Toward Implementation of Java/VRML Environment for Planning, Training and Tele-Operation of Robotic Systems

Kaveh E. Afshari and Shahram Payandeh

Experimental Robotics Laboratory (ERL)
School of Engineering Science, **S**imon **F**raser **U**niversity
Burnaby, B.C., V5A 1S6, Canada

ABSTRACT

This paper presents an approach toward implementing Java and VRML environments for training and Tele-operation of Robotic Systems. Three methods were explored for such experiments, for communicating and message passing between Java and VRML programs. The first method includes routing approach. Using VRML script, all the nodes of the virtual world are routed to the Java applet. The Java program then controls the nodes in order to model a robot. In the second method all the VRML objects are made in the Java applet, and then the whole virtual world is sent as text to the VRML environment. The third method uses a pre-structured template of a generic robot. All the nodes of the Robot are defined with DEF command, and they are accessed by Java though the browser. After gaining access to the nodes, Java has full control over the nodes, and can customize the robot. Then the robot can be used as a virtual model, and can be controlled as a graphical user interface of a real robot.

Keywords: web-based application, Tele-robotics, Java, VRML, External Authoring Interface (EAI),

1. INTRODUCTION

Since the gradual birth of the first Robots, Tele-operation has been evolving to an evermore significant subject of robotic control systems. As the robotics technology enhances, sophisticated robots (multi-robots) are becoming more challenging to coordinate and control (Tele-controlled).

Universal Kinematics Visualization Training Tools (UKVT) for purpose of planning, controlling and defining models of robots is slowly growing popularity in robotics community.

This paper presents a web-based approach for defining, interfacing and further controlling robotic manipulating systems. Other related works can be summarized as following.

At the Universita della Calabria (Italy), L. Nigro et.al. [1] have proposed a modular approach to real-time programming using actors and Java. They proposed DART, an actor-based architecture for real-time applications, which has been designed to be exploitable in popular object oriented languages.

Also at the Universita di Pisa, S. Piccinocchi et.al. [2] have presented an interactive benchmark for planning Algorithms on the web. They present an interactive environment available on the World Wide Web intended to allow fair and thorough comparison of different techniques to solve a basic problem in non-holonomic motion planning.

At the University of Western Australia, K. Taylor et.al. [3] have demonstrated remote operation and programming of an industrial robot by linking remote users to an industrial six- axis robot through the World Wide Web which is part of the Internet computer network. The remote user receives still video images of the robot's work space and can move its gripper in x, y, z, and roll pitch and yaw directions. At the Agency of Industrial Science and Technology, H. Hirukawa et.al. [4] have been developing a prototype of a Tele-operation system via a standard protocol with a standard human interface, where an enhanced VRML is employed as the protocol and a Web browser as the human interface.

This paper presents an application of VRML/Java to the design of interactive web-based robotics. It is arranged as follows: Section 2 presents an overview of web-based application, and introduces some universal 3D Graphics environments. Section 3 is about visualization of robotic mechanism, which mentions some advantages of VRML and a brief history of VRML, and section 3.1.1 and 3.1.2 introduce VRML script and EAI respectively. Then we move to interfacing VRML with Java (section 4), and we describe some different methods of approaching this goal. Finally we show an example (section 5) and we present our future plans (section 6).

2. WEB BASED APPLICATION

Remote Tele-Operation requires a reliable, fast and secure network connection. One of the approaches, which may offer such connection, is 'The Internet'. Although such connection has its general disadvantages, i.e. low bandwidth and variable time-delay for some

applications and security, but it seems like the web-based application will have more advantages than many other network connections,
Using a web-based environment requires *platform independent* software or *server based applications*.

