OPENGL

OpenGL

- What Is OpenGL?
  - a software interface to graphics hardware
  - consists of about 150 distinct commands
    - to specify the objects and operations needed to produce interactive three-dimensional applications
    - GL Reference :
      http://www.opengl.org/sdk/docs/

  - Latest version : 4.6 (2017.7.31)
  - Windows built-in: 1.1
  - Vulkan. (2016.2.16)
    - next generation OpenGL initiative (glNext)
    - Feb. 2016
    - similar to Direct3D and Metal
    - Cross-platform: windows, linux, android, Nintendo Switch
    - MacOS, iOS – by MoltenVK (2018) https://moltengl.com/
      - enables Vulkan to run on top of Metal

OpenGL

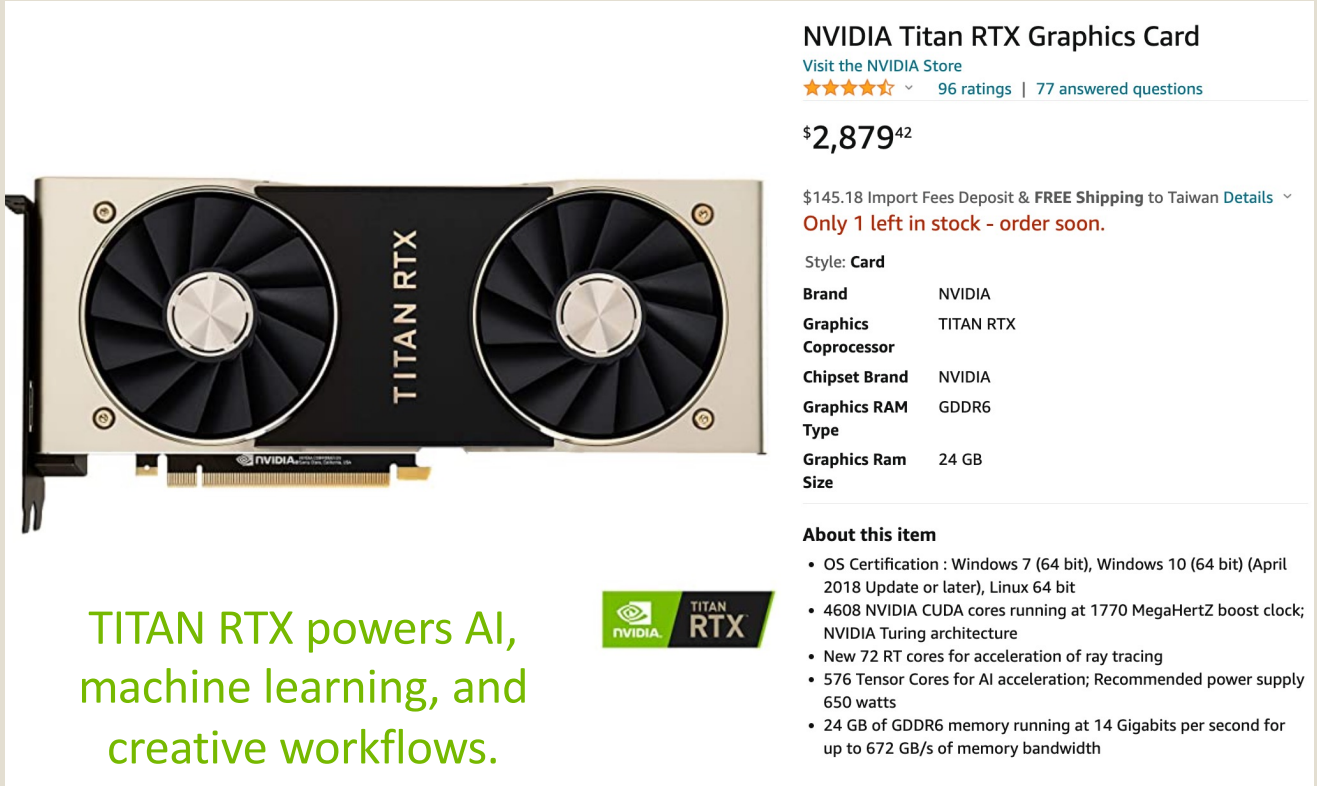
- GL:
  - Graphics library calls
- GLU:
  - OpenGL Utility Library
  - a set of functions to create texture mipmaps from a base image, map coordinates between screen and object space, and draw quadric surfaces and NURBS
  - E.g., gluPerspective()、gluOrtho2D() 、gluLookAt()
  - http://msdn.microsoft.com/en-us/library/windows/desktop/dd374158(v=vs.85).aspx
- GLUT:
  - OpenGL Utility Toolkit
  - a window system independent toolkit
  - Functions performed include window definition, window control, pop-up menus and monitoring of keyboard and mouse input
  - E.g., glutCreateMenu(), glutKeyboardFunc() , glutSolidTeapot(GLdouble size);
  - http://freeglut.sourceforge.net/docs/api.php

OpenGL

- OpenGL is designed as a hardware-independent interface to be implemented on many different hardware platforms.
  - So, no commands for performing windowing tasks or obtaining user input are included in OpenGL
  - Fixed pipeline: OpenGL 1.x
  - Latest version: 4.6

OpenGL Driver Support

- https://developer.nvidia.com/opengl-driver



NVIDIA provides full OpenGL 4.6 support and functionality on NVIDIA GeForce and Quadro graphics card with one of the following GPU architecture:

