How to Set up OpenGL and GLUT in Visual Studio (C++)

3 Dec 2016

Based on

<https://m.youtube.com/watch?v=8p76pJsUP44>

download

http://www.mediafire.com/file/cmlnr0pj0pyha5d/Glew+and+Glut.zip

unzip into c:\ Glew and Glut

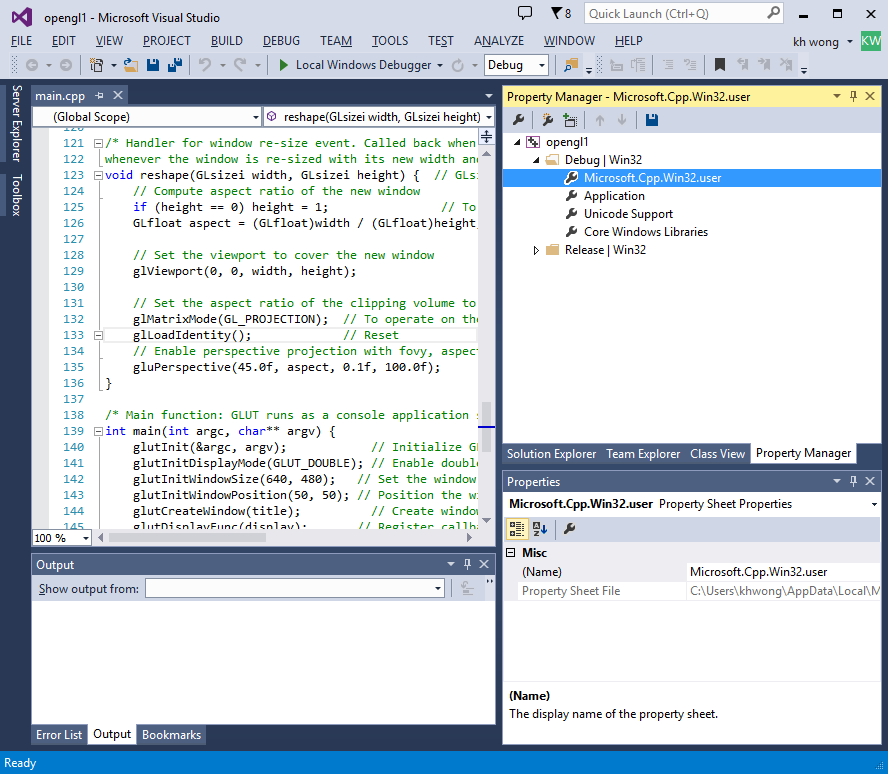
SETUP, only do it once

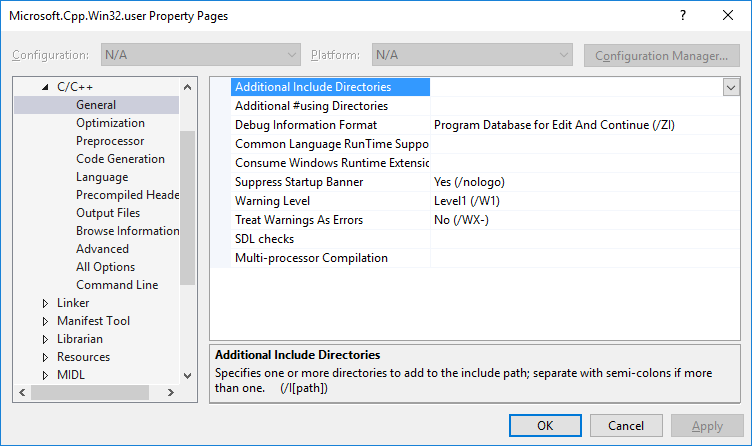
Create a win32 console project

IN vs2013>>VIEW>>Property windows>>(lower right corner , select property manager>> >> open Debug|Win32>> double click Microsoft.Cpp.Win32.user>> : in Microsoft.Cpp.Win32.user .property pages, 3 changes

If you cannot see “property manager”. Try

(Vs2013 Root menu)TOOLS>> options>>property manager>> select something etc.





http://www.mediafire.com/file/cmlnr0pj0pyha5d/Glew+and+Glut.zip

1. C++ >>general >> Additional Include directories

C:\Glew and Glut\freeglut\include;C:\Glew and Glut\glew-1.11.0\include

1. Linker >> General

C:\Glew and Glut\freeglut\lib;C:\Glew and Glut\glew-1.11.0\lib

1. Linker>>input>>Additional dependence>>

freeglut.lib;glew32.lib;

