

Using OpenGL in Java with JOGL

David Wolff (daw@plu.edu)

Department of Computer Science and Computer Engineering
Pacific Lutheran University

Consortium for Computing Sciences in Colleges, 2005

Outline

1 Introduction

2 Tutorial

- Installing JOGL
- Tutorial Files
- Example 1: Basic Framework
- Example 2: Hello JOGL
- Example 3: Swing and JOGL
- Example 4: Animation
- Example 5: Composable Pipelines
- Example 6: Textures
- Example 7: PBuffers
- Example 8: Multiple Canvases and Shared Data
- Example 9: Vertex Arrays
- Example 10: Screen Capture

3 Summary and Discussion

Java Bindings Under Active Development

Incomplete List

- LWJGL: Light Weight Java Game Library (<http://www.lwjgl.org>)
 - Focused on game development
 - Usually full-screen oriented
 - Minimal AWT/Swing integration
 - BSD style license.
- JOGL: (<http://jogl.dev.java.net>)
 - Good integration with AWT/Swing
 - Under development by employees at Sun
 - Will be used as basis for JSR-231 for integration into Java standard distribution (Java 6.0?)
 - Under BSD license.

Java Bindings Under Active Development

Incomplete List

- LWJGL: Light Weight Java Game Library (<http://www.lwjgl.org>)
 - Focused on game development
 - Usually full-screen oriented
 - Minimal AWT/Swing integration
 - BSD style license.
- JOGL: (<http://jogl.dev.java.net>)
 - Good integration with AWT/Swing
 - Under development by employees at Sun
 - Will be used as basis for JSR-231 for integration into Java standard distribution (Java 6.0?)
 - Under BSD license.

JSR-231

- **Java Community Process** (<http://jcp.org>)
- JSR: Java Specification Request
- JSR-231: Java bindings to OpenGL
(<http://www.jcp.org/en/jsr/detail?id=231>)
 - JOGL codebase recently forked to JSR-231.
 - Large and small API changes.
 - Package name moving to `javax.media.opengl`.
(Currently: `net.java.games.jogl`)
 - Future releases after JOGL 1.1.1 will be under the JSR-231 API.

JSR-231

- Java Community Process (<http://jcp.org>)
- JSR: Java Specification Request
- JSR-231: Java bindings to OpenGL
(<http://www.jcp.org/en/jsr/detail?id=231>)
 - JOGL codebase recently forked to JSR-231.
 - Large and small API changes.
 - Package name moving to `javax.media.opengl`.
(Currently: `net.java.games.jogl`)
 - Future releases after JOGL 1.1.1 will be under the JSR-231 API.

JSR-231

- Java Community Process (<http://jcp.org>)
- JSR: Java Specification Request
- JSR-231: Java bindings to OpenGL
(<http://www.jcp.org/en/jsr/detail?id=231>)
 - JOGL codebase recently forked to JSR-231.
 - Large and small API changes.
 - Package name moving to `javax.media.opengl`.
(Currently: `net.java.games.jogl`)
 - Future releases after JOGL 1.1.1 will be under the JSR-231 API.



Outline

- 1 Introduction
- 2 **Tutorial**
 - **Installing JOGL**
 - Tutorial Files
 - Example 1: Basic Framework
 - Example 2: Hello JOGL
 - Example 3: Swing and JOGL
 - Example 4: Animation
 - Example 5: Composable Pipelines
 - Example 6: Textures
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data
 - Example 9: Vertex Arrays
 - Example 10: Screen Capture
- 3 Summary and Discussion

The JOGL Components

- **Java class library: `jogl.jar` (platform independent)**
- JOGL shared native libraries (platform specific)
 - Windows: `libjogl.dll` and `libjogl_cg.dll`.
 - Linux/UNIX: `libjogl.so` and `libjogl_cg.so`
 - Mac OS X: `libjogl.jnilib` and `libjogl_cg.jnilib`
- Usage:
 - Place `jogl.jar` in `CLASSPATH`.
 - Place directory containing native libraries in system property `java.library.path`.
 - Specify `java.library.path` on command line, or
 - Place native libs in default location.



The JOGL Components

- Java class library: `jogl.jar` (platform independent)
- JOGL shared native libraries (platform specific)
 - Windows: `libjogl.dll` and `libjogl_cg.dll`.
 - Linux/UNIX: `libjogl.so` and `libjogl_cg.so`
 - Mac OS X: `libjogl.jnilib` and `libjogl_cg.jnilib`
- Useage:
 - Place `jogl.jar` in `CLASSPATH`.
 - Place directory containing native libraries in system property `java.library.path`.
 - Specify `java.library.path` on command line, or
 - Place native libs in default location.

