CSCI3260 PRINCIPLES OF COMPUTER GRAPHICS

Tutorial 1

Colman Leung

Outline

- Course & Tutorial information
- What is GPU and OpenGL ?
- Why OpenGL ?
- How to program with OpenGL ?

About This Course

- Leung Ho Man Colman (hmleung@cse.cuhk.edu.hk)
  Office: SHB 1024
  Office Hour: WED 2:30pm – 4:30pm

- Tutorial time:
  Tuesday 10:30am – 11:15am LSK LT3

- Program Platform :
  Visual C++ 2008 / 2010 / 2012

About Tutorial

- You will learn:

  2D & 3D programming

  - OpenGL (Open Graphics Library)

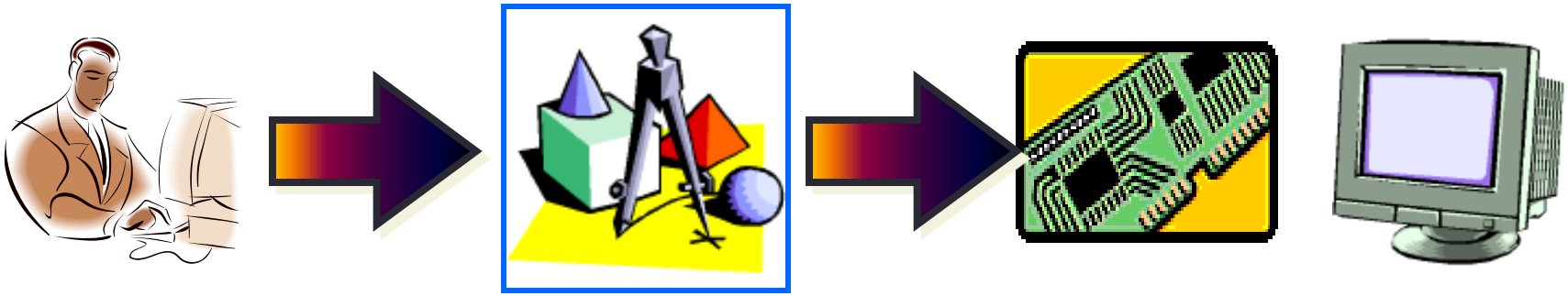
    A cross-platform standard for 2D/3D graphics programming

What is GPU

- GPU – Graphics Processing Unit
- Real-time and realistic games with hairy characters, realistic water/fires, movie-like game scenes

What is OpenGL?

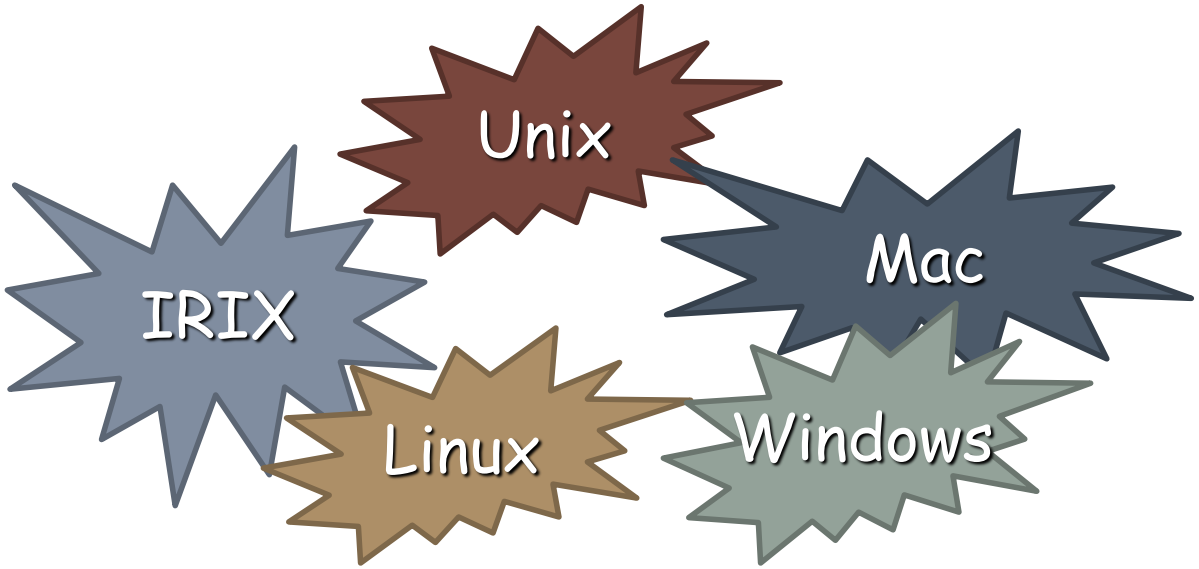
- OpenGL (Open Graphics Library)
- A software interface to graphics hardware (GPU)
- API (Application Programming Interface) for developing interactive 2D and 3D graphics applications



Developer | **OpenGL API** | **Graphics Hardware**

Why OpenGL?