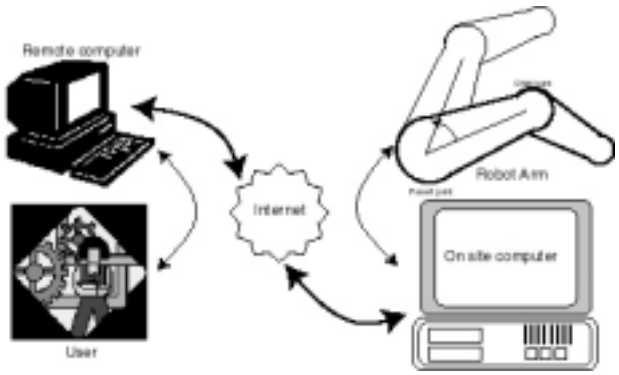

Figure 1-Remote Tele Operation via Internet

An example of a system with Internet connection is shown in Figure 1.
In this figure an onsite computer (server) can be connected to the Robot arm. The server has access to the Robot I/O ports. The user can login to the control server from any computer via the internet. The web based control software is loaded at the user's computer, and the user can control the robot arm. A 3D model of the robot enables the user to visualize and control the motion of the arm.

2.1 Graphics

Visualizing a simple world can be very difficult in a 2D projecting monitor. 3D graphic tools that have been developed and are being enhanced every day, create a more realistic interface. Some of these environments are more dedicated to high precision and dedicated modeling: ACAD and SolidWorks, while some other are more graphics oriented, like: VRML, OpenGL.
High precision 3D modeling and design software are usually more expensive, while some graphics oriented environments are available through the web free of cost. VRML and OpenGL are two examples of these kinds. Yet OpenGL is a C++ graphics library, and therefore the final application would be platform dependent.
VRML is platform independent, and it runs usually as a plug-in to standard web browsers (Microsoft Internet Explorer, and Netscape Navigator). The plug-in is available for most systems, and can be downloaded from the Internet for free. The two most widely used plug-ins are Cosmoplayer (http://www.cosmosoftware.com/), and Blaxxun Ccpro (http://www.blaxxun.com/). Both support Microsoft Internet Explorer and Netscape Navigator, and they have common standard functionality, but different interface. They both have some non-standard features that add to their capabilities.
Another platform independent environment is Java 3D. The Java 3D™ 1.1 API is a set of classes for writing three-dimensional graphics applications and 3D applets. It gives developers high level constructs for creating and manipulating 3D geometry and for constructing the structures used in rendering that geometry. Application developers can describe very large virtual worlds using these constructs, that provide Java 3D with enough information to render these worlds efficiently [5].

3. VISUALIZATION OF ROBOTIC MECHANISM

Most robot arms are multi-chained, and they have several different joints and links. Revolute and Prismatic joints are just two of the simpler and more practical representations. In this representation, one joint is the parent and the other joints are its children. So any motion in the parent joint will affect the children respectively. So for example if the first joint rotates an angle (θ) the coordinate system of all the children will change (Figure 2). Similarly if the first joint translates by a vector $X=(x,y,z)^T$, all the coordinate systems of the children will be changed accordingly.

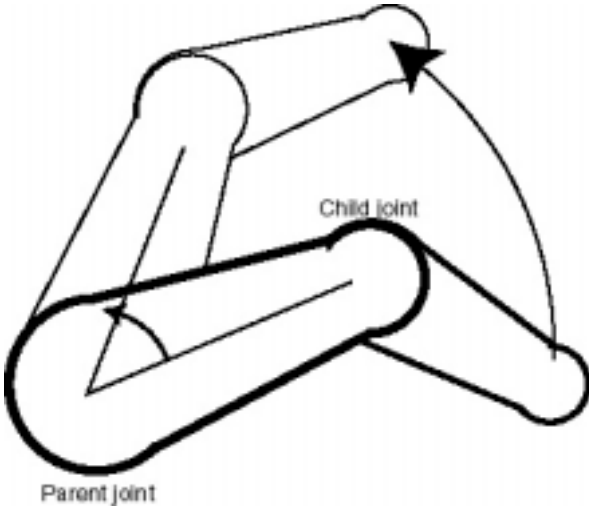

Figure 2-Two joint arm

Consequently, it is beneficial that the modeling system be able to perform these calculations implicitly. Since VRML already has parent-child *Nodes* (Listing 1) built in the infrastructure, designing a robot with real-life features is facilitated.

Listing 1

```
DEF ROBOT_ARM Group
{
children
[
DEF JOINT_1 Transform
{
   translation x1 y1 z1
   rotation x1 y1 z1 θ1
   children
   [
   USE JOINT_1_SHAPE
```

```
      USE LINK_1_TRANSFORM
      DEF JOINT_2 Transform
      {
        translation x2 y2 z2
        rotation x2 y2 z2 θ2
        children
        [
        USE JOINT_2_SHAPE
        USE LINK_2_TRANSFORM
        ]
      }
      ]
  }
  ]
  }
```

In the above code, JOINT_1 is parent with respect to JOINT_2; therefore, any transformation (rotation and translation for our case) on JOINT_1 automatically transforms the coordinate system of JOINT_2.

