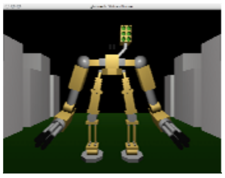
433-380 Graphics and Computation

Introduction to OpenGL

Some images in these slides are taken from "The OpenGL Programming Manual",
which is copyright Addison Wesley and the OpenGL Architecture Review Board.
http://www.opengl.org/documentation/red_book_1.0/

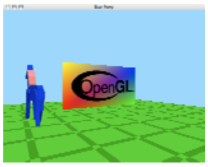
Outline

- OpenGL Background and History
- Other Graphics Technology
- Drawing
- Viewing and Transformation
- Lighting
- JOGL and GLUT
- Resources

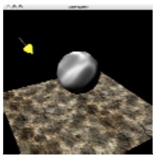


OpenGL Background and History

- OpenGL = Open Graphics Library
- Developed at Silicon Graphics (SGI)
- Successor to IrisGL
- Cross Platform
  (Win32, Mac OS X, Unix, Linux)
- Only does 3D Graphics. No Platform Specifics
  (Windowing, Fonts, Input, GUI)
- Version 1.4 widely available
- Two Libraries
  - GL (Graphics Library)
  - GLU (Graphics Library Utilities)



Other Graphics Technology



- Low Level Graphics
- OpenGL
- Scene Graphs, BSPs
  - OpenSceneGraph, Java3D, VRML, PLIB
- DirectX (Direct3D)
- Can mix some DirectX with OpenGL (e.g OpenGL and DirectInput in Quake III)

Platform Specifics

- Platform Specific OpenGL Interfaces
  - Windows (WGL)
  - X11 (GLX)
  - Mac OS X (CGL/AGL/NSOpenGL)
  - Motif (GLwMwidget)
  - Qt (QGLWidget, QGLContext)
- Java (JOGL)
- GLUT (GL Utility Library)

The Drawing Process

```
ClearTheScreen();
DrawTheScene();
CompleteDrawing();
SwapBuffers();
```

- In animation there are usually two buffers. Drawing usually occurs on the background buffer. When it is complete, it is brought to the front (swapped). This gives a smooth animation without the viewer seeing the actual drawing taking place. Only the final image is viewed.
- The technique to swap the buffers will depend on which windowing library you are using with OpenGL.

Clearing the Window

```
glClearColor(0.0, 0.0, 0.0, 0.0);
glClear(GL_COLOR_BUFFER_BIT);
```

- Typically you will clear the colour and depth buffers.

```
glClearColor(0.0, 0.0, 0.0, 0.0);
glClearDepth(0.0);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Setting the Colour

- Colour is specified in (R,G,B,A) form [Red, Green, Blue, Alpha], with each value being in the range of 0.0 to 1.0.
- There are many variants of the glColor command

```
    glColor4f(red, green, blue, alpha);
    glColor3f(red, green, blue);

glColor3f(0.0, 0.0, 0.0);      /* Black    */
glColor3f(1.0, 0.0, 0.0);      /* Red      */
glColor3f(0.0, 1.0, 0.0);      /* Green    */
glColor3f(1.0, 1.0, 0.0);      /* Yellow   */
glColor3f(1.0, 0.0, 1.0);      /* Magenta  */
glColor3f(1.0, 1.0, 1.0);      /* White    */
```

Complete Drawing the Scene

- Need to tell OpenGL you have finished drawing your scene.