The JOGL Components

- Java class library: `jogl.jar` (platform independent)
- JOGL shared native libraries (platform specific)
 - Windows: `libjogl.dll` and `libjogl_cg.dll`.
 - Linux/UNIX: `libjogl.so` and `libjogl_cg.so`
 - Mac OS X: `libjogl.jnilib` and `libjogl_cg.jnilib`
- Usage:
 - Place `jogl.jar` in **CLASSPATH**.
 - Place directory containing native libraries in system property `java.library.path`.
 - Specify `java.library.path` on command line, or
 - Place native libs in default location.

The JOGL Components

- Java class library: `jogl.jar` (platform independent)
- JOGL shared native libraries (platform specific)
 - Windows: `libjogl.dll` and `libjogl_cg.dll`.
 - Linux/UNIX: `libjogl.so` and `libjogl_cg.so`
 - Mac OS X: `libjogl.jnilib` and `libjogl_cg.jnilib`
- Usage:
 - Place `jogl.jar` in `CLASSPATH`.
 - Place directory containing native libraries in system property `java.library.path`.
 - Specify `java.library.path` on command line, or
 - Place native libs in default location.

JOGL Class Library

- Packages:
 - `net.java.games.jogl` : Core classes.
 - `net.java.games.jogl.util` : Buffer utilities and GLUT.
 - `net.java.games.cg` : Classes for use with Cg shading language.
 - `net.java.games.glugen.runtime` : Java-OpenGL JNI integration code.

JOGL Class Library

- Packages:
 - `net.java.games.jogl` : Core classes.
 - `net.java.games.jogl.util` : Buffer utilities and GLUT.
 - `net.java.games.cg` : Classes for use with Cg shading language.
 - `net.java.games.glugen.runtime` : Java-OpenGL JNI integration code.

JOGL Class Library

- Packages:
 - `net.java.games.jogl` : Core classes.
 - `net.java.games.jogl.util` : Buffer utilities and GLUT.
 - `net.java.games.cg` : Classes for use with Cg shading language.
 - `net.java.games.glugen.runtime` : Java-OpenGL JNI integration code.

JOGL Class Library

- Packages:
 - `net.java.games.jogl` : Core classes.
 - `net.java.games.jogl.util` : Buffer utilities and GLUT.
 - `net.java.games.cg` : Classes for use with Cg shading language.
 - `net.java.games.glugen.runtime` : Java–OpenGL JNI integration code.



Outline

- 1 Introduction
- 2 **Tutorial**
 - Installing JOGL
 - **Tutorial Files**
 - Example 1: Basic Framework
 - Example 2: Hello JOGL
 - Example 3: Swing and JOGL
 - Example 4: Animation
 - Example 5: Composable Pipelines
 - Example 6: Textures
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data
 - Example 9: Vertex Arrays
 - Example 10: Screen Capture
- 3 Summary and Discussion

Eclipse Project Files

- **Download zip file from:**
`http://www.cs.plu.edu/~dwolff/talks/jogl-ccsc`
- Import into Eclipse
- `jogl.jar` is included
- Execution: add to JVM command line args:
`-Djava.library.path=<path to native libs dir>`
- Pre-made Eclipse launch configs.

Eclipse Project Files

- Download zip file from:
`http://www.cs.plu.edu/~dwolff/talks/jogl-ccsc`
- Import into Eclipse
- `jogl.jar` is included
- Execution: add to JVM command line args:
`-Djava.library.path=<path to native libs dir>`
- Pre-made Eclipse launch configs.

Eclipse Project Files

- Download zip file from:
`http://www.cs.plu.edu/~dwolff/talks/jogl-ccsc`
- Import into Eclipse
- `jogl.jar` is included
- Execution: add to JVM command line args:
`-Djava.library.path=<path to native libs dir>`
- Pre-made Eclipse launch configs.



Eclipse Project Files

- Download zip file from:
`http://www.cs.plu.edu/~dwolff/talks/jogl-ccsc`
- Import into Eclipse
- `jogl.jar` is included
- Execution: add to JVM command line args:
`-Djava.library.path=<path to native libs dir>`
- Pre-made Eclipse launch configs.



Eclipse Project Files

- Download zip file from:
`http://www.cs.plu.edu/~dwolff/talks/jogl-ccsc`
- Import into Eclipse
- `jogl.jar` is included
- Execution: add to JVM command line args:
`-Djava.library.path=<path to native libs dir>`
- Pre-made Eclipse launch configs.



