DESIGN OF AN INTEGRATED ROBOT MANIPULATOR SIMULATOR FOR REMOTE LEARNING APPLICATIONS

by

Brendon J. Wilson

A THESIS PROPOSAL SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF APPLIED SCIENCE
In The School

of

Engineering Science

SIMON FRASER UNIVERSITY
September 30, 1997

Abstract

In response to rising equipment costs and reduced funding, the REMOTE (Really Exciting Manipulator Object Tele-learning Experience) project is investigating new methods of providing hands-on experience to students using tele-learning. Tele-learning is a method of providing distance education using multimedia technologies via any combination of telephone, video-conferencing, or Internet connections. The purpose of this project is to create a Java application that enables students to explore robotics programming via interactive experimentation.

This thesis project will develop a complete simulation application, extending work by Ron Racine and Scott Branden for ENSC 439. The completed environment will feature a user interface that displays visual simulation feedback and allows users to edit, program, and debug simulation files. An Internet connection to a remote server will enable users to test their completed programs on a real manipulator, and view the result using CU-See-Me Internet video software. The application's underlying simulation engine will control the internal representation and simulation of the robot manipulator.

This project is of particular importance to future ENSC 439 classes, which will use the simulator to provide students with more access to robot programming experience at a lower cost. To suit the purposes of ENSC 439, the software will be required to simulate a Scorbot ER III manipulator and interface via the Internet to the existing Scorbot manipulator available in the Engineering Science lab. By careful design and implementation, the completed application should permit addition of supplemental features, allowing the application to suit the present and future demands of ENSC 439.

Approval

Name:	Brendon Wilson
Degree:	Bachelor of Applied Science
Title of Thesis:	Design of an integrated robot manipulator simulator for remote learning applications
	Dr. John Jones Director School of Engineering Science, SFU
Examining Committee:	
Technical and Academic Supervisor:	Dr. John Dill Professor School of Engineering Science, SFU
Committee Member:	Dr. Kamal Gupta Associate Professor School of Engineering Science, SFU
Committee Member:	Dr. Brian Fisher Research Associate Center for Systems Science, SFU
Communication Lecturer:	Steve Whitmore Communication Lecturer School of Engineering Science, SFU
	Date Approved

Table of Contents

Introduction

As the cost of providing quality education has continued to rise over the past few years, many new methods of providing meaningful, hands-on experience to students are being investigated. One proposed method of improving the student-to-cost ratio is tele-learning: providing distance education using multimedia technologies via any combination of telephone, video-conferencing, or Internet connections. The purpose of this project is to create a Java application that enables students to learn robotics programming via interactive experimentation. This thesis will extend original work started by Ron Racine and Scott Branden as an ENSC 439 project.

The goal of the REMOTE (Really Exciting Manipulator Object Tele-learning Experience) project is to enable students to explore and learn about programming a jointed robot. This thesis project is to develop an application that achieves this goal in two stages: the first stage is a software simulator that allows users to program and debug a robot manipulator; the second stage is a client-server connection to a real manipulator which allows users to test their programs and view the results using CU-See-Me Internet video. In effect, this simulator will give a larger number of students access to a limited resource, and the mechanism to conduct remote experiment in robotics.

This project is of particular importance to future ENSC 439 classes, which will use the simulator to provide students with more access to robot programming experience at a lower cost. To suit the purposes of ENSC 439, the software will be required to simulate a Scorbot ER III manipulator and interface via the Internet to the existing Scorbot manipulator available in the Engineering Science lab.

Overview of Project Requirements

The completed project application will consist of several subsections:

- Graphical User Interface (GUI): responsible for providing the user with a way to interact with the simulator. The GUI will be expected to represent the state of the simulation to the user and allow them to edit and run the simulation, perform file operations, configure their preferences, and send their program to the remote server.
- The Simulation Engine: responsible for interpreting the robot model, constructing internal data structures to represent the robot, and manipulating the robot's joint angles, as specified by simulation commands and the robot's inverse-kinematics.
- Simulation File Parsers: responsible for parsing files for simulation commands, descriptions of the physical form of the manipulator, and pre-solved inverse-kinematics equations. These files control the internal model of the robot used to simulate and control the manipulator.