```
glFinish();
```

or

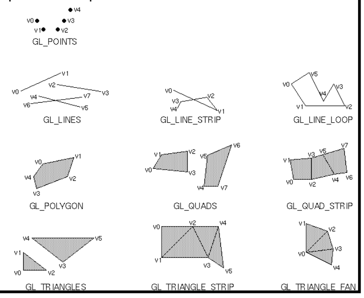
```
glFlush();
```

- For more information see: http://www.rush3d.com/reference/opengl-redbook-1.1/chapter02.html

Drawing in OpenGL

- Use glBegin() to start drawing, and glEnd() to stop.
- glBegin() can draw in many different styles (see figure)
- glVertex3f(x,y,z) specifies a point in 3D space.

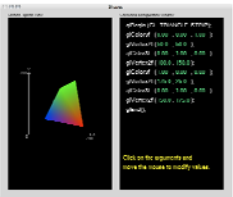
```
glBegin(GL_POLYGON);
glVertex3f(0.0, 0.0, 0.0);
glVertex3f(0.0, 3.0, 1.0);
glVertex3f(3.0, 3.0, 3.0);
glVertex3f(4.0, 1.5, 1.0);
glVertex3f(3.0, 0.0, 0.0);
glEnd();
```



Mixing Geometry with Colour

- Specifying vertices can be mixed with colour and other types of commands for interesting drawing results.

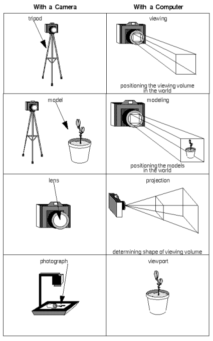
```
glBegin(GL_POLYGON);
glColor3f(1.0, 0.0, 0.0);
glVertex3f(0.0, 0.0, 0.0);
glColor3f(0.0, 1.0, 0.0)
glVertex3f(3.0, 1.0, 0.0);
glColor3f(0.0, 0.0, 1.0);
glVertex3f(3.0, 0.0, 0.0);
glEnd();
```

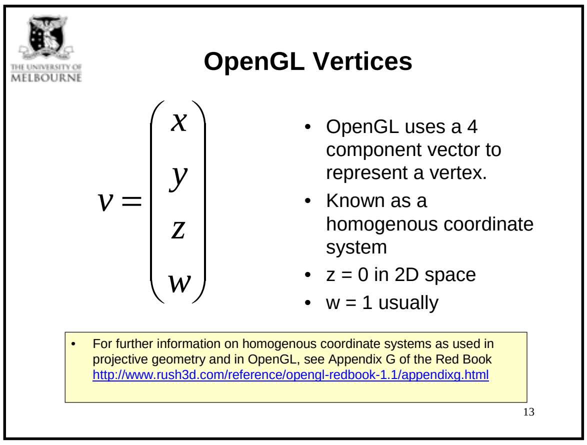


