PitA-Board PDA

QUARTERLY REPORT

Henry Kyle Bygott
Henry.Bygott@colorado.edu
Undergraduate Research Apprentice
L^{3}D

First Quarter, 2002

SCOPE

The PitA-Board (Participate In The Action Board) is a human-computer-interface with primary goals of increasing simultaneous interaction and group collaboration. It does this in a system utilizing simulations and dynamic information spaces to provide transparent access to problem solving resources in order to increase user understanding of the problem being dealt with.

INTRODUCTION

The PitA-Board is designed to be a human-computer interface that is much more transparent and easy to use than a typical keyboard and mouse (or even touch-screen) system. It is an improvement to the action space in the Envisionment and Discovery Collaboratory (EDC). The PitA-Board is, at its core, a chessboard sensor net designed by DGT Projects. This sensor has the ability to detect the type of any chess piece (plus 3 unreleased pieces – a total of 15) placed in any one of the squares (64 per board). Using these sensor nets and the tags from within the chess pieces, the PitA-Board system was designed using the Smalltalk programming language in Squeak.

Using the PitA-Board allows information to be seemingly imbedded into specific pieces. In the earlier EDC design, a user had to understand interface issues in the system to use the system effectively. In order to place a house, the user has to click draw, click house, and click where to put the house. With the PitA-Board system all one has to do to is pick up the house model and place the house wherever it is needed. Although the house model actually has no ability to actively store information, it does have one very important characteristic—a resonant frequency identified by the system as #8. This allows the PitA-Board to determine the piece's identity as a house, and take appropriate action.

The PitA-Board also allows for simultaneous interaction much better than the earlier version of the EDC. The previous EDC utilized a touch screen that only allowed one user to interact at a single time; if two tried to use the screen simultaneously it would place the touched point somewhere between the two users. The PitA-Board scans one square at a time, identifying pieces in that square. This means that any number of users can be using the PitA-Board at once, even using same pieces in certain cases. This is a great stride forward in simultaneous interaction.

PROGRESS SUMMARY

Strong headway was made in the first quarter of 2002 in integrating PDAs into the PitA-Board system. In January, much of the focus was on continuing abstractions in the Squeak implementation of the PitA-Board software. With the new abstractions completed, creating a simulation in Squeak for the PitA-Board is now even easier. A new statistics and control feature was also added allowing statistics to be viewed and more precise control to be had over each individual sensor pad in the PitA-Board system. In February, much of the month was spent diagnosing and repairing a hardware problem with one of the iPaq's expansion sleeves, and configuring the iPaq to properly connect to the L3D network. In March, however, all efforts focused on creating a PDA program to interact with the PitA-Board system. At its current state, the PDA can specify what type of residence to place on a numbered grid position, and a program on the PC can receive this data and interpret and display it. Although Squeak and the simulations therein have no part as of yet, a class has been written in Squeak to allow Squeak to connect to the Pebbles program (which runs the PC to PDA communications) and early PDA simulation interactions should be completed by the end of April.

PROGRESS SPECIFICS

- Further PitaController class
- Integrate “Walking Distance” simulation
- Create GridSim class
- Create Statistics and Control Interface
- Diagnose and replace PDA expansion sleeve
- Research and acquire Pebbles software
- Acquire Visual C++ and Embedded C++
- Create PDA PitA program
- Create PitA Pebbles plugin
- Create Pebbles class in Squeak

Furthering of PitaController Class

The PitaController is a class designed to abstract out the tasks of dealing with the serial port and low-level board interaction from the simulation. A simulation creates a PitaController instance and registers itself with the controller as a listener. The simulation then knows nothing of the actual board interaction, it is just informed by the controller when pieces are added or removed.

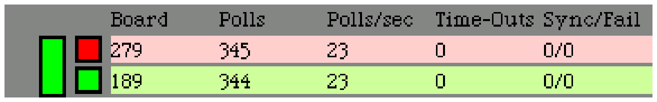
The PitaController is now able to automatically configure and properly interact with both the 2-board and the 4-board systems. It creates a coordinate system based on the setup it discovers and reports the information back to the simulation. Simulations have been generalized such that they are now able to configure and display properly for either board layout. This allows the same code-base to be utilized on either board layout and it will automatically configure itself to adjust.