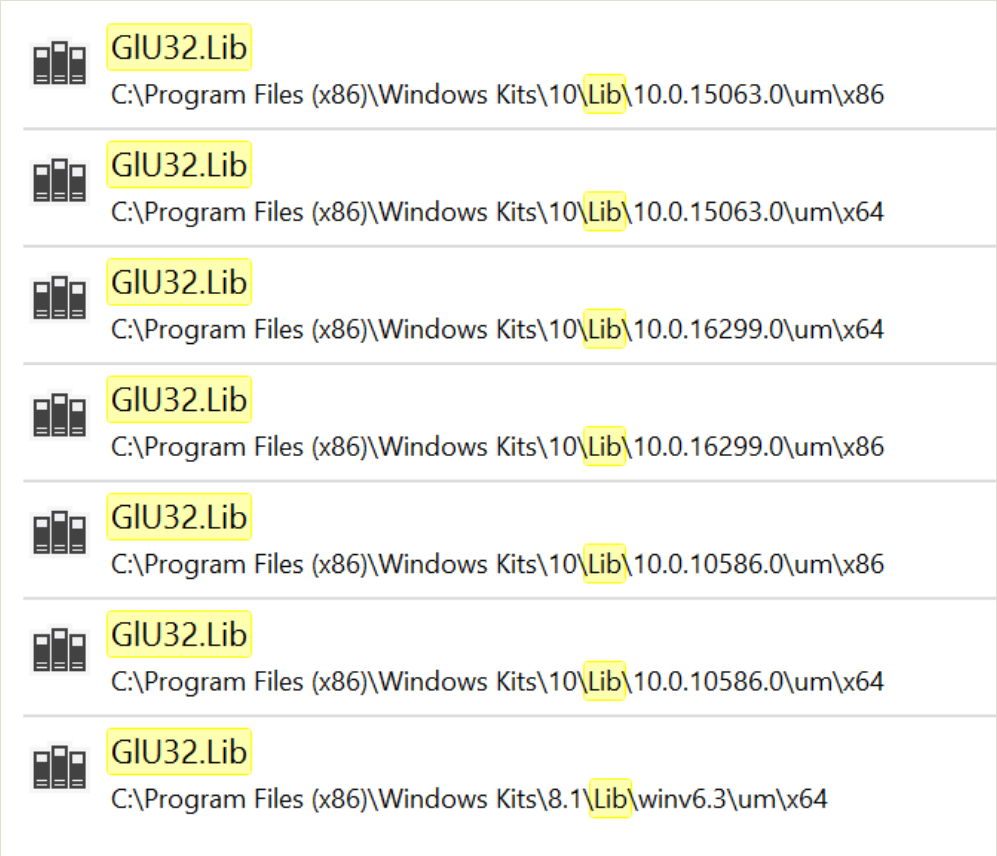
- Turing, Volta, Pascal, Maxwell (first or second generation) or Kepler based GPUs

OpenGL

- gl.h
- glu.h
- glut.h
- opengl32.lib
- glu32.lib
- glut32.lib
- opengl32.dll
- glu32.dll
- glut32.dll

OpenGL

- Some of them are already in Microsoft SDK
  - E.g, glu32.lib



Search OpenGL in win 7

- GL.h
  - C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Include\gl
- OpenGL32.Lib
  - C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Lib\x64
- opengl32.dll
  - C:\Windows\System32
- Glu.h
  - C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Include\gl
- Glu32.lib
  - C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Lib\IA64
- Glu32.dll
  - C:\Windows\System32

OpenGL

What is not included?

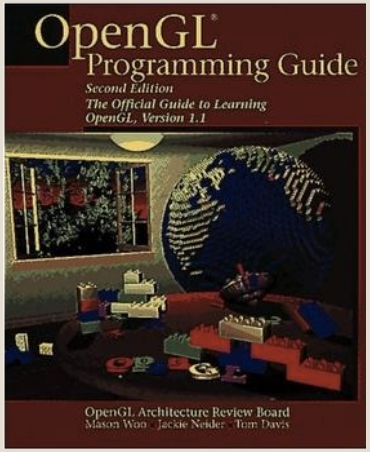
- gl.h
- glu.h
- **glut.h**
- opengl32.lib
- glu32.lib
- **glut32.lib**
- opengl32.dll
- glu32.dll
- **glut32.dll**

GLUT (OpenGL Utility Toolkit)

- Provides functionality common to all window systems
  - Open a window
  - Get input from mouse and keyboard
  - Menus
  - Event-driven

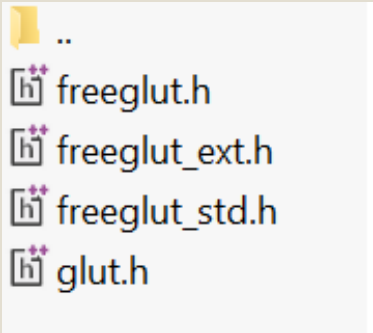
GLUT (OpenGL Utility Toolkit)

- GLUT was originally written by *Mark Kilgard* to support the sample programs in the second edition OpenGL 'RedBook'
  - https://www.opengl.org/resources/libraries/glut/
- Latest version
  - glut-3.7.6-bin.zip (117 KB) since 2001!
- Strict license
- Sweet alternative
  - freeglut
  - http://freeglut.sourceforge.net/docs/api.php



Preliminaries

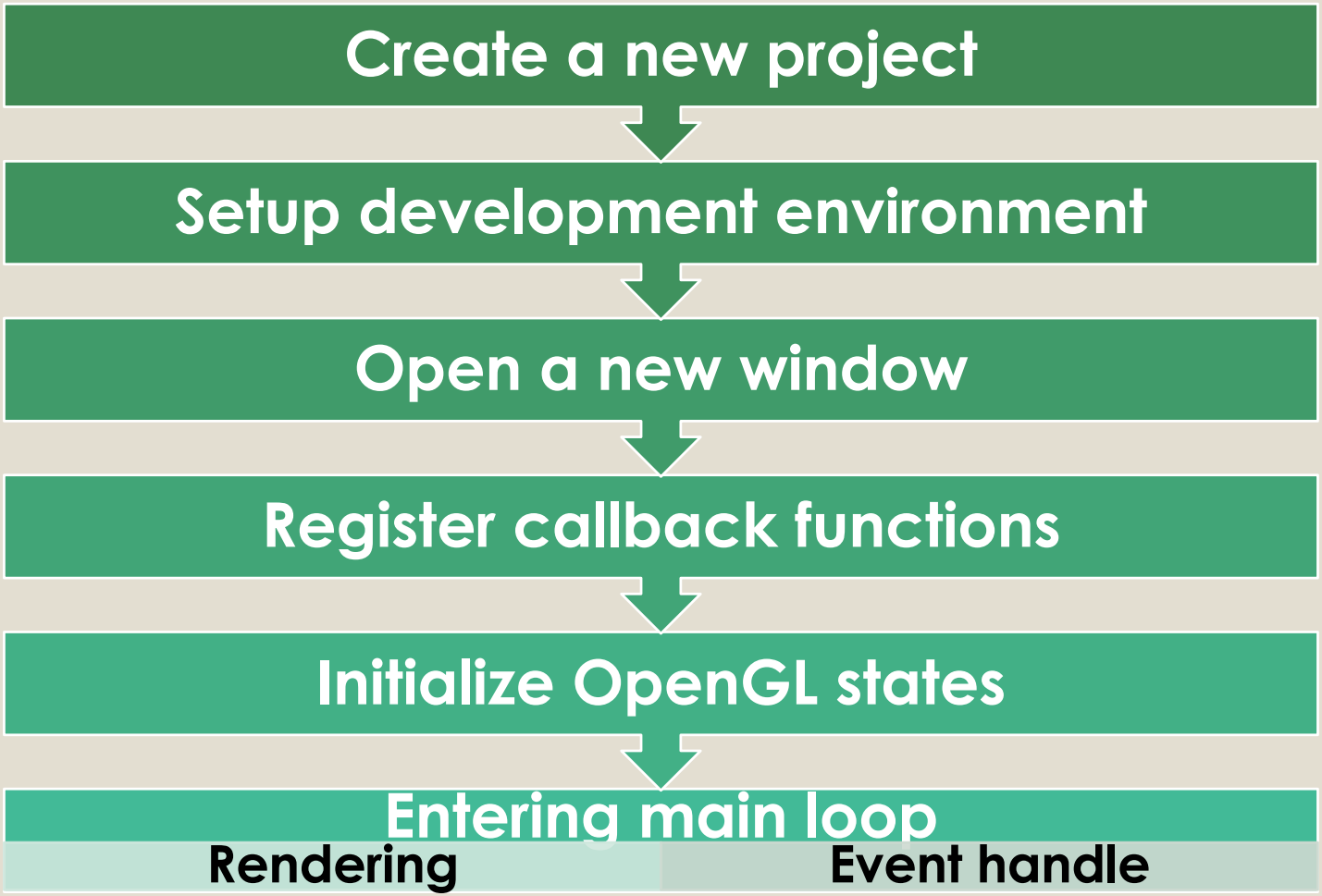
- GLUT library (freeglut 3.7)
  - Download:
    - freeglut.tar.gz
  - Header files
    - glut.h
    - freeglut.h
    - freeglut_ext.h
    - freeglut_std.h