```
glBegin(GL_TRIANGLE_STRIP);
glColor3f ( [illegible] );
glVertex2f( [illegible] );
glColor3f ( 0.00 , 1.00 , 0.00 );
glVertex2f( 100.0 , 150.0 );
glColor3f ( [illegible] );
glVertex2f( [illegible] );
glColor3f ( 0.00 , 1.00 , 0.00 );
glVertex2f( [illegible] );
glEnd();
```

Click on the arguments and move the mouse to modify values.

Viewing the Scene: The Camera


With a Camera
With a Computer
tripod
viewing
positioning the viewing volume in the world
model
modeling
positioning the models in the world
lens
projection
determining shape of viewing volume
photograph
viewport



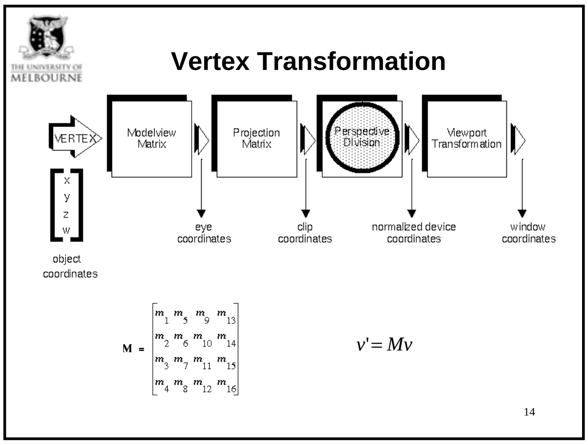
OpenGL Vertices

$$v = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

- OpenGL uses a 4 component vector to represent a vertex.
- Known as a homogenous coordinate system
- z = 0 in 2D space
- w = 1 usually

- For further information on homogenous coordinate systems as used in projective geometry and in OpenGL, see Appendix G of the Red Book http://www.rush3d.com/reference/opengl-redbook-1.1/appendixg.html

13

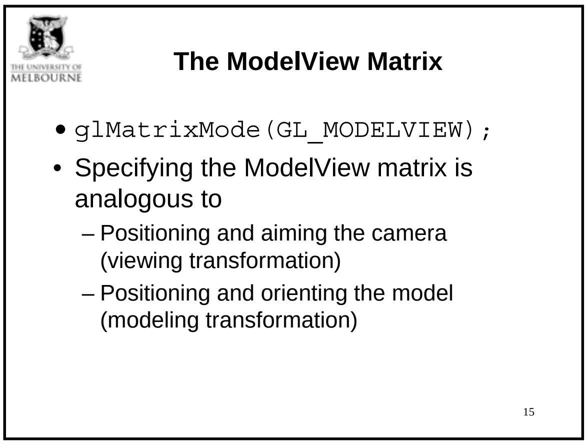


Vertex Transformation

$$\mathbf{M} = \begin{bmatrix} m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \\ m_4 & m_8 & m_{12} & m_{16} \end{bmatrix}$$

$$v' = Mv$$

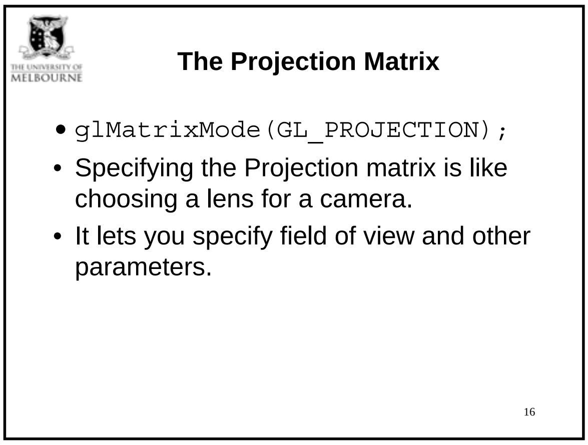
14



The ModelView Matrix

- `glMatrixMode(GL_MODELVIEW);`
- Specifying the ModelView matrix is analogous to
  - Positioning and aiming the camera (viewing transformation)
  - Positioning and orienting the model (modeling transformation)

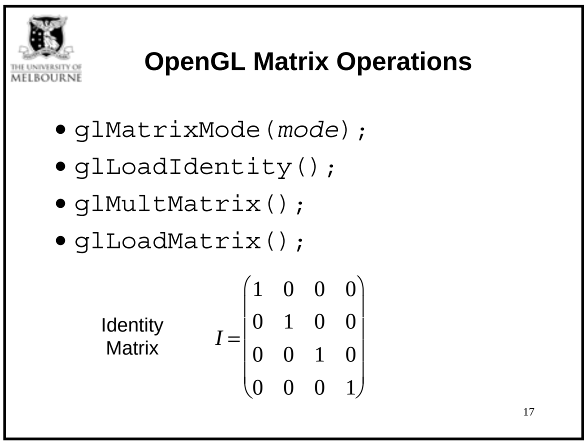
15



The Projection Matrix

- `glMatrixMode(GL_PROJECTION);`
- Specifying the Projection matrix is like choosing a lens for a camera.
- It lets you specify field of view and other parameters.

16



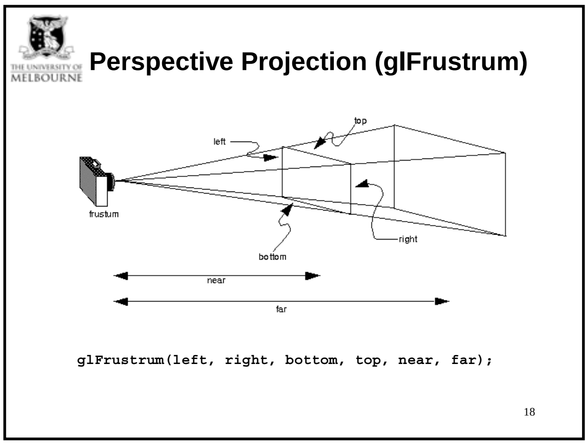
OpenGL Matrix Operations

- `glMatrixMode(`*`mode`*`);`
- `glLoadIdentity();`
- `glMultMatrix();`
- `glLoadMatrix();`

Identity Matrix

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

17

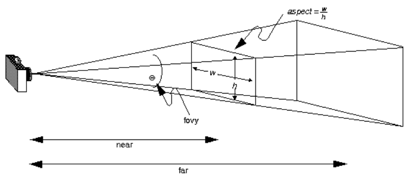


Perspective Projection (glFrustrum)

```
glFrustrum(left, right, bottom, top, near, far);
```

18

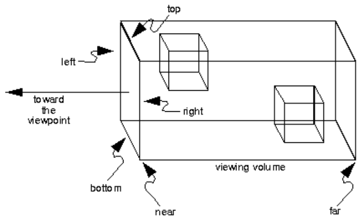
Perspective Projection (gluPerspective) from GLU



