

WAPcam – using a WAP application in student education

Frank Kargl, Torsten Illmann, Alexander Raschke, Stefan Schlott, Michael Weber
Department of Multimedia Computing
University of Ulm, Germany

INTRODUCTION

As a preparation for a lab course entitled “Web-Engineering” we had the idea to incorporate some practical work on the Wireless Application Protocol (WAP). Inspired by some earlier work from our colleagues from TECO (Karlsruhe) we decided to realize an interactive camera, controllable remotely via mobile phones: the WAPcam.

REQUIREMENTS

Before starting the actual implementation we clarified under what conditions such a project could be realized. From a technical point of view we had to define what functionality should be provided by the application. The camera should be turnable in both horizontal and vertical axis and manipulation should be possible in an intuitive way using the mobile phone. As the displays of today’s mobile phones are very small, we also need a zoom function in order to resolve smaller details. Finally as displays usually have only black-and-white displays and bad contrast we planed to integrate a contrast- or brightness adjustment.

Besides these technical aspects, which we had to consider for implementing our prototype, we had to take into account the organizational problems of using WAP in a lab course. Due to high connection costs it was evident that the students had to use a WAP simulator for development and testing most of the time [1]. As the course introduced a lot of different and in part new technologies (like HTML, CGI, Perl, PHP, Java, Servlets, Applets, XML, XSL, ...) and as WAP adds yet another set of complex protocols and concepts (WAP, WML [2], WMLS [3]), it was obvious to us that that the last assignment had to be especially appealing and exciting for the students in order to ensure their full participation. So besides the simulator work we wanted to facilitate the work on actual WAP phones that were provided by the adjacent development center of Siemens. To enhance motivation even further we constructed some kind of mars landscape around our WAPcam and hid some interesting riddles in there.

Given these reflections we were ready for implementing a first prototype to demonstrate the feasibility of our project.

SYSTEM ARCHITECTURE

Figure 1 shows the overall architecture of our WAPcam control. The camera is situated on a movable and controllable mounting [4]. Reading of camera input and control of the camera mount is done via a separate control

computer. The functionality of this system is accessible via the network using a Java RMI Interface [5]. This interface contains methods for positioning the camera to a direction specified either as absolute or relative angle. Furthermore you can reset the camera position to the starting position or read a single image from the camera. The RMI interface was provided by us and the students used the RMI functionality as a basis for their WAPcam implementation. That way they didn’t have to care about details of the camera control.

When many parties access the camera concurrently, this can lead to confusing behavior of the camera and may even harm the camera mount. To avoid such problems we have introduced a reservation procedure where a process must first register with the RMI server. Upon registration it receives a token that is needed for further communication. The token gets invalid if the user either logs-off or after a short timeout, preventing dead clients from blocking the system.

Next we have a second computer running an apache web server accessible via HTTP. Due to the very restricted bandwidth and client terminals available to any WAP application, it is highly recommended to do as much of the processing as possible on the server side. So in our case the function of the mobile phone is reduced to simply displaying the picture and sending the moving-instructions back to the server. Most of the application logic is implemented in the web server either as a CGI script [6] or as a Java servlet [7]. The following aspects need to be realized:

- Motion control according to the commands of the client.
- Retrieval of picture data from the control server and (intelligent) conversion into WBMP [8].

Especially the later should be done with regard as it is very tricky to reduce a high color and high resolution camera image to low resolution, black-and-white in a way that it remains recognizable. The source image is a 640x480 pixel JPEG image. On the receiver side we typically have only black-and-white displays with about 100x70 pixels. This dimension may vary significantly. In the preparation state, most of the people involved were very skeptical whether a reasonable result is achievable given such severe constraints.

So it is quite obvious that the image has to be specially enhanced while converting it into WBMP. With empirical tests we found out that the following steps result in an acceptable image quality:

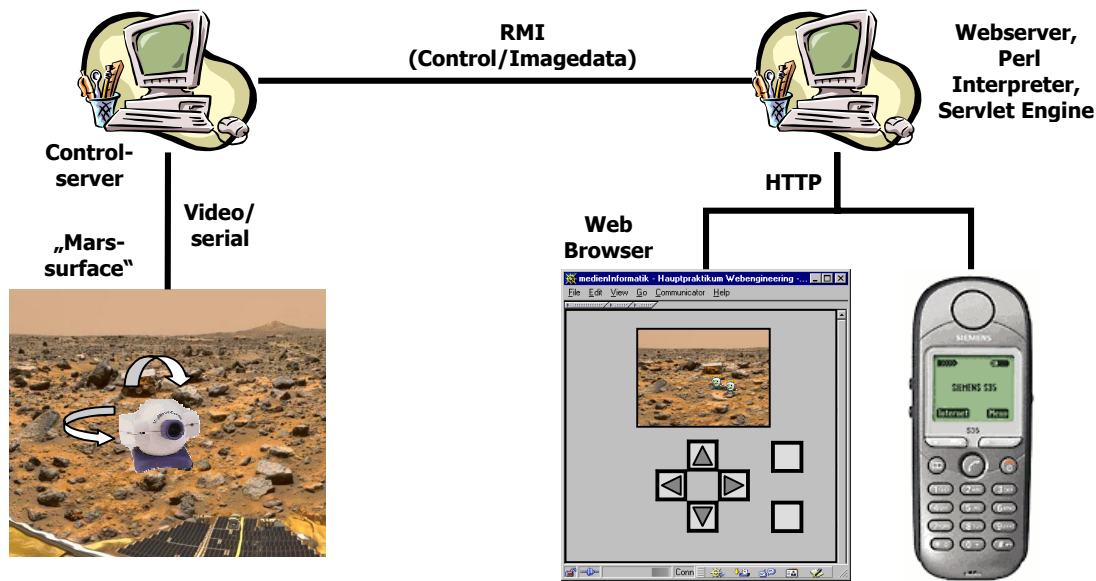


Figure 1. System Architecture

1. Conversion to grayscale image
2. Edge- and contrast enhancement
3. Clipping (this realizes the zoom function)
4. Scaling to client's resolution
5. Normalization of gray-scale image
6. Conversion to black/white using Floyd-Steinberg Dithering. An adjustable threshold realizes the brightness control.

These steps are illustrated in figure 2. Sometimes it is not that easy to predict the client's resolution. Some WAP Browsers/Gateways transmit the display size in an HTTP header field which can be used directly. If this doesn't work you can realize a database that maps the User-Agent header field to a list of known display sizes. If this fails too, you can only resort to a small sized default image or let the user choose from a number of different image resolutions. Here the standard bodies should clearly identify a common way to reliably transmit the client's capabilities to the server.

During our work we did not set up a WAP gateway ourselves. Instead we used the WAP gateways of our service providers (German D1 and D2 networks). During our tests we recognized no problems or limitations based on this decision.

IMPLEMENTATION PROTOTYPE

In order to give such an assignment in a lab course we need to demonstrate the feasibility of our design in a prototype. We decided to put the camera picture on a static WML page so the page itself can be cached at the

client. The image is generated by a Perl-CGI script. Motion control commands, zoom and brightness values may be handed to the script either as part of the URL (GET method) or in a <POSTFIELD> Tag (POST method).

In order to allow a convenient camera control we decided that the normal number of keys available to the WAP programmer is not sufficient. So we choose to use the number keys of the mobile phone. The keys 2,4,6,8 correspond to the directions up, left, right and down. With 1 and 3 you can control the zoom, with 7 and 9 the brightness of the image. Finally the key 5 resets the camera position, zoom and brightness to default. This solution is very usable and intuitively but suffers from two drawbacks:

- Assigning functionality to the number keys is not intended in WAP 1.1. So we used a proprietary extension of the phone.com browser [1] (ACCESSKEY attribute to <ANCHOR>) that is also part of WAP 1.2.
- The value of ANCHOR Tags show up as text lines under the image. After pressing a number key you eventually have to scroll up in order to see the image completely.

It is our believe that today's WAP/WML standard doesn't provide enough functionality to realize extensive interactive applications. It is foreseeable that developers will evade to proprietary extensions. The standards should be extended fast to include all reasonable I/O options of today's mobile phones (which surely includes the number keys).