- Fast communication between software and hardware
- Complete 3D hardware acceleration
- Makes real-time 3D effects possible
- Cross platforms

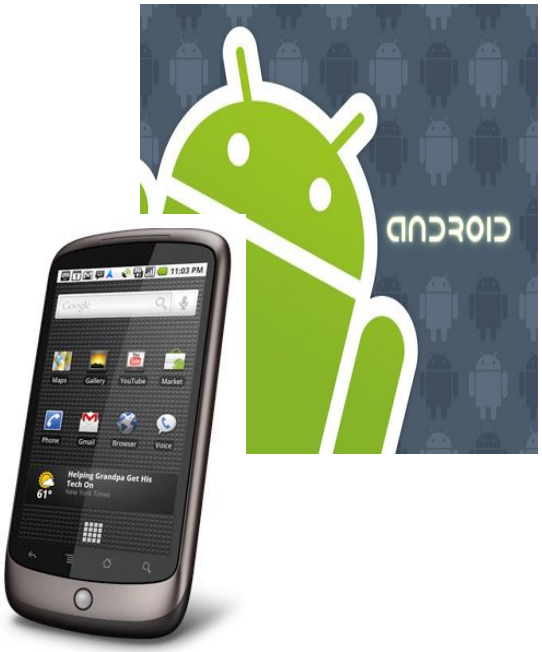


More about openGL

- OpenGL vs. DirectX
  - Both 2D/3D graphics API
  - Both for high performance graphics app.
  - OpenGL is multi-platform; DX is for Windows only
  - DX is a more complicated API (powerful for sound and video)
- OpenGL Shading Language (GLSL)
  - API for controling shader in GPU
  - Realize more realistic graphics effect
  - Design high performance parallel application for general purpose (GPGPU)

More about openGL

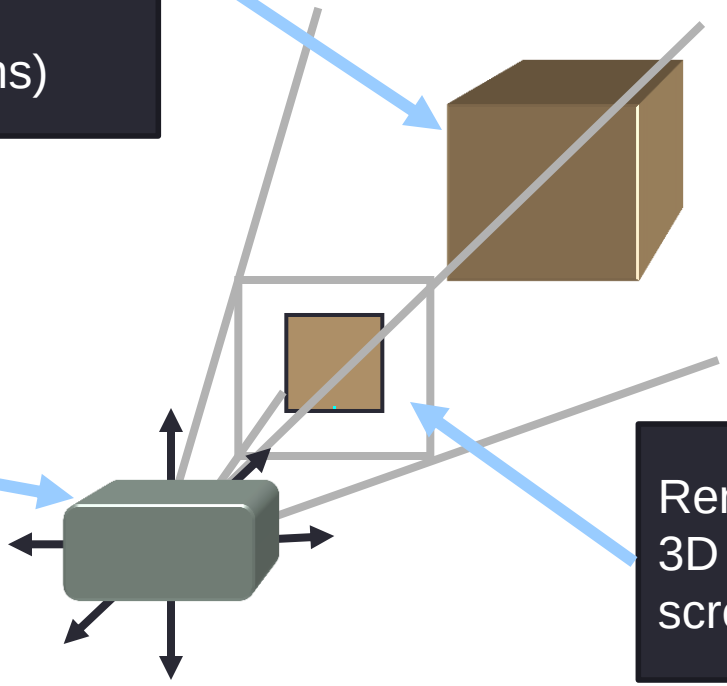
- OpenGL ES (OpenGL for Embedded Systems)
  - For portable device (cell phone, PDA, video game consoles)
  - Multiple platform (iPhone, Windows mobile, Android, …)



OpenGL Overview

Draw basic graphics primitives in 3D space (points, lines, triangles & polygons)

