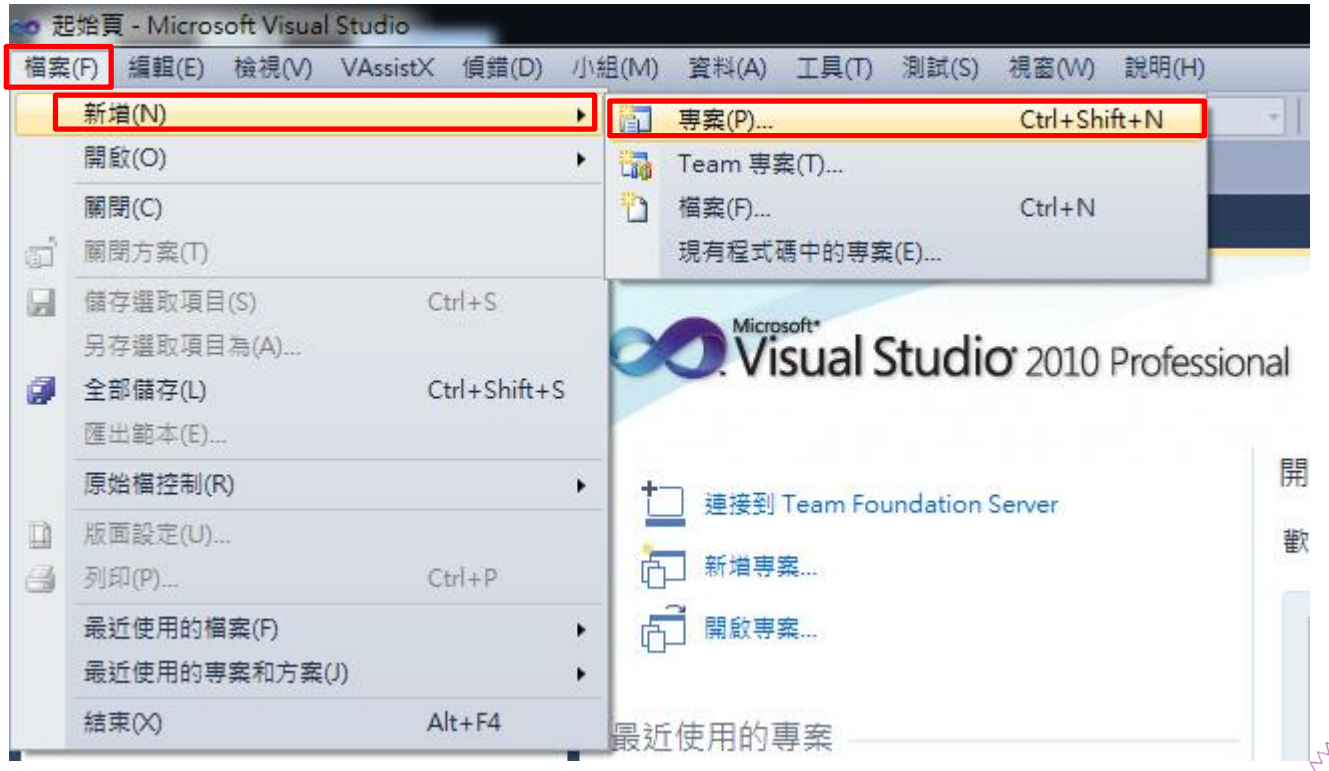
   freeglut 2.8.1 (download)

   glew 1.12.0 (download)

- CG_HW1
  - CG_HW1
    - include
      - freeglut
        - freeglut.h
        - freeglut_ext.h
        - freeglut_std.h
        - glut.h
      - GL
        - glew.h
        - glxew.h
        - wglew.h
    - lib
      - freeglut.dll
      - freeglut.lib
      - glew32.dll
      - glew32.lib