```
gluPerspective(fovy, aspect, near, far);
```

fovy = field of view angle in degrees, in the y direction.
aspect = aspect ratio that determines the field of view in the x direction (ratio of width to height).

Orthographic (Parallel) Projection (glOrtho)

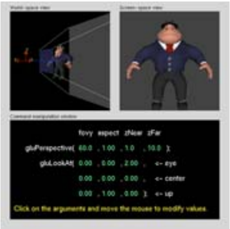


```
glOrtho(left, right, bottom, top, near, far);
```

gluLookAt

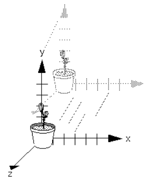
Specifies the camera position, the point where the camera is pointing, and the orientation vector of the camera.

```
gluLookAt(eyex, eyey, eyez,
          centerx, centery, centerz,
          upx, upy, upz);
```



```
              fovy  aspect  zNear  zFar
gluPerspective( 60.0 , 1.00 , 1.0  , 10.0 );
     gluLookAt( 0.00 , 0.00 , 2.00 ,   <- eye
                0.00 , 0.00 , 0.00 ,   <- center
                0.00 , 1.00 , 0.00 );  <- up
Click on the arguments and move the mouse to modify values.
```

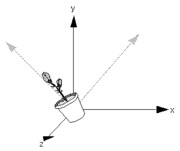
Translation Transformation



$$T = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
glTranslatef(x,y,z);
```

Rotation Transformations



$$\text{glRotate*}(\alpha, 1, 0, 0): \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos a & -\sin a & 0 \\ 0 & \sin a & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

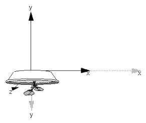
$$\text{glRotate*}(\alpha, 0, 1, 0): \begin{bmatrix} \cos a & 0 & \sin a & 0 \\ 0 & 1 & 0 & 0 \\ -\sin a & 0 & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{glRotate*}(\alpha, 0, 0, 1): \begin{bmatrix} \cos a & -\sin a & 0 & 0 \\ \sin a & \cos a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
glRotatef(angle, x, y,z);
```

Specifies an angle and an axis (x,y,z) to rotate around.

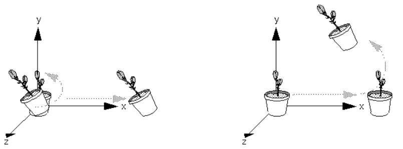
Scaling Transformations



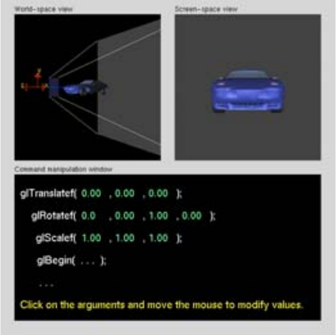
$$S = \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } S^{-1} = \begin{bmatrix} \frac{1}{x} & 0 & 0 & 0 \\ 0 & \frac{1}{y} & 0 & 0 \\ 0 & 0 & \frac{1}{z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
glScale(x,y,z);
```

Order of Transformations



Transformations in Action



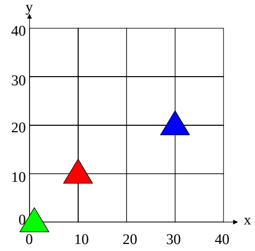
```
glTranslatef( 0.00  , 0.00  , 0.00  );
   glRotatef( 0.0   , 0.00  , 1.00  , 0.00  );
    glScalef( 1.00  , 1.00  , 1.00  );
     glBegin( . . . );
      . . .
```

Click on the arguments and move the mouse to modify values.

The Matrix Stack

- glPushMatrix() and glPopMatrix()

```
glPushMatrix();
glTranslatef(10,10,0);
DrawRedTriangle();
   glPushMatrix();
   glTranslatef(20,10,0);
   DrawBlueTriangle();
   glPopMatrix();
glPopMatrix();
DrawGreenTriangle();
```



OpenGL Lighting

- Eight available lights (individually toggled)
- Lighting types
  - Emitted
  - Ambient
  - Diffuse
  - Specular
- Material Colours and Properties
- Lighting Demo

Setting up an OpenGL Light

```
GLfloat lightAmbient[] = { 0.4, 0.5, 0.0, 1.0 };
GLfloat lightDiffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat lightSpecular[] = { 1.0, 1.0, 1.0, 1.0};
GLfloat lightPosition[] = {1.0, 1.0, 1.0, 0.0};

glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular);
glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);

glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);
```

glLight{if}[v](light, pname, param)

Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	ambient RGBA intensity of light
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	diffuse RGBA intensity of light
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	specular RGBA intensity of light
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	(*x, y, z, w*) position of light
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	(*x, y, z*) direction of spotlight
GL_SPOT_EXPONENT	0.0	spotlight exponent
GL_SPOT_CUTOFF	180.0	spotlight cutoff angle
GL_CONSTANT_ATTENUATION	1.0	constant attenuation factor
GL_LINEAR_ATTENUATION	0.0	linear attenuation factor
GL_QUADRATIC_ATTENUATION	0.0	quadratic attenuation factor

Table 6-1 : Default Values for pname Parameter of glLight*()

Material Properties

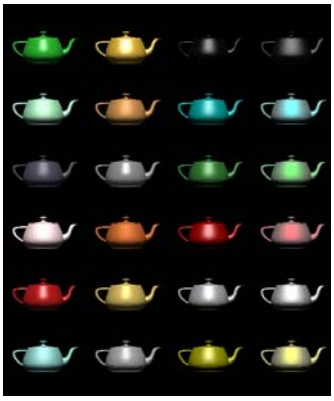
- glMaterial{if}[v](GLenum face
  GLenum pname,
  TYPE param);

```
GLfloat material[] = {0.1, 0.5, 0.8, 1.0 };
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE,
                                material);

GLfloat matSpecular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat lowShininess[] = { 5.0 };
glMaterialfv(GL_FRONT, GL_SPECULAR, matSpecular);
glMaterialfv(GL_FRONT, GL_SHININESS, lowShininess);
```

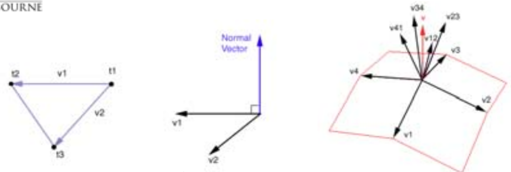
glMaterial Default Parameters

Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	ambient color of material
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	diffuse color of material
GL_AMBIENT_AND_DIFFUSE		ambient and diffuse color of material
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	specular color of material
GL_SHININESS	0.0	specular exponent
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	emissive color of material
GL_COLOR_INDEXES	(0,1,1)	ambient, diffuse, and specular color indices

Table 6-2 : Default Values for pname Parameter of glMaterial*()

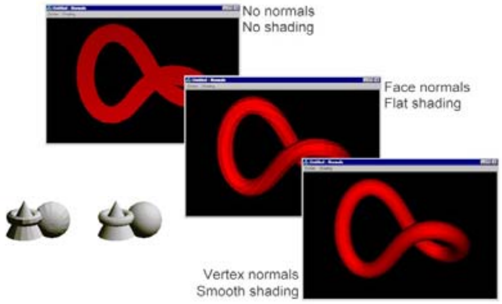


Normal Vectors



```
glBegin(GL_POLYGON);
    glNormal3fv(n0);
    glVertex3fv(v0);
    glNormal3fv(n1);
    glVertex3fv(v1);
    glNormal3fv(n2);
    glVertex3fv(v2);
    glNormal3fv(n3);
    glVertex3fv(v3);
glEnd();
```

Normal Vectors (2)



http://www.codeguru.com/Cpp/G-M/opengl/article.php/c2681/

Hidden Surface Removal

- In order for OpenGL to remove hidden surfaces, you need to enable depth testing for the depth buffer.

```
glEnable(GL_DEPTH_TEST);
while (1) {
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   GetCurrentViewingPosition();
   DrawObjectA();
   DrawObjectB();
}
```

GLUT

- GLUT = GL Utility Library
- Easy, stable, simple to use library for showing OpenGL demos.
- Limited to simple windows, mouse/keyboard input, and some simple 3D shapes.
- Most OpenGL demos and code on the web use GLUT.
- Default implementation in C (bindings for many languages available: Python, Perl etc)

Example GLUT Program in C

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>

. . .