  - Check the readme.cmake file to build this library

  - Library files
    - freeglut.lib, freeglutd.lib ←generated after build
  - Binary files
    - freeglut.dll, freeglutd.dll ←generated after build

Using GLUT : Step-by-Step

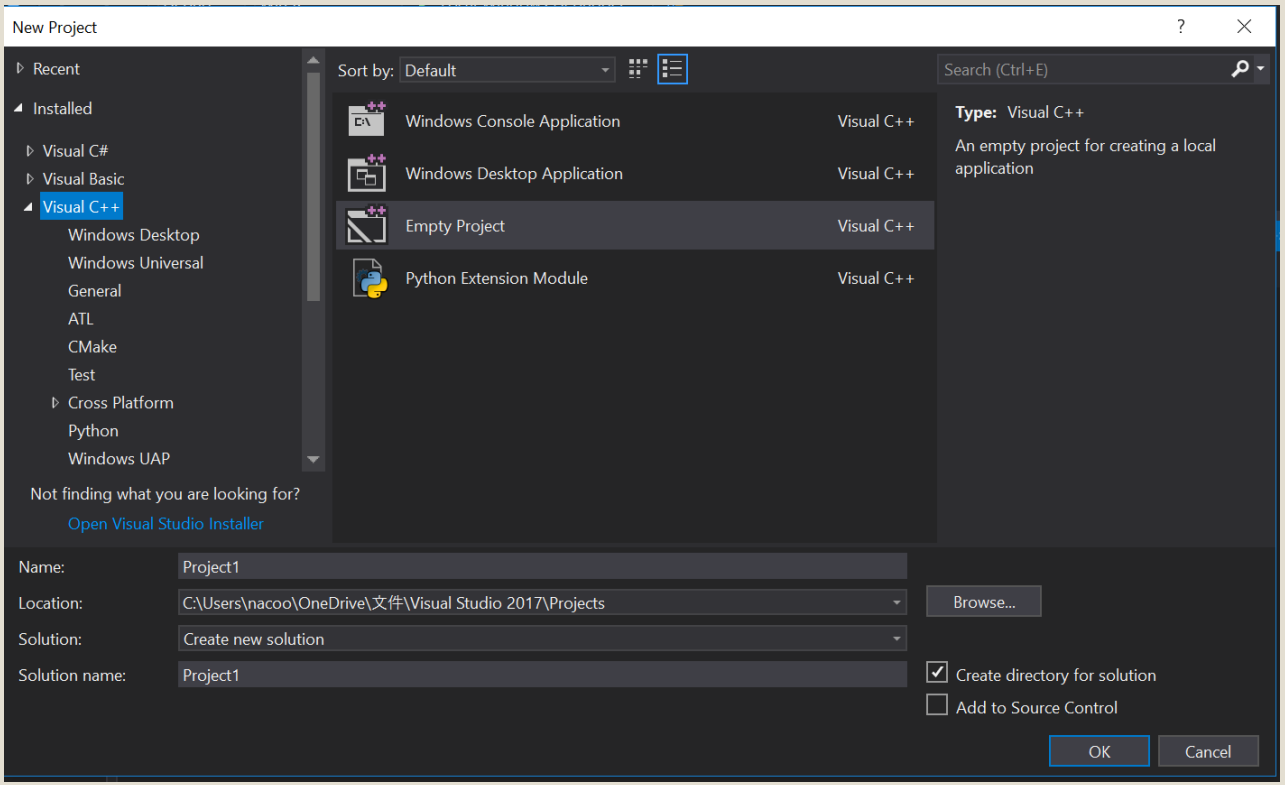


CREATE A NEW PROJECT

Visual Studio 2010

Win32 Console Application

- Visual C++\ Empty Project



SETUP DEVELOPMENT ENVIRONMENT

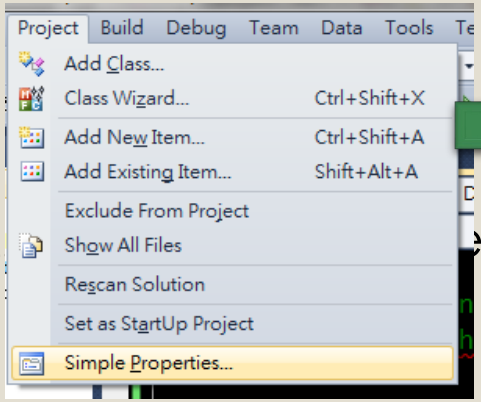
Global setup

Local setup

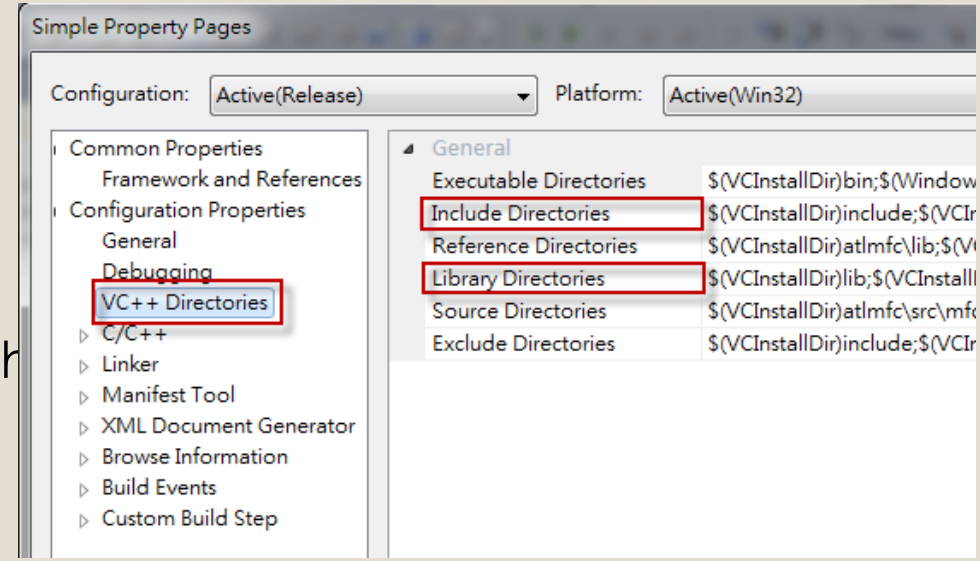
Global Setup (option1)

1. Add absolute freeglut folder path to
   - "Include Directories"
   - "Library Directories"

Put freeglut folder to somewhere you like and open the project "properties"



2 eglut h




```
#include <glut.h>
```

Global Setup (option1)

3. Copy freeglut.dll, freeglutd.dll to
   - C:\Windows\SysWOW64 folder
   - C:\Windows\System32 folder

OPEN A NEW WINDOW

Code snippet

```
int main(int argc, char** argv)
{
  //create a new GLUT Window (Initialization):
  glutInit (&argc, argv);
  glutInitWindowSize (500, 500);
  glutInitDisplayMode ( GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
  glutInitWindowPosition(100,100);
  glutCreateWindow (“Flying Teapot");

  // Register callbacks functions:
  glutDisplayFunc (display);
  glutReshapeFunc (reshape);
  glutKeyboardFunc (Keyboard);
  glutMouseFunc (MouseButton);
  glutMotionFunc (MouseMotion);
  glutIdleFunc (AnimateScene);

  //…(continue next page)
```

Code snippet

```
int main(int argc, char** argv){
  //create a new GLUT Window (Initialization):
  //....
  //Register callbacks functions:
  //....

  //Initialize OpenGL graphics state
  initGraphics();

  // Create our popup menu
  buildPopupMenu();
  glutAttachMenu(GLUT_RIGHT_BUTTON);

  // Turn the flow of control over to GLUT
  glutMainLoop();
  return 0;
}
```

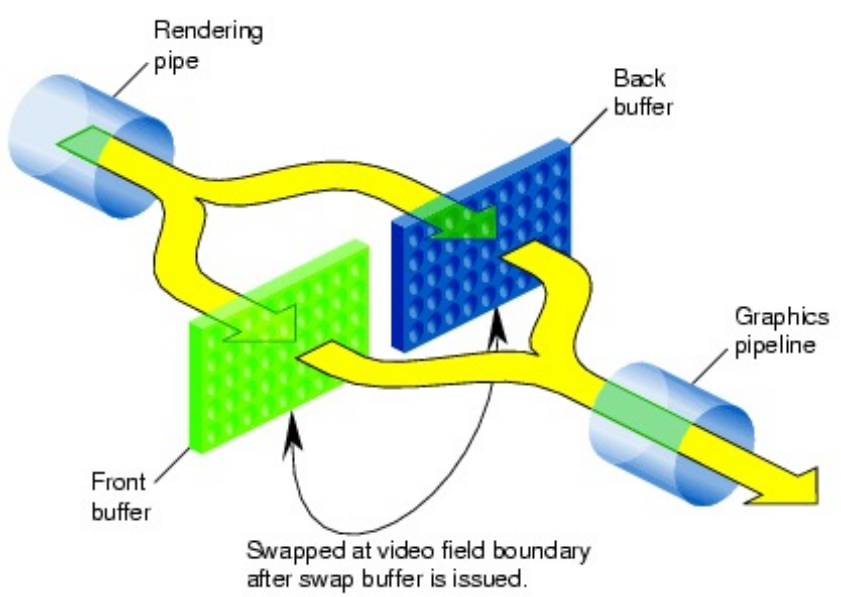
glutInitDisplayMode()

glutInitDisplayMode (GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);

- Specify required data for each pixel in frame buffer
- GLUT_RGBA
  - RGBA color mode
- GLUT_DOUBLE
  - A double-buffered window
- GLUT_DEPTH
  - Allocate depth information
- For other options:
  - http://www.opengl.org/resources/libraries/glut/spec3/node12.html#SECTION00033000000000000000

Double Buffering

- The drawing commands are actually executed on an off-screen buffer and then quickly swapped into view on the window later.
- Avoid flashing effect when animating

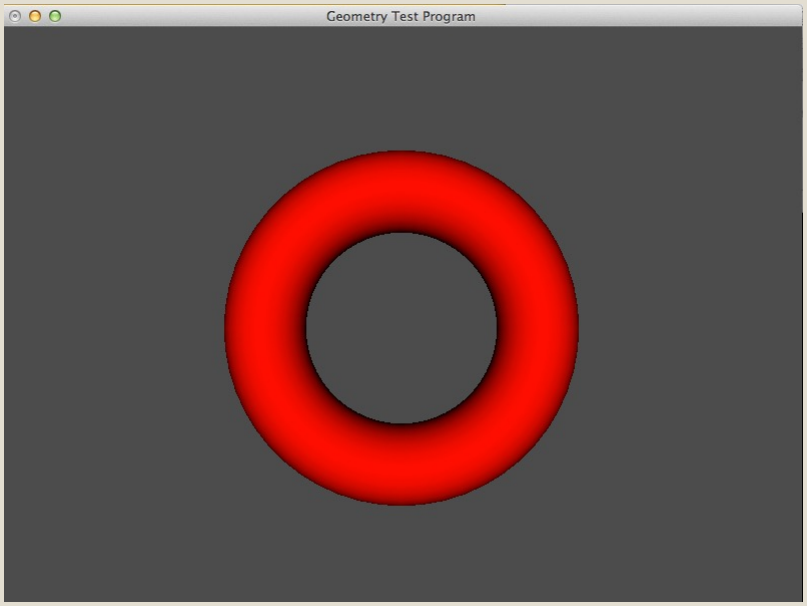


Double Buffering cont.