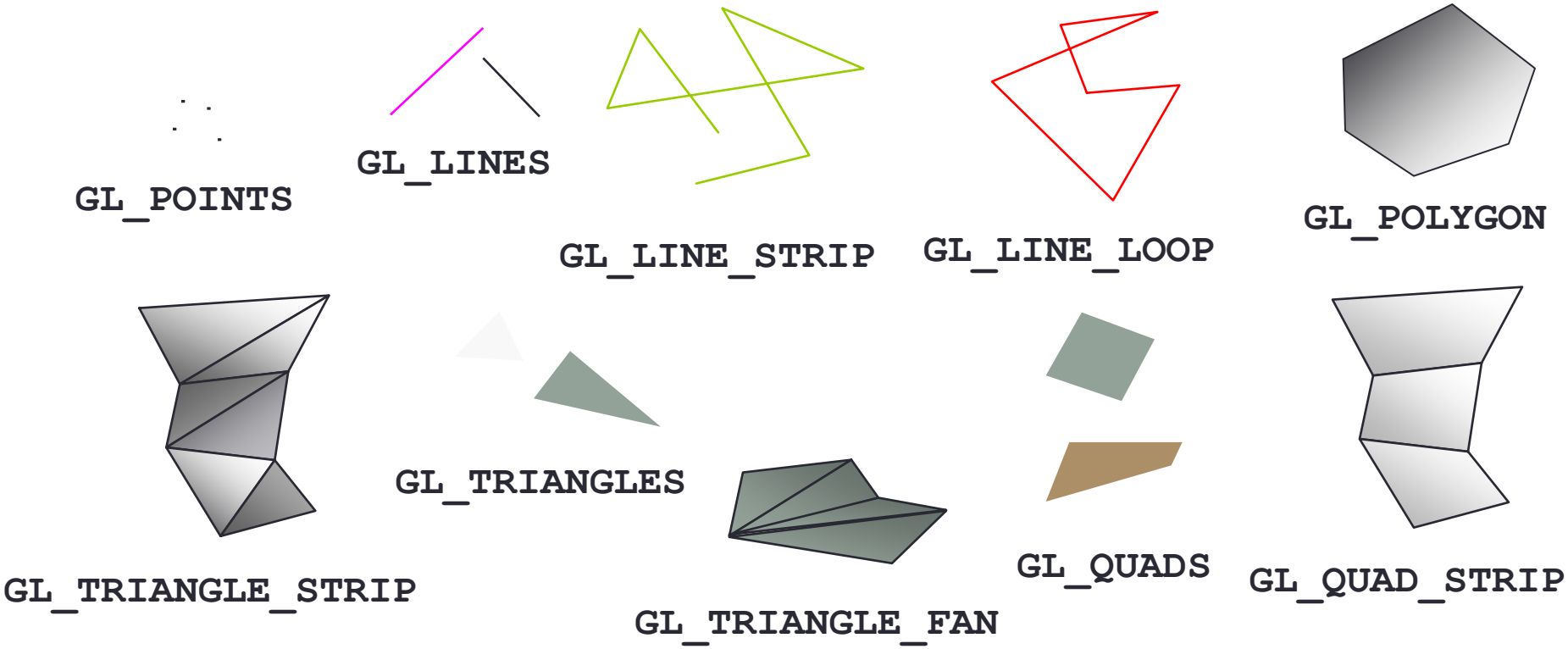
Setup the virtual camera



Render (draw the 3D scene on a 2D screen)

OpenGL Geometric Primitives

- All Geometric Primitives are specified by vertices



OpenGL Function Naming Convention

gl → Low level
glu → High
glut → Higher

Library

Command

glVertex3fv(*v*)

Number of components

2 - (x,y)
3 - (x,y,z)
4 - (x,y,z,w)

Data Type

b - byte
ub - unsigned byte
s - short
us - unsigned short
i - int
ui - unsigned int
f - float
d - double

Vector

omit "v" for scalar form

glVertex2f(x, y)

