JOGL: An OpenGL Based Graphics Domain for PtolemyII

Yasemin Demir

JOGL Domain

- JOGL is a new experimental domain under development in PtolemyII.
  - Handles geometry and transformation of 3D objects which relies on OpenGL semantics.
  - Imports utilities, semantic and functionalities of OpenGL.
  - Reproduces the usability of OpenGL in an actor based platform by retaining the OpenGL semantics.
  - Avoids complicated task of keeping track of all the transformation matrices and save more space for other calculations using the matrix-stack specification

Why is JOGL domain useful?

- This domain will be used for implementations of real-time animation synthesis.
- The specification provides users to fully utilize the potential in the communication between the rendering engine and the computation in the model.
- This bottom-up design approach allows developers to create a event-based MoC that provides user, a sophisticated and flexible specification.

Ptolemy and JOGL Domain

- 3D computer graphics turn out to be a significant tool that allows users and developers more entertaining and visual platforms.
- Modern embedded systems also puts high demands on 3D simulations that increases the heterogeneity of embedded systems.
- Heterogeneity needs special environments that supports simulating each different subsystem and enabling the interaction between these subsystems.
- Here comes Ptolemy II: a good example for such environments that provides a broad range of computational models in an actor based platform.
- Ptolemy II allows users combine different models in different domains in the specified hierarchy.

The Goal

- The goal for this project is
  - To build a a user-friendly graphic domain in PtolemyII that provides a declarative specification for the user and allows real-time 3D simulations.
  - To achieve a reasonable computation efficiency by migrating computation from the model (CPU) onto GPU's.

A typical OpenGL Implementation

```
public class Complex implements GLEventListener {

public static void main(String[ ] args) {

 Frame frame = new Frame("Basic"); // create the frame
frame.setSize(500, 500); // give it a size
 GLCanvas canvas = new GLCanvas(); // create the canvas
Complex basic = new Complex(); // create the listener
 canvas.addGLEventListener(basic); // add listener to canvas
frame.add(canvas);//add canvas to window
frame.setVisible(true); // show window

 }
public void init(GLAutoDrawable drawable) {

GL gl = drawable.getGL();

}
```

A typical OpenGL Implementation

```
public class Basic implements GLEventListener {
public void display(GLAutoDrawable drawable) {
GL gl = drawable.getGL();
gl.glBegin(GL.GL_POLYGON); // draw a square
gl.glVertex2f(-0.5f, -0.5f);
gl.glVertex2f(-0.5f, 0.5f);
gl.glVertex2f(0.5f, 0.5f);
gl.glVertex2f(0.5f, -0.5f);
gl.glEnd();
}
```

Actors in JOGL Domain

- Point3D
- Line3D
- Triangle3D
- Box3D
- Sphere3D
- Translate3D
- Rotate3D
- Display3D

JOGL Actor Implementation

- Parameter setup
  - Line3D actor parameters
    - width
    - RgbColor
    - lineStart
    - lineEnd