Outline

- 1 Introduction
- 2 **Tutorial**
 - Installing JOGL
 - Tutorial Files
 - **Example 1: Basic Framework**
 - Example 2: Hello JOGL
 - Example 3: Swing and JOGL
 - Example 4: Animation
 - Example 5: Composable Pipelines
 - Example 6: Textures
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data
 - Example 9: Vertex Arrays
 - Example 10: Screen Capture
- 3 Summary and Discussion

Example 1: Basic Framework

Interface GLDrawable

- Key implementing classes:
 - GLCanvas - “heavyweight” component. Subclass of `java.awt.Canvas`.
 - Some drawing problems when mixing with lightweight components.
 - GLJPanel - “lightweight” component. Subclass of `javax.swing.JPanel`
 - Currently implemented using PBuffers
 - Slower performance than GLCanvas
 - Expected to improve dramatically with JSR-231
- Selected methods:
 - `void addGLEventListener(GLEventListener)`
 - `GL getGL()`
 - `void setGL(GL gl)`
 - `boolean canCreateOffscreenDrawable()`
 - `GLPbuffer createOffscreenDrawable(...)`



Example 1: Basic Framework

Interface GLDrawable

- Key implementing classes:
 - GLCanvas - “heavyweight” component. Subclass of `java.awt.Canvas`.
 - Some drawing problems when mixing with lightweight components.
 - GLJPanel - “lightweight” component. Subclass of `javax.swing.JPanel`
 - Currently implemented using PBuffers
 - Slower performance than GLCanvas
 - Expected to improve dramatically with JSR-231
- Selected methods:
 - `void addGLEventListener(GLEventListener)`
 - `GL getGL()`
 - `void setGL(GL gl)`
 - `boolean canCreateOffscreenDrawable()`
 - `GLPbuffer createOffscreenDrawable(...)`



Interface GLDrawable

- Key implementing classes:
 - GLCanvas - “heavyweight” component. Subclass of `java.awt.Canvas`.
 - Some drawing problems when mixing with lightweight components.
 - GLJPanel - “lightweight” component. Subclass of `javax.swing.JPanel`
 - Currently implemented using PBuffers
 - Slower performance than GLCanvas
 - Expected to improve dramatically with JSR-231

- Selected methods:

- `void addGLEventListener(GLEventListener)`
- `GL getGL()`
- `void setGL(GL gl)`
- `boolean canCreateOffscreenDrawable()`
- `GLPbuffer createOffscreenDrawable(...)`



Interface GLDrawable

- **Key implementing classes:**
 - GLCanvas - “heavyweight” component. Subclass of `java.awt.Canvas`.
 - Some drawing problems when mixing with lightweight components.
 - GLJPanel - “lightweight” component. Subclass of `javax.swing.JPanel`
 - Currently implemented using PBuffers
 - Slower performance than GLCanvas
 - Expected to improve dramatically with JSR-231
- **Selected methods:**
 - `void addGLEventListener(GLEventListener)`
 - `GL getGL()`
 - `void setGL(GL gl)`
 - `boolean canCreateOffscreenDrawable()`
 - `GLPbuffer createOffscreenDrawable(...)`



Interface GLDrawable

- Key implementing classes:
 - GLCanvas - “heavyweight” component. Subclass of `java.awt.Canvas`.
 - Some drawing problems when mixing with lightweight components.
 - GLJPanel - “lightweight” component. Subclass of `javax.swing.JPanel`
 - Currently implemented using PBuffers
 - Slower performance than GLCanvas
 - Expected to improve dramatically with JSR-231
- Selected methods:
 - `void addGLEventListener(GLEventListener)`
 - `GL getGL()`
 - `void setGL(GL gl)`
 - `boolean canCreateOffscreenDrawable()`
 - `GLPbuffer createOffscreenDrawable(...)`



Example 1: Basic Framework

Interface GLDrawable

- Key implementing classes:
 - GLCanvas - “heavyweight” component. Subclass of `java.awt.Canvas`.
 - Some drawing problems when mixing with lightweight components.
 - GLJPanel - “lightweight” component. Subclass of `javax.swing.JPanel`
 - Currently implemented using PBuffers
 - Slower performance than GLCanvas
 - Expected to improve dramatically with JSR-231
- Selected methods:
 - `void addGLEventListener(GLEventListener)`
 - `GL getGL()`
 - `void setGL(GL gl)`
 - `boolean canCreateOffscreenDrawable()`
 - `GLPbuffer createOffscreenDrawable(...)`

Interface GLDrawable

- Key implementing classes:
 - GLCanvas - “heavyweight” component. Subclass of `java.awt.Canvas`.
 - Some drawing problems when mixing with lightweight components.
 - GLJPanel - “lightweight” component. Subclass of `javax.swing.JPanel`
 - Currently implemented using PBuffers
 - Slower performance than GLCanvas
 - Expected to improve dramatically with JSR-231
- Selected methods:
 - `void addGLEventListener(GLEventListener)`
 - `GL getGL()`
 - `void setGL(GL gl)`
 - `boolean canCreateOffscreenDrawable()`
 - `GLPbuffer createOffscreenDrawable(...)`



Creating a GLCanvas

- **Factory method from GLDrawableFactory class.**
 - `GLDrawableFactory.getFactory().createGLCanvas(capabilities)`
- Requires an instance of `GLCapabilities`.
 - Specifies a list of requested capabilities (e.g. hw acceleration, bit depth, etc.)
 - Will pick defaults appropriate for driver.
 - No guarantees.
 - Choose alternatives with `GLCapabilitiesChooser` object.