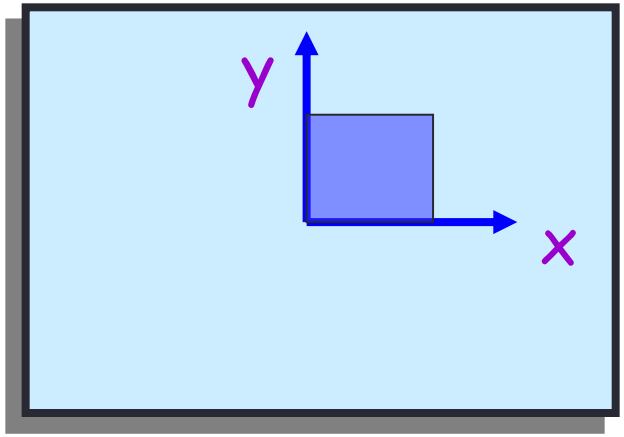
Coordinate System

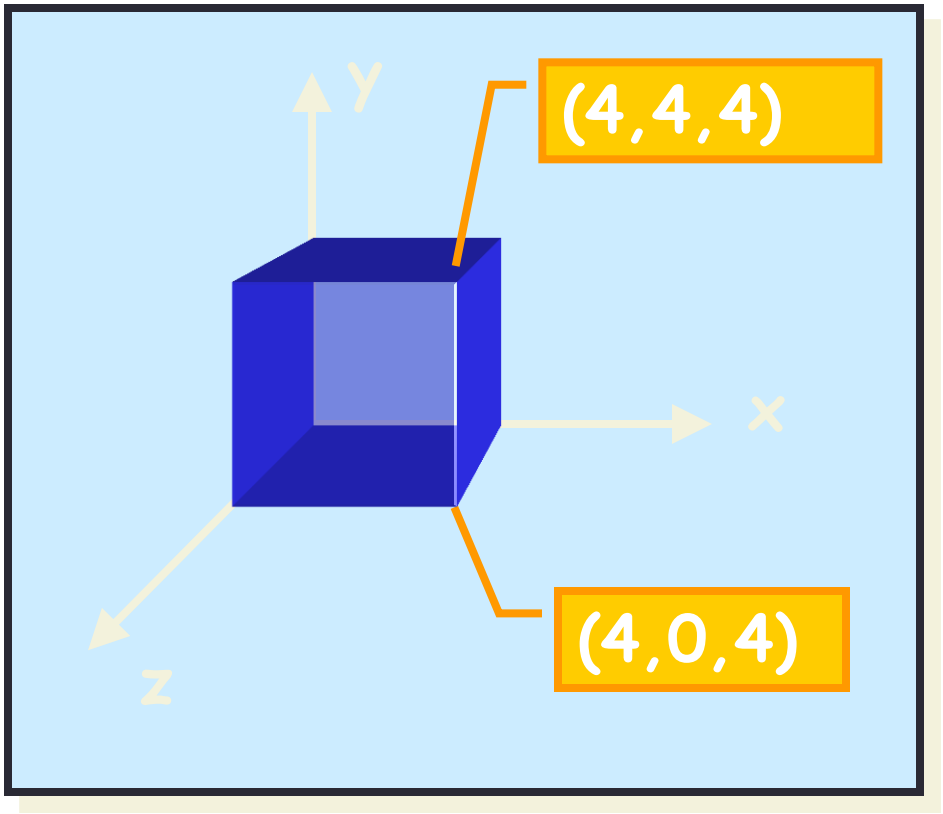
2D point (a, b)

→ glVertex2*(a, b)

3D vertex (a, b, c);

→ glVertex3*(a, b, c);

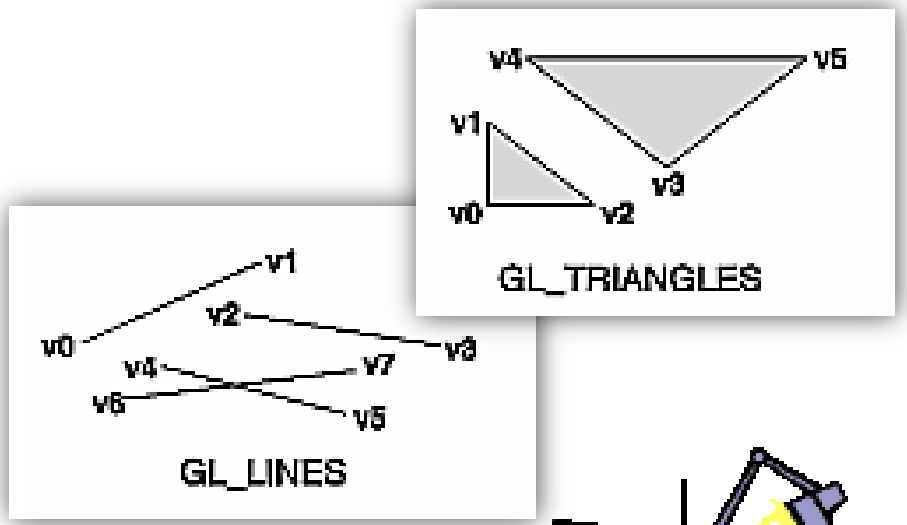




Primitive Drawing in OpenGL

- To draw a set of points, lines, triangles or polygon use glBegin & glEnd

```
glBegin(mode);
glVertex3*(x0, y0, z0);
glVertex3*(x1, y1, z1);
...
glEnd();
```

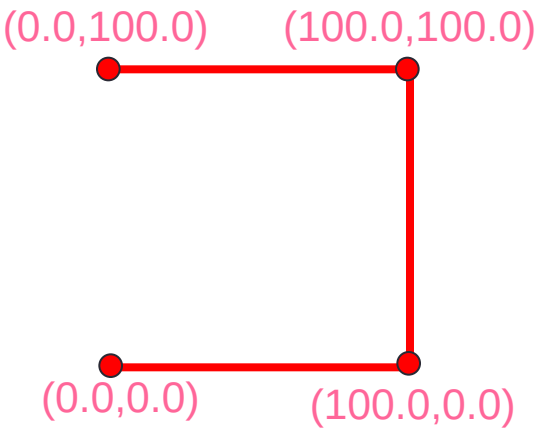


mode can be

GL_POINTS, GL_LINES, GL_POLYGON, GL_TRINGLES, GL_LINE_STRIP, GL_LINE_LOOP, GL_QUADS, GL_QUADS_STRIP, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN

