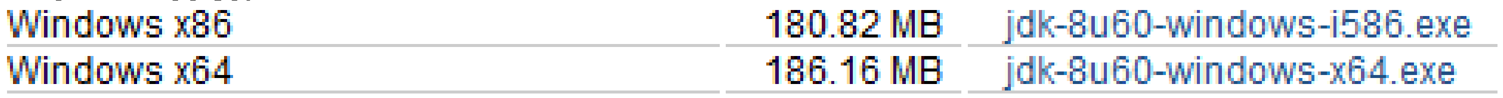
Set up OpenGL Programming Environment

Professor Jim X. Chen
Department of Computer Science
George Mason University
Fairfax, VA 22030

Setting Up Working Environment for JOGL

- http://www.cs.gmu.edu/~jchen/graphics/setup.html
  - Instsall JDK (which is included in Eclipse)
    - If otherwise, unistall Java first; download JDK and install



Windows x86	180.82 MB	jdk-8u60-windows-i586.exe
Windows x64	186.16 MB	jdk-8u60-windows-x64.exe

  - Installing a Java IDE
    - (**Eclipse**, jGRASP, Netbeans, or JBuilder)
  - Installing JOGL libraries

Windows simplified instructions

- Download Eclipse or jGRASP; http://www.eclipse.org/downloads/
- Download the examples and unzip them: joglExamples2011 ; http://www.cs.gmu.edu/~jchen/graphics/setup.html
- Download the corresponding libraries that matches your platform:
  - http://jogamp.org/wiki/index.php/Downloading_and_installing_JOGL#Downloading_the_latest_stable_version
- A working set is at:
  - http://www.cs.gmu.edu/~jchen/graphics/jogl/notes/jogl-1.1.1a-windows-amd64.zip (for 64-bit)
  - http://www.cs.gmu.edu/~jchen/graphics/jogl/notes/jogl-1.1.1a-windows-i586.zip
  - http://www.cs.gmu.edu/~jchen/graphics/jogl/notes/jogl-1.1.1a-macosx-ppc.zip
  - http://www.cs.gmu.edu/~jchen/graphics/jogl/notes/jogl-1.1.1a-macosx-universal.zip

Windows simplified instructions

Eclipse

- Start Eclipse using a working directory other than the sample programs'.
- Create a project corresponding to the directory: joglExamples2011.
- Remove the two dead "*.jar" file names if you see them. They are created when I compile my program under my specific directory.
- Under "Project->Properties-> (Java Build Path) -> Libraries", you can use (Add External JARs) to add "*.jar" (the two file names in the downloaded Jogl lib directory).
  - Click the triangle at the beginning of "gluegen-rt.jar" and "jogl.jar", you have Native Library Location, add the directory of your "*.jar" files, which allows access to the "*.dll" files.

jGRASP

- In the project under "Settings->PATH/CLASSPATH->Workspace", you can add the directory of the "*.dll" files to the system PATH window, and add "*.jar" files with full path to the CLASSPATH window.

Now you can try the examples. They should run from here.

Mac OS

- http://www.cs.gmu.edu/~jchen/graphics/jogl/notes/joglSetup.html
  1. Download Eclipse;
  2. Download Mac OS X: *.jar and *.jnilib (which are packaged in the jar file: jogl-1.1.1a-macosx-ppc.zip or jogl-1.1.1a-macosx-universal.zip depending on your platform)
  3. Copy all downloaded files into directory /System/Library/Java/Extensions/
  4. Download the examples and unzip them: joglExamples2011
  5. Start Eclipse, create a project corresponding to the directory: joglExamples201
  6. In eclipse, you can add "*.jar" (the two file names) under "Project->Properties->Libraries". Remove the two dead "*.jar" file names if you see them. They are created when I compile my program under my specific directory.
  7. Try the examples. They should run from here.

http://jogamp.org/

- JOGL's new home
- Tutorial http://jogamp.org/wiki/index.php/Jogl_Tutorial
- You can follow this page for the latest

OpenAL

- **OpenAL** (**Open** **A**udio **L**ibrary) is a cross-platform audio application programming interface (API). It is designed for efficient rendering of multichannel three dimensional positional audio. Its API style and conventions deliberately resemble those of OpenGL.

OpenCL

- **Open Computing Language (OpenCL)** is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs) and other processors.