Creating a GLCanvas

- **Factory method from GLDrawableFactory class.**
 - `GLDrawableFactory.getFactory().createGLCanvas(capabilities)`
- **Requires an instance of GLCapabilities.**
 - Specifies a list of requested capabilities (e.g. hw acceleration, bit depth, etc.)
 - Will pick defaults appropriate for driver.
 - No guarantees.
 - Choose alternatives with `GLCapabilitiesChooser` object.



Creating a GLCanvas

- Factory method from `GLDrawableFactory` class.
 - `GLDrawableFactory.getFactory().createGLCanvas(capabilities)`
- Requires an instance of `GLCapabilities`.
 - Specifies a list of requested capabilities (e.g. hw acceleration, bit depth, etc.)
 - Will pick defaults appropriate for driver.
 - No guarantees.
 - Choose alternatives with `GLCapabilitiesChooser` object.



Creating a GLCanvas

- Factory method from `GLDrawableFactory` class.
 - `GLDrawableFactory.getFactory().createGLCanvas(capabilities)`
- Requires an instance of `GLCapabilities`.
 - Specifies a list of requested capabilities (e.g. hw acceleration, bit depth, etc.)
 - Will pick defaults appropriate for driver.
 - No guarantees.
 - Choose alternatives with `GLCapabilitiesChooser` object.

Creating a GLCanvas

- Factory method from `GLDrawableFactory` class.
 - `GLDrawableFactory.getFactory().createGLCanvas(capabilities)`
- Requires an instance of `GLCapabilities`.
 - Specifies a list of requested capabilities (e.g. hw acceleration, bit depth, etc.)
 - Will pick defaults appropriate for driver.
 - No guarantees.
 - Choose alternatives with `GLCapabilitiesChooser` object.



The GLEventListener Interface

- Follows the standard Java event listener paradigm.
Subinterface of `java.util.EventListener`.
- Methods
 - `public void init(GLDrawable d)`
 - Called once on creation.
 - `public void reshape(GLDrawable d, int x, int y, int width, int height)`
 - Called after `init` and on resize event. (x and y are always zero?).
 - `public void display(GLDrawable d)`
 - Called for each refresh.
 - `public void displayChanged(GLDrawable d, boolean modeChanged, boolean deviceChanged)`
 - Intended for handling multiple monitors, and for changing of resolution or bit depth
 - Currently unimplemented (likely to be removed).



The GLEventListener Interface

- Follows the standard Java event listener paradigm.
- Subinterface of `java.util.EventListener`.

● Methods

- `public void init(GLDrawable d)`
 - Called once on creation.
- `public void reshape(GLDrawable d, int x, int y, int width, int height)`
 - Called after `init` and on resize event. (x and y are always zero?).
- `public void display(GLDrawable d)`
 - Called for each refresh.
- `public void displayChanged(GLDrawable d, boolean modeChanged, boolean deviceChanged)`
 - Intended for handling multiple monitors, and for changing of resolution or bit depth
 - Currently unimplemented (likely to be removed).



The GLEventListener Interface

- Follows the standard Java event listener paradigm.

Subinterface of `java.util.EventListener`.

- Methods

- `public void init(GLDrawable d)`

- Called once on creation.

- `public void reshape(GLDrawable d, int x, int y, int width, int height)`

- Called after `init` and on resize event. (x and y are always zero?).

- `public void display(GLDrawable d)`

- Called for each refresh.

- `public void displayChanged(GLDrawable d, boolean modeChanged, boolean deviceChanged)`

- Intended for handling multiple monitors, and for changing of resolution or bit depth

- Currently unimplemented (likely to be removed).

The GLEventListener Interface

- Follows the standard Java event listener paradigm.

Subinterface of `java.util.EventListener`.

- Methods

- `public void init(GLDrawable d)`

- Called once on creation.

- `public void reshape(GLDrawable d, int x, int y, int width, int height)`

- Called after `init` and on resize event. (x and y are always zero?).

- `public void display(GLDrawable d)`

- Called for each refresh.

- `public void displayChanged(GLDrawable d, boolean modeChanged, boolean deviceChanged)`

- Intended for handling multiple monitors, and for changing of resolution or bit depth
- Currently unimplemented (likely to be removed).