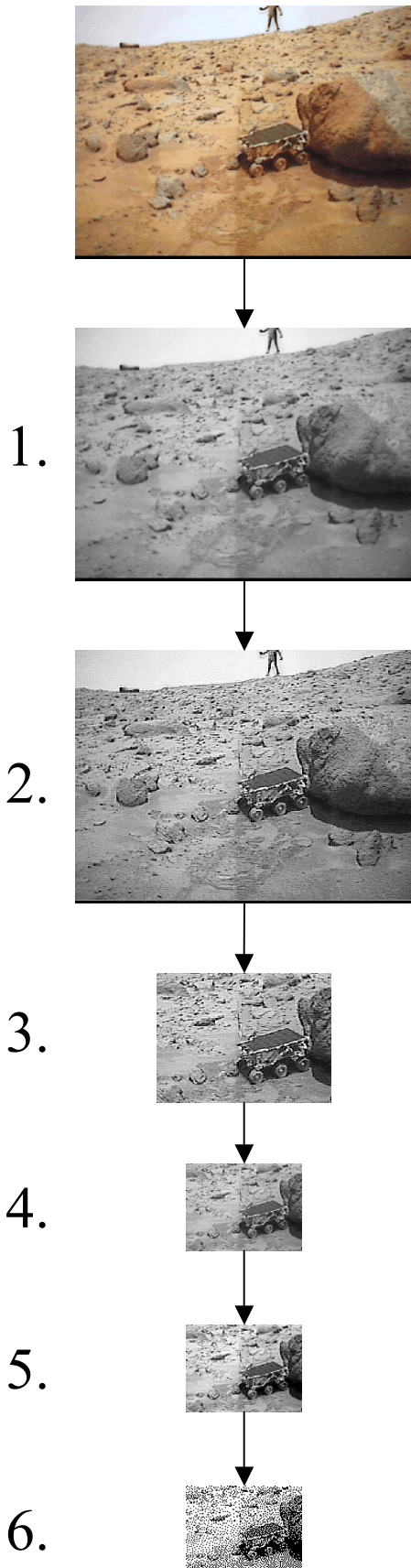


Figure 2. Image Conversion

Although implementation itself was quite straight-forward, we discovered a number of implementation errors or incompatibilities between different products that slowed down the implementation process significantly. The phone.com Browser in mobile phones evaluates all WML variables within WML attributes to the empty string. So it is not possible to hand such variable values to server scripts using GET methods (HREF="...?variable=\$value") The phone.com simulator works as expected. The suggesting solution is to use the POST method. Other problems detected are inconsistent caching behavior etc.

EXPERIENCES AND APPLICATION DOMAINS

First some technical aspects. Although initially many people thought that building an interactive camera using WAP was either impossible or at least not reasonable, the feedback from people that used our final prototype was throughout very positive. We found that the response of people depended to nearly 100 percent on the image quality. Given today's mobile phones we are able to reach a quality ranging from acceptable to good, depending on the motive. But future developments leading to devices with high-resolution color displays will solve this problem in the foreseeable future.

But even with the quality we can reach now there are a number of reasonable applications. A surveillance system could monitor private homes allowing their owners to check if everything within their houses is okay or if some water is draining or a break-in has occurred. Combined with other control functions (like e.g. shutters, irrigation systems etc.) this transforms the mobile phone into a comfortable control station. Of course there are problems waiting to be solved. We already mentioned the limited client devices. Furthermore today's GSM networks have a very limited data transfer rate. GRPS and UTMS promise improvements. Today using WAP is very costly, so services need to offer a solid benefit in order to be used. As price rates and billing models change although applications that offer a mere convenience will be used.

Anyhow our students that took the lab course were very eager and interested in the WAP exercise. The numerous flaws in the current WAP implementations and the limitations of the standard are of course a little bit problematic in an actual course, but given a high motivation of the participants, this isn't really disturbing the results.

REFERENCES

- [1] Phone.com: UP.SDK Development Kit, Release 4.1, 2000.
URL: <http://developer.phone.com/>
- [2] WAP Forum: Wireless Markup Language Specification, April 30, 1998.
URL: <http://www.wapforum.org/>

[3] WAP Forum: WMLScript Specification, April 30, 1998.

URL: <http://www.wapforum.org/>

[4] Directed Perceptions Inc.: Pan-Tilt Tracking Mount – C Programmer’s Interface, Version 1.0707b, 1995.

[5] Sun Microsystems Inc.: RMI Specification. URL: <http://java.sun.com/j2se/1.3/docs/guide/rmi/spec/rmiTOC.html>

[6] E. Wilde: Wilde’s WWW. Technical Foundations of the World Wide Web, Springer, 1999.

[7] Sun Microsystems Inc.: Java Servlets Specification, 2001.

URL: <http://java.sun.com/products/servlets/2.2/>

[8] WAP Forum: Wireless Application Environment Specification Version 1.1.

URL: <http://www.wapforum.org/>