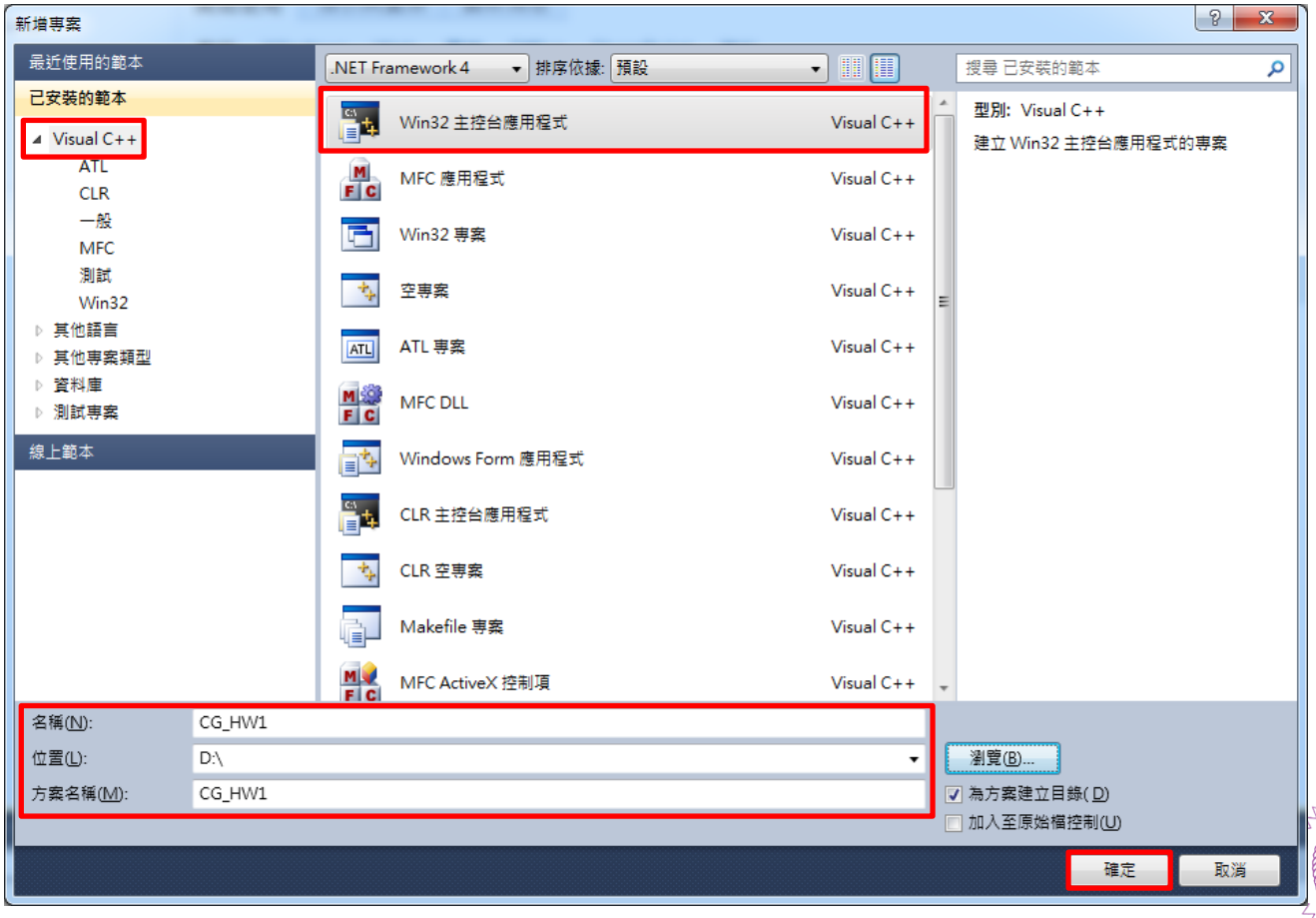
# How to create an OpenGL project?

1. Create an empty project (e.g. M$ VS C++ 2010)



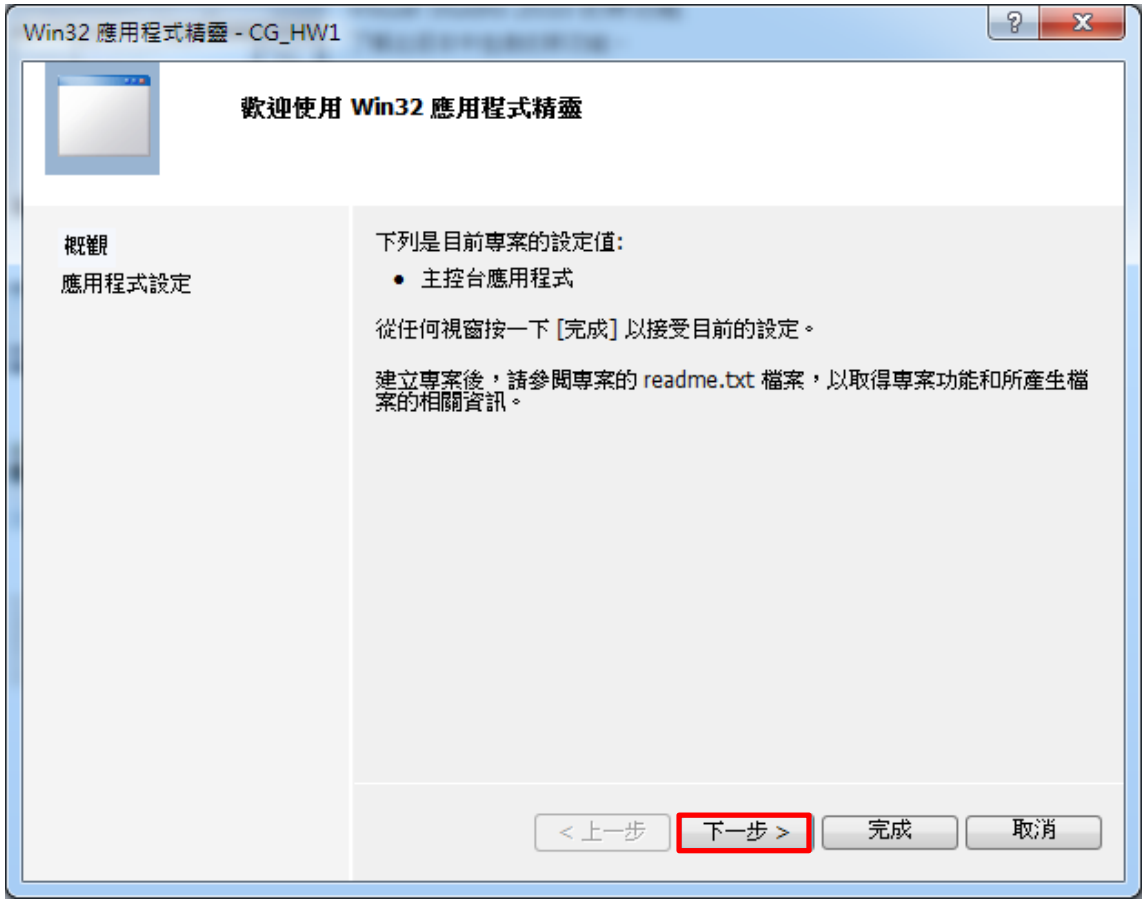
# How to create an OpenGL project?