3.1 About VRML

Virtual Reality Modeling Language (VRML) is an international standard (ISO/IEC 14772) file format for describing interactive 3D multimedia on the Internet [6]. VRML 1.0 was developed at Silicon Graphics based on *Open Inventor*. After reviews by VRML moderated email discussion group (www-vrml@vrml.org), VRML 2.0 was released.

VRML97 is the informal name of the International Standard (ISO/IEC 14772-1:1997). It is almost identical to VRML 2.0, but with many editorial improvements to the document and a few minor functional differences. VRML97 is also the name of the VRML Technical Symposium that took place in February 1997 in Monterey, CA [7].

Web3D is the new generation of these modeling languages. It combines a run-time delivery engine and VRML 97-inspired file format with XML bindings, with strong industry support for a proven standardization process [7]. While all of these modeling languages are backward compatible, they add numerous advantages for the programmer. VRML now supports internal scripting as well as External Authoring Interface.

3.1.1 VRML Script

3D modeling is just the base of VRML. This language supports internal scripting, which makes the virtual world interactive. VRML script provides a great environment for non-scientific virtual worlds, but due to its limitations, it is not a great candidate for programming control systems.

3.1.2 JAVA/VRML External Authoring Interface (EAI)

Another way of controlling the VRML world is through the External Authoring Interface. Using VRML libraries, we can control our virtual world with a programming language. And the most commonly used language, which is multi-platform that can also interface with VRML, is JAVA.

4. INTERFACING VRML WITH JAVA

There are a number of methods that can be followed for developing interactive interface with VRML representation of a robotic system.

4.1 Method 1

One way of implementing this idea is to use *Routing*. With a small VRML-script, a Java class can be called and specific nodes in the pre-coded virtual world are routed to the Java program (Figure 3).

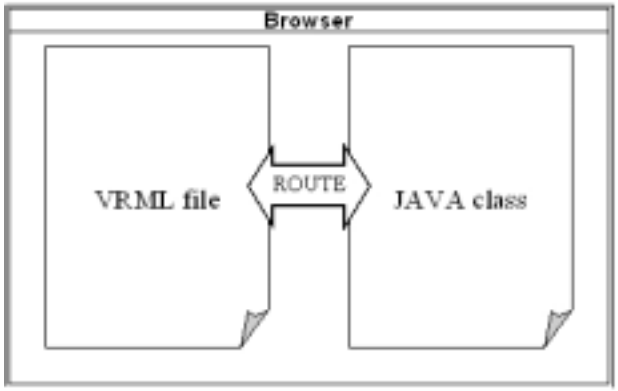

Figure 3-Direct VRML JAVA routing

Here the Java applet can pass messages to and from the virtual world. A message send to a Node is *EventIn*, and a message from the node in *EventOut*. Some Nodes also have fields that contain structural information. But only exposed fields can be accessed from an external interface. Listing 2 shows a sample node with all the fields, their types and their default value(s).

Listing 2

```
Collision
{
eventIn        MFNode   addChildren
eventIn        MFNode   removeChildren
exposedField   MFNode   children          []
exposedField   SFBool   collide           TRUE
field          SFVec3f  bboxCenter        0 0 0
field          SFVec3f  bboxSize          -1 -1 -1
field          SFNode   proxy             NULL
eventOut       SFTime   collideTime
}
```

In this method there is no HTML web page, the browser directly opens the virtual world. When the VRML code runs, it can initiate the JAVA class. Listing 3 shows a sample VRML script that initiates a Java class, and routes some nodes to the internal Java variables.

Listing 3

```
DEF Control Script
{
```

```
    url "Control.class"
        mustEvaluate TRUE

    eventOut SFRotation outEv_Rotation_theta
    eventOut SFVec3f outEv_Translation_z
    eventOut SFBool outEv_ViewPoint
}

ROUTE Control.outEv_Rotation_theta TO X_R.rotation
ROUTE Control.outEv_Translation_z TO Z_T.translation
ROUTE Control.outEv_ViewPoint TO MYVP.set_bind
```

In the VRML file the node properties are routed to the corresponding variables in the Java class. Now the Java class can send and receive information to the desired node, and the virtual world becomes fully interactive.