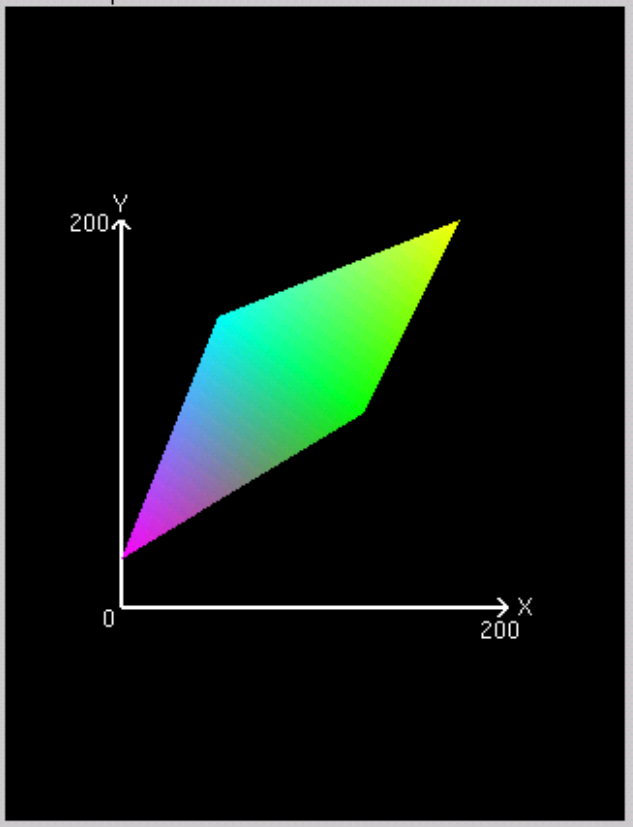
Sample OpenGL commands

```
DrawLines(){
  glColor3f(1.0,0.0,0.0);//R,G,B
  glBegin(GL_LINE_STRIP)
       glVertex2f(0.0, 0.0);
       glVertex2f(100.0, 0.0);
       glVertex2f(100.0, 100.0);
       glVertex2f(0.0, 100.0);
   glEnd();
}
```



An Example - color

Screen-space view



Command manipulation window

```
glBegin (GL_TRIANGLE_STRIP);
glColor3f  (1.00  , 0.00  , 1.00  );
glVertex2f (0.0   , 25.0  );
glColor3f  (0.00  , 1.00  , 1.00  );
glVertex2f (50.0  , 150.0 );
glColor3f  (0.00  , 1.00  , 0.00  );
glVertex2f (125.0 , 100.0 );
glColor3f  (1.00  , 1.00  , 0.00  );
glVertex2f (175.0 , 200.0 );
glEnd();
```

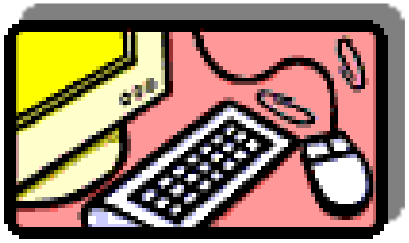
OpenGL is a state machine

- Rendering attributes are encapsulated in state
  - Primitive parameters; rendering styles; shading; lighting; texture
- Appearance is controlled by current state
- One particular state remains unchanged until you change it by various OpenGL commands
- E.g. executing … glColor3f(1.0,1.0,1.0);

Color of objects drawn afterwards will become white until another glColor is called

OpenGL Overview

- What OpenGL (solely) doesn't do?
  - No high-level commands to describe complex object. (e.g. NURBs curve/surface)
  - No commands for performing windowing events (e.g. mouse click or key stroke).



To develop an interactive graphics application, other OpenGL related utility libraries are required.

OpenGL Related Libraries

- OpenGL Core Library (GL)
  - Core OpenGL commands
- OpenGL Utility Library (GLU)
  - Higher level routines using a set of basic OpenGL commands
  - Implemented with core OpenGL, provide auxiliary features, like surface and curves, projection, etc
- OpenGL Utility Toolkit Library (GLUT)
  - Similar to GLU but more powerful
  - Addressing the problems of interfacing with the window system

Using GLUT!

- OpenGL Utility Toolkit (GLUT)
  - A window system independent toolkit for writing OpenGL programs.
  - Window creating, window destroying and event handling which are not included in OpenGL.
  - Create more complicated 3-D objects, such as a sphere, a torus, a teapot…

Application Structure

Configure and open window

Initialize OpenGL state

Register input callback function

- ❖render
- ❖resize
- ❖input: keyboard, mouse, etc.