- Instead of one color buffer, we use two
  - Front Buffer: one that is displayed but not written to
  - Back Buffer: one that is written to but not displayed
- Program then requests a double buffer in
  - glutInitDisplayMode(GL_RGB | GL_DOUBLE)
  - At the end of the display callback buffers are swapped using ***glutSwapBuffers()*** command

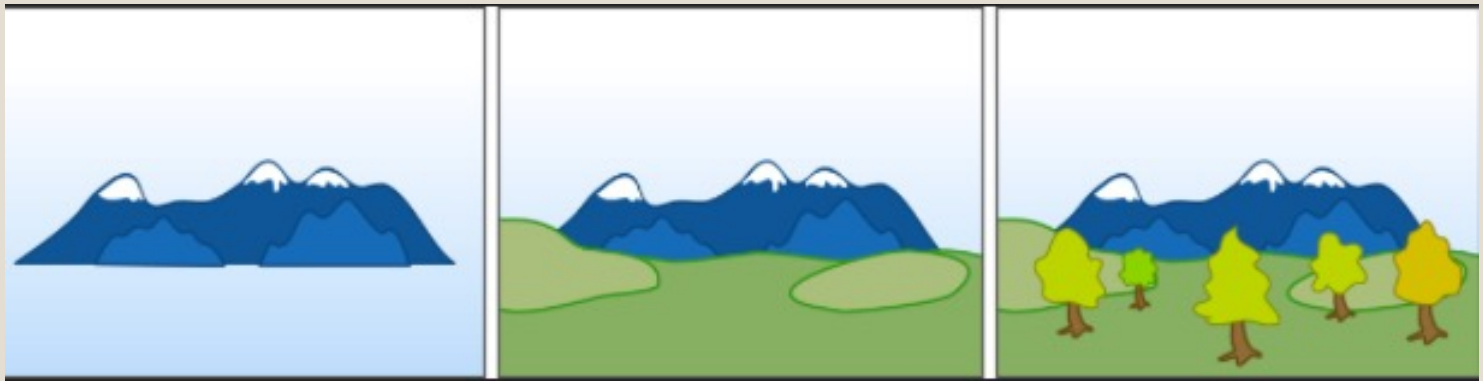
Depth testing

- Chapter03/GeoTest
  - Depth test



Depth Testing
Painter's algorithm

- For hidden surface removal (inefficient!)
  - sort the triangles
  - render the ones farther away first
  - render the nearer triangles on top of them.
- Why it is inefficient?
  - must write to every pixel twice wherever any geometry overlaps, and writing to memory slows things down
  - sorting individual triangles would be prohibitively expensive



Depth testing

- For hidden surface removal (efficient!)
  - when a pixel is drawn, a z value is assigned
    - to denotes its distance from the viewer's perspective
  - when another pixel needs to be drawn to that screen location, their z values are compared
  - If the z value is higher
    - it is closer to the viewer → redrawn
- To request a depth buffer
  - glutInitDisplayMode(GL_RGB | GL_DOUBLE | GLUT_DEPTH)
  - A buffer with storage for a depth value for every pixel on the screen
- To enable depth testing
  - glEnable(GL_DEPTH_TEST);
    - If you do not have a depth buffer, then enabling depth testing will just be ignored

ENTERING MAIN LOOP

GLUT Event Loop

- Recall that the last line in a program using GLUT must be:

glutMainLoop();

  - which puts the program in an infinite event loop
  - In each pass through the event loop, GLUT looks at the events in the queue
    - for each event in the queue, GLUT executes the appropriate callback function if one is defined
      - E.g., keyboard function, mouse function,..etc
      - if no callback is defined for the event, the event is ignored

Code snippet

```
int main(int argc, char** argv)
{
  //create a new GLUT Window (Initialization):
  glutInit (&argc, argv);
  glutInitWindowSize (g_Width, g_Height);
  glutInitDisplayMode ( GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
  glutInitWindowPosition(100,100);
  glutCreateWindow ("Flying Teapot");

  // Initialize OpenGL graphics state
  initGraphics();

  // Register callbacks function:
  glutDisplayFunc (display);
  glutReshapeFunc (reshape);
  glutKeyboardFunc (Keyboard);
  glutMouseFunc (MouseButton);
  glutMotionFunc (MouseMotion);
  glutIdleFunc (AnimateScene);

  // Create our popup menu
  buildPopupMenu ();
  glutAttachMenu (GLUT_RIGHT_BUTTON);

  // Turn the flow of control over to GLUT
  glutMainLoop ();
  return 0;
}
```


```
// Turn the flow of control over to GLUT
glutMainLoop ();
return 0;
```

REGISTER CALLBACK FUNCTIONS

Callbacks

- Programming interface for event-driven input
- Define a ***callback function*** for each type of event the graphics system recognizes
- This user-supplied function is executed when the event occurs

GLUT callbacks

- GLUT recognizes a subset of the events recognized by any particular window system (Windows, X, Macintosh)
  - glutDisplayFunc
  - glutMouseFunc
  - glutReshapeFunc
  - glutKeyboardFunc
  - glutTimerFunc
  - glutIdleFunc

Code snippet

```
int main(int argc, char** argv)
{
  //create a new GLUT Window (Initialization):
  glutInit (&argc, argv);
  glutInitWindowSize (g_Width, g_Height);
  glutInitDisplayMode ( GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
  glutInitWindowPosition(100,100);
  glutCreateWindow ("Flying Teapot");

  // Initialize OpenGL graphics state
  initGraphics();

  // Register callbacks function:
  glutDisplayFunc (display);
  glutReshapeFunc (reshape);
  glutKeyboardFunc (Keyboard);
  glutMouseFunc (MouseButton);
  glutMotionFunc (MouseMotion);
  glutIdleFunc (AnimateScene);

  // Create our popup menu
  buildPopupMenu ();
  glutAttachMenu (GLUT_RIGHT_BUTTON);

  // Turn the flow of control over to GLUT
  glutMainLoop ();
  return 0;
}
```

```
// Register callbacks function:
glutDisplayFunc (display);
glutReshapeFunc (reshape);
glutKeyboardFunc (Keyboard);
glutMouseFunc (MouseButton);
glutMotionFunc (MouseMotion);
glutIdleFunc (AnimateScene);
```

```
/// Register callback functions
//////////////////////////////////////////////////////////
glutReshapeFunc ( My_Reshape );
glutDisplayFunc ( My_Display );
glutKeyboardFunc( My_Keyboard );
glutMouseFunc   ( My_Mouse );
glutSpecialFunc ( My_SpecialKeys );
glutTimerFunc   (timer_speed, My_Timer, timer_flag);
//////////////////////////////////////////////////////////

/// Initialize OpenGL
```