1. Create an empty project



# How to create an OpenGL project?

1. Create an empty project

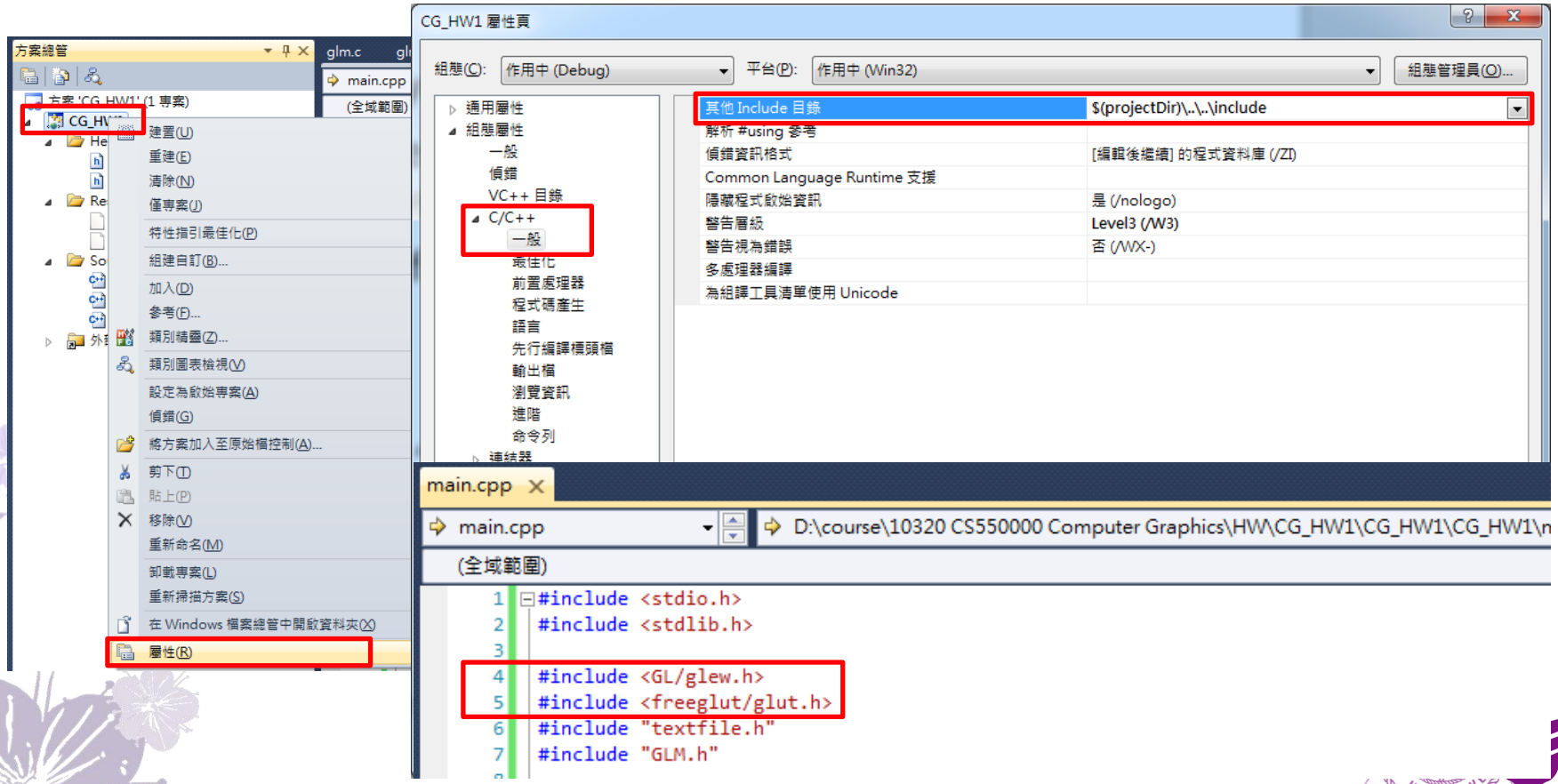