Enter event processing loop

Sample Program

```
#include <glut.h>
void main(int argc, char **argv){
 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_RGB|GLUT_DEPTH);
 glutInitWindowSize(250,250);
 glutInitWindowPosition(100,100);
 glutCreateWindow("Hello");

 init();

 glutDisplayFunc(display);
 glutReshapeFunc( resize );
 glutKeyboardFunc( key );
 glutIdleFunc( idle );

 glutMainLoop();
}
```

Configure and open window

Initialize OpenGL state

Register several event callback functions

Go into forever looping

Initialize a Window

```
#include <glut.h>
void main(int argc, char **argv){
 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_RGB|GLUT_DEPTH);
 glutInitWindowSize(250,250);
 glutInitWindowPosition(100,100);
 glutCreateWindow("Hello");

 init();

 glutDisplayFunc(display);
 glutReshapeFunc( resize );
 glutKeyboardFunc( key );
 glutIdleFunc( idle );

 glutMainLoop();
 }
```

Initialize GLUT and process any command line arguments, be called first

Specify color model, buffer

the size in pixel of your window

location of the upper-left corner of window

The title of your window

Initialize the State

```
#include <glut.h>
void main(int argc, char **argv){
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGB|GLUT_DEPT);
  glutInitWindowSize(250,250);
  glutInitWindowPosition(100,100);
  glutCreateWindow("Hello");

  init();

  glutDisplayFunc(d
  glutReshapeFunc(
  glutKeyboardFunc(
  glutIdleFunc( idl

  glutMainLoop();
}
```

```
void init( void )
{
  glClearColor( 0.0, 0.0, 0.0, 1.0 );
  glClearDepth( 1.0 );

  glEnable( GL_LIGHT0 );
  glEnable( GL_LIGHTING );
  glEnable( GL_DEPTH_TEST );
}
```

Clear the color buffer

Clear the depth buffer

Enable the light condition

Enable the depth condition

Register Callback Function

```
#include <glut.h>
void main(int argc, char **argv){
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGB|GLUT_DEPT);
  glutInitWindowSize(250,250
  glutInitWindowPosition(100
  glutCreateWindow("Hello");

  init();

  glutDisplayFunc(display);
  glutReshapeFunc( resize );
  glutKeyboardFunc( key );
  glutIdleFunc( idle );

  glutMainLoop();
}
```

Routine to call when something happens

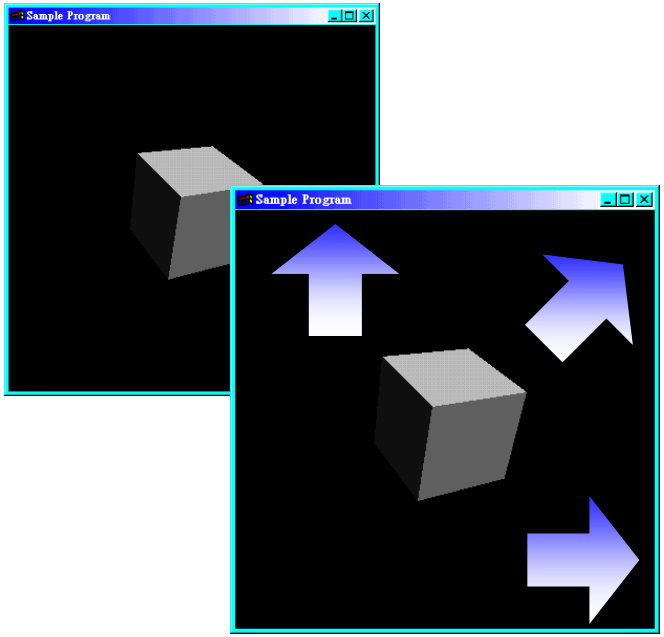
- **window resize or redraw**
- **user input**
- **animation**

"Register" callbacks with GLUT

GLUT CALLBACK Functions

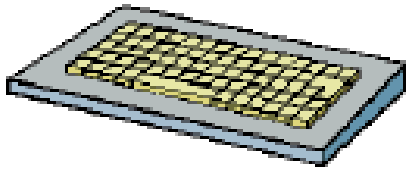
- draw the scene whenever the window is created, moved…etc
  - e.g. glutDisplayFunc(display);
  - Do all of your drawing here
  - "Whenever the window needed redraw, call the function display( );"

- handle reshape event
  - e.g. glutReshape(reshape);
  - "When the user resizes the window,
  - call function reshape( );"



GLUT CALLBACK Functions

- Handle keyboard and mouse input event
  - e.g. glutKeyboardFunc(keyboard); glutMouseFunc()
  - Process user input
  - "When any key is pressed, call the function keyboard( );"