The Reshape callback

```
// Register callbacks function:
glutDisplayFunc (display);
glutReshapeFunc (reshape);
glutKeyboardFunc (Keyboard);
glutMouseFunc (MouseButton);
glutMotionFunc (MouseMotion);
glutIdleFunc (AnimateScene);
```

```
void reshape(GLint width, GLint height)
{
   g_Width = width;
   g_Height = height;
   glViewport(0, 0, g_Width, g_Height);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(65.0, (float)g_Width / g_Height, g_nearPlane, g_farPlane);
   glMatrixMode(GL_MODELVIEW);
}
```

- void reshape(int w, int h)
  - Returns width and height of new window (in pixels)
  - A redisplay is posted automatically at end of execution of the callback
  - GLUT has a default reshape callback but you probably want to define your own
- The reshape callback is good place to put viewing functions because it is invoked when the window is first opened

My_Reshape() - Code snippet

- Whenever the window is resized glut calls this function

```
//////////////////////////////////////////////////////////////
// Called by GLUT library when the window has changed size
void My_Reshape(int w, int h)
{
    GLfloat aspectRatio;

    // Prevent a divide by zero
    if(h == 0)
        h = 1;

    // Set Viewport to window dimensions
    glViewport(0, 0, w, h);

    // Reset coordinate system
    glMatrixMode(GL_PROJECTION);
```

Width and height of new window

The Display Callback

- The display callback is executed whenever GLUT determines that the window should be refreshed, for example
  - When the window is first opened
  - When the window is reshaped
  - When a window is exposed
  - When the user program decides it wants to change the display
- Every GLUT program must have a display callback

My_Display() - Code snippet

- Call ***glutSwapBuffers*** to swap back/front buffers

```
////////////////////////////////////////////////////////////////////////
// Called to draw scene
void My_Display(void)
{
    /// Clear the window with current clearing color
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    /// Draw a triangle
    glBegin(GL_TRIANGLES);
    glColor3ub(timer_cnt, 0, 255-timer_cnt);
    glVertex3fv(tri_v1);
    glColor3ub(255, timer_cnt, 255-timer_cnt);
    glVertex3fv(tri_v2);
    glColor3ub(255-timer_cnt, 0, timer_cnt);
    glVertex3fv(tri_v3);
    glEnd();

    /// Flush drawing commands
    glutSwapBuffers();
}
```

Your First Triangle

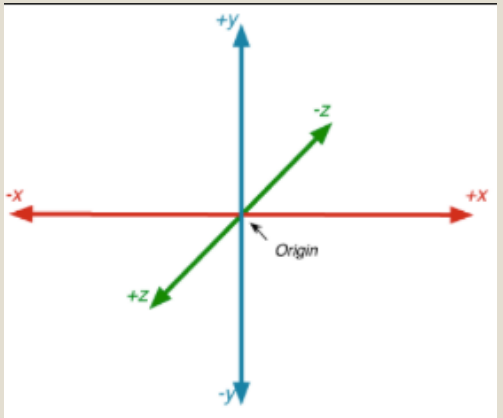
```
#include <glut.h>



// Main entry point for GLUT based programs
int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode( GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH | GLUT_STENCIL);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Triangle");
    glutReshapeFunc(ChangeSize);
    glutDisplayFunc(RenderScene);
    glClearColor(0.0f, 0.0f, 1.0f, 1.0f );
    glutMainLoop();
    return 0;
}
```

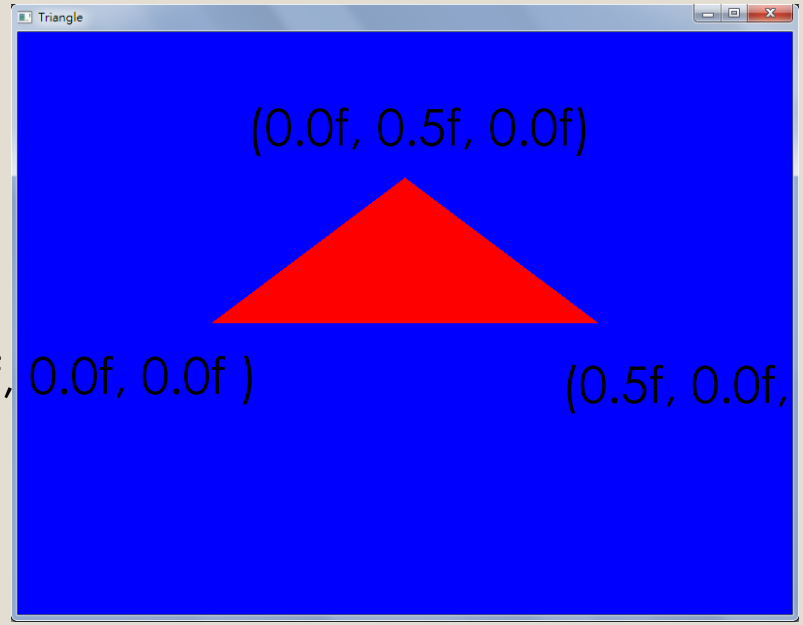
Your First Triangle

```
GLfloat vVerts[3][3] = {{-0.5f, 0.0f, 0.0f},
                        {0.5f, 0.0f, 0.0f},
                        {0.0f, 0.5f, 0.0f}};
GLfloat vColor[] = { 1.0f, 0.0f, 0.0f};
////////////////////////////////////////////////////
////////////////////

// Window has changed size, or has just
been created. In either case, we need to
use the window dimensions to set the
viewport and the projection matrix.

void ChangeSize(int w, int h)
{
    glViewport(0, 0, w, h);
}
```