int main(int argc, char** argv)
{
   glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);
   glutInitWindowSize(1024, 768);
   glutInitWindowPosition(0, 0);
   glutInit(argc, argv);

   glutCreateWindow(“OpenGL Demo”);
   glutReshapeFunc(reshape);
   glutDisplayFunc(display);
   glutIdleFunc(display);
   glutKeyboardFunc(keyboard);
   glutMouseFunc(mouse);
   glutMotionFunc(motion);
   InitGL();

   glutMainLoop();
}
```

Typical InitGL() Function

```
void InitGL(void)
{
   glClearColor(0.0, 0.0, 0.0, 0.0);
   glClearDepth(1.0);
   glDepthFunc(GL_LEQUAL);
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_COLOR_MATERIAL);
   glShadeModel(GL_SMOOTH);
   glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);
   glEnable(GL_BLEND);

   glLightModel(GL_LIGHT_MODEL_AMBIENT, [0.5, 0.5, 0.5, 1.0]);
   glLightfv(GL_LIGHT0, GL_AMBIENT,  (0.4, 0.4, 0.4, 1.0));
   glLightfv(GL_LIGHT0, GL_DIFFUSE,  (0.4, 0.4, 0.4, 1.0));
   glLightfv(GL_LIGHT0, GL_POSITION, (0.0, 0.0, -100.0, 1.0));
   glEnable(GL_NORMALIZE);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);

   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(fovy, aspect, zNear, zFar);
   glMatrixMode(GL_MODELVIEW);
}
```

Typical GLUT Reshape Function

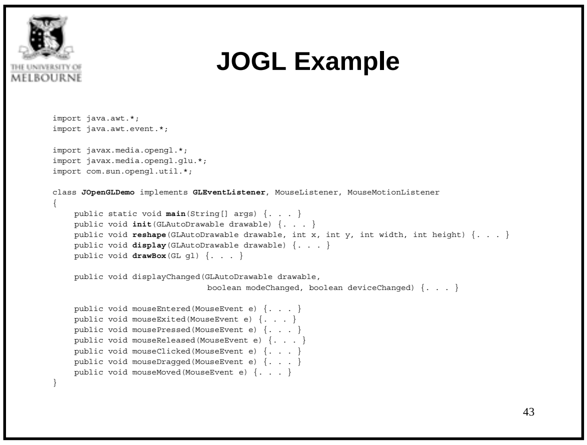
```
void reshape(int width, int height);
{
   glViewport(0, 0, width, height);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(fovy, aspect, zNear, zFar);
}
```

Typical GLUT Display Function

```
void display(void)
{
   UpdateWorld();

   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(fovy, aspect, zNear, zFar);

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   gluLookAt(0.0, 0.0, 150.0,
         0.0, 0.0, 0.0,
         0.0, 1.0, 0.0);

   RenderScene();

   glutSwapBuffers();
}
```

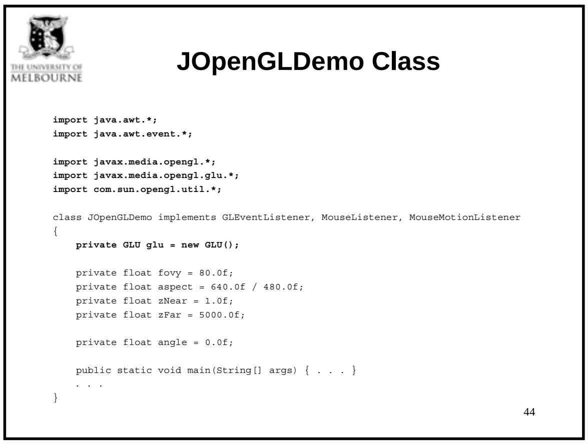
JOGL

- Java bindings for OpenGL use JNI.
- Still undergoing active development.
- Some tutorials (e.g. NeHe) on the web have JOGL code which is not compatible with the latest version.
- For this class JSR 231 beta 03 (Feb 17 2006)
- http://jogl.dev.java.net/
- Will need to prefix OpenGL functions with gl and constants with GL when using JOGL/Java.
  - `glLineWidth(1.0)` becomes `gl.glLineWidth(1.0);`
  - `GL_SMOOTH` becomes `GL.GL_SMOOTH`



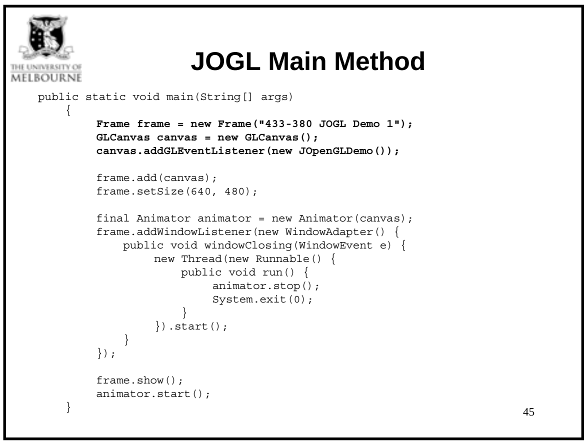
JOGL Example