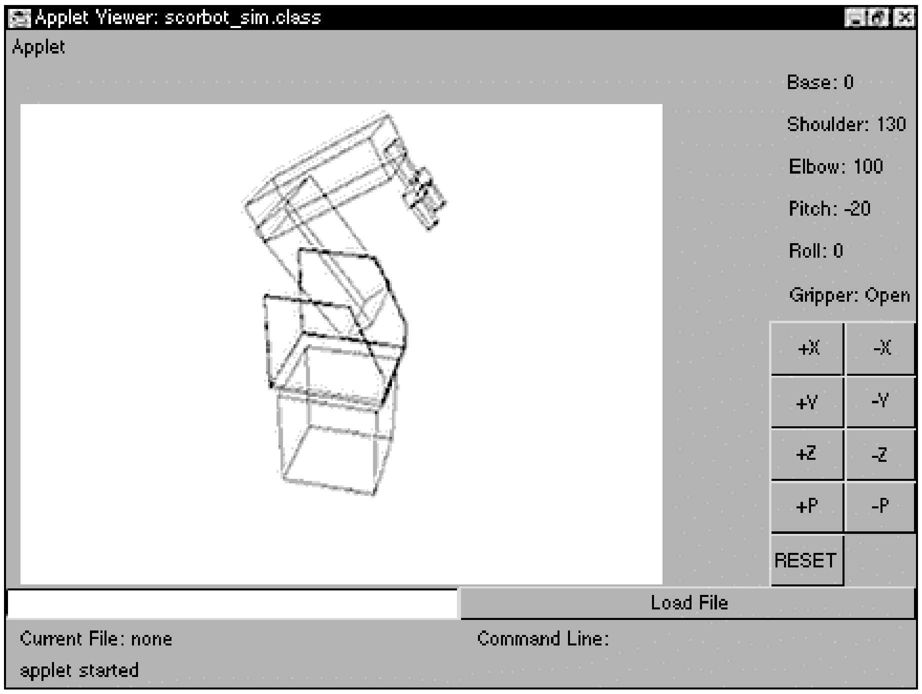
- Remote Server Communications: responsible for allowing the simulator application to communicate with a server, enabling the simulator to control the real manipulator remotely via the server.

In addition to the parsers that will be created, the simulator will require a framework to allow feature extensibility. This framework will ensure that additional parsers for different model formats and robot languages can be easily programmed and added to the application's existing features.

To discuss the project requirements outlined above, the main issues have been grouped into several subsections: Design of the Graphical User Interface, Integration of Interactive Programming Features, The Simulation Engine, Server Communications, and Simulator Generalization. In addition to the core tasks discussed in these subsections, an additional subsection justifies the proposed use of the Java programming language.

Design of the User Interface

The original simulator, shown in Figure 1, is a stand alone Java program with a simple user interface, providing minimal functionality in a non-intuitive layout. One of the main goals of the project will be to improve the GUI to be more user-friendly and intuitive; this includes adding functionality to give users the ability to control all aspects of the simulator, instead of those few controls presently provided.



Base: 0
Shoulder: 130
Elbow: 100
Pitch: -20
Roll: 0
Gripper: Open

Current File: none
Command Line:
applet started

Figure 1: Original REMOTE Simulator

One particular aspect of the interface that may be redesigned is the current wire-frame representation of the robot. The expected delivery of the Java 3D API in the spring of 1998 could be used to replace the current wire frame model, and provide users with a clearer three-dimensional representation of the manipulator. However, this work cannot be undertaken until the arrival of the 3D API, making it a low priority for the project. More important will be the proper implementation of other user interface elements.

An important design issue that will need to be addressed is the interface's usability. Testing of several designs will be required to ensure that the layout and functionality of the GUI correspond to an average user's intuition and expectations. Part of the usability testing will be conducted in conjunction with the ENSC 439 class occurring in the 98-1 semester; this class will be using the simulator to complete one or more assignments, and will provide an opportunity to test the simulator's interface with real users.

Integration of Interactive Programming Features

At present, the simulator provides no features that allow the user to edit file of simulation commands from within the simulator. Simulations are currently programmed externally in a text-editing application and then loaded into the simulator; once a simulation is loaded into the current simulator, the simulation cannot be stopped, once completed it cannot be rerun, and a new simulation cannot be loaded without restarting the simulator. This approach defeats the intended interactive nature of the application, and restricts the user's learning process.

Using the simulator, the user should be able to follow a typical software development cycle, shown in Figure 2. A user should be able to write a program, run the simulation, and view the simulation results, repeating this cycle until the desired results are achieved. All of these actions should occur within the simulator environment, without using external applications for program development.