# How to create an OpenGL project?

1. Create an empty project

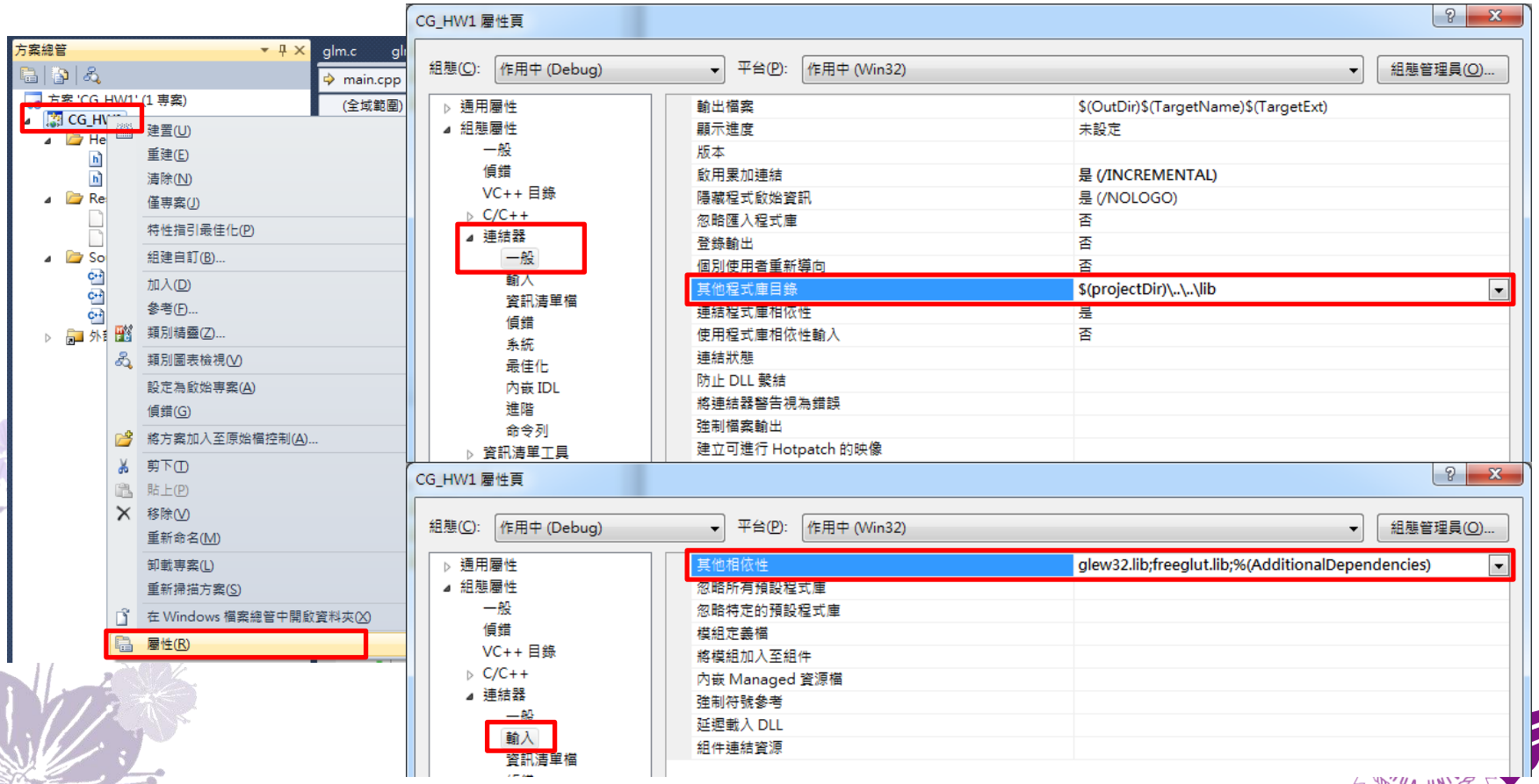
# How to create an OpenGL project?

