The Mars Rover Mockup

Forest Balk	Paul Dadah	Jeff McGough
Computer Science Dept.	Computer Science Dept.	Project Advisor
SDSM&T	SDSM&T	SDSM&T
Rapid City, SD 57701	Rapid City, SD 57701	Rapid City, SD 57701
Forest.Balk@ mines.sdsmt.edu	Paul.Dadah@ mines.sdsmt.edu	Jeff.McGough@ sdsmt.edu

Abstract

The Journey Museum in Rapid City, South Dakota, is interested in a proof-of-concept rover as part of a larger NASA based science exhibit. The preliminary documents explore a combination of software and hardware solutions used for the development, construction and implementation of the final project. The Primary user interface will be an intricate web application accessible via customized user-role access and capable of streaming live video from a variety of cameras. Most importantly, the user interface will also provide access to client applications capable of controlling the rover itself. In conjunction, there will be multiple background programs employed on the server to make communication between the client and the rover possible. From a black box perspective, the final product presents a practical, simplistic solution to a complex problem. Transcending user machine specifications, we provide the same experience to everyone, barring some minimum client hardware and software requirements.

Forest Balk, Paul Dadah and Jeff McGough
Department of Mathematics and Computer Science
South Dakota School of Mines and Technology
Rapid City, SD 57701
Jeff.McGough@sdsmt.edu

1 Introduction

We've designated the project name to be "The Mars Rover Mockup". The goal is to design and implement a rover capable of traversing a landscape. The rover will have at least one mounted arm that can manipulate the environment, one on-board webcam capable of streaming live video and will have the capability of being controlled via the internet through a website accessible from anywhere in the world.

1.1 Definitions, Acronyms and Abbreviations

The following terms are defined with respect to their context as referenced within this document.

- IIS: Internet Information Services
- JDK: Java Development Kit
- JRE: Java Runtime Environment
- Servo: A small device with an output shaft that can be positioned to specific angular positions by sending the servo a coded signal via an input line.

1.2 Overview of Rest of the Document

The rest of the document will describe, in more detail, the individual components of the original problem, our approach to satisfying each of the requirements and the specifics of the methods used in those approaches.

2 The Problem Statement In Detail and Our Approach

Once the scope of the problem was clear, the team's first objective was to nail down the requirements for the project. We pinpointed seven critical and independent requirements, all of which had to be fulfilled. For clarity, they're listed below:

- The rover shall be controlled remotely via a web user interface.
- The rover will have a movement library for arm and driving control.
- The rover will carry at least one mounted video camera.
- The website will be accessed via user/manager role authentication.
- There shall be support for both real-time and batch movement.

- The GUI must be simple to use for all ages, including children.
- The website must provide a signup and queuing feature to regulate exhibit usage.

The requirements list is very important, as it enables us to identify the different areas of focus for development. This also allowed us, as a team, to decide who had the most expertise and familiarity with certain topics so we could distribute the work. Lastly, it allowed us to further pinpoint some of the specifics, such as what platform(s) we can develop on, what libraries we can use, etc.

The team made the decision that the server (host) operating system would be Windows Server 2008. This made Internet Information Services 7 a viable candidate for website deployment, in which we're able to serve up web pages in a quick, reliable manner. However, website deployment was not the lone reason for choosing Windows Server 2008, as this also opened up the potential for utilizing the .NET Framework and Microsoft's SQL Server. As you'll see in more detail later on, we kept portability in mind and we wanted to make sure no potential end-user would be left in the dark.

This brings us to the next phase of our document, where we'll discuss in detail our approaches to the various problems and the specifics for each approach.

2.1 Remote Control Via Web Interface

To control the rover, we needed two things:

1. A way for the user to send movement (wheels and/or arm) commands.
2. A movement library for interpreting and sending distinguished commands.

To accomplish this, we needed a platform independent mechanism that was relatively intuitive and friendly to interact with. Among solutions such as Active X controls and downloadable client utilities, we opted for a Java Applet that utilizes the JFC/Swing component architecture. This made it possible to embed application functionality inside the client's web browser where we could establish a socket connection, effectively creating a tunnel to the server, where commands could be sent back and forth.

More details with respect to the Java Application are provided in the upcoming sections including details related to the cameras, arm movement and real-time/batch movement commands.

2.2 Movement Library for Arm and Driving Controls

The rover will have one arm that can manipulate the environment mounted to the front of the base of the rover. The arm can rotate at the base side to side and has three joints and one claw to pick objects up with. The manipulation arm has a wide range of motion which requires a large amount of error handling to prevent damage to the servo motors and the rover itself.