4.2 Method 2

Another method is to create a web page that contains the VRML file and the JAVA applet. The VRML file contains only an empty ROOT node (Listing 4).

Listing 4

```
#VRML V2.0 utf8
DEF Camera Viewpoint
{ position 0 0 7 }
DEF ROOT Group {}
```

Using EAI the rest of the virtual world can be built through Java. The capabilities of EAI are vast, yet sometimes very limited. For example it was found that one could control the translation, rotation, and scale of objects. New objects can be added as children to a node, but only the last child added can be accessed.

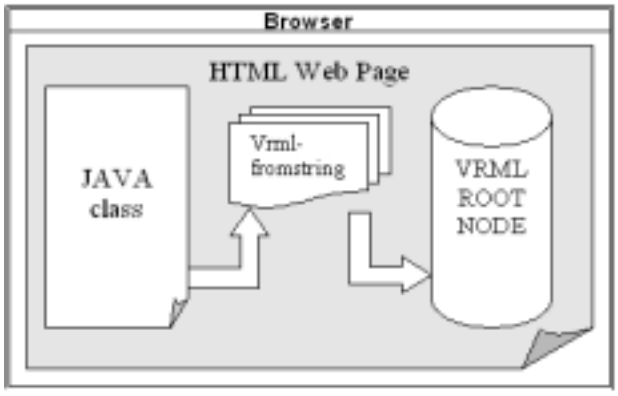

Figure 4-HTML embedded Java/Vrml

A Java applet communicates with a VRML world by first obtaining an instance of the **Browser** class. This class is the Java encapsulation of the VRML world. It contains the entire Browser Script Interface as well as the **getNode()** method, which returns a **Node** when given a DEF name string. Only DEF names in the base file are accessible. Names in **Inline** files and those created with **createVRMLFromString()** or **createVRMLFrom-URL()** are not accessible. Since VRML files can have multiple occurrences of the same DEF name, only the node with the *last* occurrence of a given name is accessible [8].

Listing 5

```
//this function creates a vrml Node from String
public void draw_my_Cone()
  {
       my_Cone = browser.createVrmlFromString(
   "Transform\n" +
   "{\n" +
   " translation 0 1 0\n"+
   " children Shape \n" +
   " {\n" +
   "           appearance Appearance \n" +
   "           {\n" +
   "                    material Material\n" +
   "                    {\n" +
   "           diffuseColor  0.5 1 0.8\n"+
   "                    }\n" +
   "           }\n" +
   "           geometry Box \n" +
   "           {\n" +
   "                    size 1 1 1 \n"+
   "           }\n" +
   " }\n" +
   "}\n");
  }
...
...
...
//calling the function to setup the Cone
draw_my_Cone()
//adding the Cone node to the VRMl file
addChildren.setValue(my_Cone);
```

Thus, if we design and build our robot using Java, we have to send the model as text to the VRML program (Listing 5), but when VRML displays the world, Java cannot access the desired nodes to move the robot arm.
We can use TouchSensors to overcome this problem, but now the user has to click on the joint, which may require repositioning the viewpoint. After clicking (or touching) that joint, Java can access that node and move the joint. These extra intermediate steps make the interface very user-unfriendly.

4.3 Method 3

Another way of building an interactive control system is creating an HTML page that includes the VRML model (.wrl) and the JAVA applet (.class) which controls the world. A universal template of a robot arm is coded in the VRML file, and each node is assigned a name.