3. Include headers…



# How to create an OpenGL project?

4. Link libraries…



# How to create an OpenGL project?

5. Start programming!


main.cpp

processNormalKeys | void processNormalKeys(unsigned char key, int x, int y)

(全域範圍)

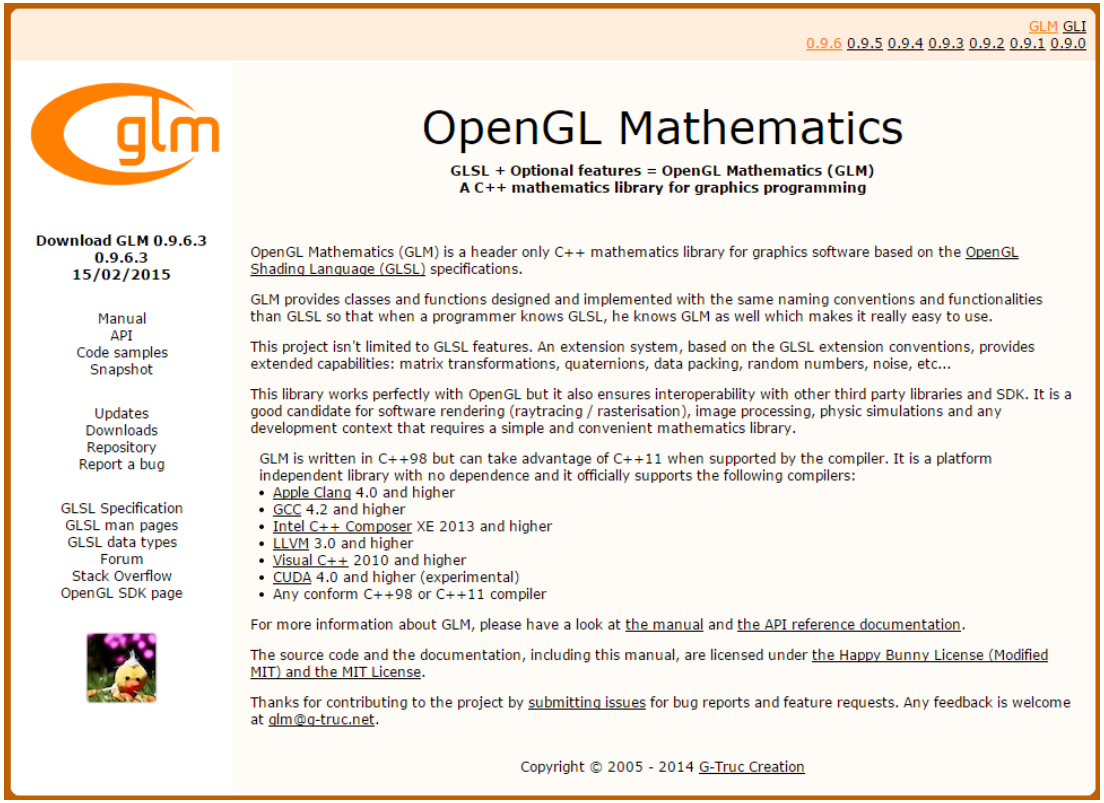
```
235
236 int main(int argc, char **argv) {
237     // glut init
238     glutInit(&argc, argv);
239     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
240
241     // create window
242     glutInitWindowPosition(460, 40);
243     glutInitWindowSize(800, 800);
244     glutCreateWindow("10320 CS550000 CG HW1 TA");
245
246     glewInit();
247     if(glewIsSupported("GL_VERSION_2_0")){
248         printf("Ready for OpenGL 2.0\n");
249     }else{
250         printf("OpenGL 2.0 not supported\n");
251         system("pause");
252         exit(1);
253     }
254
255     // load obj models through glm
256     loadOBJModel();
257
258     // register glut callback functions
259     glutDisplayFunc (renderScene);
260     glutIdleFunc    (idle);
261     glutKeyboardFunc(processNormalKeys);
262     glutMouseFunc   (processMouse);
263     glutMotionFunc  (processMouseMotion);
264
265     glEnable(GL_DEPTH_TEST);
266
267     // set up shaders here
268     setShaders();
269
270     // main loop
271     glutMainLoop();
272
273     // free
274     glmDelete(OBJ);
275
276     return 0;
277 }
278
279
```

# API & Library

CS 550000 Computer Graphics
March 8, 2017
Department of Computer Science
National Tsing Hua University


# GLM

- OpenGL Mathematics
  - http://glm.g-truc.net/0.9.6/index.html



GLM GLI
0.9.6 0.9.5 0.9.4 0.9.3 0.9.2 0.9.1 0.9.0

glm

Download GLM 0.9.6.3
0.9.6.3
15/02/2015

Manual
API
Code samples
Snapshot

Updates
Downloads
Repository
Report a bug

GLSL Specification
GLSL man pages
GLSL data types
Forum
Stack Overflow
OpenGL SDK page

## OpenGL Mathematics

**GLSL + Optional features = OpenGL Mathematics (GLM)**
**A C++ mathematics library for graphics programming**

OpenGL Mathematics (GLM) is a header only C++ mathematics library for graphics software based on the OpenGL Shading Language (GLSL) specifications.