**\*\*\* DO not touch “ VC++ directories”**

**Make sure**

**VC++ directories: included directories: $(VC\_IncludePath);$(WindowsSDK\_IncludePath);**

**And**

**VC++ directories: included libraries: $(VC\_LibraryPath\_x86);$(WindowsSDK\_LibraryPath\_x86);**

**Otherisw youa re in trouble**

--Now create a project (or using existing project)

Copy

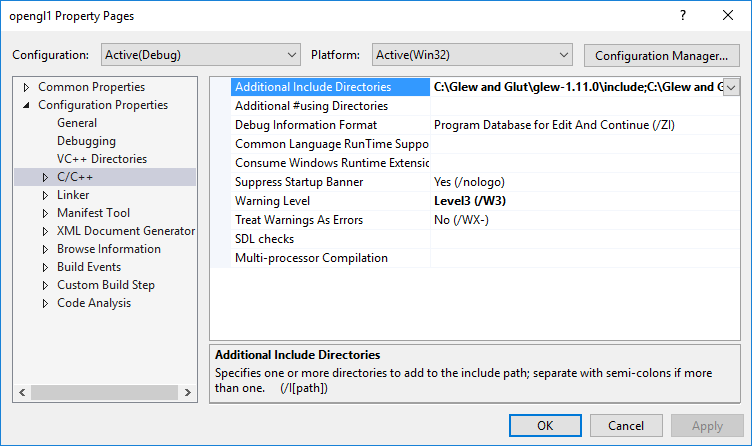
The 2 .dll’s

From C:\Glew and Glut\glew-1.11.0\bin\Release\Win32\glew.dll and

C:\Glew and Glut\freeglut\bin\freeglut.dll

to

C:\Users\khwong\Documents\Visual Studio 2013\Projects\opengl1\opengl1



C:\Glew and Glut\freeglut\include

C:\Glew and Glut\glew-1.11.0\include

* Now create a source file main.c , copy the following content to main.c

Then build and run, it should be ok, finish.

//////////////////// end of opengl demo -- main.c

//Jeff Chastine

#include "stdafx.h"

#include <Windows.h>

#include <GL\glew.h>

#include <GL\freeglut.h>

#include <iostream>

/\*

\* OGL01Shape3D.cpp: 3D Shapes

\*/

#include <windows.h> // for MS Windows

#include <GL/glut.h> // GLUT, include glu.h and gl.h

/\* Global variables \*/

char title[] = "3D Shapes";

/\* Initialize OpenGL Graphics \*/

void initGL() {

glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque

glClearDepth(1.0f); // Set background depth to farthest

glEnable(GL\_DEPTH\_TEST); // Enable depth testing for z-culling

glDepthFunc(GL\_LEQUAL); // Set the type of depth-test

glShadeModel(GL\_SMOOTH); // Enable smooth shading

glHint(GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_NICEST); // Nice perspective corrections

}

/\* Handler for window-repaint event. Called back when the window first appears and

whenever the window needs to be re-painted. \*/

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); // Clear color and depth buffers

glMatrixMode(GL\_MODELVIEW); // To operate on model-view matrix

// Render a color-cube consisting of 6 quads with different colors

glLoadIdentity(); // Reset the model-view matrix

glTranslatef(1.5f, 0.0f, -7.0f); // Move right and into the screen

glBegin(GL\_QUADS); // Begin drawing the color cube with 6 quads

// Top face (y = 1.0f)

// Define vertices in counter-clockwise (CCW) order with normal pointing out

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(1.0f, 1.0f, -1.0f);

glVertex3f(-1.0f, 1.0f, -1.0f);

glVertex3f(-1.0f, 1.0f, 1.0f);

glVertex3f(1.0f, 1.0f, 1.0f);

// Bottom face (y = -1.0f)

glColor3f(1.0f, 0.5f, 0.0f); // Orange

glVertex3f(1.0f, -1.0f, 1.0f);

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f(-1.0f, -1.0f, -1.0f);

glVertex3f(1.0f, -1.0f, -1.0f);

// Front face (z = 1.0f)