JOGL Actor Implementation

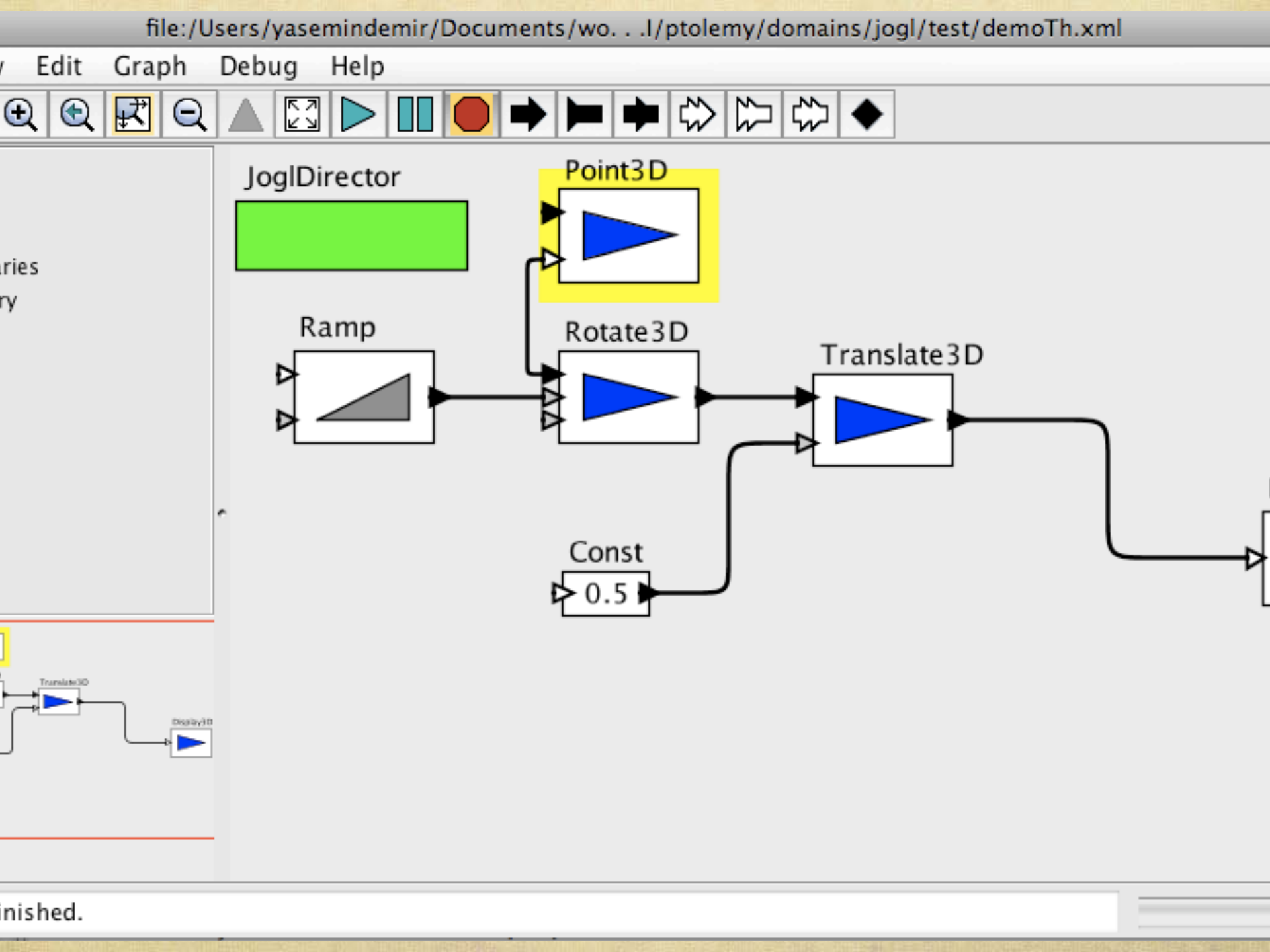
- fire() method implementation
  - Line3D Illegal action exception control mechanism

```
if (input.hasToken(0)) {
        ObjectToken inputToken = (ObjectToken)input.get(0);
        Object inputObject = inputToken.getValue();
        if (!(inputObject instanceof GL)) {
            throw new IllegalActionException(this,
                    "Input is required to be an instance of GL. Got "
                    + inputObject.getClass());
        }
```