The GLEventListener Interface

- Follows the standard Java event listener paradigm.
Subinterface of `java.util.EventListener`.
- Methods
 - `public void init(GLDrawable d)`
 - Called once on creation.
 - `public void reshape(GLDrawable d, int x, int y, int width, int height)`
 - Called after init and on resize event. (x and y are always zero?).
 - `public void display(GLDrawable d)`
 - Called for each refresh.
 - `public void displayChanged(GLDrawable d, boolean modeChanged, boolean deviceChanged)`
 - Intended for handling multiple monitors, and for changing of resolution or bit depth
 - Currently unimplemented (likely to be removed).

Outline

- 1 Introduction
- 2 Tutorial**
 - Installing JOGL
 - Tutorial Files
 - Example 1: Basic Framework
 - Example 2: Hello JOGL**
 - Example 3: Swing and JOGL
 - Example 4: Animation
 - Example 5: Composable Pipelines
 - Example 6: Textures
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data
 - Example 9: Vertex Arrays
 - Example 10: Screen Capture
- 3 Summary and Discussion

Calling GL Functions

- **First param for each GLEventListener method: GLDrawable.**
 - **The GLJPanel or GLCanvas.**
- GLDrawable provides access to GL and GLU objects.
- All GL functions available through GL object.
 - ```
GL gl = drawable.getGL();
gl.glPushMatrix();
gl.glBegin(GL.GL_TRIANGLES);
...
gl.glEnd();
gl.glPopMatrix();
```
  - Recommendation: always retrieve GL object, rather than store as instance variable.

# Calling GL Functions

- **First param for each GLEventListener method:** GLDrawable.
  - The GLJPanel or GLCanvas.
- **GLDrawable provides access to GL and GLU objects.**
- **All GL functions available through GL object.**
  - `GL gl = drawable.getGL();`  
`gl.glPushMatrix();`  
`gl.glBegin( GL.GL_TRIANGLES );`  
`...`  
`gl.glEnd();`  
`gl.glPopMatrix();`
  - Recommendation: always retrieve GL object, rather than store as instance variable.

# Calling GL Functions

- First param for each `GLEventListener` method: `GLDrawable`.
  - The `GLJPanel` or `GLCanvas`.
- `GLDrawable` provides access to GL and GLU objects.
- All GL functions available through GL object.
  - ```
GL gl = drawable.getGL();  
gl.glPushMatrix();  
gl.glBegin( GL.GL_TRIANGLES );  
...  
gl.glEnd();  
gl.glPopMatrix();
```
 - Recommendation: always retrieve GL object, rather than store as instance variable.

Calling GL Functions

- First param for each `GLEventListener` method: `GLDrawable`.
 - The `GLJPanel` or `GLCanvas`.
- `GLDrawable` provides access to GL and GLU objects.
- All GL functions available through GL object.
 - ```
GL gl = drawable.getGL();
gl.glPushMatrix();
gl.glBegin(GL.GL_TRIANGLES);
...
gl.glEnd();
gl.glPopMatrix();
```
  - Recommendation: always retrieve GL object, rather than store as instance variable.



# Calling GL Functions

- First param for each `GLEventListener` method: `GLDrawable`.
  - The `GLJPanel` or `GLCanvas`.
- `GLDrawable` provides access to GL and GLU objects.
- All GL functions available through GL object.
  - ```
GL gl = drawable.getGL();
gl.glPushMatrix();
gl.glBegin( GL.GL_TRIANGLES );
...
gl.glEnd();
gl.glPopMatrix();
```
 - Recommendation: always retrieve GL object, rather than store as instance variable.

Outline

- 1 Introduction
- 2 **Tutorial**
 - Installing JOGL
 - Tutorial Files
 - Example 1: Basic Framework
 - Example 2: Hello JOGL
 - **Example 3: Swing and JOGL**
 - Example 4: Animation
 - Example 5: Composable Pipelines
 - Example 6: Textures
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data
 - Example 9: Vertex Arrays
 - Example 10: Screen Capture
- 3 Summary and Discussion

Heavyweight component: GLCanvas

- Good performance compared to GLJPanel
- Integrates with Swing well in nearly all situations.
- Exceptions:
 - `JInternalFrames`
 - `JPopupMenu` and swing tooltips.

- Workarounds:

`JPopupMenu`:

- `setLightWeightPopupEnabled(false)`
- `setDefaultLightWeightPopupEnabled(false)`

`ToolTipManager`

- `setLightWeightPopupEnabled(false)`

Heavyweight component: GLCanvas

- Good performance compared to GLJPanel
- Integrates with Swing well in nearly all situations.
- Exceptions:

- `JInternalFrames`
- `JPopupMenu` and swing tooltips.

- Workarounds:

`JPopupMenu`:

- `setLightWeightPopupEnabled(false)`
- `setDefaultLightWeightPopupEnabled(false)`

`ToolTipManager`

- `setLightWeightPopupEnabled(false)`

Heavyweight component: GLCanvas

- Good performance compared to GLJPanel
- Integrates with Swing well in nearly all situations.
- Exceptions:
 - JInternalFrames
 - JPopupMenu and swing tooltips.

