NASA Ames uses Eclipse RCP for real-time situational awareness of remote robots.

Tamar Cohen
Intelligent Robotics Group
NASA Ames Research Center
NASA Ames Research Center
Intelligent Robotics Group
presents

VERVE

a Visual Environment for Remote Virtual Exploration

featuring the K10 Rover at Haughton Crater, Devon Island, Canada
Summer 2010

Tamar Cohen
Intelligent Robotics Group
NASA Ames Research Center
NASA Ames uses Eclipse RCP for real-time situational awareness of remote robots
The Fun Parts: How we did it

This presentation will explore implementation details of the following:

• Java 3D View based on Ardor3D
• Communicating with robots
• Complex KML loading based on EMF
• Generating UI components with the databinding framework
• Real-time telemetry display
The “VERVE 3D View” is an Eclipse view with contents rendered by the Java Ardor3D libraries which we have wrapped in a plug-in.

Ardor3D provides user interface support within the view, eg the compass rose.

Ardor3D provides hooks for mouse control and keyboard input, along with typical controls for a 3D graphics library (ie cameras, scene, lighting, etc)

A model of K10Black
Robots in the Scene Graph

Ardor3D scene graph is comprised of **Spatials**, which define geometry and rendering settings

**Nodes**, which are spatials with parents and children

For each type and instance of a robot, we create a RobotNode (extends Node) which contains nodes that represent its model (3D object) and child nodes for representation of scientific data

We have a reference between our conceptual representation of a robot and each of its parts, and the Ardor3D nodes which represent each concept.
The Scene Graph Tree View

We have a standard Eclipse view which includes a tree populated with the contents of the scene graph. This tree can be extremely deep.

Since elements are dynamically added to and removed from the 3D scene, this tree is actually populated with WeakReferences, to support garbage collection.

When various events happen, the tree refreshes asynchronously.

Checkboxes show and hide 3D models in the Ardor3D view.
Connecting the Canvas to Ardor3D

Within our createPartControl for the VERVE 3D View, we make explicit calls to use Ardor3D to render the contents of the canvas

```java
public void createPartControl(Composite parent) {
    VerveScene scene = getScene(getSceneName()); // an Ardor3D scene graph
    LwjglCanvasRenderer canvasRenderer = new LwjglCanvasRenderer(scene); // an LWJGL Ardor3D renderer
    final GLData glData = new GLData(); // SWT class
    m_a3dCanvas = new Ardor3dCanvas(canvasRenderer, parent, SWT.NONE, glData);
    // Ardor3dCanvas extends GLCanvas implements com.ardor3d.framework.Canvas, which defines the Ardor3D “View,” owns the rendering phase, controls all interactions with renderer
    Ardor3D.getFrameHandler().addCanvas(m_a3dCanvas);
    // get Ardor3D frame handler from singleton to do the work needed in a given frame
    Ardor3D.getLogicalLayerUpdater().registerLogicalLayer(m_a3dCanvas.getLogicalLayer());
    // register for triggers for updates to scene
    scene.addScenePickListener(this);
    // our own construct for pick delegation handling – we set the camera to look at what was picked
}
```
We have data coming in too fast to render, so we throttle that back:

```java
Timer timer = new Timer(); // an Ardor3D timer which uses System.nanoTime
FrameHandler frameHandler = new FrameHandler(timer);
frameHandler.addUpdater(new LogicalLayerUpdater());
frameHandler.addUpdater(new CameraControlUpdater()); // we expose camera controls in the Eclipse UI
RenderUpdateThread renderUpdateThread = new RenderUpdateThread(applicationPlugin, frameHandler, timer);
renderUpdateThread.start();
```

Our very simple render update thread runs as follows:

Once the system and display are ready (until the display is disposed)

Check the time that the last change occurred;

if the elapsed time is enough, asynchronously run an update

```java
frameHandler.updateFrame();
```
Communicating With Robots

We use CORBA (Common Object Request Broker Architecture, a standard) to communicate with our robots. While we could command the robots, we primarily monitor them from VERVE.

CORBA uses IDL (Interface Definition Language) to express classes and interfaces. Via IDL, we have compatible classes passed between the robots and our Java classes.

CORBA uses ORBs (Object Request Brokers) to connect to and communicate with the robots.

We expose CORBA settings through a preferences page, and have a CORBA status view to support reconnecting and showing status.

With CORBA we can communicate with more than one robot at a time.

**CORBA Preferences**

**CORBA Channel**

<table>
<thead>
<tr>
<th>K10Black Naming Context:</th>
<th>K10Black</th>
</tr>
</thead>
<tbody>
<tr>
<td>K10Black EventChannel:</td>
<td>EventChannel</td>
</tr>
<tr>
<td>K10Red Naming Context:</td>
<td></td>
</tr>
<tr>
<td>K10Red EventChannel:</td>
<td></td>
</tr>
</tbody>
</table>

*Multiple CORBA channels*
Managing CORBA Events

CORBA Events come over the CORBA channel.