```
import java.awt.*;
import java.awt.event.*;

import javax.media.opengl.*;
import javax.media.opengl.glu.*;
import com.sun.opengl.util.*;

class JOpenGLDemo implements GLEventListener, MouseListener, MouseMotionListener
{
    public static void main(String[] args) {. . . }
    public void init(GLAutoDrawable drawable) {. . . }
    public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {. . . }
    public void display(GLAutoDrawable drawable) {. . . }
    public void drawBox(GL gl) {. . . }

    public void displayChanged(GLAutoDrawable drawable,
                               boolean modeChanged, boolean deviceChanged) {. . . }

    public void mouseEntered(MouseEvent e) {. . . }
    public void mouseExited(MouseEvent e) {. . . }
    public void mousePressed(MouseEvent e) {. . . }
    public void mouseReleased(MouseEvent e) {. . . }
    public void mouseClicked(MouseEvent e) {. . . }
    public void mouseDragged(MouseEvent e) {. . . }
    public void mouseMoved(MouseEvent e) {. . . }
}
```



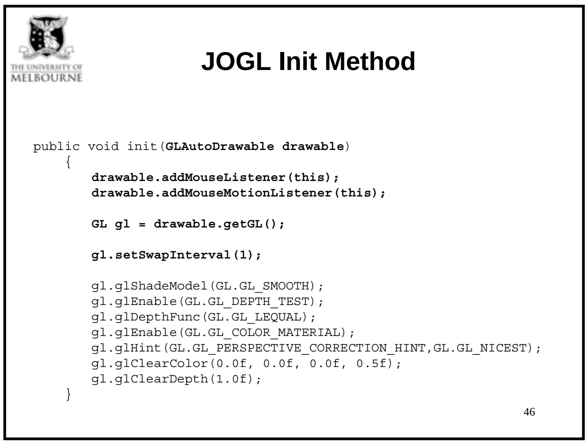
JOpenGLDemo Class

```
import java.awt.*;
import java.awt.event.*;

import javax.media.opengl.*;
import javax.media.opengl.glu.*;
import com.sun.opengl.util.*;

class JOpenGLDemo implements GLEventListener, MouseListener, MouseMotionListener
{
    private GLU glu = new GLU();

    private float fovy = 80.0f;
    private float aspect = 640.0f / 480.0f;
    private float zNear = 1.0f;
    private float zFar = 5000.0f;

    private float angle = 0.0f;

    public static void main(String[] args) { . . . }
    . . .
}
```



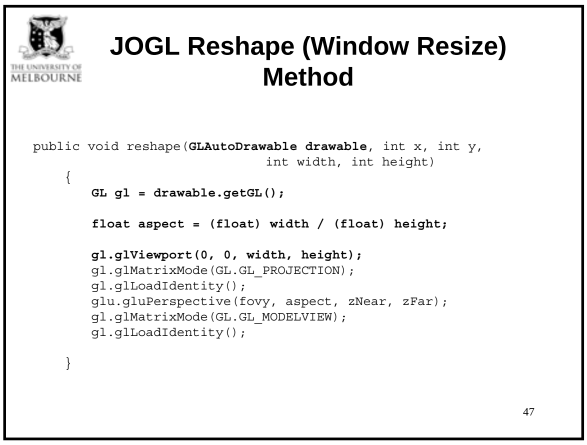
JOGL Main Method