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f(1.0f, 1.0f, 1.0f);

glVertex3f(-1.0f, 1.0f, 1.0f);

glVertex3f(-1.0f, -1.0f, 1.0f);

glVertex3f(1.0f, -1.0f, 1.0f);

// Back face (z = -1.0f)

glColor3f(1.0f, 1.0f, 0.0f); // Yellow

glVertex3f(1.0f, -1.0f, -1.0f);

glVertex3f(-1.0f, -1.0f, -1.0f);

glVertex3f(-1.0f, 1.0f, -1.0f);

glVertex3f(1.0f, 1.0f, -1.0f);

// Left face (x = -1.0f)

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(-1.0f, 1.0f, 1.0f);

glVertex3f(-1.0f, 1.0f, -1.0f);

glVertex3f(-1.0f, -1.0f, -1.0f);

glVertex3f(-1.0f, -1.0f, 1.0f);

// Right face (x = 1.0f)

glColor3f(1.0f, 0.0f, 1.0f); // Magenta

glVertex3f(1.0f, 1.0f, -1.0f);

glVertex3f(1.0f, 1.0f, 1.0f);

glVertex3f(1.0f, -1.0f, 1.0f);

glVertex3f(1.0f, -1.0f, -1.0f);

glEnd(); // End of drawing color-cube

// Render a pyramid consists of 4 triangles

glLoadIdentity(); // Reset the model-view matrix

glTranslatef(-1.5f, 0.0f, -6.0f); // Move left and into the screen

glBegin(GL\_TRIANGLES); // Begin drawing the pyramid with 4 triangles

// Front

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f(0.0f, 1.0f, 0.0f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(-1.0f, -1.0f, 1.0f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(1.0f, -1.0f, 1.0f);

// Right

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f(0.0f, 1.0f, 0.0f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(1.0f, -1.0f, 1.0f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(1.0f, -1.0f, -1.0f);

// Back

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f(0.0f, 1.0f, 0.0f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(1.0f, -1.0f, -1.0f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(-1.0f, -1.0f, -1.0f);

// Left

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex3f(0.0f, 1.0f, 0.0f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex3f(-1.0f, -1.0f, -1.0f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex3f(-1.0f, -1.0f, 1.0f);

glEnd(); // Done drawing the pyramid

glutSwapBuffers(); // Swap the front and back frame buffers (double buffering)

}

/\* Handler for window re-size event. Called back when the window first appears and

whenever the window is re-sized with its new width and height \*/

void reshape(GLsizei width, GLsizei height) { // GLsizei for non-negative integer

// Compute aspect ratio of the new window

if (height == 0) height = 1; // To prevent divide by 0

GLfloat aspect = (GLfloat)width / (GLfloat)height;

// Set the viewport to cover the new window

glViewport(0, 0, width, height);

// Set the aspect ratio of the clipping volume to match the viewport

glMatrixMode(GL\_PROJECTION); // To operate on the Projection matrix

glLoadIdentity(); // Reset

// Enable perspective projection with fovy, aspect, zNear and zFar

gluPerspective(45.0f, aspect, 0.1f, 100.0f);

}

/\* Main function: GLUT runs as a console application starting at main() \*/

int main(int argc, char\*\* argv) {

glutInit(&argc, argv); // Initialize GLUT

glutInitDisplayMode(GLUT\_DOUBLE); // Enable double buffered mode

glutInitWindowSize(640, 480); // Set the window's initial width & height

glutInitWindowPosition(50, 50); // Position the window's initial top-left corner

glutCreateWindow(title); // Create window with the given title

glutDisplayFunc(display); // Register callback handler for window re-paint event

glutReshapeFunc(reshape); // Register callback handler for window re-size event

initGL(); // Our own OpenGL initialization

glutMainLoop(); // Enter the infinite event-processing loop

return 0;

}

//////////////////// end of opengl demo

///////////////////// example 2 begin

//

// File: mycube.c

// Author: Matt Daisley

// Created: 4/25/2012

// Project: Source code for Make a Cube in OpenGL

// Description: Creates an OpenGL window and draws a 3D cube

// That the user can rotate using the arrow keys

//

// Controls: Left Arrow - Rotate Left

// Right Arrow - Rotate Right

// Up Arrow - Rotate Up

// Down Arrow - Rotate Down

// ----------------------------------------------------------

// Includes

// ----------------------------------------------------------

#include "stdafx.h"