How to setup OpenGL working environment

On Windows 95, 98, NT system, or 2000

- Download GLUT GUI library:

  http://www.xmission.com/~nate/glut.html

  Or http://cs.gmu.edu/~jchen/cs451/notes/glutdlls.zip

  to download precompiled glutdlls.zip for Windows 98-XP.

1. OpenGL/Glut Configuration Tutorial On Visual C++ 6.0

I assume you have Microsoft Visual C++ installed to the default directory, c:\msdev (For your system, you may start 'find' tool to find a VC++ file like gl.h, and then determine the default directory for your system.)

A. Install OpenGL and Glut

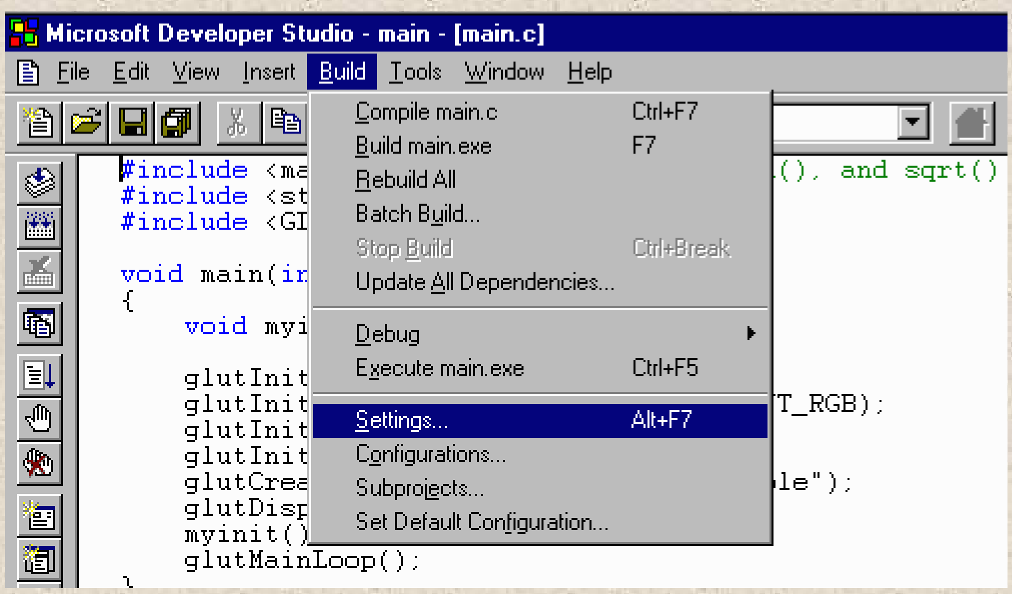
• Create a temporary directory and extract the glutdlls.zip there. Copy all *.h files to c:\msdev\include\gl; all *.lib files to c:\msdev\lib; and all *.dll files to c:\windows\system32.

You are now ready to enter the fun and exciting world of OpenGL and Glut programming!

B. Set OpenGL or Glut in Visual C++:

• First, make sure that you've included all the necessary *.h files in your program. You need <GL/glut.h> for Glut; or <GL/gl.h> and <afxwin.h>, and possibly <GL/glu.h> and/or <GL/glaux.h> for OpenGL.

•You need to let Visual C++ know that you want it to consult all the necessary libraries. To do this, you first need to choose Settings under the Project or Build menu item. The Settings button will appear after you compile your program once.