GLM provides classes and functions designed and implemented with the same naming conventions and functionalities than GLSL so that when a programmer knows GLSL, he knows GLM as well which makes it really easy to use.

This project isn't limited to GLSL features. An extension system, based on the GLSL extension conventions, provides extended capabilities: matrix transformations, quaternions, data packing, random numbers, noise, etc...

This library works perfectly with OpenGL but it also ensures interoperability with other third party libraries and SDK. It is a good candidate for software rendering (raytracing / rasterisation), image processing, physic simulations and any development context that requires a simple and convenient mathematics library.

GLM is written in C++98 but can take advantage of C++11 when supported by the compiler. It is a platform independent library with no dependence and it officially supports the following compilers:

- Apple Clang 4.0 and higher
- GCC 4.2 and higher
- Intel C++ Composer XE 2013 and higher
- LLVM 3.0 and higher
- Visual C++ 2010 and higher
- CUDA 4.0 and higher (experimental)
- Any conform C++98 or C++11 compiler

For more information about GLM, please have a look at the manual and the API reference documentation.

The source code and the documentation, including this manual, are licensed under the Happy Bunny License (Modified MIT) and the MIT License.

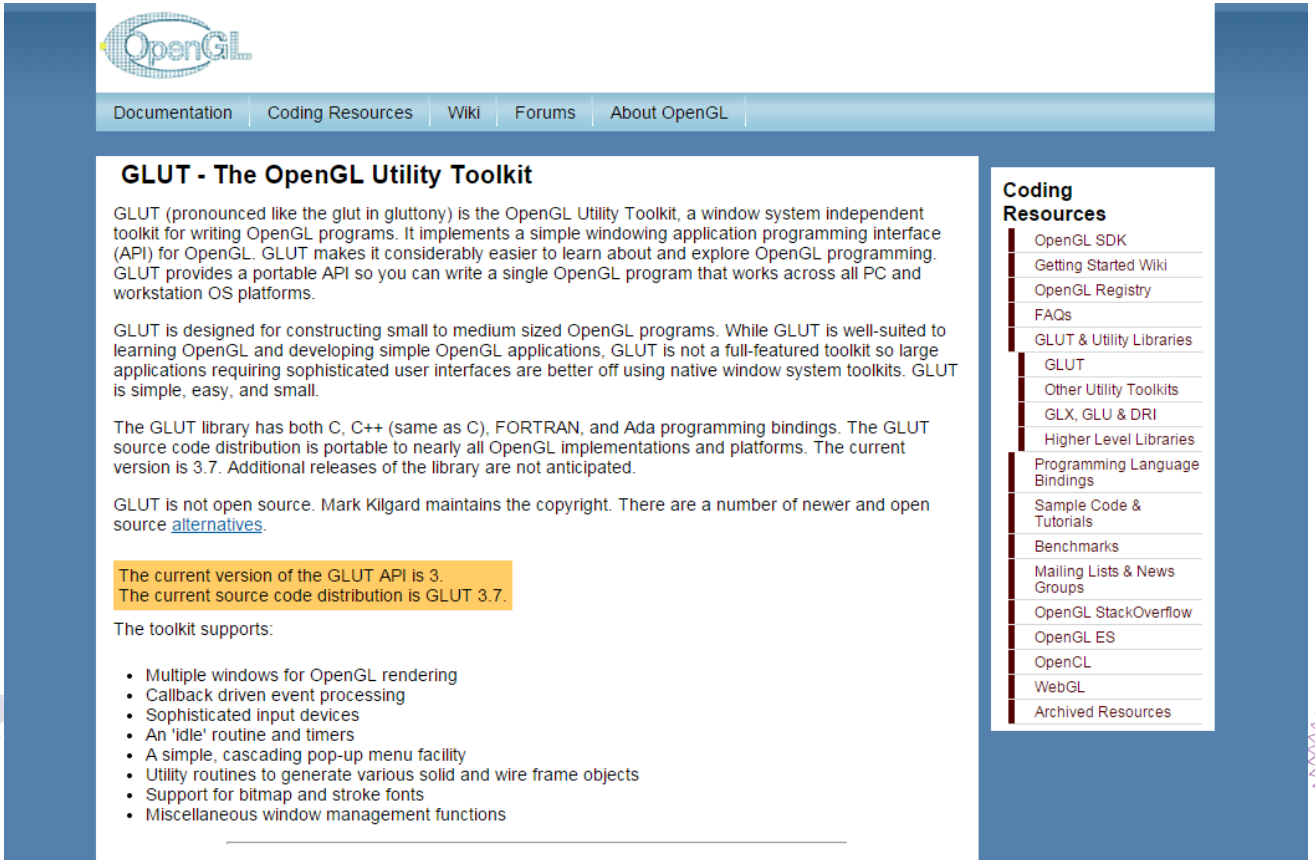
Thanks for contributing to the project by submitting issues for bug reports and feature requests. Any feedback is welcome at glm@g-truc.net.