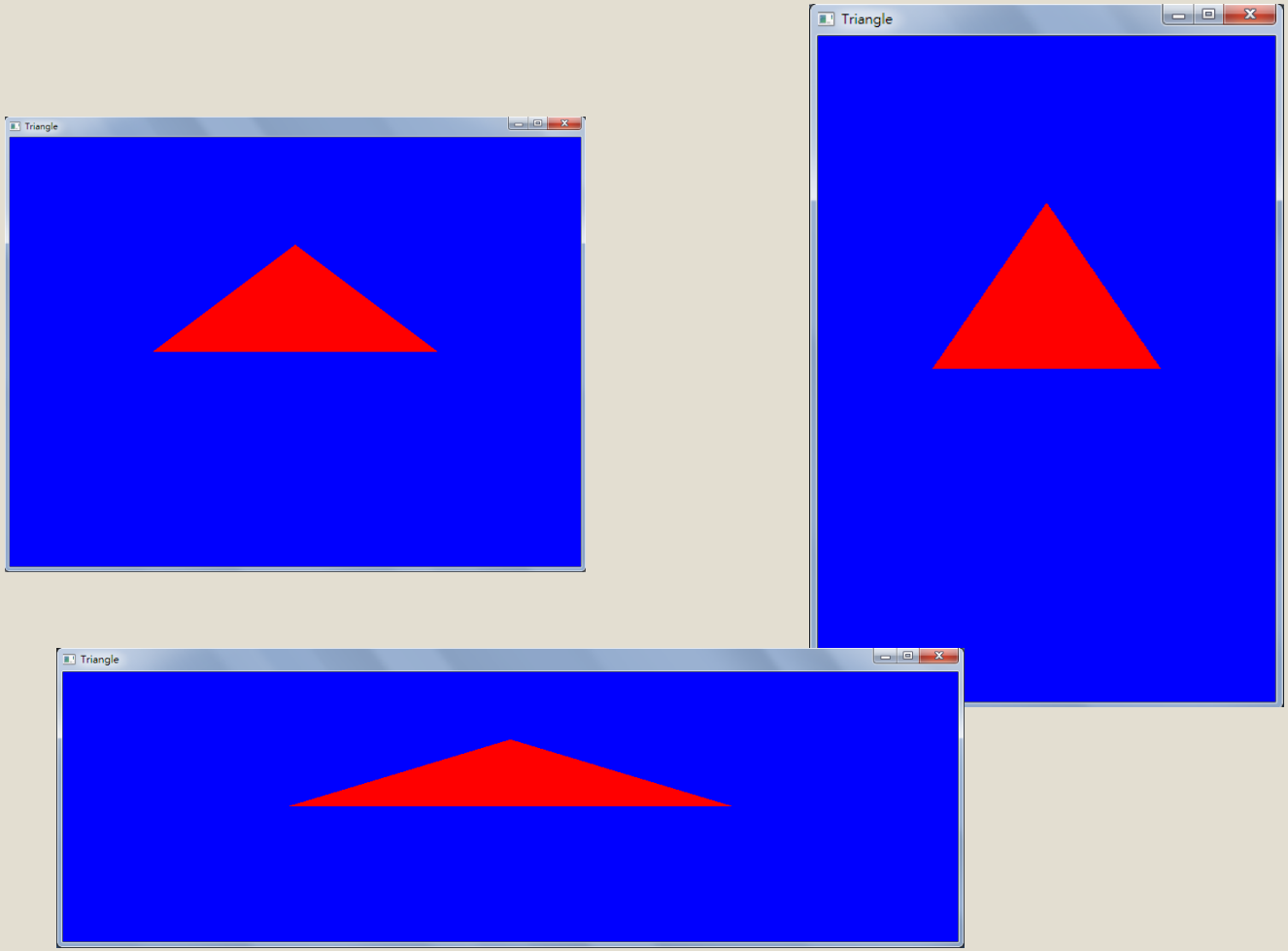
```
////////////////////////////////////////////////////
////////////////////
// Called to draw scene
void RenderScene(void)
{
  // Clear the window with current clearing
color
  glClear( GL_COLOR_BUFFER_BIT |
           GL_DEPTH_BUFFER_BIT |
           GL_STENCIL_BUFFER_BIT);
  glBegin(GL_TRIANGLES);
  glColor3fv(vColor);
  glVertex3fv(vVerts[0]);
  glVertex3fv(vVerts[1]);
  glVertex3fv(vVerts[2]);
  glEnd();
  // buffer swap to display the back buffer
  glutSwapBuffers();
}
```

Output





Change window ratio



The Mouse Callback

```
glutMouseFunc(mymouse)
void mymouse(GLint button, GLint state, GLint x, GLint y)
```

- Returns
  - which button
    - `GLUT_LEFT_BUTTON`
    - `GLUT_MIDDLE_BUTTON`
    - `GLUT_RIGHT_BUTTON`
  - state of that button
    - `GLUT_UP`
    - `GLUT_DOWN`
  - Position in window

Positioning

- The position in the screen window is usually measured in pixels with the origin at the top-left corner
  - Consequence of refresh done from top to bottom
- OpenGL uses a world coordinate system with origin at the bottom left
  - Must invert y coordinate returned by callback by height of window
  - y = h – y;

My_Mouse() - Code snippet

- Handle the mouse events

```
////////////////////////////////////////////////////////////
// Called by GLUT library when the mouse event is triggered
void My_Mouse(int button, int state, int x, int y)
{
    switch (button)
    {
    case GLUT_LEFT_BUTTON:
        if (state == GLUT_DOWN)
            cout << "Mouse left button down" << endl;
        break;
    case GLUT_MIDDLE_BUTTON:
        if (state == GLUT_DOWN)
            cout << "Mouse middle button down" << endl;
        break;
    case GLUT_RIGHT_BUTTON:
        if (state == GLUT_DOWN)
```

Which button ?

Up or down ?

Screen coordinates

The Keyboard Callback

- glutKeyboardFunc(mykey)
- void mykey(unsigned char key, int x, int y)
  - Returns ASCII code of key depressed and mouse location
- Can also check of one of the modifiers is pressed by glutGetModifiers()
  - GLUT_ACTIVE_SHIFT
  - GLUT_ACTIVE_CTRL
  - GLUT_ACTIVE_ALT
  - Allows emulation of three-button mouse with one- or two-button mice

My_Keyboard() - Code snippet

- Handle the keyboard events

```
////////////////////////////////////////////////////////////
// Called by GLUT library when the keyboard event is triggered
void My_Keyboard( unsigned char key, int x, int y )
{
    switch( key ) {
    case 'q' : case 'Q' :
        exit(0); /// quit the program
        break;
    case 'f' : case 'F' :
        /// enter/leave full-screen mode
        glutFullScreenToggle();
        break;
    case 'p' : case 'P':
        /// stop/resume timer
        if(timer_flag == 0)
```