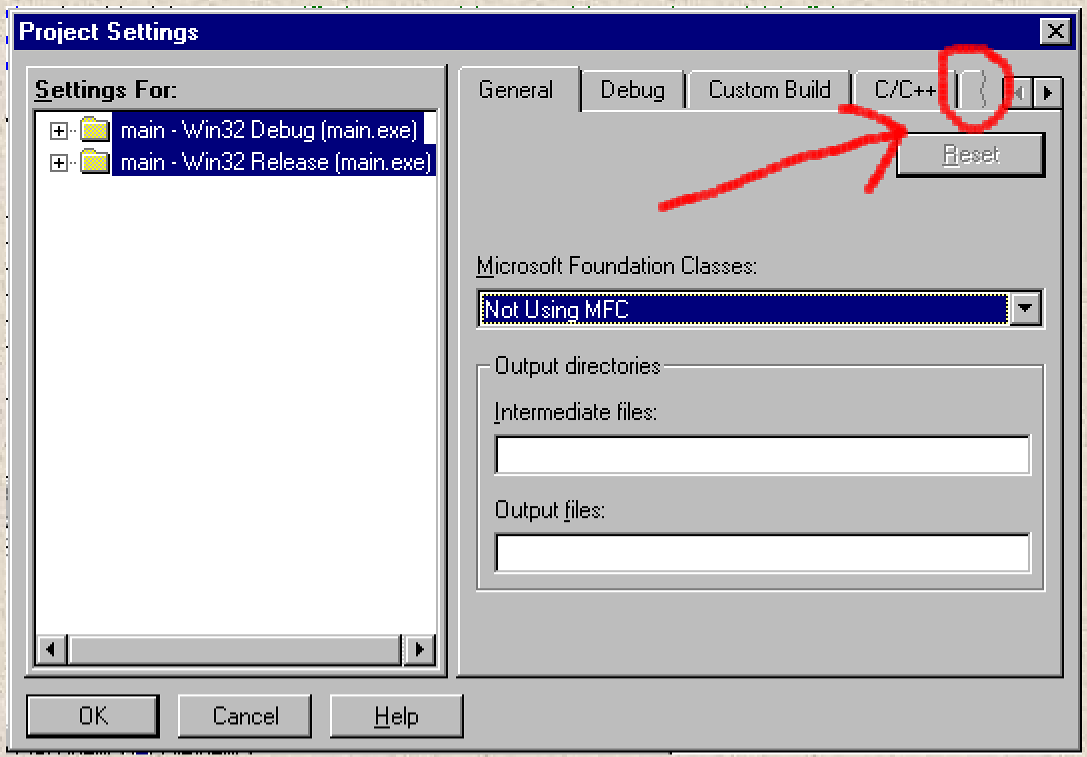


```
#include <ma                                    (), and sqrt()
#include <st
#include <GI

void main(ir
{
    void myi

    glutInit
    glutInit                                    T_RGB);
    glutInit
    glutInit
    glutCrea                                    le");
    glutDisp
    myinit()
    glutMainLoop();
```

• A Project Settings dialog box will now pop up. You now need to select the Link tab. Unfortunately, the Link tab may be hidden. Click the half-a-tab circled below...



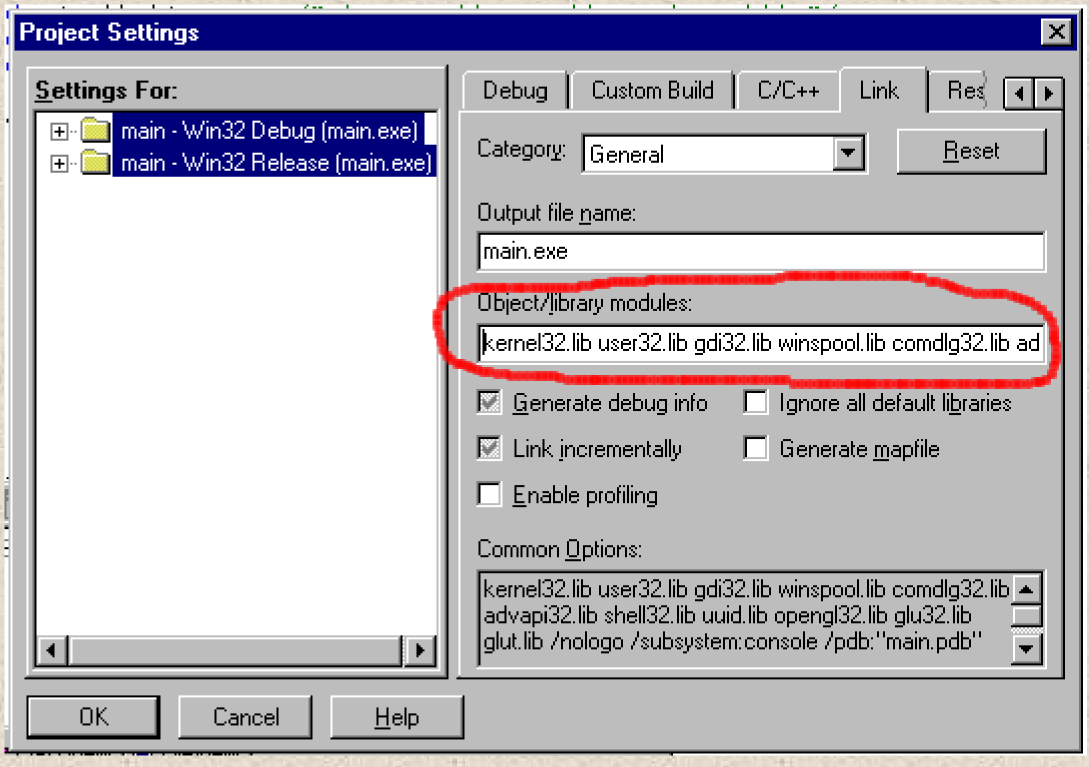
Microsoft Foundation Classes: Not Using MFC