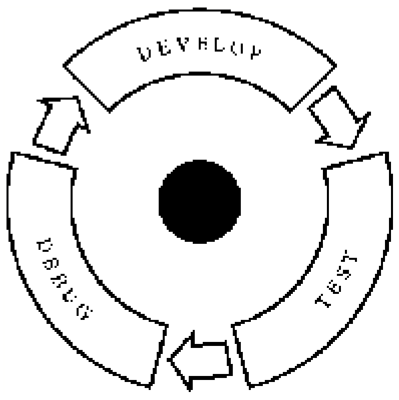


Figure 2: Ideal Simulation Program Development Cycle

The completed simulation environment will need to incorporate features that allow users to load, run, edit, rerun, partially run, and debug their programs. The environment will need to include the ability to start, stop, reset, and step the simulation, and also edit the simulation program while viewing the results. In addition to programming, users should also be able to debug their simulations; incorporation of features such as simulation stepping and perhaps the ability to add breakpoints to programs would permit intuitive debugging, similar to other programming environments.

These features will be accessible to the user as a part of the GUI, allowing the user to interact and manipulate the underlying simulation model of the robot. As outlined before, implementation of these features will need to focus on both the demands of the interface for usability and the ability of the simulation engine to provide the required output.

The Simulation Engine

The core processes of reading in robot command files, robot model definition files, performing the required inverse kinematics calculations, and manipulating the state of the internal model are the purpose of the Simulation Engine. The responsibilities of the Simulation Engine can be broken down into the following five tasks:

- Reading files that define the robot, and constructing the robot's internal representation
- Reading command files in preparation for simulation
- Manipulating the representation in accordance with the command file and the robot's pre-defined inverse kinematics equations
- Communicating with the GUI to obtain the user's updates to the simulation file
- Representing the current state of the model, which can be used by the GUI to provide useful information to the user in whichever form it deems appropriate.

The Simulation Engine should not be responsible for the lower level tasks of parsing the models and the simulation files; it should only act as a coordinator over other 'plug-in' type modules which carry out the parsing and provide the data in the appropriate format when required. Similarly, the Simulation Engine should not actually do the inverse-kinematics calculations itself, again relying on a 'plug-in' module to provide the required information to manipulate the internal model of the robot.

The reason that the Simulation Engine will act only as a coordinator is simple: by abstracting as many of the tasks that are outside running the simulation, we can make the simulator more maintainable. The lower-level tasks of data format interpretation can be modularized so that future formats and models can be incorporated into the simulator without rewriting the central simulator; new task modules would only need to be programmed according to the abstract model. This marginal effort will ensure that the simulator can be easily reused to simulate other manipulators at a later date.

Addition of Server Communications

Client-server communications is a requirement of this project to allow communication between the simulator and the remote manipulator. This connection will allow students to use their completed programs to command the remote robot via the robot server; use of the commercial CU-See-Me software will allow students to view the results of their programs running on the manipulator via the Internet.

Programming the application to set up network connections and exchange data between itself and the remote server is only one portion of the communication problem. In addition to requiring a protocol to define the format that the commands are sent to the remote server, communication between the server and the robot hardware will also need to be implemented. For this thesis, only client side programming will be implemented, with another member of the REMOTE project completing the server side programming.

A proof-of-concept server program has already been implemented by Sean Lavin, to show that it is possible to command the Scorbot hardware via a simple Java client-server connection. Due to Java's inability to communicate with hardware directly, the proof-of-concept uses a C++ library to bridge between the Java server and the Scorbot hardware via the server's serial port.

Simulator Generalization

Eventually, this application could be used to simulate and conduct remote experiments using other manipulators, and other types of devices. With this in mind, the simulator should incorporate a sizable amount of abstraction in the software design; in effect, this would be a 'plug-in' style architecture allowing various features of the simulator to be replaced or supplemented.

Producing a completely general inverse-kinematics engine would be a thesis in itself; instead of attempting to tackle that problem, the REMOTE project will attempt to make the job of adding pieces to upgrade the simulator easier. For example, if someone calculated the inverse kinematics solution for a four-link manipulator, they could extend the inverse kinematics engine by simply programming another inverse kinematics module; the simulator could then be used to simulate that manipulator.

Besides different manipulators, incorporating the ability to read different file formats for the simulator data should also be easy. The data parsers include parsers for different 3D formats for representing the robot, and parsers for different robot languages.

Producing a simulator that incorporates flexibility will require that portions of the original code for the simulator be updated or rewritten; in some cases, newer features of the Java language will replace obsolete or 'deprecated' methods. Upgrading to the new version of the Java Developer's Kit (JDK) ensures that the simulator can take advantage of the newest features of the language, improve the application's speed, and implement several bug fixes.