Copyright © 2005 - 2014 G-Truc Creation

```
65
66  /* GLMmodel: Structure that defines a model.
67    */
68  typedef struct _GLMmodel {
69    GLfloat scale;                /* the scale to make the model's max width as 2 unit */
70
71    char*    pathname;            /* path to this model */
72    char*    mtllibname;          /* name of the material library */
73
74    GLuint   numvertices;         /* number of vertices in model */
75    GLfloat* vertices;            /* array of vertices  */
76    GLfloat* colors;
77
78    GLuint   numnormals;          /* number of normals in model */
79    GLfloat* normals;             /* array of normals */
80
81    GLuint   numtexcoords;        /* number of texcoords in model */
82    GLfloat* texcoords;           /* array of texture coordinates */
83
84    GLuint   numfacetnorms;       /* number of facetnorms in model */
85    GLfloat* facetnorms;          /* array of facetnorms */
86
87    GLuint       numtriangles;    /* number of triangles in model */
88    GLMtriangle* triangles;       /* array of triangles */
89
90    GLuint       nummaterials;    /* number of materials in model */
91    GLMmaterial* materials;       /* array of materials */
92
93    GLuint       numgroups;       /* number of groups in model */
94    GLMgroup*    groups;          /* linked list of groups */
95
96    GLfloat position[3];          /* position of the model */
97  } GLMmodel;
98
```

# GLUT

- The OpenGL Utility Toolkit
  - https://www.opengl.org/resources/libraries/glut/



```
235
236 int main(int argc, char **argv) {
237     // glut init
238     glutInit(&argc, argv);
239     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
240
241     // create window
242     glutInitWindowPosition(460, 40);
243     glutInitWindowSize(800, 800);
244     glutCreateWindow("10320 CS550000 CG HW1 TA");
245
246     glewInit();
247     if(glewIsSupported("GL_VERSION_2_0")){
248         printf("Ready for OpenGL 2.0\n");
249     }else{
250         printf("OpenGL 2.0 not supported\n");
251         system("pause");
252         exit(1);
253     }
254
255     // load obj models through glm
256     loadOBJModel();
257
258     // register glut callback functions
259     glutDisplayFunc (renderScene);
260     glutIdleFunc     (idle);
261     glutKeyboardFunc(processNormalKeys);
262     glutMouseFunc    (processMouse);
263     glutMotionFunc   (processMouseMotion);
264
265     glEnable(GL_DEPTH_TEST);
266
267     // set up shaders here
268     setShaders();
269
270     // main loop
271     glutMainLoop();
272
273     // free
274     glmDelete(OBJ);
275
276     return 0;
277 }
278
```