Output directories

Intermediate files:

Output files:

...and the Link page will magically appear:



Category: General

Output file name:

main.exe

Object/library modules:

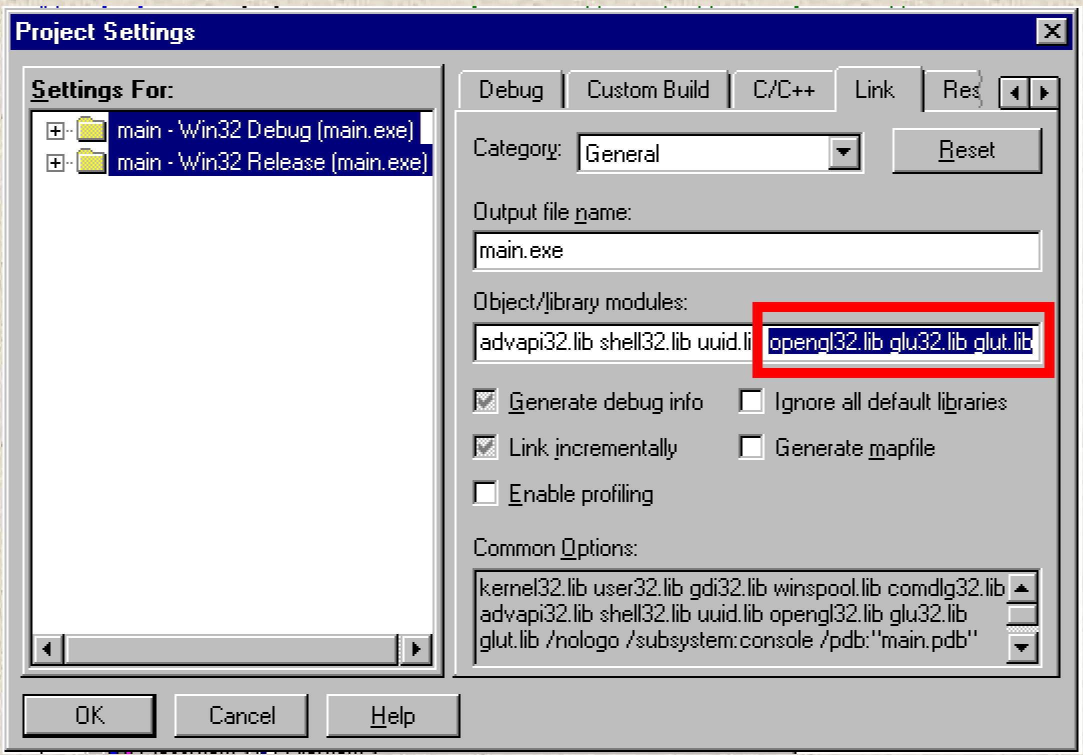
kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib ad

[✓] Generate debug info
[ ] Ignore all default libraries
[✓] Link incrementally
[ ] Generate mapfile
[ ] Enable profiling

Common Options:

```
kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib uuid.lib opengl32.lib glu32.lib
glut.lib /nologo /subsystem:console /pdb:"main.pdb"
```

• The Object/library modules section of this page which you should be interested in is circled in red above. Scroll to the end of this line and add the necessary *.lib files to the list.