```
public static void main(String[] args)
    {
        Frame frame = new Frame("433-380 JOGL Demo 1");
        GLCanvas canvas = new GLCanvas();
        canvas.addGLEventListener(new JOpenGLDemo());

        frame.add(canvas);
        frame.setSize(640, 480);

        final Animator animator = new Animator(canvas);
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                new Thread(new Runnable() {
                    public void run() {
                        animator.stop();
                        System.exit(0);
                    }
                }).start();
            }
        });

        frame.show();
        animator.start();
    }
```



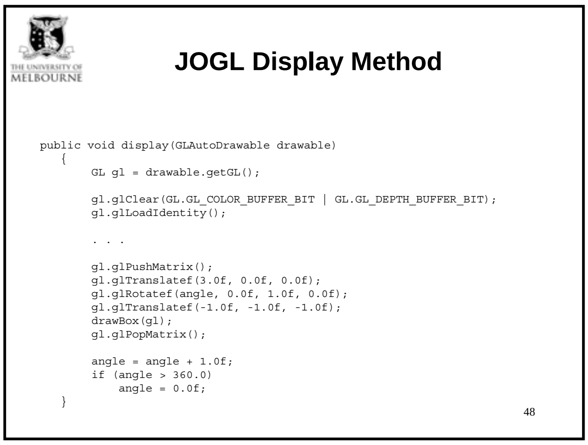
JOGL Init Method

```
public void init(GLAutoDrawable drawable)
    {
        drawable.addMouseListener(this);
        drawable.addMouseMotionListener(this);

        GL gl = drawable.getGL();

        gl.setSwapInterval(1);

        gl.glShadeModel(GL.GL_SMOOTH);
        gl.glEnable(GL.GL_DEPTH_TEST);
        gl.glDepthFunc(GL.GL_LEQUAL);
        gl.glEnable(GL.GL_COLOR_MATERIAL);
        gl.glHint(GL.GL_PERSPECTIVE_CORRECTION_HINT,GL.GL_NICEST);
        gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
        gl.glClearDepth(1.0f);
    }
```



JOGL Reshape (Window Resize) Method

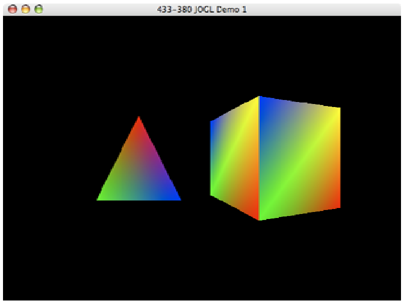
```
public void reshape(GLAutoDrawable drawable, int x, int y,
                           int width, int height)
    {
        GL gl = drawable.getGL();

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

        gl.glViewport(0, 0, width, height);
        gl.glMatrixMode(GL.GL_PROJECTION);
        gl.glLoadIdentity();
        glu.gluPerspective(fovy, aspect, zNear, zFar);
        gl.glMatrixMode(GL.GL_MODELVIEW);
        gl.glLoadIdentity();

    }
```



JOGL Display Method

```
public void display(GLAutoDrawable drawable)
    {
        GL gl = drawable.getGL();

        gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
        gl.glLoadIdentity();

        . . .

        gl.glPushMatrix();
        gl.glTranslatef(3.0f, 0.0f, 0.0f);
        gl.glRotatef(angle, 0.0f, 1.0f, 0.0f);
        gl.glTranslatef(-1.0f, -1.0f, -1.0f);
        drawBox(gl);
        gl.glPopMatrix();

        angle = angle + 1.0f;
        if (angle > 360.0)
            angle = 0.0f;
    }
```

JOGL Demo



JOGL on Mac OS X

- Precompiled JOGL for Mac OS X as Universal Binaries (PPC/Intel) available (based on March 06, 2006 snapshot):
  - http://homepage.mac.com/gziemski/projects/
- On Mac OS X copy .jar and .jnilib files into /Library/Java/Extensions
- Download the source to the demos, put in another directory and start with:
  - `java demos.gears.Gears`

OpenGL Books

OpenGL Programming Guide
OpenGL Reference Manual
OpenGL Programming for the X Window System
OpenGL Programming for Windows 95 and Windows NT
OpenGL Shading Language
OpenGL SUPERBIBLE

Resources

- OpenGL Home Page: www.opengl.org
- OpenGL Tutors http://www.xmission.com/~nate/tutors.html
- NeHe Tutorials: http://nehe.gamedev.net
- Game Programming Wiki OpenGL Tutorial: http://gpwiki.org/index.php/Category:OpenGL
- OpenGL Red Book (Programming Manual) http://www.opengl.org/documentation/red_book_1.0/
- OpenGL Blue Book (Reference Manual) http://www.opengl.org/documentation/blue_book_1.0/

Project Preparation

- Read Chapters 1-6 of OpenGL Red Book
- Familiarise yourself with OpenGL Blue Book
- Play with OpenGL Tutors
- Learn about JOGL
- Do NeHe Tutorial Lessons 1-5 (with JOGL)