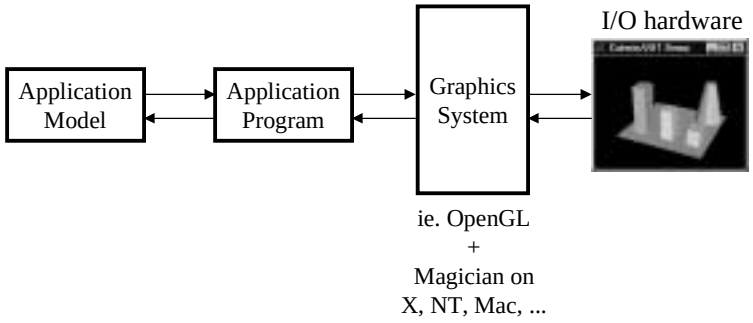
Handouts: Raster Graphics Hardware

Building Interactive Applications using OpenGL in Java using Magician

- Today: Magician
- Friday: OpenGL

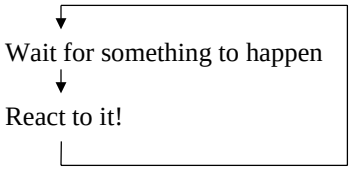
Conceptual Application Model



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

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.hermetica.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();
    }
}
```

GLComponent display callback

```
/** called each time the screen needs to be redrawn */
public void display(GLDrawable component) {
        /* 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 ourself */
        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.setColourBits(24);
        cap.setPixelType(GLCapabilities.RGBA);

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

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

Other GLComponent Methods

```
/** called once when component GL context is initialized */
public void initialize( GLDrawable component ) {...}
        gl_gl.ClearColor( 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 ) {
        gl_gl.Viewport( component, x, y, width, height );
        gl_gl.MatrixMode( GL.GL_PROJECTION );
        gl_gl.LoadIdentity();
        gl_gl.Ortho( 0.0, 1.0, 0.0, 1.0, -1.0, 1.0 );
        gl_gl.MatrixMode( GL.GL_MODELVIEW );
}

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

Input

- Uses standard AWT event mechanisms

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