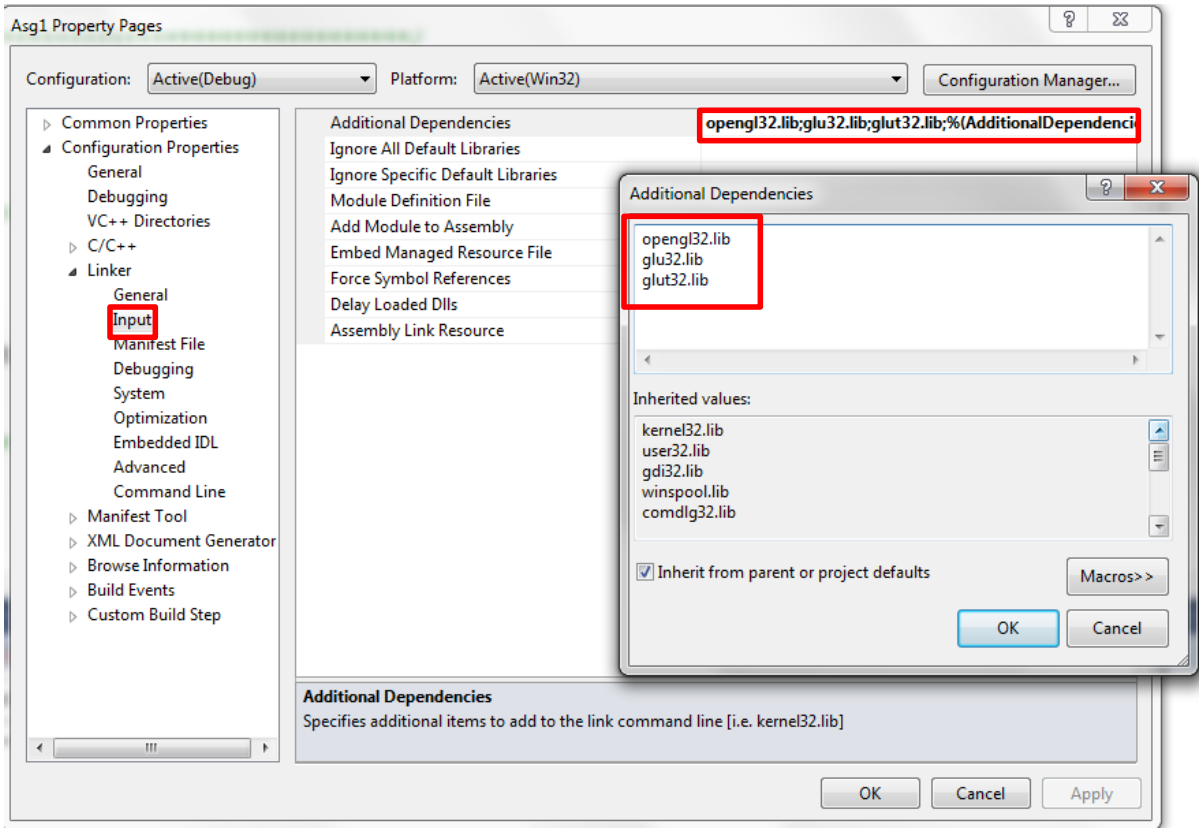
- Handle idle state
  - e.g. glutIdleFunc(idle);
  - Use for animation and continuous update
  - "When the program is idle (no events), call the function idle( );"

Setup OpenGL Environment

- You may use Visual C++ (VS2010 is recommended)
- Place opengl32.dll, glu32.dll, glut32.dll in C:\windows\system32
- Place gl.h, glu.h, glut.h in PATH\include\GL\
  Visual C++ is installed under the directory PATH
  e.g. C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\GL
- Place opengl32.lib, glu32.lib, glut32.lib in PATH\lib
- e.g. C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\lib
- Please note that the GLUT library and header files do not come with OpenGL. You have to download them by yourself.
- You can download everything you need on http://www.opengl.org/