Project Settings

Settings For:
- main - Win32 Debug (main.exe)
- main - Win32 Release (main.exe)

Category: General

Output file name: main.exe

Object/library modules: advapi32.lib shell32.lib uuid.li opengl32.lib glu32.lib glut.lib

Generate debug info [✓]
Ignore all default libraries [ ]
Link incrementally [✓]
Generate mapfile [ ]
Enable profiling [ ]

Common Options:

```
kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib uuid.lib opengl32.lib glu32.lib
glut.lib /nologo /subsystem:console /pdb:"main.pdb"
```

• There are four libraries that you might need to add: opengl32.lib, glu32.lib, glaux.lib, and glut32.lib. Since it doesn't hurt much to have extra libs listed, I suggest you add all of them.

C. Compile and run the following program

Source code examples can be downloaded from

ftp://ftp.sgi.com/opengl/opengl12.zip

Or

http://cs.gmu.edu/~jchen/cs451/notes/opengl12.zip

/* Example 1.1.point.c: draw multiple randomly generated points */

// by Jim X. Chen; September 2000

Points.c

Points

OpenGL/GLUT with Visual Studio.net

1. **Extract the glutdlls.zip**

- Copy all *.h files to **C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\include\gl\;** (if gl directory is not exist, create one); all *.lib files to **C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\lib;** and all *.dll files to c:\windows\system32.

2. **Start Visual Studio**

- Under the file menu select New Project
- In the "New Project Window", select "Visual C++ Projects" on the left hand side and select "Win32 console project" on the right hand side

  Specify name and location of the project
- In the "Win32 Application Wizard Window ", select the "Application Settings"; then click "Empty Project"

- Now the project is open; right click on the "Source Files" folder (on the right); select "Add" "Add New Item" "C++ file"

  Name the file, and copy whatever code to this file, and run.

2. To build and run your project, hit "F5"

3. Point Visual Studio to the relevant GLUT files

(Optional, if you put your glut under C:/my/glut/)

- Right click on your *project name* (on the right hand side) or pull down the Project menu

  Select "Properties" "C/C++" properties

  Select "General"and add the following line to "Additional Include Directories" "C:/my/glut/include"

- Select the "Linker" properties "General" and add the following line to "Additional Library Directories" "C:/my/glut/"

2. OpenGL/Glut Tutorial On Borland C++

Please visit web site for detail:

http://home.clara.net/paulyg/ogl.htm
http://www.gnt.net/~heiman/opengl.htm
http://www.opengl.org/Coding/Coding.html

How to setup OpenGL working environment

On SGI Workstations (ST Rm228 and Graphics Lab.)

- The system administrator should have OpenGL library installed. Check this with your system administrator
- Download GLUT GUI library for SGI: Go to web site:

  http://reality.sgi.com/opengl/glut3/glut3.html
- Get GLUT documents from the same place
- Following the instructions to compile and configure GLUT on your system
- GLUT Online help: http://reality.sgi.com/opengl/spec3/spec3.html
- Sample code: ftp://ftp.sgi.com/opengl/opengl12.tar.Z

How to setup OpenGL working environment

On SUN Workstations (ST2 SITE Lab.)

• First we'll create directories for AUX and then copy AUX files from system directories. (Here we use aux GUI library)

```
mkdir $HOME/OpenGL
mkdir $HOME/OpenGL/aux
mkdir $HOME/OpenGL/aux/examples
mkdir $HOME/OpenGL/aux/libaux
mkdir $HOME/OpenGL/aux/libtk
cd /usr/openwin/share/src/GL/contrib/utils/aux
cp ./aux/examples $HOME/OpenGL/aux/examples
cp ./aux/libaux $HOME/OpenGL/aux/libaux
cp ./aux/libtk $HOME/OpenGL/aux/libtk
```

OR just run the following command:

```
cp -r ~xwang1/OpenGL  $HOME/OpenGL
```

•After that, you have your environment ready. Now you can compile and run some example program.

•First change to the examples directory under your account by:

```
cd $HOME/OpenGL/aux/examples
```

•then compile an example program (movelight.c) by

```
make movelight
```

•finnally, run ‘movelight’, and you will see the result.

Reference web site:

http://sun.com/software/graphics/OpenGL/

Microsoft .NET development environment and C#

- C# (pronounced "C sharp") for the new Microsoft .NET development environment provides tools and services that fully exploit both computing and communications.
- C# is generally regarded as being a successor to C++ that incorporates and improves upon the major innovations made in the Java programming language.

C# OpenGL Wrapper

- There is not yet a version of .NET which supports OpenGL. But you may find several C# OpenGL wrappers that are freely available. Google search with the key words “C# OpenGL wrapper”.
- The simplest C# OpenGL wrapper will be “**Colin Fahey's C# OpenGL Wrapper**”. Detailed information about this wrapper may be found at http://www.colinfahey.com/opengl/csharp.htm.