#include <stdio.h>

#include <stdarg.h>

#include <math.h>

#define GL\_GLEXT\_PROTOTYPES

#ifdef \_\_APPLE\_\_

#include <GLUT/glut.h>

#else

#include <GL/glut.h>

#endif

// ----------------------------------------------------------

// Function Prototypes

// ----------------------------------------------------------

void display();

void specialKeys();

// ----------------------------------------------------------

// Global Variables

// ----------------------------------------------------------

double rotate\_y = 0;

double rotate\_x = 0;

// ----------------------------------------------------------

// display() Callback function

// ----------------------------------------------------------

void display(){

// Clear screen and Z-buffer

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

// Reset transformations

glLoadIdentity();

// Other Transformations

// glTranslatef( 0.1, 0.0, 0.0 ); // Not included

// glRotatef( 180, 0.0, 1.0, 0.0 ); // Not included

// Rotate when user changes rotate\_x and rotate\_y

glRotatef(rotate\_x, 1.0, 0.0, 0.0);

glRotatef(rotate\_y, 0.0, 1.0, 0.0);

// Other Transformations

// glScalef( 2.0, 2.0, 0.0 ); // Not included

//Multi-colored side - FRONT

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.0, 0.0); glVertex3f(0.5, -0.5, -0.5); // P1 is red

glColor3f(0.0, 1.0, 0.0); glVertex3f(0.5, 0.5, -0.5); // P2 is green

glColor3f(0.0, 0.0, 1.0); glVertex3f(-0.5, 0.5, -0.5); // P3 is blue

glColor3f(1.0, 0.0, 1.0); glVertex3f(-0.5, -0.5, -0.5); // P4 is purple

glEnd();

// White side - BACK

glBegin(GL\_POLYGON);

glColor3f(1.0, 1.0, 1.0);

glVertex3f(0.5, -0.5, 0.5);

glVertex3f(0.5, 0.5, 0.5);

glVertex3f(-0.5, 0.5, 0.5);

glVertex3f(-0.5, -0.5, 0.5);

glEnd();

// Purple side - RIGHT

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.0, 1.0);

glVertex3f(0.5, -0.5, -0.5);

glVertex3f(0.5, 0.5, -0.5);

glVertex3f(0.5, 0.5, 0.5);

glVertex3f(0.5, -0.5, 0.5);

glEnd();

// Green side - LEFT

glBegin(GL\_POLYGON);

glColor3f(0.0, 1.0, 0.0);

glVertex3f(-0.5, -0.5, 0.5);

glVertex3f(-0.5, 0.5, 0.5);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(-0.5, -0.5, -0.5);

glEnd();

// Blue side - TOP

glBegin(GL\_POLYGON);

glColor3f(0.0, 0.0, 1.0);

glVertex3f(0.5, 0.5, 0.5);

glVertex3f(0.5, 0.5, -0.5);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(-0.5, 0.5, 0.5);

glEnd();

// Red side - BOTTOM

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.0, 0.0);

glVertex3f(0.5, -0.5, -0.5);

glVertex3f(0.5, -0.5, 0.5);

glVertex3f(-0.5, -0.5, 0.5);

glVertex3f(-0.5, -0.5, -0.5);

glEnd();

glFlush();

glutSwapBuffers();

}

// ----------------------------------------------------------

// specialKeys() Callback Function

// ----------------------------------------------------------

void specialKeys(int key, int x, int y) {

// Right arrow - increase rotation by 5 degree

if (key == GLUT\_KEY\_RIGHT)

rotate\_y += 5;

// Left arrow - decrease rotation by 5 degree

else if (key == GLUT\_KEY\_LEFT)

rotate\_y -= 5;

else if (key == GLUT\_KEY\_UP)

rotate\_x += 5;

else if (key == GLUT\_KEY\_DOWN)

rotate\_x -= 5;

// Request display update

glutPostRedisplay();

}

// ----------------------------------------------------------

// main() function

// ----------------------------------------------------------

int main(int argc, char\* argv[]){

// Initialize GLUT and process user parameters

glutInit(&argc, argv);

// Request double buffered true color window with Z-buffer

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

// Create window

glutCreateWindow("Awesome Cube");

// Enable Z-buffer depth test

glEnable(GL\_DEPTH\_TEST);

// Callback functions

glutDisplayFunc(display);

glutSpecialFunc(specialKeys);

// Pass control to GLUT for events

glutMainLoop();

// Return to OS

return 0;

}