A Graphical User Interface for a Wireless Intercom System

Magali Figueroa
SUPERB 99 Project Report

ABSTRACT

The goal of the project was to design and implement a graphical user interface (GUI) for an intercom system. The design of the GUI was influenced by the capabilities of an existing graphics library. The resulting design of the GUI also incorporated the capabilities and limitations of the hardware and software. A significant element of the project was implementation of ANSI 'C' code to design the graphical interface and emulate the environment of the intercom system. Another key milestone was the use of bench testing to gain a solid understanding of the graphics library and implement the design of the GUI.

1 INTRODUCTION

The project encompassed the design and implementation of a graphical user interface. The graphical user interface was created for a hand-held wireless Intercom unit. Independent thinking and learning drove the process to reaching the project goals. These goals involved research and design to create a graphical user interface that enables a user to operate the Wireless Intercom System. The graphical user interface is a prototype used for reaching the completion of the ideal system.

The following are the two main roles of the graphical user interface:

- Initial source of interaction with the system to control connections and communications with voice channels
- Component to display the progress of the project at conferences

2 PROJECT OUTLINE AND BACKGROUND

To understand the design and implementation of the graphical user interface required an understanding of the basics of the Intercom System. The Berkeley Wireless Research Center is currently developing the Intercom System project.

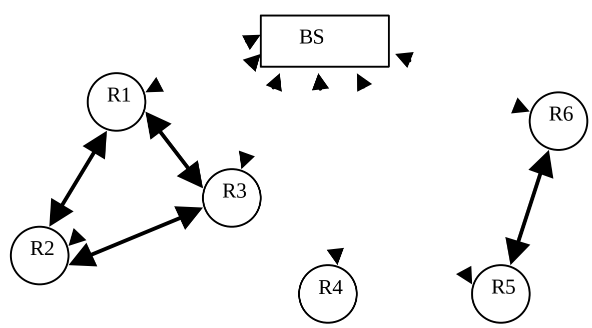

Figure 2: Intercom System Functionality

2.1 Intercom System Design

The Intercom project is aimed at building a prototype of a wireless communication system. The basic system is composed of a single base station and two or more remote nodes. The base station maintains control of the system by setting up connections while voice communication is between nodes. In Figure 2 the dashed lines represent the connection of the group of nodes with the base station. The solid lines represent examples of conversation from node to node.

2.2 InfoPad

A factor that led to the project's development was an earlier research project called InfoPad. The InfoPad is a device developed and completed in early 1997 by students and faculty in Electrical Engineering and Computer Science at the University of California, Berkeley. The Intercom project is based on principles developed for InfoPad, such as processor hardware architecture, display capabilities, and the physical enclosure.

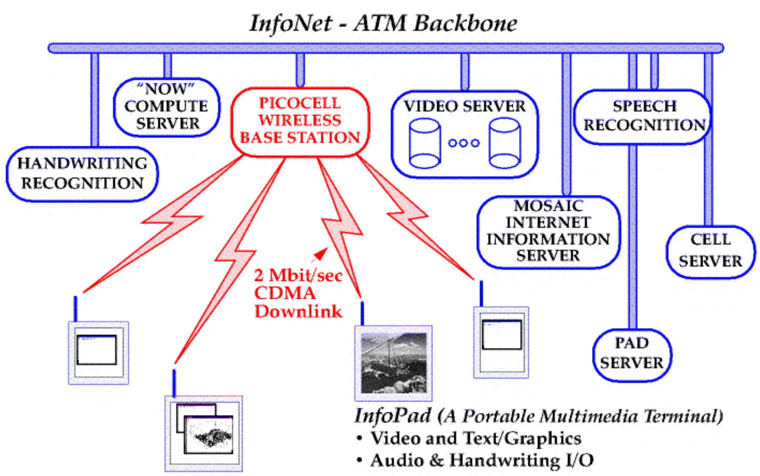

Figure 3: InfoPad Diagram

The features of the user device were important in the design of the test board. The components adopted from InfoPad are the ARM microprocessor and the Xilinx FPGA (Field Programmable Gate Array) architecture. For convenience in testing, the Intercom's test board was designed within the InfoPad's physical dimensions, such as the case, LCD, and connectors. The InfoPad's LCD panel and the touch-screen were the principle components addressed in the development of the graphical user interface.