C# OpenGL Wrapper

- Another open source wrapper **CsGL** at http://csgl.sourceforge.net/.
- It is recommended to use Colin Fahey's C# OpenGL Wrapper because it’s simpler and rather easier to install.
- **If you will be using Microsoft .NET to do your project, please make sure to include the information about the type of wrapper you are using and the steps of running your project when submitting it.**

Other References on Setting up your Graphics programming environment

- http://www.cs.uml.edu/~hmasterm/Environments/environment.html
- Set up DirectX programming environment (Visual C++)

Computer Graphics

Jim X. Chen, Ph.D.

Director of Computer Graphics Lab

George Mason University

Reference

- Jim. X Chen and Chunyang Chen, *Foundations of 3D Graphics Programming*, Springer Verlag, Second Edition, required text book.
- Jim. X Chen, *Guide to 3D Graphics Tools*, Second Edition, Springer Verlag, companion in C with a survey of graphics tools.
- Mason Woo, Jackie Neider, and Tom Davis, *OpenGL Programming Guide*, Addison Wesley. Programming examples
- James D. Foley, Andries van Dam, Steven K. Feiner and John F. Hughes, *Computer Graphics: Principles and Practice*, Addison Wesley. Detailed theory in depth.
- Donald Hearn and M. Pauline Baker, *Computer Graphics*, Printice-Hall. Easy to read textbook for undergraduates. Contains more than enough.

Objects and Models

- ***Objectives***
  - Introduce basic graphics concepts
    - object, model, image, graphics library, frame buffer, scan-conversion, clipping, and anti-aliasing
  - Set up OpenGL programming environment
  - Understand simple OpenGL programs

Computer Graphics

Points · Lights · Raytracing · nfaces

- CG displays or animates real or imaginary objects from their computer-based models;
  - Modeling creating and modifying a model
- image processing treats the inverse process: the analysis of images, pattern recognition, or the reconstruction of 2D or 3D objects from images.

Display

- A graphics *display* is a drawing area comprised of an array of fine points called pixels (picture elements).
- At the heart of a graphics system there is a magic pen,
  - move at lightning speed to a specific pixel
  - draw the pixel with a specific color — a red, green, and blue (RGB) vector value.
  - Computer graphics is about using this pen automatically through programming.

Object, Model, and Image

- A real or imaginary *object* is represented in a computer as a model, and is displayed as an image.
- A *model* is an abstract description of the object’s shape (vertices) and attributes (colors),
  - which can be used to find all the points on the object corresponding to the pixels in the drawing area.
  - Given a model, the application program will control the pen through a graphics library to generate the corresponding image.
- An *image* is simply a 2D array of pixels.

Interactive

•User controls the creation, modification, and animation of the models through input devices (keyboard & mouse)

spider

Primitive and Graphics Library

Primitive Lab

- A *graphics library* provides a set of graphics commands or functions.
  - bound in *C*, *Java*, or other programming languages on different platforms.
  - specify primitive 2D and 3D geometric models to be digitized and displayed.
- *Primitive models* or simply *primitives* stand for some simple shapes (such as points, lines, and polygons)
- *OpenGL* is a graphics library; *DirectX* includes a graphics library *Direct3D*

OpenGL Programming

- OpenGL is the most widely used graphics library (GL) or application programming interface (API)
- Compile and run Example *J1_0_Point.java*
  - To change the font size in Eclipes: Window->Preferences… General->Colors and Fonts->Basic->Text Font, then select.
  - Corresponding example in C: *1.1.point.c*
- links to all the example programs, and setting up working environments on different platforms:
  - http://www.cs.gmu.edu/~jchen/graphics/