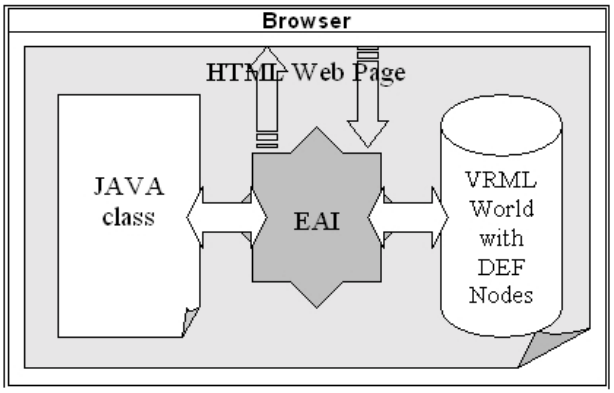

Figure 5-JAVA VRML interaction using EAI

The JAVA applet can find the VRML *world* through the browser (Figure 5). After finding the ROOT it will look for the named nodes and connects them to the corresponding Java variables.
Then all the EventIn and EventOut`s are defined for each node, and they are attached to corresponding Java objects with correct types (Listing 6).

Listing 6

```
…
//gets the browser
browser = (Browser)
   vrml.external.Browser.getBrowser(this)
//gets the nodes through the browser
joint_1 = browser.getNode("JOINT_1");
endeffector = browser.getNode("ENDEFFECTOR");
…
//setting up the proper Events for each node
joint_1_set_translation = (EventInSFVec3f)
   joint_1.getEventIn("translation")
joint_1_set_rotation = (EventInSFRotation)
   joint_1.getEventIn("rotation")
endeffector_sensor = (EventOutSFBool)
   endeffector.getEventOut("isActive");
…
```

Now Java can access the Nodes to pass information to and from the virtual world (Listing 7). Using a Java GUI we can initiate interaction with Event handlers.

Listing 7

```
…
//Java Event handling
…
//sending information to Nodes
joint_1_set_translation.setValue (joint_1_translation);
joint_1_set_rotation.setValue (joint_1_rotation);
…
//receiving information from nodes
sensor_state = endeffector_sensor.getValue();
…
```

After starting the communication between VRML and Java, our universal template of the robot arm can be modified to fit our specifications. Subsequent to the customization, the arm can be fully controlled through Java.

This interactive modeling system can be connected to a real robot through a server via Internet. And the robot can be remotely controlled, using this 3D modeling environment.

4.4 Discussion

The three methods that we examined in our research each have their pros and cons. Method 1 requires programming in both Java and VRML side, and therefore to expand the software, you have to access both codes.
Method 2 is not a good candidate for this purpose, since the nodes designed from Java cannot be accessed again. Or they have to be activated from the VRML side, which requires clicking on the object. This makes the user interface very unfriendly. But this method requires programming mainly in the Java side. So changing and upgrading the code would be more reliable.
We used method 3 to design our research. There is no programming in the VRML side, yet you have to build a very complex virtual robot in the VRML file. Also, since all the sections are embedded in a web page, it expands our limitations. May be a combination of all the three methods can be used to optimize the software.

5. EXAMPLE

Using the EAI methods with the predefined robot template, we created the Dyno Robo. Dyno Robo is a virtual robot that can be customized to match a real robot. Currently, it has 4 joints and a basic end-effector. Each of the 4 joints can be independently configured. The initial orientation of the joints can be set, and the length of the links can also be changed. Figure 6 shows some sample modes of Dyno Robo. Figure 6-a shows a robot with two links set to length of zero, where the end-effector has two degrees of freedom by itself. In Figure 6-b link 1 is doubled, and link 3 is set to zero. The end-effector has one degree of freedom. Figure 6-c shows a robot with all the links extended, and in Figure 6-d joint 3 is also rotated by 135 degrees.

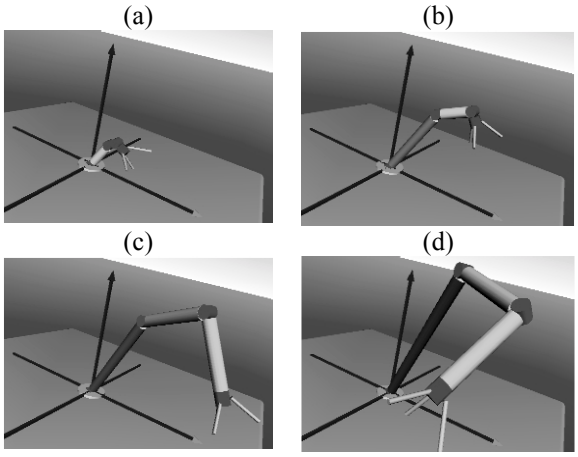

Figure 6-Different customizations of Dyno Robo

After this process, the user can move all the joints, and change the view position to the end-effector view. It can grasp an object and place it on one of the two squares.

