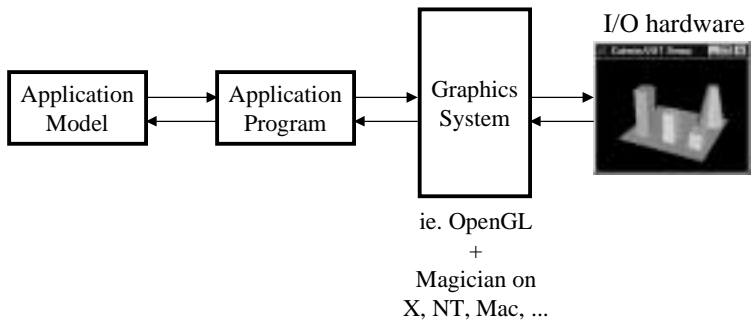


Handouts: Raster Graphics Hardware

Building Interactive Applications using OpenGL in Java using Magician

- Today: Magician
- Friday: OpenGL

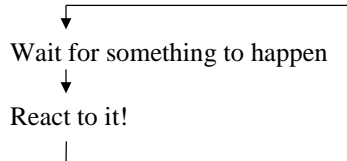
Conceptual Application Model



Handouts: Raster Graphics Hardware

Application Control Flow

- Interactive programs are *event driven*



- What are *events*?
 - User input
 - Window system, application generated

OpenGL

- 3D graphics library
 - Output only: render graphics
 - Only knows about "graphics contexts"
- Platform independent: No support for
 - Window creation
 - GLX
 - GLW
 - Input

Handouts: Raster Graphics Hardware

Magician: Java OpenGL bindings



- OS independent Java/OpenGL programs:
 - Multiple windows for OpenGL rendering.
 - Java AWT-style events.
 - Other features
 - Composable pipelines (debugging and profiling).
 - Support for animation and timers.
 - Utility routines to generate various objects (built on the GLUT toolkit).
 - Support for bitmap and stroke fonts.

Example Program:



```
import java.awt.*;
import com.hemetic.magician.*;

public class WhiteRectangle extends Frame implements GLEventListener {
    CoreGL gl_ = new CoreGL();
    GLComponent glc = null;

    public WhiteRectangle() {...}

    /** methods for GLEventListener */
    public void reshape(GLDrawable c, int x, int y, int width, int height) {...}
    public void initialize(GLDrawable component) {...}
    public void display(GLDrawable component) {...}
    public GL getGL() {...}

    /** typical main. Create a new "me" */
    public static void main(String argv[]) {
        WhiteRectangle t = new WhiteRectangle();
    }
}
```

Handouts: Raster Graphics Hardware

GLComponent display callback



```
/** called each time the screen needs to be redrawn */
public void display( GLDrawableComponent ) {
    /* clear the screen */
    gl.glClear( GL.GL_COLOR_BUFFER_BIT );

    /* set the current color */
    gl.glColor3f( 1.0f, 1.0f, 1.0f );

    /* draw a polygon */
    gl.glBegin( GL.GL_POLYGON );
    gl.glVertex3f( 0.25f, 0.25f, 0.0f );
    gl.glVertex3f( 0.75f, 0.25f, 0.0f );
    gl.glVertex3f( 0.75f, 0.75f, 0.0f );
    gl.glVertex3f( 0.25f, 0.75f, 0.0f );
    gl.glEnd();
}
```

Sample Constructor



```
public WhiteRectangle() {
    setLayout( new BorderLayout() );

    /* create a new component and add it to ourselves */
    glc = (GLComponent)GLDrawableFactory.createGLComponent( 200, 200 );
    add( "Center", glc ); pack(); show();

    /* specify the characteristics of the GLComponent */
    GLCapabilities cap = glc.getContext().getCapabilities();
    cap.setDoubleBuffered( GLCapabilities.DOUBLEBUFFER );
    cap.setDepthBits( 12 );
    cap.setColorBits( 24 );
    cap.setPixelFormat( GLCapabilities.RGBA );

    /* add this frame as a GL event listener, so we get OpenGL events */
    glc.addGLEventListener( this );

    /* go! */
    glc.initialize();
}
```

Handouts: Raster Graphics Hardware

Other GLComponent Methods

```
/** called once when componentGL context is initialized */
public void initialize( GLDrawable component ) {...}
    glClearColor( 0.0f, 0.0f, 0.0f, 1.0f );
}

/** called when window is resized */
public void reshape( GLDrawable component, int x, int y, int width, int height ) {
    glViewport( component, x, y, width, height );
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    glOrtho( 0.0, 1.0, 0.0, 1.0, -1.0, 1.0 );
    glMatrixMode( GL_MODELVIEW );
}

/** called by Magician to obtain OpenGL rendering context */
public GL getGL() {
    return gl;
}
```

Input

- Uses standard AWT event mechanisms

Handouts: Raster Graphics Hardware

Composable Pipelines

- CoreGL, CoreGLU
- TraceGL, TraceGLU
- ProfileGL, ProfileGLU
- ErrorGL, ErrorGLU

Assignment #1

- Hand out Friday, due Friday Sept. 1st
- Purpose:
 - learn how to write Java/OpenGL programs