Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

WEB BASED ROBOT SIMULATION USING VRML

Martin Rohrmeier

Innovation
IXOS Software AG
Pappenheimstraße 7
80335 Munich, Germany

ABSTRACT

The Virtual Reality Modeling Language (VRML) enables the integration of interactive 3D graphics into the web. At the German Aerospace Center we have been using the new language in robotic applications from its beginning on. The shown project is an example of the possibilities of using it in web-based simulations. Specialized and expensive hard- or software is not needed, any web browser with a vrml viewer is able to run the program which makes the application independent from any underlying hardware platform. There was put a special attention to building an efficient and usable interface for the standard pointing device. The functionalities provided by the 3D GUI are easy to use and internationalized because self-explaining symbols were used instead of language. Together with Java and VRML's External Authoring Interface (EAI) the simulation can also be used for visualizing and telemanipulating real robots which was the original intention behind the development of this work.

1 INTRODUCTION

When talking of simulation applications one usually thinks of large monolithic program packages that are installed on a workstation to run stand-alone. When VRML 1.0 was replaced by the more flexible VRML 2.0 it provided for the first time the technology to build small but complex three dimensional sceneries running in a web environment. At this point, we started the project discussed here to evaluate the possibilities of implementing a web-based simulation and telemanipulation environment for industrial robots.

The simulation should benefit of all the strengths of VRML. The result should be a platform independent software that runs completely inside a web browser, it therefore has to be small enough to be loaded from the web server every time when needed. With the exception of the VRML plugin no installation should be necessary. As the complete user interface is also integrated into the visualization area it also must benefit from the additional degree of freedom and be more intuitive than existing interfaces. As a severe restriction for a robotic application the only possibility for user interaction in VRML worlds is the use of the mouse. Because 6-DOF input devices cannot be used, a way has to be found to enter position and rotation values with the 2D pointing device.

For the telemanipulation of industrial robots a connection must be somehow established between the controlling workstation and the real manipulator. The most comfortable way to access the values in a VRML scene is via a java applet. Therefore, we can use the External Authoring Interface (EAI) which can register an applet to be notified when certain events occur and vice versa events can be generated and sent to the virtual world.

2 IMPLEMENTATION

In this section, we describe details of the implementation. First we will find an abstract description of the simulated robot which is as abstract as possible to work with different types of industrial manipulators. Based on this description we will then implement a VRML object `robot` that includes the complete functionality of the simulation. In a last step we gain access to the simulation and build the telemanipulation interface.

2.1 Defining the object robot

To define an abstract object, we need to determine the input and output parameters as well as the properties describing an instance of this object. To do this, we want to have a short glance at robotics theory here.

An industrial robot is a tool to position a tool in a limited working space. The tool is mounted at the end of its movable arm and when speaking of the tool's or robot's position we always refer to the origin of the tool coordinate system. The robot arm consists of a set of linear or rotational joints connected by the segments of the arm. The different joints enable various degrees of freedoms (DOF) which is a motion in a different direction to reach all points in the usually three dimensional working space. Three DOFs are necessary to reach any position in space and three for the rotation, so to move freely in space the robot needs at least six joints. As linear joints are not able to change the orientation at least three of them must be rotational joints. In this project we consider only robots with six rotational joints, so called 6R-manipulators. Though the approach is similar for other types. A robot can be defined by Denavit-Hartenberg parameters. This set of parameters describes the arrangement of joints and is the basis for any kinematic calculation. There are four values needed for every joint: *a*, *d*, α and θ. Additionally we specify values for the minimum and the maximum of the joint rotation: *min* and *max*. For a more detailed introduction to robotics we recommend reading the according literature. As they don't change after the instatiation of the robot object we can define a field of six float values in VRML for each parameter.