3 DEVELOPMENT and PROCEDURES

The graphical user interface comprised an essential element in the development of the Intercom remote unit. Gaining an understanding of how the Intercom system works and the role of the graphical user interface were the backbone to the next set of tasks:

- Understand an existing graphics library in detail
- Design and implement the GUI
- Write emulation code for parts of the Intercom system that were not available, such as the base station
- Test the code on the bench using the Intercom test board

3.1 Graphics Library

The graphical user interface consists of a set of simple graphical primitives, representing buttons and lamps. The basic set has seven buttons and five lamps. Application of code to implement the buttons and lamps was first approached by gaining a solid understanding of an existing graphics library. Several steps were taken to achieve this goal. The graphics library was coded in ANSI ‘C’ and contained useful operations to draw images. These images include lines, arcs, text, color options, and styling features. Studying the functions resulted in gaining a basic understanding of their role.

***Example*:** Function: **gdImageLine(im, x1, y1, x2, y2, color);**

Action: Specifying two points in a coordinate system and the color of the line drew a line.

The technique of studying functions was complemented by the use of a text editor in UNIX and reading the body of functions. These techniques were helpful by providing an environment to analyze the graphics library. An example of this process was when a function in the graphics library was studied.

The first step was to see what arguments were present and if any other functions, structures, pointers, etc… were used within the function body. The text editor complemented this process of dissecting the function by accessing information about the function effectively. One technique was finding key words within the graphics library, such as variable names, and tracking their usage. Through this process a general idea of the use of most functions was grasped. The final product of the exploration was a detailed documentation of the principle functions in the graphics library. These techniques cleared the path to the next step: displaying the graphical objects on the LCD screen to see what they look like and how they behave.

3.2 Interface Design and Development

Analyzing the target graphical features, coordination of events, and the capabilities and limitations of the hardware and software influenced the design and development of the graphical user interface. The features of the interface were six buttons and five lamps with the option for an additional button and display to handle text.

The actions and properties of the buttons were the following:

- Connect to the base station for the local unit
- Disconnect the local unit from the base station
- Open a full-duplex voice channel to a remote unit
- Close the voice channel to a remote unit
- Send a query request to the base station

The typical sequence of actions for a session is the following:

- Connect to the base station
- Open a channel
- Talk
- Query the base station
- Close a channel
- Terminate connection

Some visual representations implemented in the interface were the lamps. The purpose of including lamps was to indicate a response to an action. For example, when a connection is requested, a lamp visually indicates to the user that the base station received the connection request. The lamps were polygon-shaped icons and were located near the buttons for ease of interpretation.

The buttons were also a source of visual representation for the user. They were designed as boxes with a word of text that indicate their principal function. Buttons to connect or open channels were constructed with a visually brighter design compared to the buttons to disconnect and close channels.

Their importance and the ease of recognition to the user as well as the flow of any actions defined the position and size of buttons and lamps. Boxes enveloped and grouped the buttons to open and close channels. From left to right the buttons were arranged to make a connection, followed by the option to open a channel then the option to close a channel. The button for disconnection was located below the connection button. The query button was placed apart from this chain of requests with a text box beside it to display the query results.

These elements defined the following design of the graphical user interface:

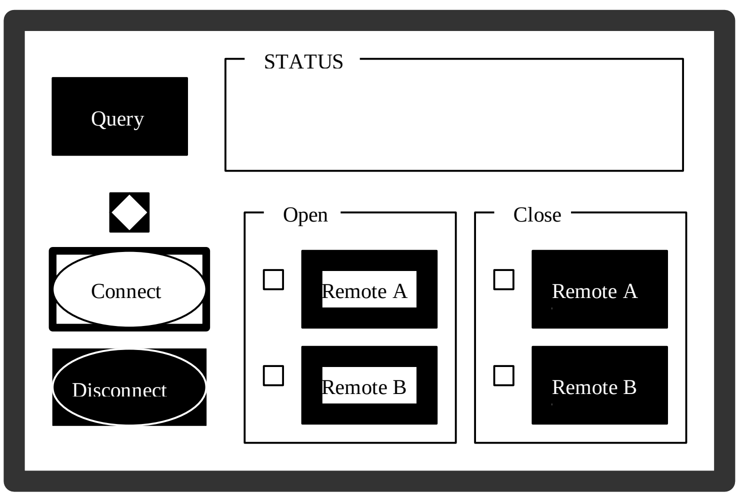

Figure 5: Sketch of the graphical user interface

3.3 GUI Interaction Mechanisms

Once the appearance of the interface was established, the next task involved implementation in ANSI ‘C’ code. Factors addressed were the sequence of events, the user-screen interaction environment, and the visual communication with the user.