Use of the Java Programming Language

There are several reasons that the programming language for this project has already been selected. The main reason is that the original program completed by Ron Racine and Scott Branden was written in Java; attempts to use what has already been done will reduce the time to complete the project. Features that make Java the candidate for this project include:

- Java applications can be run on any platform that has an implementation of the Java interpreter (known as the Java Virtual Machine).
- Java provides instant access to a collection of pre-existing networking, security, user interface, graphics and other libraries.
- Java libraries are being created and improved at a phenomenal rate.
- The use of Java means that a user is presented with familiar user interface elements.

The features of Java eliminate code porting and ensure that the final product can reach the widest audience of users, while reducing programming outside the scope of the central task. Java provides a user interface that is consistent with the native platform, taking advantage of the user's previous knowledge of their platform's interface. Using Java means results are quickly achieved, while maintaining the flexibility required to incorporate new features as additional libraries become available.

For the given reasons, it seems appropriate that the simulator development should continue using Java, extending the original simulator programmed.

Project Planning Horizon

The Gantt chart in Figure 3 summarizes the estimated project milestones.

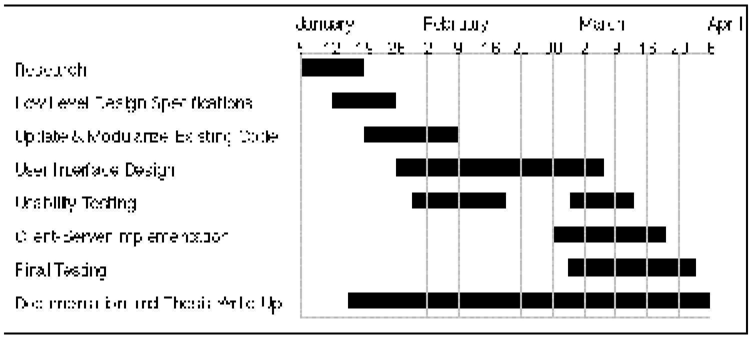


Figure 3: Gantt Chart of Project Milestones

At the start of the thesis project in January, a functional specification and a high-level design specification will already complete, and significant amount of time already invested in the project during my year abroad in Britain. Additional time is required to add features that could not be added due to pending software release dates (such as the Java 3D API), to properly design and test the GUI, and to implement the remote communications portion of the application. The final piece of work required to complete my thesis will be the drafting and revision of the thesis document.

Project work will be supervised by Dr. John Dill (Technical and Academic Supervisor), Dr. Kamal Gupta (Committee Member), and Brian Fisher (Committee Member). Other members of the REMOTE project will provide feedback and input to refine and revise the application. The evaluation and guidance of the thesis committee members and the students of ENSC 439 in 98-1 will guide the development of the simulator to completion.

Project Resource Requirements

The project requires access to computing facilities capable of compiling and running the Java Virtual Machine and the Java compiler in a timely fashion. For this purpose, I have already secured access to the Sun workstation resources available in the School of Engineering Science, in addition to my own Macintosh PowerBook computer. The compiler, interpreter, documentation, tutorials and all other required software for Java are available free of cost on the Internet from JavaSoft and other corporations.

Resources required for the remote server include the Scorbot ER III, the server computer, the video camera, and the CU See Me software. Funding for the hardware portion of this project has already been provided to Dr. Dill by the Ministry of Labour and Training Innovation Fund, and the required hardware has already been obtained.

In addition to the material resources required to complete the project, a suitable student to design, program, test, and document the project will be required. This project will require a background in object-oriented programming, preferably in the Java language, and familiarity with the concepts of robotics. From my recent work experience at ICS Net, I have acquired close to two years of experience with object-oriented programming in the Java language, and I have also completed an equivalent to ENSC 438 ("Introduction to Robotics") at the University of Sussex. I believe my experience, coupled with my experience as an undergraduate engineering student, is more than enough to carry this project through to completion.

Conclusion

This project is designed to fill the immediate needs of Dr. Dill's ENSC 439 course, namely providing a system to allow students to interactively program a robot simulation which can also be used to conduct robotics experiments remotely. In turn, this project also seeks to incorporate flexibility in its design that will allow the environment to be easily updated to simulate other manipulators in accordance with foreseeable future requirements. The project calls for a redesign of the current user interface, and the addition of elements to allow users to interactively simulate, debug, and remotely operate a robot using the proposed improved simulator environment.

The project timeline is being set to accommodate my year abroad at the University of Sussex, with the provision that additional work on the project will be required on my return. The thesis will be completed during the 98-1 semester, with the final thesis defense expected to take place at the end of the spring term of 1998.

All project requirements, including required hardware, software, and the knowledge requirements have been addressed; the procurement of the appropriate facilities and my background are more than adequate to carry this project through to its objectives: to create a tool that enables useful robotics tele-learning.