The object can be manipulated by events. There are two different possibilities in moving the robot arm to a new position. Either the joint values can be set directly which will affect the tool's position and its rotation or more probably we give a target position or a target rotation and let the object calculate and set the according joint values. The joint states are represented by a six dimensional vector of value `MFFloat` and instead of a 3x4-matrix we will use the VRML data types `SFVec3f` and `SFRotation` for position and rotation values. When an `eventIn` is received the changed values will be sent by `eventOut` again. so the behavior of the object is as follows.

```
1. IN set_position  >  OUT joint_changed
2. IN set_rotation  >  OUT joint_changed
3. IN set_joint     >  OUT position_changed +
                       OUT rotation_changed
```

In VRML syntax the prototype for the robot object can be defined like shown below. The additional parameter arm contains the URL to the VRML file for each segment.

```
PROTO Robot [
  eventIn  SFVec3f    set_position
  eventIn  SFRotation set_rotation
  eventIn  MFFloat    set_joint

  eventOut SFVec3f    position_changed
  eventOut SFRotation rotation_changed
  eventOut MFFloat    joint_changed

  field    MFFloat    a       [0,0,0,0,0,0]
  field    MFFloat    d       [0,0,0,0,0,0]
  field    MFFloat    alpha   [0,0,0,0,0,0]
  field    MFFloat    theta   [0,0,0,0,0,0]
  field    MFFloat    min     [0,0,0,0,0,0]
  field    MFFloat    max     [0,0,0,0,0,0]
  field    MFString   arm     []
]
```

2.2 Implementing the VRML Object Robot

The implementation task was divided up into three separate problems: a visualization object to display the robot model on the screen, a calculation object to convert position and rotation into joint values with inverse kinematics and vice versa and finally an interface that enables the user to move the robot in 6 DOFs by using a default 2D pointing device like the mouse. How data is transmitted between the modules is shown below in Figure 1.

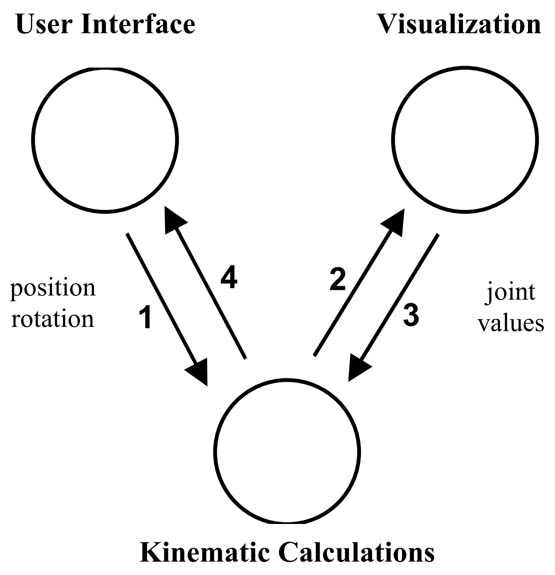

Figure 1: Data Flow Between Components

These three components are implemented as three different VRML objects. In fact the complete robot object consists of more than three components but these are the essential ones and build together a fully functional interactive robot model. We are now going to describe these three modules beginning with the most interesting one, the visualization component.

2.2.1 Visualization

This section describes how the 3D robot model is structured in VRML. Each segment of the arm moves in a coordinate system that is determined by the position of the previous segments. This chain can be modeled by a tree of nodes, where each segment node is a child of the previous segment node. The distances for assembling the chain are taken from the Denavit- Hartenberg parameters which also have to be passed to the module. Note that it doesn't need to know any single position, every segment knows only the according joint's value. As shown in Figure 1, the input for the VRML prototype `rob_geo` is an `MFFloat` array of six joint values. When such an `eventIn` is received, a script node is called that distributes the values among the segments. Therefore a `SFRotation eventOut` is generated and sent to the according joint, the incoming arrow in Figure 2 illustrates this.