This involved setting up procedures to receive and process the choice of the user. Since the test board was not available to use, code needed to be implemented that mimicked the touch-screen interface. Initially the code was set up to read a character inputted by the user that represents a button, go to that case scenario, and call functions from the graphics library to inform the user the request was processed. Ultimately the code needed to be set up to receive input through a touch-screen. Therefore instead of actually touching the display there were prompts on the laptop screen for a coordinate x and a coordinate y based on the screen axis. As a result, code was implemented to check if the input points were within the boundaries of a button and to assign a value for the choice that represents the request made. After this assignment the case setup mentioned previously was initiated. Through this process the code closely resembled the actual process of selecting a button on a touch-screen.

Another issue addressed was mimicking the communication to the base station once the case scenario was found. This was implemented by using stubs: small functions that mimic the behavior of the base station. This technique was effective in establishing a call to a function in each case scenario that accesses the base station, which in turn calls a function to communicate to the user that the base station received the request. The form of communication was the active blinking of the lamp corresponding to the choice made. The blinking action of the lamps was created using stubs that mimic timers normally provided by the operating system.

3.4 Bench Testing

To further broaden an understanding of the existing graphics library "hands on" bench testing was incorporated. Testing the graphics library required the use of the Intercom test board, the ARM Debugger for Windows (ADW), and Angel (a debug monitor on the test board). The ideal Intercom test board for the project was not accessible for use. As an alternative, an ARM processor development platform called the Brutus board was used to test the graphics library. The Brutus board was useful because it contained a testing platform for the StrongARM microprocessor and a LCD panel. Learning the concepts and operations of these materials was fundamental in the bench testing approach.

Laptop tools

- ARM Project Manager
- ARM Debugger
  - ADW

ARM software

- Angel
  - Exception Handler
- Application Code

Hardware

- ARM microprocessor
- LCD screen
- PROM (Programmable Read-Only Memory)

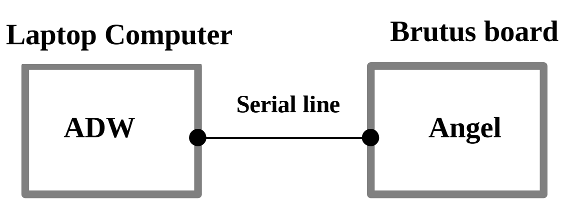

Figure 4: Bench testing environment

Procedures

- Turn on power to Brutus board
- Angel runs on ARM, on Brutus board
  - Angel is located in PROM at memory location 0
- Start ADW on laptop
  - Debugger communicates with Angel via serial port
    - Loads configuration code to RAM (Random Access Memory)
- Load Application
  - ADW and Angel cooperate in loading application code from laptop hard disk into RAM on Brutus board at 0x8000
- Execute Application
  - Debugger changes ARM PC (Program Counter) to 0x8000, causing the application to take control

Code was available to test some of the functions in the graphics library. A general use of drawing rectangles, arcs and text was supplemented by testing this existing code. Modifications of the code were made to test other features of the graphics library. These features included line styling, polygon drawing, and border implementation in polygons. Modifying or completely changing code also contributed to acquiring a broader knowledge of the general use of the functions available in the graphics library. These steps established a basic knowledge of the principle functions in the graphics library. The method of bench testing was trial and error. This process involved the use of existing functions in the graphics library to establish their corresponding images. Bench testing was instrumental in the development and completion of the graphical user interface. The procedure was used to test the design of and modifications to the graphics library.

4 CONCLUSIONS

The project encompassed independent thinking, analysis, reasoning, and flexibility. These elements lead to the establishment of techniques and mechanisms to create the graphical user interface for a Wireless Intercom Unit. Understanding an existing graphics library, implementing ANSI ‘C’ code, and bench testing enhanced the design and implementation of the graphical user interface. These goals contributed to building a GUI prototype for the intercom system.