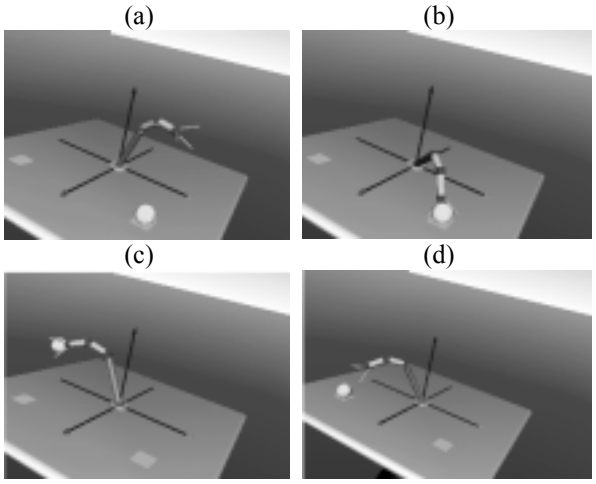

Figure 7-Dyno Robo In Action

Sensors implemented in the world enable the robot to detect the objects; therefore, it can grab the object only if it is within the range. To release the object we used proximity sensors to detect the place of the object, so it won't release the object unless it is safe to do so.

In Figure 7 these sequence is shown in 4 pictures. The robot moves to grab a ball (Figure 7-a). When the sensors detect the ball, the ball is picked up (Figure 7-b). The robot moves to the second square (Figure 7-c), and when it is in the permitted proximity, the ball is released (Figure 7-d).

Figure 8 shows the interface of Dyno Robo 3.0 . You can navigate within the virtual world using the standard VRML navigation features, and also you can change your view point through the Java interface, so you can have the view point right at the end-effector.

This is just a basic version, and it can be expanded so all the features are customizable, and the number and type of joints are configurable.