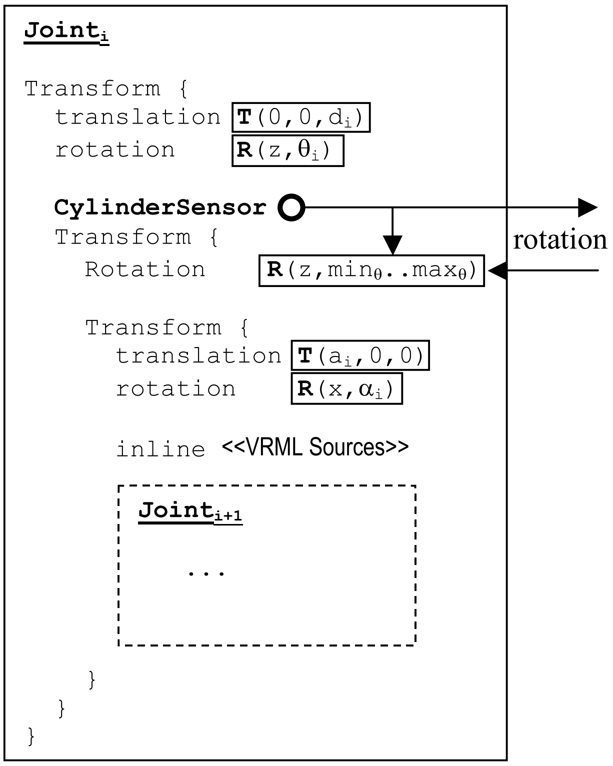

Figure 2: Structure of a Single Arm Segment in VRML

As one can see there is also a break with the data flow model. Joint value control by grabbing the arm at any segment and rotate the according joint which should be a task of the user interface is better included directly in the model because the tree is already existing here. The generated events are directly routed to the joint and will also be sent to the script which again writes the values into an array. This array is the output of the `rob_geo` module. The prototype is defined as follows.

```
PROTO rob_geo [
  eventIn  MFFloat    set_joint

  eventOut MFFloat    joint_changed

  field    MFFloat    a      [0,0,0,0,0,0]
  field    MFFloat    d      [0,0,0,0,0,0]
  field    MFFloat    alpha  [0,0,0,0,0,0]
  field    MFFloat    theta  [0,0,0,0,0,0]
  field    MFFloat    min    [0,0,0,0,0,0]
  field    MFFloat    max    [0,0,0,0,0,0]
]
```

2.2.2 Kinematic Calculations

Almost every interaction with the model produces a lot of calculations which are done by this component. For details about how it works please read the according literature. While mathematically quite complex its task is very easy. Position and rotation have to be converted into joints and joints have to be converted into positions and rotations. So it contains two script nodes: one for the forward and one for the inverse kinematics. Another script node receives the module's `eventIns` and calls the needed calculation routine. All of these are based on the Denavit-Hartenberg parameters. So `rob_kin`, the prototype of the module, needs with the exception of the VRML descriptions all the information of the `Robot` object and therefore has almost the same definition.

```
PROTO rob_kin [
  eventIn  SFVec3f    set_position
  eventIn  SFRotation set_rotation
  eventIn  MFFloat    set_joint

  eventOut SFVec3f    position_changed
  eventOut SFRotation rotation_changed
  eventOut MFFloat    joint_changed

  field    MFFloat    a      [0,0,0,0,0,0]
  field    MFFloat    d      [0,0,0,0,0,0]
  field    MFFloat    alpha  [0,0,0,0,0,0]
  field    MFFloat    theta  [0,0,0,0,0,0]
  field    MFFloat    min    [0,0,0,0,0,0]
  field    MFFloat    max    [0,0,0,0,0,0]
]
```

2.2.3 User Interface

The user interface is a collection of 3D elements with sensors that allows the user to manipulate the robot in its working space comfortably. No matter which of these is used, the output is always a position and/or a rotation value that is passed to the inverse kinematics subroutine. As the given target point may be outside of the working space the set position values are always sent back to the interface as feedback. As no other variables are needed the prototype is quite simple.

```
PROTO rob_ctrl [
  eventOut SFVec3f    position_changed
  eventOut SFRotation rotation_changed

  eventIn  SFVec3f    set_position
  eventIn  SFRotation set_rotation
]
```