- Workarounds:

JPopupMenu:

- `setLightWeightPopupEnabled(false)`
- `setDefaultLightWeightPopupEnabled(false)`

ToolTipManager

- `setLightWeightPopupEnabled(false)`

Heavyweight component: GLCanvas

- Good performance compared to GLJPanel
- Integrates with Swing well in nearly all situations.
- Exceptions:
 - JInternalFrames
 - JPopupMenu and swing tooltips.

- Workarounds:

JPopupMenu:

- `setLightWeightPopupEnabled(false)`
- `setDefaultLightWeightPopupEnabled(false)`

ToolTipManager

- `setLightWeightPopupEnabled(false)`

Heavyweight component: GLCanvas

- Good performance compared to GLJPanel
- Integrates with Swing well in nearly all situations.
- Exceptions:
 - JInternalFrames
 - JPopupMenu and swing tooltips.
- Workarounds:
 - JPopupMenu:
 - `setLightWeightPopupEnabled(false)`
 - `setDefaultLightWeightPopupEnabled(false)`
 - ToolTipManager
 - `setLightWeightPopupEnabled(false)`

Capturing AWT Events

- GL context current only in `GLEventListener` methods.
 - GL/GLU objects should not be used outside.
- Store changes and retrieve on next call to `display()`
- Ways to refresh:
 - `GLDrawable.display()`: blocking.
 - `Container.repaint()`: non-blocking.
- Both methods available in `GLCanvas`.

Capturing AWT Events

- GL context current only in `GLEventListener` methods.
 - GL/GLU objects should not be used outside.
- Store changes and retrieve on next call to `display()`
- Ways to refresh:
 - `GLDrawable.display()`: blocking.
 - `Container.repaint()`: non-blocking.
- Both methods available in `GLCanvas`.

Capturing AWT Events

- GL context current only in `GLEventListener` methods.
 - GL/GLU objects should not be used outside.
- Store changes and retrieve on next call to `display()`
- Ways to refresh:
 - `GLDrawable.display()`: blocking.
 - `Container.repaint()`: non-blocking.
- Both methods available in `GLCanvas`.

Threading Issues

- All of the `GLEventListener` methods are executed on the AWT event dispatching thread.
- Future versions may change this.
- `setRenderingThread()` is currently a no-op.
- This is a change from previous versions in response to a variety of threading issues.

Outline

- 1 Introduction
- 2 **Tutorial**
 - Installing JOGL
 - Tutorial Files
 - Example 1: Basic Framework
 - Example 2: Hello JOGL
 - Example 3: Swing and JOGL
 - **Example 4: Animation**
 - Example 5: Composable Pipelines
 - Example 6: Textures
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data
 - Example 9: Vertex Arrays
 - Example 10: Screen Capture
- 3 Summary and Discussion

The Animator Class

- Causes continual calls to `display()`
- As fast as possible, no FPS limit.
- Give `GLDrawable` instance to `Animator` upon creation.
- Can be started and stopped repeatedly.
- Can not be started until `GLCanvas` is “realized”
- Starting at the end of `init()` seems to work well

A Clocked Animator: FPSAnimator

- Uses `java.util.Timer`
- Calls `GLDrawable.display()` every `x` microseconds based on `fps`.
- JOGL community's `FPSAnimator` possibly unstable.
 - Seems to fail when stopping and restarting on some OSs.

Outline

- 1 Introduction
- 2 **Tutorial**
 - Installing JOGL
 - Tutorial Files
 - Example 1: Basic Framework
 - Example 2: Hello JOGL
 - Example 3: Swing and JOGL
 - Example 4: Animation
 - **Example 5: Composable Pipelines**
 - Example 6: Textures
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data
 - Example 9: Vertex Arrays
 - Example 10: Screen Capture
- 3 Summary and Discussion

Composable Pipelines

- Add additional behaviors to GL pipeline.
- “Wrap” the GL object with new pipeline
 - `drawable.setGL(new <Pipeline>(drawable.getGL()))`
- Should be done at the beginning of `init()`

DebugGL and TraceGL

● DebugGL

- `drawable.setGL(new
DebugGL(drawable.getGL()))`
- Calls `glGetError()` after each OpenGL call
- Throws `GLException` when an error is found
- This is a distinct advantage over C

● TraceGL

- `drawable.setGL(new
TraceGL(drawable.getGL()))`
- Prints logging info after each OpenGL call.

DebugGL and TraceGL

- DebugGL

- `drawable.setGL(new
DebugGL(drawable.getGL()))`
- Calls `glGetError()` after each OpenGL call
- Throws `GLException` when an error is found
- This is a distinct advantage over C

- TraceGL

- `drawable.setGL(new
TraceGL(drawable.getGL()))`
- Prints logging info after each OpenGL call.