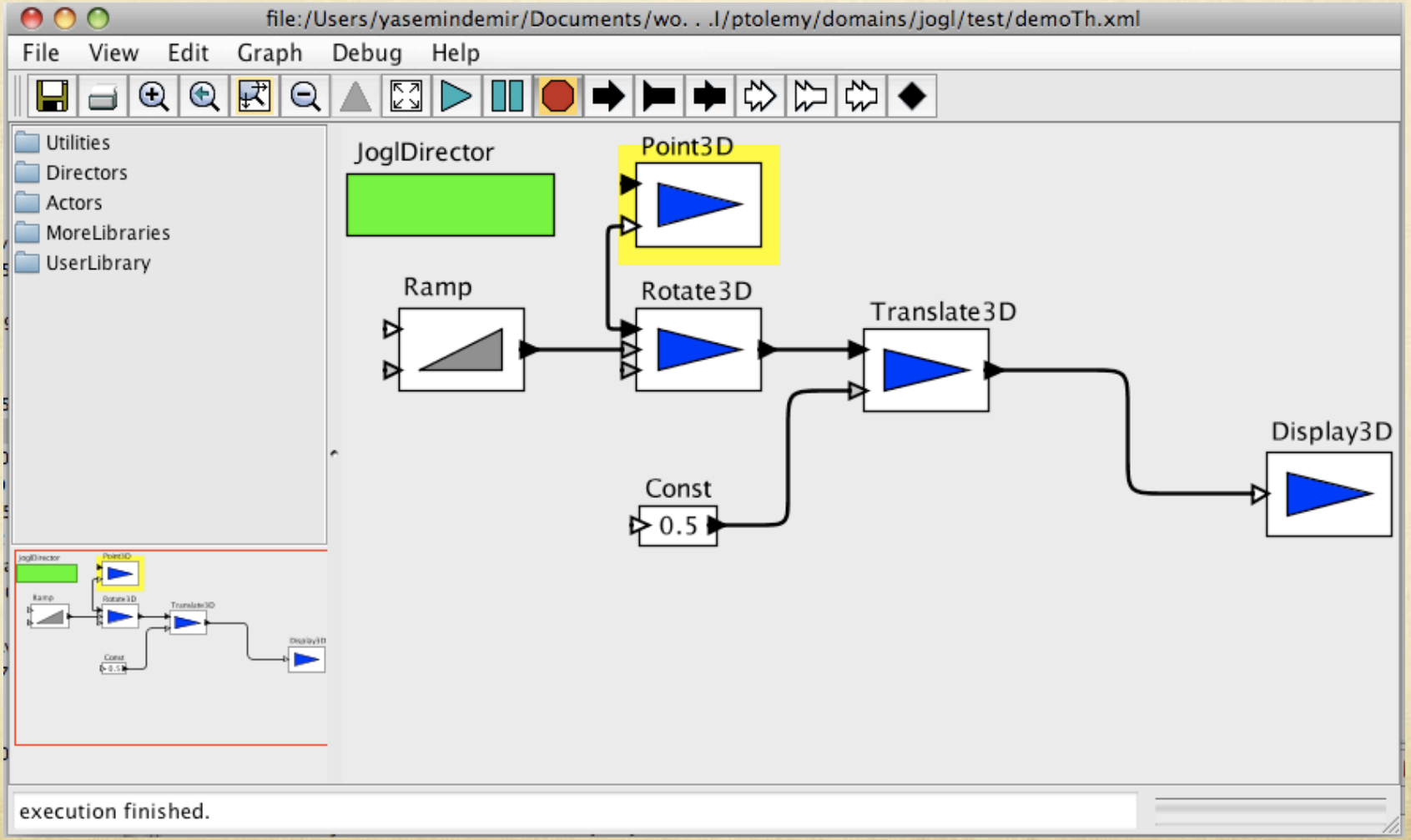
JOGL Actor Implementation

- Line3D fire() method implementation

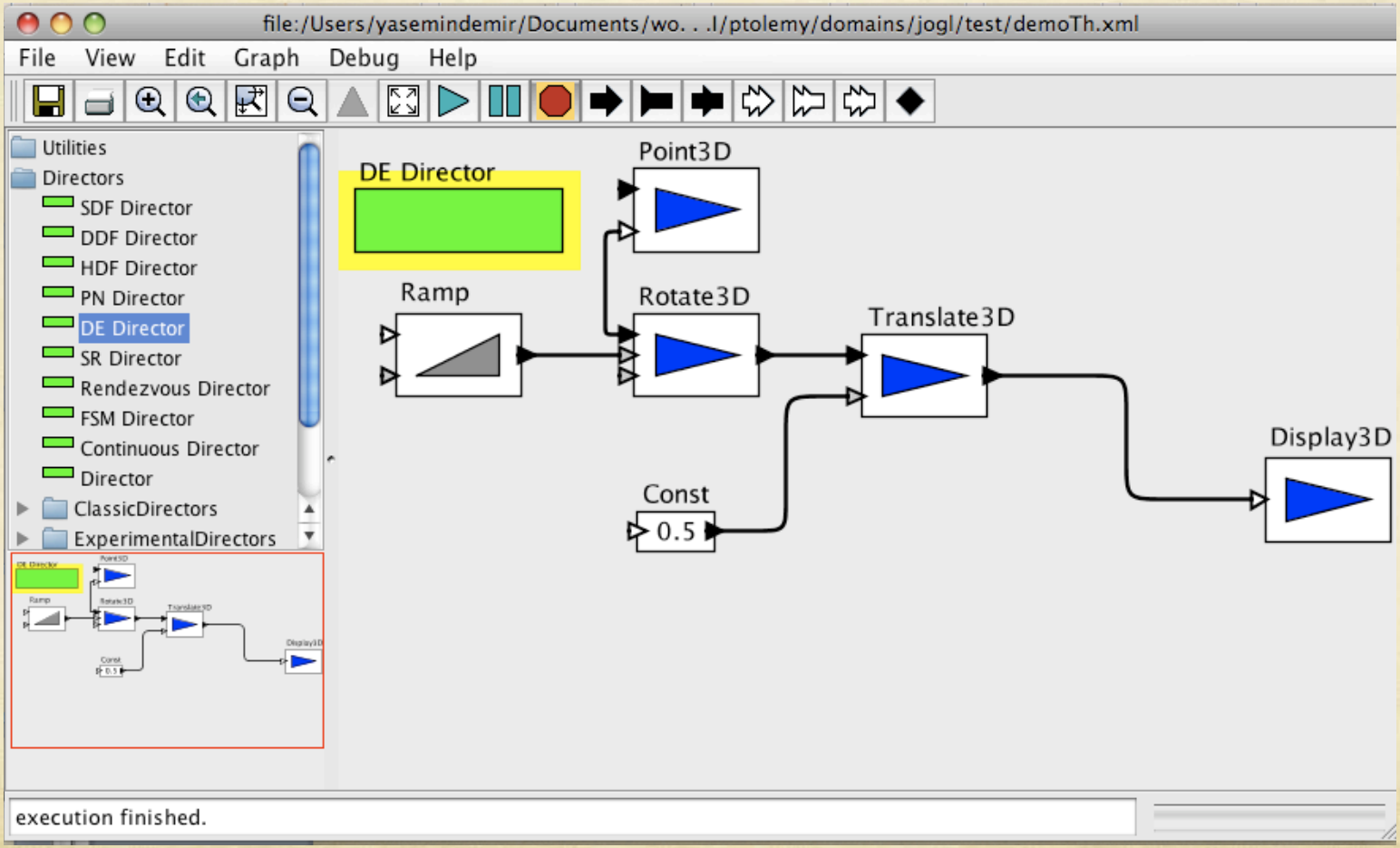
```
GL gl = (GL)inputObject;

    ArrayToken lineStartToken = ((ArrayToken) lineStart.getToken());
    ArrayToken lineEndToken = ((ArrayToken) lineEnd.getToken());
    ArrayToken rgbColorValue = ((ArrayToken) rgbColor.getToken());
    DoubleToken widthValue = (DoubleToken) width.getToken();
    gl.glLineWidth((float) widthValue.doubleValue());
    gl.glBegin(GL.GL_LINES);
    gl.glColor3d(
            ((DoubleToken) rgbColorValue.getElement(0)).doubleValue(),
            ((DoubleToken) rgbColorValue.getElement(1)).doubleValue(),
            ((DoubleToken) rgbColorValue.getElement(2)).doubleValue());
    // origin of the line
    gl.glVertex3d(
            ((DoubleToken) lineStartToken.getElement(0)).doubleValue(),
            ((DoubleToken) lineStartToken.getElement(1)).doubleValue(),
            ((DoubleToken) lineStartToken.getElement(2)).doubleValue());
    // ending point of the line
    gl.glVertex3d(
            ((DoubleToken) lineEndToken.getElement(0)).doubleValue(),
            ((DoubleToken) lineEndToken.getElement(1)).doubleValue(),
            ((DoubleToken) lineEndToken.getElement(2)).doubleValue());
    gl.glEnd( );
```


file:/Users/yasemindemir/Documents/wo. . .l/ptolemy/domains/jogl/test/demoTh.xml
Edit
Graph
Debug
Help
JoglDirector
Point3D
Ramp
Rotate3D
Translate3D
Const
0.5
inished.

A Simple Implementation in PtolemyII



A Simple Implementation in PtolemyII



An Experimental System

- Using OptiTrack System
  - Triangulate the 3D position of markers
  - Passive Marker (Retroreflective Markers)
- Build human interactive environments