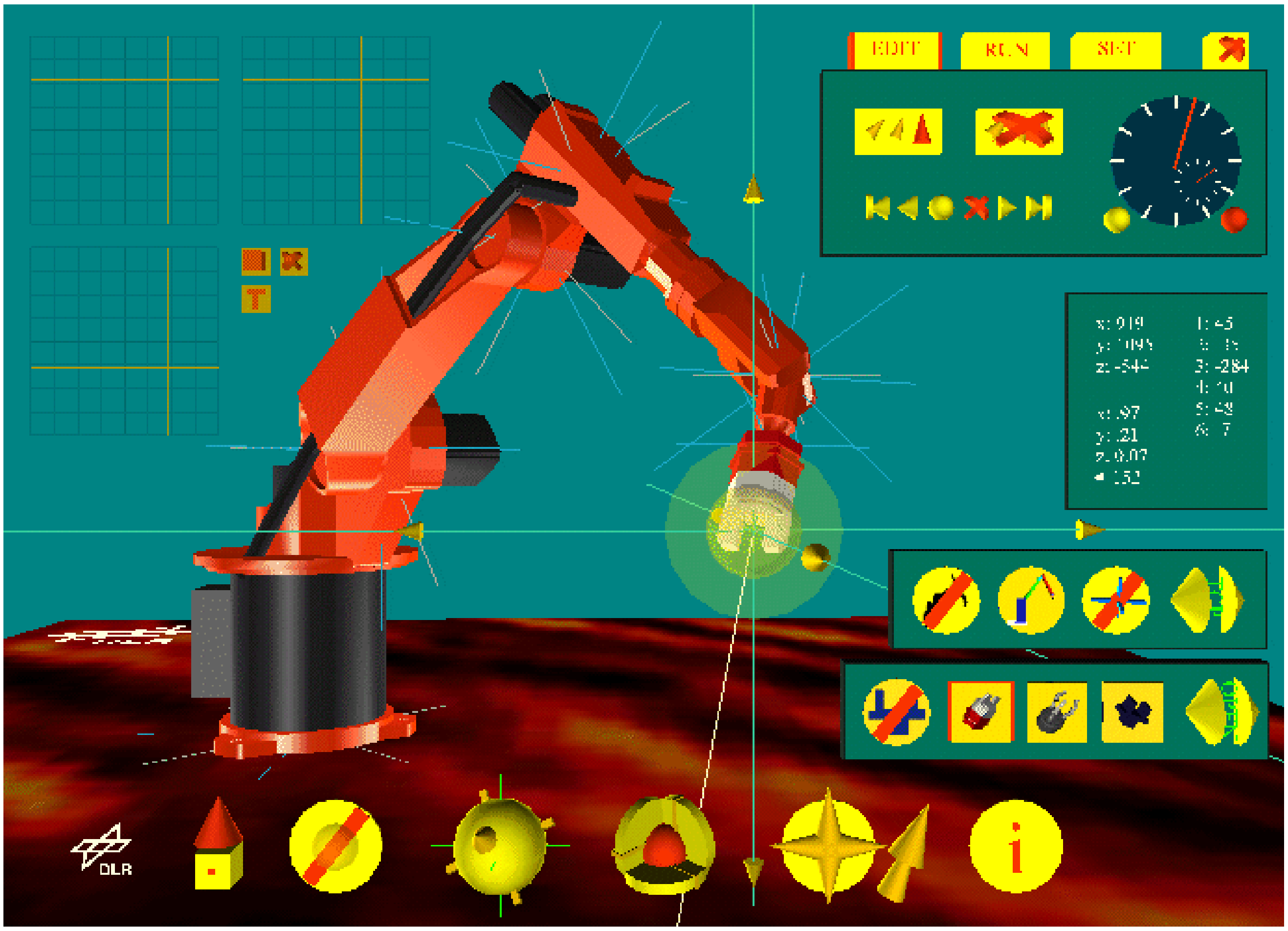

Figure 3: A Screenshot of the VRML Simulation

3 CONCLUSION

With the VRML based simulation of industrial robots via WWW some significant improvements towards known solutions could be achieved. The very high level language VRML can run on any underlying platform. The only requirements for running the program is a web browser with a plugin installed that is available for free. So there are no costs for additional specialized hard- or software. The completely three dimensional interface is mostly self-explanatory and easy to use. This makes the program also international because the functionality is indicated by signs instead of text. Another plus is that the controller con freely choose his viewpoint of the scenery. Although at the moment a video stream transmitting the movements of the real robot to the user is still needed, in the future a whole virtual environment is possible which will drastically reduce the required bandwidth for the telemanipulation. Figure 3 shows a screenshot of the application.

REFERENCES

Carey R. and Bell G. 1997. *The Annotated VRML 97 Reference Manual*. Reading, MA: Addison-Wesley.

Craig, J.J. 1989. *Introduction to Robotics*. Reading, MA: Addison-Wesley.

Rohrmeier, M. 1997. Telemanipulation von Robotern mittels VRML 2.0 und Java. Master Thesis, Department of Robotics and System Dynamics, German Aerospace Center (DLR), Weßling, Germany.

Siegert H.-J. and Bocionek S. 1996. *Robotik: Programmierung intelligenter Roboter*. Berlin: Springer.

AUTHOR BIOGRAPHY

MARTIN ROHRMEIER is a project manager in the Innovation department at IXOS Software AG. He received his diploma in computer science at the Technical University of Munich. The described project was his master thesis at the German Aerospace Center (DLR). His email address is <rohrmeier@web.de>.