We have enumerated the types of events we expect (KnEvent)

```java
public class KnEvent {
    public enum Type {
        Battery ("Battery.SBattery", Sbattery.class, SbatteryHelper.class),
        PoseEstimate ("Pose.SPoseEstimate", SPoseEstimate.class, SPoseEstimateHelper.class);
    
    public String type_name;
    public final Class dataType;
    public final Class eventHelper;
    ...
}
```

We have listeners for our events.

```java
public interface KnEventListener {
    public void knEventReceived(KnRover.Name source, KnEvent.Type type, Object msg, String type_name);
}
```

We have an event collector for each robot

```java
collector = new KnEventCollectorAsapQueue(rover);
Orb.addListener(collector);
```

We expose the possible events to subscribe to in the preferences.

---

NASA Ames uses Eclipse RCP for real-time situational awareness of remote robots
Example: connect events coming in with transformations for 3D nodes.

```java
public class K10PoseProvider implements IPoseProvider, KnEventListener {
    final KnRover.Name m_knRoverName;
    Vector3 m_xyz = new Vector3();
    Vector3 m_rpy = new Vector3();

    public void connectToTelemetry() {
        KnEventCollector telemetry = KnEventCollector.instance(m_knRoverName);
        telemetry.addListener(this, new KnEvent.Type[] { KnEvent.Type.PoseEstimate });
    }

    // when a new event is received, handle it; cache the latest information
    public synchronized void knEventReceived(KnRover.Name source, KnEvent.Type type, Object msg) {
        switch (type) {
            case PoseEstimate: m_poseEstimate = (SPoseEstimate)msg; break;
            ...
        }
    }

    // When it comes time to update the frame, our conceptual representation of a robot part will get the
    // latest information from the provider, and use it to update the 3D Model if it is visible and dirty.
    public abstract class K10PartOrientationSensor extends AbstractRobotPart {
        public void handleFrameUpdate(long currentTime) {
            if(isActive()) {
                getK10Robot().getK10PoseProvider().getCenterXyz(m_xyz);
                m_ring.setTranslation(m_xyz);
                if(isDirty()) {
                    synchronized(m_rpy) {
                        m_ring.setRotation(K10PoseProvider.rotFromRpy(m_rpy, m_rot));
                    }
                    setDirty(false);
                }
            }
        }
    }
```
Loading KML Files

KML is an XML file format to store geographic data used by Google Earth. Our scientists are used to working with KML; we support it for base mapping and markup.

We have created an EMF (Eclipse Modeling Framework) representation of the KML format. After a file is loaded, we construct the Ardor3D spatial.

KML includes network links, which reference external or remote files and reload at specified intervals. We support this, updating our 3D geometry based on newly loaded KML.
VERVE is designed to support different robots, which will have different models and instrumentation.

We generate user interface controls bound to these models based the Java classes representing robot concepts.

We use reflection to explore classes.

```java
for (Method method : m_class.getMethods()){
    if (MethodUtil.isGetMethod(method) || MethodUtil.isIsMethod(method)){
        createField(method, container);
    }
}
```

We have created custom Java annotations to control generated databound widgets, for example:

```java
@ReadOnly
public Vector3 getVector3() {
    return m_vector3;
}
```

Changes made with these widgets will automatically be reflected in the Ardor3D view.
Huge amounts of telemetry are constantly streaming in from the robots.

VERVE presents this information in a clear way, so users can quickly understand the information and know where to look for problems.

Warning and error thresholds and colors are set for each type of telemetry.
We have created a `TableBarListener` to handle drawing informative color blocks in table cells

```java
public class TableBarListener implements Listener {
    public void handleEvent(Event event) {
        GC gc = event.gc;

        // use reflection to get the percentage value for the data for the cell
        percent = (Number) getPercentMethod().invoke(getMethodContainer(), item.getData());

        // make the string pretty
        String percentString = getFormatter().format(percent);

        // look up the color based on thresholds set for this table bar listener
        Color barColor = getBarColor(percent);

        // figure out how wide the bar should be
        int width = (getBarColumn().getWidth() - 2) * (percent.intValue()) / m_spread;

        // use the graphics context to fill the rectangle
        gc.fillRectangle(event.x-1, event.y, width, event.height-2);

        // draw the text of the percentage
        gc.drawText(percentString, event.x+2, event.y+offset, true);
    }
}
```
VERVE for Haughton Mars Project (HMP) 2010
http://www.youtube.com/watch?v=bA8GkyK0ZpQ

Intelligent Robotics Group at NASA Ames Research Center
http://irg.arc.nasa.gov

Blog about HMP 2010
http://lunarscience.nasa.gov/robots/2010/

HMP
http://www.marsonearth.org/

Ardor3D
http://ardor3d.com
Questions?

Intelligent Robotics Group
Intelligent Systems Division
NASA Ames Research Center

http://irg.arc.nasa.gov