Outline

- 1 Introduction
- 2 **Tutorial**
 - Installing JOGL
 - Tutorial Files
 - Example 1: Basic Framework
 - Example 2: Hello JOGL
 - Example 3: Swing and JOGL
 - Example 4: Animation
 - Example 5: Composable Pipelines
 - **Example 6: Textures**
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data
 - Example 9: Vertex Arrays
 - Example 10: Screen Capture
- 3 Summary and Discussion

Using ImageIO For Loading Images

- `ImageIO.read(...)` returns `BufferedImage`
 - Supports jpg, png, gif
 - Plugins available for tga images.
- Slow?

Converting BufferedImage to OpenGL Format

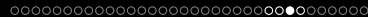
- `BufferedImage` provides access to pixel data in a variety of formats.
- JOGL prefers `java.nio` direct buffers.
- This example uses `ByteBuffer`:
 - `ByteBuffer.allocateDirect(nBytes);`
- Unpack pixels from `BufferedImage` and pack into `ByteBuffer`.
- `BufferedImage.getRGB(row, col)` returns `int` pixel in ARGB format.

Outline

- 1 Introduction
- 2 Tutorial**
 - Installing JOGL
 - Tutorial Files
 - Example 1: Basic Framework
 - Example 2: Hello JOGL
 - Example 3: Swing and JOGL
 - Example 4: Animation
 - Example 5: Composable Pipelines
 - Example 6: Textures
 - Example 7: PBuffers**
 - Example 8: Multiple Canvases and Shared Data
 - Example 9: Vertex Arrays
 - Example 10: Screen Capture
- 3 Summary and Discussion

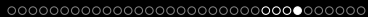
Creating an Offscreen Drawable

- Support depends on graphics card
- JOGL API is experimental, may change
- Check for support:
`drawable.createOffscreenDrawable()`
- `drawable.createOffscreenDrawable(caps, w, h)`
- Returns `GLPbuffer`
- The main drawable, and the Pbuffer may have separate `GLEventListeners`.
- Texture data and display lists are shared.



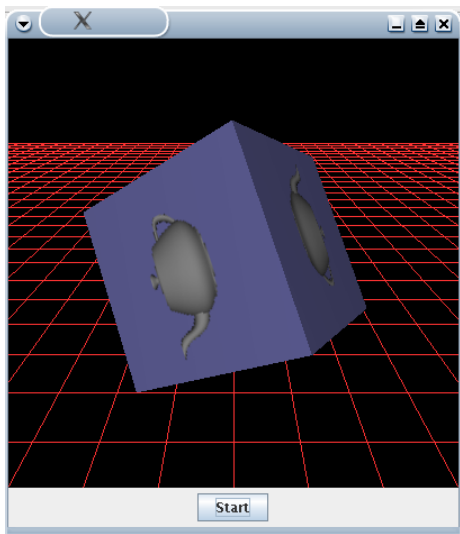
Using the PBuffer as a Texture

- Direct render to texture is supported in limited hardware.
- This example: render to Pbuffer, copy pixels to texture.
- At end of `display()` in Pbuffer:
`glCopyTexImage2D(...)`



Example 7: PBuffers

Screenshot



Outline

- 1 Introduction
- 2 Tutorial**
 - Installing JOGL
 - Tutorial Files
 - Example 1: Basic Framework
 - Example 2: Hello JOGL
 - Example 3: Swing and JOGL
 - Example 4: Animation
 - Example 5: Composable Pipelines
 - Example 6: Textures
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data**
 - Example 9: Vertex Arrays
 - Example 10: Screen Capture
- 3 Summary and Discussion

Creating Multiple Canvases with Shared Data

- Sharing of display lists, texture data, etc.
- Second parameter to `createGLCanvas()`, is canvas to share with.
 - `...createGLCanvas(caps, otherCanvas)`
- This example shows sharing of display list and two textures.

Outline

- 1 Introduction
- 2 **Tutorial**
 - Installing JOGL
 - Tutorial Files
 - Example 1: Basic Framework
 - Example 2: Hello JOGL
 - Example 3: Swing and JOGL
 - Example 4: Animation
 - Example 5: Composable Pipelines
 - Example 6: Textures
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data
 - **Example 9: Vertex Arrays**
 - Example 10: Screen Capture
- 3 Summary and Discussion

Using `java.nio` Buffers for Vertex Arrays

- Helper methods for creating buffers:
`net.java.games.jogl.util.BufferUtils`
- Vertex, normal arrays: `java.nio.DoubleBuffer`
- Index array: `java.nio.IntBuffer`
- Creation: `vertexBuffer =
BufferUtils.newDoubleBuffer(nDoubles)`
- Append triple: `vertexBuffer.put(vertex)`
- Give to OpenGL: `gl.glVertexPointer(3,
GL.GL_DOUBLE, 0, vertexBuffer)`