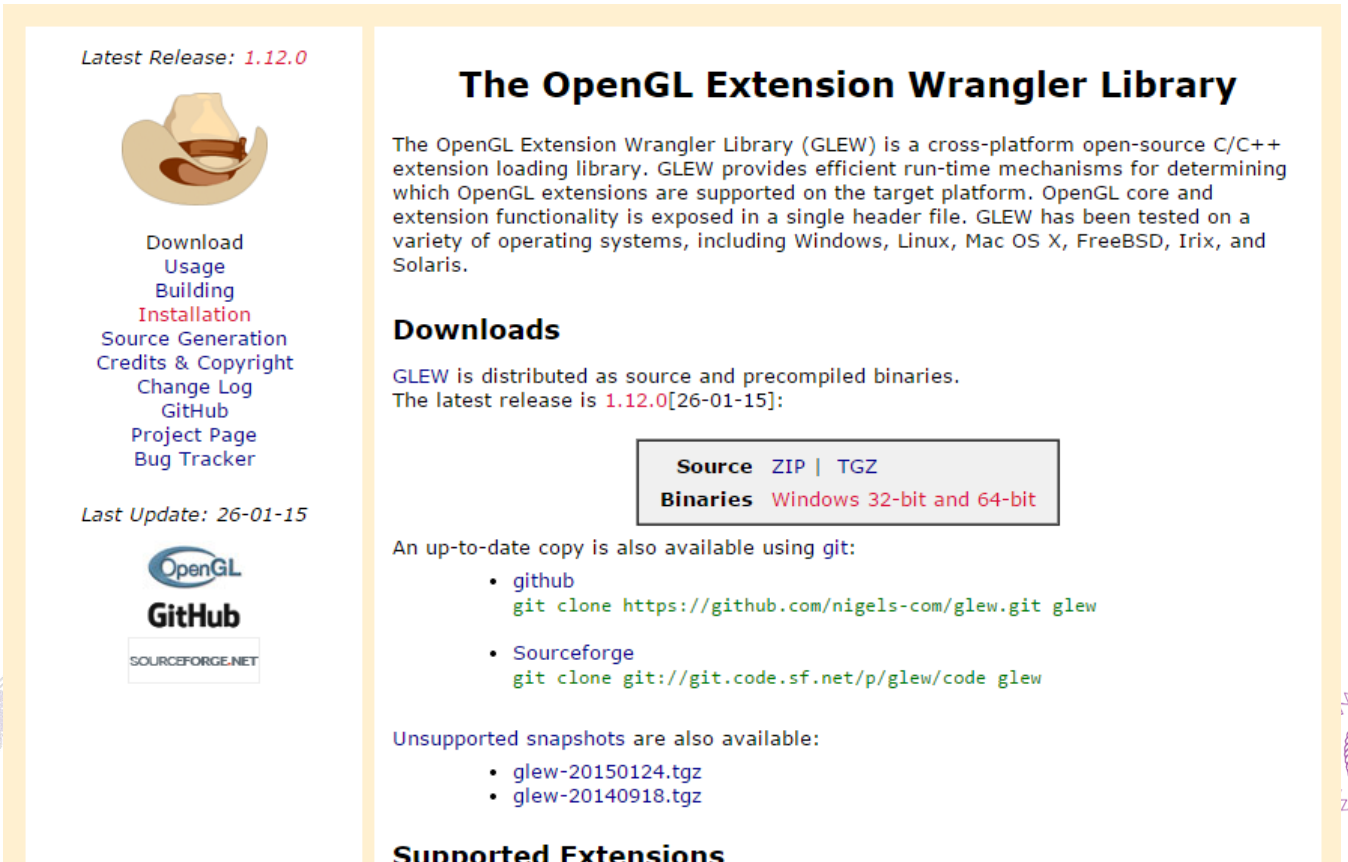
```
// glut init
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);

// create window
glutInitWindowPosition(460, 40);
glutInitWindowSize(800, 800);
glutCreateWindow("10320 CS550000 CG HW1 TA");
```

```
// register glut callback functions
glutDisplayFunc (renderScene);
glutIdleFunc     (idle);
glutKeyboardFunc(processNormalKeys);
glutMouseFunc    (processMouse);
glutMotionFunc   (processMouseMotion);
```

```
// main loop
glutMainLoop();
```

# GLEW

- The OpenGL Extension Wrangler Library
  - http://glew.sourceforge.net/



*Latest Release: 1.12.0*

Download
Usage
Building
Installation
Source Generation
Credits & Copyright
Change Log
GitHub
Project Page
Bug Tracker

*Last Update: 26-01-15*

OpenGL
GitHub
SOURCEFORGE.NET

## 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 1.12.0[26-01-15]:

| | |
|---|---|
| **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`
- Sourceforge
  `git clone git://git.code.sf.net/p/glew/code glew`

Unsupported snapshots are also available:

- glew-20150124.tgz
- glew-20140918.tgz

### Supported Extensions

```
108 void renderScene(void)
109 {
110     // clear canvas
111     glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
112     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
113
114     glEnableVertexAttribArray(iLocPosition);
115     glEnableVertexAttribArray(iLocColor);
116
117     static GLfloat triangle_color[] = {
118         1.0f, 0.0f, 0.0f,
119         0.0f, 1.0f, 0.0f,
120         0.0f, 0.0f, 1.0f
121     };
122
123     static GLfloat triangle_vertex[] = {
124          1.0f, -1.0f, 0.0f,
125          0.0f,  1.0f, 0.0f,
126         -1.0f, -1.0f, 0.0f
127     };
128
129     glVertexAttribPointer(iLocPosition, 3, GL_FLOAT, GL_FALSE, 0, triangle_vertex);
130     glVertexAttribPointer(   iLocColor, 3, GL_FLOAT, GL_FALSE, 0, triangle_color);
131
132     // draw the array we just bound
133     glDrawArrays(GL_TRIANGLES, 0, 3);
134
135     glutSwapBuffers();
136 }
```

clear frame buffer & z buffer

$x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3, \ldots\ldots$

$r_1, g_1, b_1, r_2, g_2, b_2, r_3, g_3, b_3, \ldots\ldots$

stride

start index

num of vertices

draw the entire array


Q
&
A