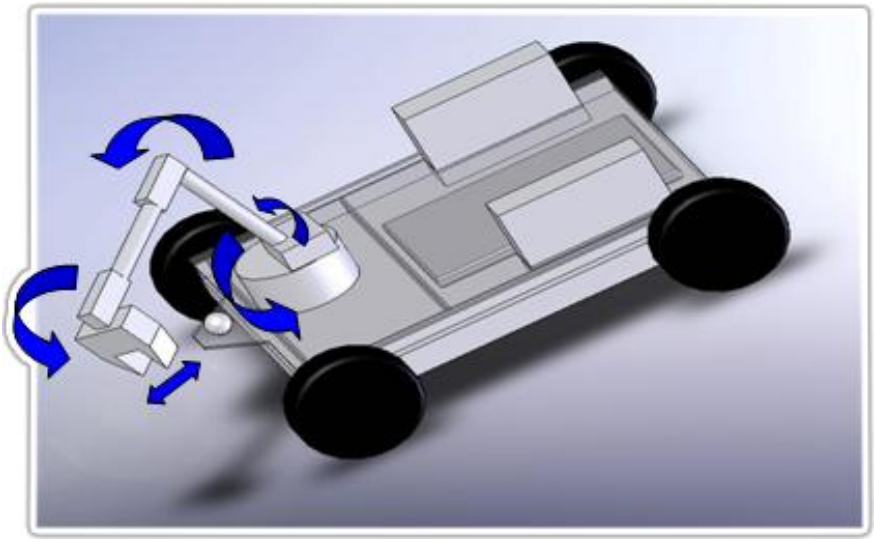

Figure 2.3.1: SolidWorks Rover Design Demonstrating Arm Movement Direction

2.3 The Video Cameras

To satisfy the need for a video camera on the rover we decided to mount a Logitech USB webcam to the front of the rover and connect it to an Asus EeePC netbook running Linux. We are using the Unicap [1] library to interface with the camera and the gdk-pixbuf Library which is part of the GTK+ Project [2] to convert and save the unicap_data_buffer to a jpeg file. The jpegs can either be saved locally on the netbook or remotely on the server. The server then streams the jpegs to the web client with a customizable delay to simulate communication with the actual Mars Rover.

Though not a requirement, we decided to add an environmental camera to monitor the rover from a distance. We choose a DLink IP camera and communicate with it by using .Net and making Http web requests directly from the server.

2.4 Website User/Manager Role Authentication

Due to the nature of this project, the website needed to be secure so that users are restricted to using the website as we, the developers, project advisor and project sponsor, originally intended. In addition, a need for delegating access rules (roles) through a GUI existed so that someone without knowledge of the underlying system could delegate the who, what and when.

In order to provide for all this, we decided to go with a username/role system. ASP provides an excellent starting tool known as ASP Membership [3]. This groundwork allows developers to "greatly reduce the amount of code to authenticate users." We essentially grabbed the default database generated (tables and stored procedures) and customized it for the various subsections of our site. We also wrote a number of stored procedures to allow for password/username retrieval, role assignment and automatically/temporarily locking out users. The result is a secure and sophisticated login system tailored to suit our needs

2.5 Real-Time and Batch Movement

This requirement has a fairly simplistic concept behind it in that the end user shall be able to do one of two things:

1. Input a set of commands to be executed. Once executed, the rover would automatically interpret and react to each one.
2. Control the rover in real-time, so as commands are sent through the GUI the Rover will respond immediately (as soon as possible).

The development approach for this was to focus on batch movement. The reasoning behind this is that real-time movement is nothing more than some multiple of movement commands in some unit of time. Once we were able to send our first array of commands successfully, implementing real-time movement was nothing more than checking every unit-time (i.e. one second) for a queued list of movement commands to be executed.

2.6 Simple To Use GUI

Many Computer Scientists don't consider website development to lie within the realm of actual Computer Science, but most will agree that a quality GUI can be essential in the deployment of any project. Luckily for us, browsing the web has become common practice for all people of all ages from all around the world. It's cross-platform friendly

and (relatively speaking) hardware independent. This makes a web browser **the** ideal candidate for people to connect remotely to our rover.

Now, although choosing the type of operating system to host on the server was our decision, we had to be sure it wouldn't isolate any potential end users. In conjunction with standardized web development techniques, we decided to build the client side application using Java development tools. The JDK, as some of you may know, is a widely available and largely supported language toolkit for developing cross-platform software applications. The end user, whether they be on Mac, Linux or Windows and using Internet Explorer, Firefox or Opera (just to name a few) simply needs to have JRE installed on their machine in order to interface with our website applications.

Figure 2.1.1: The Main Page from the Remote Mars Rover Mockup Website

Figure 2.1.1 shows what our website will look like (in general) on every page. This basic layout of HTML, JavaScript and CSS has been thoroughly tested on both Internet Explorer and Firefox to ensure that end users can enjoy the same experience, without regards to their browser choices.

2.7 Signup and Queuing Feature

Because the original project intended for the Mars Rover Mockup to be part of a larger exhibit at the Journey Museum, people from all over the world (theoretically) could use

the rover at any time and a need for a signup and queuing feature was an obvious necessity.

On the website, once a user logs in, they have the option of looking at the Rover's current schedule, which is saved in the server's database. In order to control the rover, any particular user may apply for a time slice within any of the daily time constraints (set by the website administrator) where the administrator can either approve or deny that user's preferred time.

Similarly, the administrator can schedule (or set) "free mode" where a first come first serve scheduler is enabled. Global time limits can also be placed so that anyone gaining control of the rover cannot deadlock the system and prevent other users from getting their turn.

Lastly, at any point, an administrator may boot a user from the driver's seat, particularly if they see the user attempting to damage the rover or if the user engages in any other action against the rules.

3 Conclusion

Overall, the project has been a very interesting experience which has tested a wide variety of the knowledge we've learned throughout our time at school. We've written an assortment of applications in a variety of programming languages to bring hardware and software together into a fairly organized and intricate package.

References

[1] Unicap. <http://www.unicap-imaging.org/>
[2] GTK+ Project. <http://www.gtk.org/documentation.html>
[3] ASP Membership. < http://msdn.microsoft.com/en-us/library/ms998347.aspx>