My_SpecialKeys() - Code snippet

- Handle the special keyboard events

```
////////////////////////////////////////////////////////////
// Called by GLUT library when the special keyboard event is triggered
void My_SpecialKeys( int key, int x, int y )
{
    switch( key ) {
    case GLUT_KEY_F1 :
        cout << "This is F1 key" << endl;
        break;
    case GLUT_KEY_PAGE_UP :
        cout << "This is PageUp key" << endl;
        break;
    case GLUT_KEY_LEFT :
        cout << "This is Left key" << endl;
        break;
    }
}
```

```
glutKeyboardFunc(My_Keyboard);
glutSpecialFunc(My_SpecialKeys);
```

```
void glutTimerFunc( unsigned int msecs ,
                    void (*func)(int value),
                    value);
```

Animation in GLUT

glutTimerFunc

- Registers a timer callback to be triggered in a specified number of milliseconds
- Only called once!
- Multiple timer callbacks at same or differing times

- Call ***glutPostRedisplay*** to refresh the screen

```
void TimerFunc(int value)
{
    glutPostRedisplay();
    glutTimerFunc(100, TimerFunc, 1);
}
```

http://openglut.sourceforge.net/group__idletimer.html

Animation in GLUT alternative

```
void glutIdleFunc ( void (*func)());
```

glutIdleFunc

- Sets the global idle callback.
- Only one idle function
- Can be easily stopped by

```
glutIdleFunc ( NULL );
```

- Call ***glutPostRedisplay*** to refresh the screen

My_Timer() - Code snippet

```
////////////////////////////////////////////////////////////
// Called by GLUT library when the special keyboard event is triggered
void My_Timer( int value )
{
    if(value == 0) return;

    timer_cnt++;
    timer_cnt = timer_cnt % 256;

    glutPostRedisplay();
    glutTimerFunc(timer_speed, My_Timer, timer_flag);
}
```

Stop / Resume timer

```
case 'p' : case 'P':
    /// stop/resume timer
    if(timer_flag == 0)
    {
        timer_flag = 1;
        glutTimerFunc(timer_speed, My_Timer, timer_flag);
    }
    else
        timer_flag = 0;
    break;
```

INITIALIZE OPENGL STATES

Code snippet

```
////////////////////////////////////////////////////////////
// Main program entry point
int main(int argc, char* argv[])
{
    /// Create a new window
    ////////////////////////////////////////////////////////
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowPosition(100,100);
```

```
    /// Initialize OpenGL
    InitGL();
```

```
    glutSpecialFunc ( My_SpecialKeys );
    glutTimerFunc   (timer_speed, My_Timer, timer_flag);
    ////////////////////////////////////////////////////////

    /// Initialize OpenGL
    InitGL();

    /// Entering main loop
    glutMainLoop();

    return 0;
}
```

InitGL() - Code snippet

- Setup OpenGL initial states

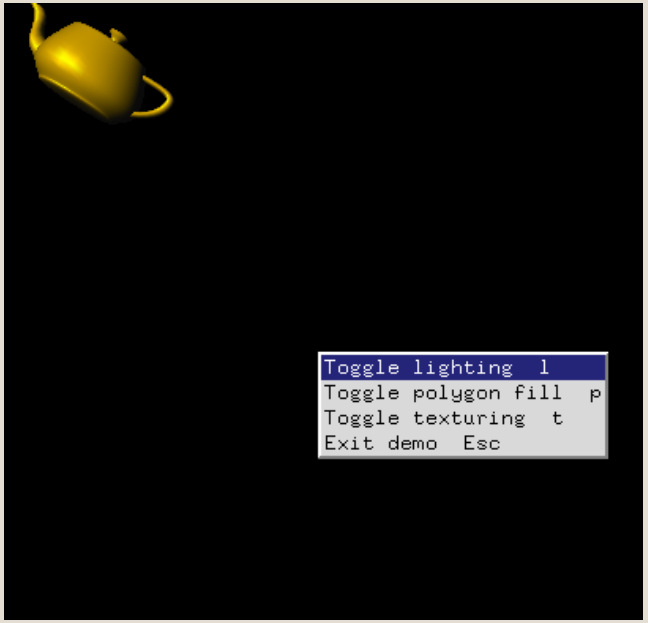
```
////////////////////////////////////////////////////////////
// Setup the rendering state
void InitGL(void)
{
    /// Setup background color
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

    /// Enable depth testing
    glEnable(GL_DEPTH_TEST);

    /// Disable lighting
    glDisable(GL_LIGHTING);
}
```

CREATE GLUT CONTEXT MENU

Menus

- GLUT supports pop-up menus
  - A menu can have submenus
- Three steps
  - Define entries for the menu
  - Define action for each menu item
    - Action carried out if entry selected
  - Attach menu to a mouse button

Popup menu




```
#define MENU_LIGHTING 0
#define MENU_POLYMODE 1
#define MENU_TEXTURING 2
#define MENU_EXIT 3

int buildPopupMenu (void)
{
  int menu;
  menu = glutCreateMenu (selectFromMenu);
  glutAddMenuEntry ("Toggle lighting  l", MENU_LIGHTING);
  glutAddMenuEntry ("Toggle polygon fill  p", MENU_POLYMODE);
  glutAddMenuEntry ("Toggle texturing  t", MENU_TEXTURING);
  glutAddMenuEntry ("Exit demo  Esc", MENU_EXIT);
  return menu;
}
```

```
void selectFromMenu (int option)
{
	switch (option) {
		case MENU_LIGHTING:
			//……
			break;
		case MENU_POLYMODE:
			//….
			break;
		case MENU_TEXTURING:
			//….
			break;
		case MENU_MENU_EXIT:
			//….
			break;

		default:
			break;

	}
}
```

```
int main(int argc, char** argv)
{
  //create a new GLUT Window (Initialization):
  glutInit (&argc, argv);
  glutInitWindowSize (g_Width, g_Height);
  glutInitDisplayMode ( GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
  glutInitWindowPosition(100,100);
  glutCreateWindow ("Flying Teapot");

  // Initialize OpenGL graphics state
  initGraphics();

  // Register callbacks function:
  glutDisplayFunc (display);
  glutReshapeFunc (reshape);
  glutKeyboardFunc (Keyboard);
  glutMouseFunc (MouseButton);
  glutMotionFunc (MouseMotion);
  glutIdleFunc (AnimateScene);

  // Create our popup menu
  buildPopupMenu ();
  glutAttachMenu (GLUT_RIGHT_BUTTON);

  // Turn the flow of control over to GLUT
  glutMainLoop ();
  return 0;
}
```

Next Tuesday

- Rendering Pipeline