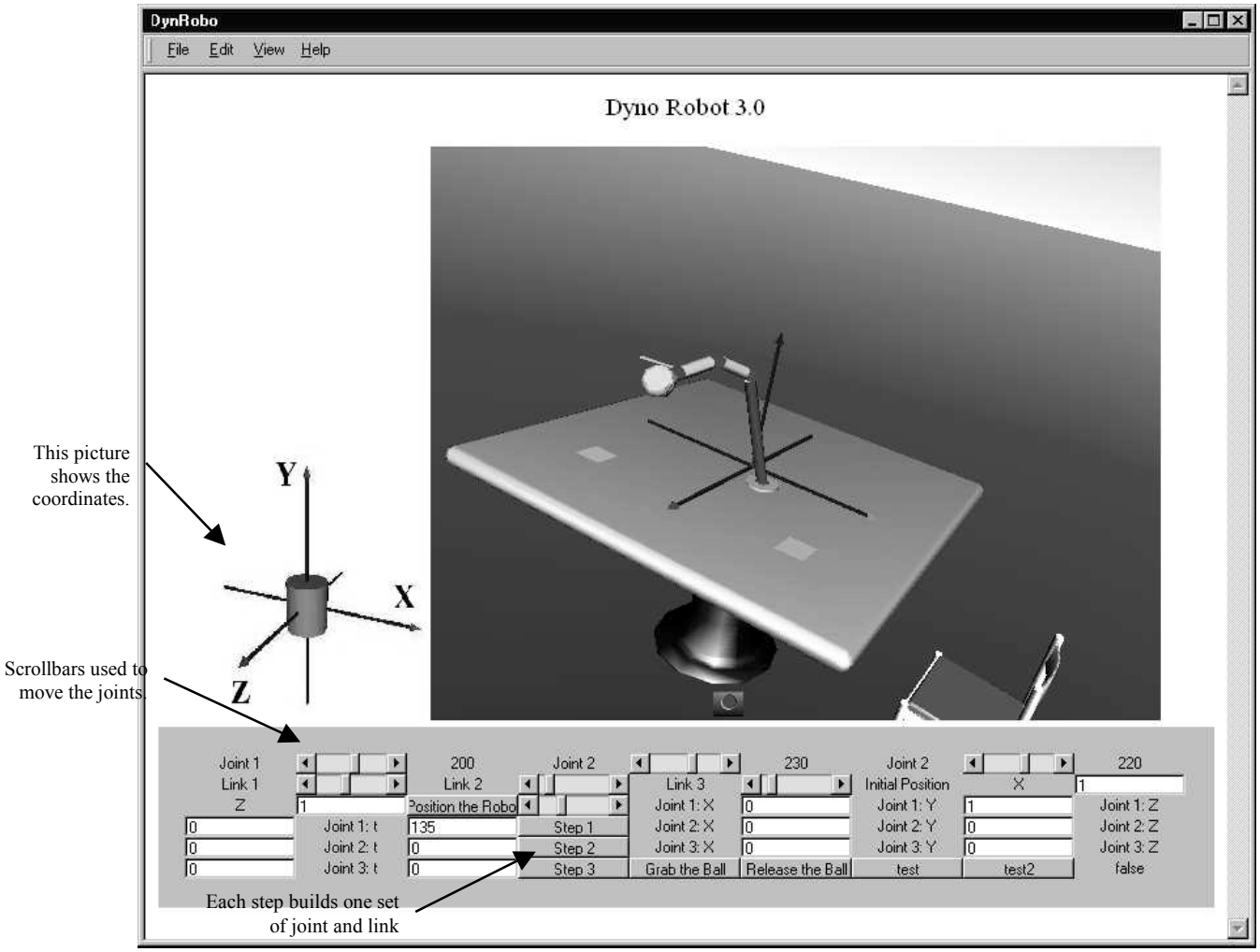

Figure 8-Dyno Robot

6. FUTURE PLANS

After having a fully functional Robot modeling system and with full features, a dynamic representation to the model will be added, so the virtual model has mass and inertia. And all joints follow real robot specifications. Also we have to design client to server socket connection and implement I/O ports to the Java applet. We may also implement some Java 3D in our interface.

For example, this robotics system can be used for training. Since it is web-based and interactive, it is a good candidate for teaching basic robotics. The software can be extended to study dexterous and reachable workspace of a robot. Trajectory planning is another advantage of this tool. It can help to investigate and visualize feasible trajectories in a Tele-robotics environment.

Adding physical-based modeling properties to the graphics model provides a path for haptic rendering and haptic interaction.

7. CONCLUSIONS

In this research, we discussed the different 3D environments that are possible candidate for Tele-operation of robotic systems. We explained some different methods of implementing Java and VRML interface. Finally, we described a software which is being developed in our research Lab.

8. ACKNOLEDGEMENTS

This project was partially funded by the Institute of Robotics and Intelligent Systems (IRIS) of Canada.

9. REFERENCES

[1]L. Nigro, F. Pupo, “A modular approach to real-time programming using actors and Java”, Control Engineering Practice 6 (1998) 1485-1491

[2]S. Piccinocchi, M. Ceccaredlli, F. Piloni, A. Bicchi, “Interactive Benchmark for planning Algorithms on the Web”, Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico – April 1997, Page 399

[3]http://telerobot.mech.uwa.edu.au/ROBOT/telerobo.htm, visited 4,1999

[4]h. Hirukawa, T. Matsui, S. Hirai, A Prototype of standard Tele-operation Systems on an Enhanced VRML, 1997 International Conference on Inteligent Robots and Systems, Vol 3, page 1801-6

[5] http://java.sun.com/products/java-media/3D/

[6]http://www.web3d.org/technicalinfo/specifications/specifications.htm , visited 4,1999

[7]http://www.web3d.org/technicalinfo/x3d/x3d_process.htm htm , visited 4,1999

[8]http://cosmosoftware.com/developer/moving-worlds/spec/ExternalInterface.html, visited 4,1999

[9]http://cosmosoftware.com/developer/eai.html

[10]http://cosmosoftware.com/developer/moving-worlds/spec/ExternalInterface.html

[11]http://www-winfo.uni-siegen.de/vrmlHistory/docs/index.html

[12]http://www-robotics.cs.umass.edu/robotics.html

[13]http://wwwipr.ira.uka.de/~germ_rob/

[14]http://telerobot.mech.uwa.edu.au/

[15]http://www.rosl.com/

[16]http://www.robotics.org/