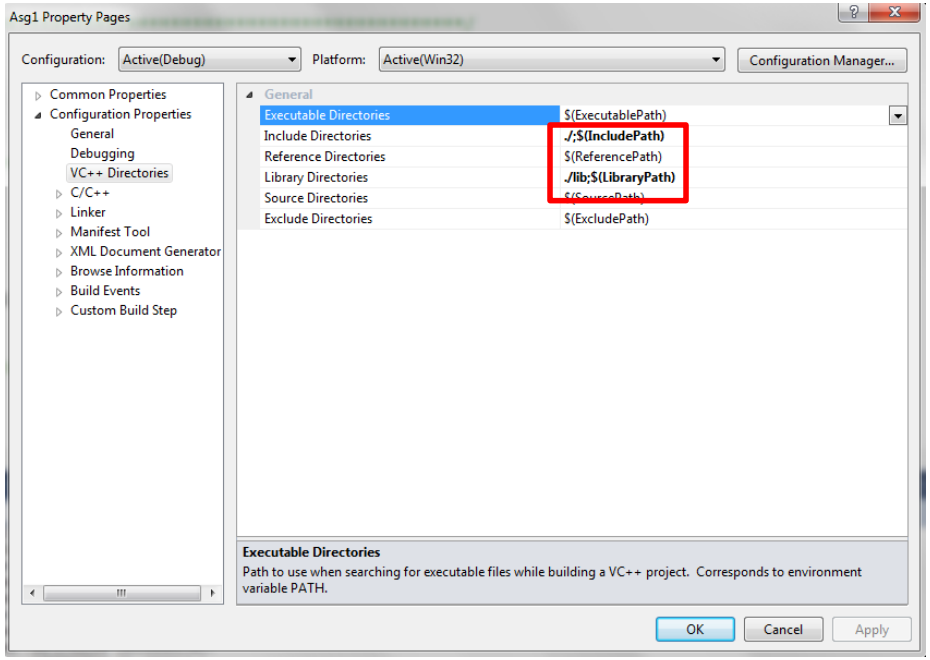
Setup OpenGL Environment

- Project Settings:
  - Include all necessary libraries in project setting, link option--*opengl32.lib*, *glu32.lib*, and *glut32.lib*



Setup OpenGL Environment (For Lab Machines)

- Place glut32.dll in project directory
- Place glut.h in PROJECT_DIR\GL\
- Place glut32.lib in PROJECT_DIR\lib
- Add the corresponding include and library directories



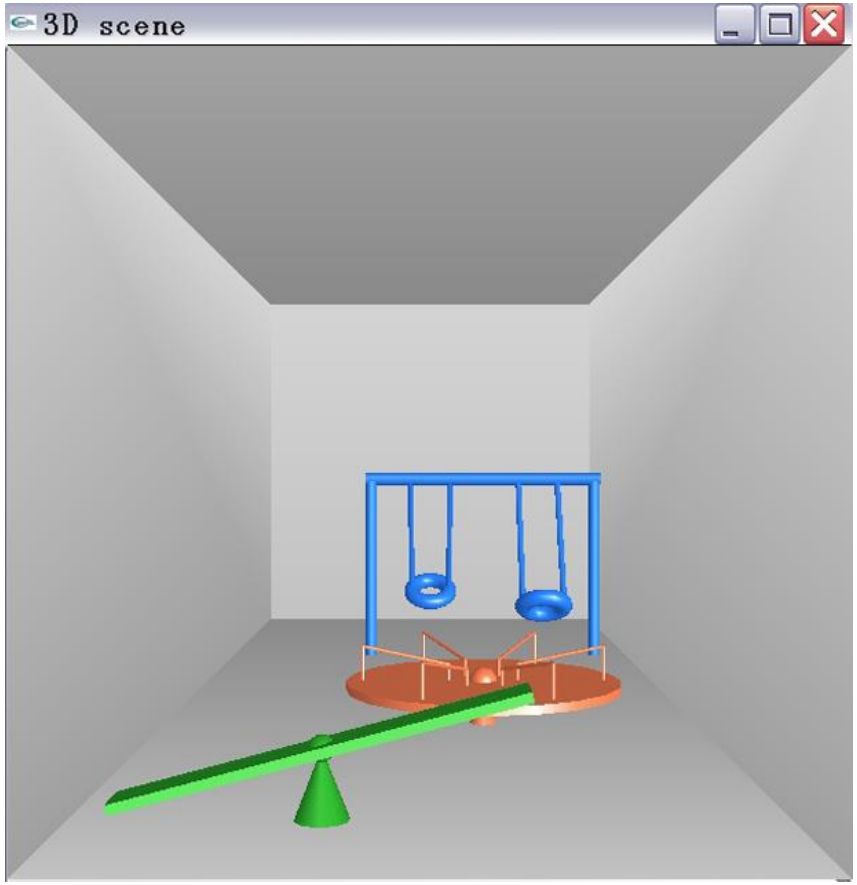
Include Directories: Add “./”
Library Directories: Add “./lib”

Setup OpenGL Environment

- Then .. Standard Programming steps:
  - Start with a new Win32 console project
  - Include glut header file in your program i.e. #include <GL/glut.h> (including glut.h will also include gl.h and glu.h)
  - Your coding
  - Build the program
  - Run/Debug it inside Visual C++

Assignment 1

- Creating a 3D scene
  - Draw objects using different geometric primitives
  - Simple animation
- Demo Program



The End