Example 10: Screen Capture

Outline

- 1 Introduction
- 2 **Tutorial**
 - Installing JOGL
 - Tutorial Files
 - Example 1: Basic Framework
 - Example 2: Hello JOGL
 - Example 3: Swing and JOGL
 - Example 4: Animation
 - Example 5: Composable Pipelines
 - Example 6: Textures
 - Example 7: PBuffers
 - Example 8: Multiple Canvases and Shared Data
 - Example 9: Vertex Arrays
 - **Example 10: Screen Capture**
- 3 Summary and Discussion

Example 10: Screen Capture

Retrieving the Frame Buffer Using GLReadPixels

- Copy frame buffer into `ByteBuffer`
 - Allocate buffer:

```
BufferUtils.newByteBuffer(w*h*3)
```
 - Copy pixels:

```
gl.glReadPixels(..., GL.GL_RGB,  
GL.GL_UNSIGNED_BYTE, ...)
```
- Move from buffer into `BufferedImage`
 - Swap bytes, pack into `int[]`
 - Set data in `BufferedImage`: `img.setRGB(...)`
- Save image to file using `ImageIO`:
 - `ImageIO.write(img, "PNG", file)`

Retrieving the Frame Buffer Using `GLReadPixels`

- Copy frame buffer into `ByteBuffer`
 - Allocate buffer:


```
BufferUtils.newByteBuffer(w*h*3)
```
 - Copy pixels:


```
gl.glReadPixels(..., GL.GL_RGB,  
GL.GL_UNSIGNED_BYTE, ...)
```
- Move from buffer into `BufferedImage`
 - Swap bytes, pack into `int[]`
 - Set data in `BufferedImage`: `img.setRGB(...)`
- Save image to file using `ImageIO`:
 - `ImageIO.write(img, "PNG", file)`

Retrieving the Frame Buffer Using GLReadPixels

- Copy frame buffer into ByteBuffer
 - Allocate buffer:
`BufferUtils.newByteBuffer(w*h*3)`
 - Copy pixels:
`gl.glReadPixels(..., GL.GL_RGB,
GL.GL_UNSIGNED_BYTE, ...)`
- Move from buffer into BufferedImage
 - Swap bytes, pack into int[]
 - Set data in BufferedImage: `img.setRGB(...)`
- Save image to file using ImageIO:
 - `ImageIO.write(img, "PNG", file)`

Java Web Start

- Deploy JOGL apps via web link.
- JNLP (Java Network Launch Protocol) file:
 - XML based
 - Point to jar file and JOGL libraries.
 - Pre-defined jnlp for JOGL libs hosted at:
`https://jogl.dev.java.net/webstart/jogl.jnlp`
 - Will select appropriate native libs.
 - Avoids security headaches (signed by Sun with VeriSign cert.)
- Web server must supply correct MIME type:
`application/x-java-jnlp-file`

Scenegraph Support



- Scenegraph: Xith 3D (<http://xith.org>)
 - Built on JOGL (or LWJGL)
 - Provides access to OpenGL commands
 - Alternative to Java3D, using similar structure
 - Includes 3ds loader.

Game Programming

- Quake 2 engine developed by Clark et al. for instructional purposes, written in JOGL. (JCSC V. (20) 2, December 2004)
- Full featured Quake 2 engine (open source) called Jake 2, developed using JOGL.
 - <http://www.bytonic.de>
 - Benchmarks: "Fast JOGL" 250 fps vs Original C code 315 fps.

Java in Intro. Graphics

● Pro

- Spend less time on C++ review, and more on graphics
- Can use the Java Collection classes instead of the STL
- No need for a windowing toolkit such as GLUT, Qt, etc.
- Graphics is difficult enough without C++

● Con

- C++ is industry standard for 3D graphics
- Performance
- It's "good for 'em"

Java in Intro. Graphics

- Pro

- Spend less time on C++ review, and more on graphics
- Can use the Java Collection classes instead of the STL
- No need for a windowing toolkit such as GLUT, Qt, etc.
- Graphics is difficult enough without C++

- Con

- C++ is industry standard for 3D graphics
- Performance
- It's "good for 'em"



JOGL in Intro. Computer Graphics at PLU

- Previous experiments: GLUT, Qt
- At PLU: CS1/2 are taught in Java.
- STL skills?
- Java Collection API: Familiar, no extra instruction
- Student job market preparation



Thanks!

- Tutorial materials:
<http://www.cs.plu.edu/~dwolff/talks/jogl-ccsc>
- Contact: daw@plu.edu
- Questions?