/* draw a point */

```
/* Java's supplied classes are "imported". Here the awt (Abstract
     Windowing Toolkit) is imported to provide "Frame" class, which
     includes windowing functions
*/
import java.awt.*;

// JOGL: OpenGL functions
import net.java.games.jogl.*;
```

```
/* Java class definition: “extends” means “inherits”. So J1_0_Point is a subclass
     of Frame, and it inherits Frame’s variables and methods. “implements” means
     GLEventListener is an interface, which only defines methods (init(),
     reshape(), display(), and displaychanged()) without implementation. These
     methods are actually callback functions handling events. J1_0_Point will
     implement GLEventListener’s methods and use them for different events.
*/
public class J1_0_Point extends Frame implements GLEventListener {

       static int HEIGHT = 400, WIDTH = 400;
       static GL gl; //interface to OpenGL
       static GLCanvas canvas; // drawable in a frame

       …; // methods
}
```

```
public class J1_0_Point extends Frame implements GLEventListener {

      …

      public J1_0_Point() { // constructor

            //1. specify a drawable: canvas
            GLCapabilities capabilities = new GLCapabilities();
            canvas =
    GLDrawableFactory.getFactory().createGLCanvas(capabilities);

            //2. listen to the events related to canvas: reshape
            canvas.addGLEventListener(this);
            …
    }
    …
}
```

```
 public J1_0_Point() { // constructor
        …
        //3. add the canvas to fill the Frame container
        add(canvas, BorderLayout.CENTER);
        /* In Java, a method belongs to a class object.
        Here the method “add” belongs to J1_0_Point’s
        instantiation, which is frame in “main” function.
        It is equivalent to use “this.add(canvas, ...)” */

        //4. interface to OpenGL functions
        gl = canvas.getGL();
}
public static void main(String[] args) {

        J1_0_Point frame = new J1_0_Point();
        …
}
```

```
public class J1_0_Point extends Frame implements GLEventListener {
    …
    public static void main(String[] args) {
    J1_0_Point frame = new J1_0_Point();

    //5. set the size of the frame and make it visible
    frame.setSize(WIDTH, HEIGHT);
    frame.setVisible(true);
    }

    // Called once for OpenGL initialization
    public void init(GLDrawable drawable) {

    //6. specify a drawing color: red
    gl.glColor3f(1.0f, 0.0f, 0.0f);
    }
    …
}
```

```
public class J1_0_Point extends Frame implements GLE ventListener {

    …

    // Called for handling reshaped drawing area
    public void reshape(GLDrawable drawable, int x, int y,
            int width, int height) {

            //7. specify the drawing area (frame) coordinates

            gl.glMatrixMode(GL.GL_PROJECTION);
            gl.glLoadIdentity();
            gl.glOrtho(0, width, 0, height, -1.0, 1.0);
    }

    …
}
```

```
public class J1_0_Point extends Frame implements GLEventListener {
    …

    // Called for OpenGL rendering every reshape
    public void display(GLDrawable drawable) {

        //8. specify to draw a point
        gl.glBegin(GL.GL_POINTS);
                gl.glVertex2i(WIDTH/2, HEIGHT/2);
        gl.glEnd();
    }

    // called if display mode or device are changed
    public void displayChanged(GLDrawable drawable,
        boolean modeChanged, boolean deviceChanged) {
    }
}
```

```
/* Example 1.1.point.c: draw randomly generated points */
      #include <stdlib.h>
      #include <GL/glut.h>

      #define Height 400
      #define Width 400
```

Points

```
void display(void)
{
        int x, y;

        //a. generate a random point
        x = rand() % Width;
        y = rand() % Height;
        //b. specify a drawing color: red
        glColor3f(1, 0, 0);

        //c. specify to draw a point
        glBegin(GL_POINTS);
                glVertex2i (x,y);
        glEnd();

        //d. start drawing
        glFlush();
}
```

```
static void reshape(int w, int h)
{       //e. specify the window's coordinates
         glMatrixMode (GL_PROJECTION);
         glLoadIdentity ();
         glOrtho(0, Width, 0, Height, -1.0, 1.0);
}

int main(int argc, char **argv)
{       //f. initialize a drawing area
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE);
        glutInitWindowSize(Width, Height);
        glutCreateWindow("Example 1.1.point.c");

         //g. specify event callback functions
         glutReshapeFunc(reshape);
         glutDisplayFunc(display);
         glutIdleFunc(display);
         glutMainLoop();
}
```

Set up your programming environment (due before next class)

1. Set up your programming environment, and run the first sample program that draws a point (or randomly generated points).
2. If you failed, schedule a time to come to my office or bring your computer to the next class.
3. You don't have to let me know if you succeed in setting up your working environment.
4. You have to let me know if you cannot get the sample programs to run within a week. Please come to my office during my office hours or make an appointment with me or my TA.