Integrated “Walking Distance” Simulation

One of the most time consuming tasks in January was to integrate the “Walking Distance” simulation, written by Hal, into my generalized PitaController. For various reasons, most notably a rush to complete the simulation by CSCL, Hal and my code became quite divergent. Rectifying these differences allowed both the original (the “BusSim”) and the newer Walking Distance simulation to run from the same code base. The new simulation was also changed to allow an automatic configuration between the 2-board and 4-board layouts.

Creation of GridSim Class

With the walking distance and bus simulations running from the same code-base it became an obvious task to merge similarities between the two into a new parent class—the GridSim. This class assumes that a simulation will be running with the typical grid-based display and control systems that all PitA-Board simulations to this date have used. This class creates a 2-dimensional array to store board information, initializes basic graphics (the board and grid), initializes a PitaController and sets up to receive board events properly. This removes mundane and repetitive tasks from the simulations that use this system and allows more time to be spent on the simulation itself.



	Board	Polls	Polls/sec	Time-Outs	Sync/Fail
🟥	279	345	23	0	0/0
🟩	189	344	23	0	0/0

Statistics and Control Interface

The PitaController and SingleBoard classes (SingleBoard objects manage a single sensor-net in a PitA-Board system) now have a graphical representation, allowing administrator information and control. This interface allows statistics such as board polls/second, time-outs, and resynchronizations to be viewed in a table format. It can also singly or universally cause boards to ignore piece movements. In ignore mode, a board reports no piece movement to the PitaController. An example of when this would be useful is in the Walking Distance simulation, during a phase change. Since certain pieces have no meaning (or even counter-intuitive meanings) in different phases, the boards could be disabled while old pieces are collected and new ones distributed. This can prevent much confusion during the phase shifts of the simulation. In the above screenshot, board 279 is ignoring while board 189 is listening.

PDA Expansion Sleeve

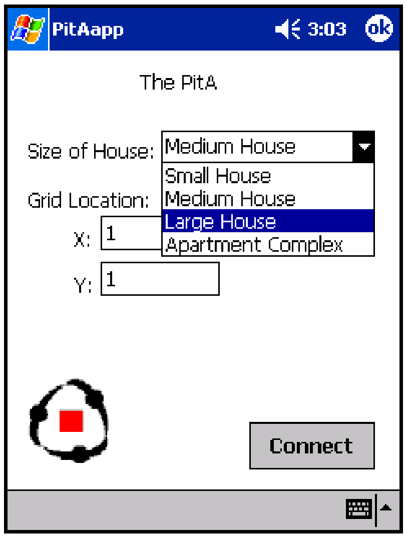
Upon installing the wireless Ethernet card into the new iPaq problems became readily apparent. The PDA intermittently detected the expansion card, and never successfully connected to the Internet. It was eventually determined that the expansion sleeve shipped with the iPaq was faulty. Once this was discovered and replaced, the iPaq could detect the network card reliably, but still couldn't connect to the Internet. After tweaking some obscure network settings on the iPaq this was repaired and the iPaq could finally connect to the network and Internet reliably.

Pebbles PDA Software

The Pebbles project at Carnegie Mellon University in Pittsburgh (http://www-2.cs.cmu.edu/~pebbles/) provides a useful start on the PDA project. They have created a piece of software (Pebbles) that monitors the network (and other connection techniques) for PDAs. A PDA sends a connection request to Pebbles over the network, along with a request to load a certain plugin. Assuming the plugin exists; the Pebbles software loads it and acts as a mediator. In this role it converts PDA messages to Windows events sent to the plugin and converts Windows events from the plugin and sends them to the PDA. The software is able to connect multiple PDAs concurrently, and allows them to run different programs. Using this software, custom plugins and PDA programs can be created which use the Pebbles program to communicate. This removes the difficult communication portion of the project, and allows focus to be placed on creating usable programs.

Create PDA PitA program

Upon learning how to properly utilize the Microsoft Foundation Classes (MFC) in Visual C++, a program to test communication from the PDA to the PC was created. It simply consists of a "connect" button, which changes to "disconnect" upon successful connection to the PC, dialog boxes to enter the grid location to be edited, and a drop-down dialog allowing to selection of a size of residency. Upon the selection of a residency size, the program sends grid location and house size information to the PC plugin. This allows a simple proof of concept to be created in the PitA environment. A screenshot of the program can be seen at right.



The PitA

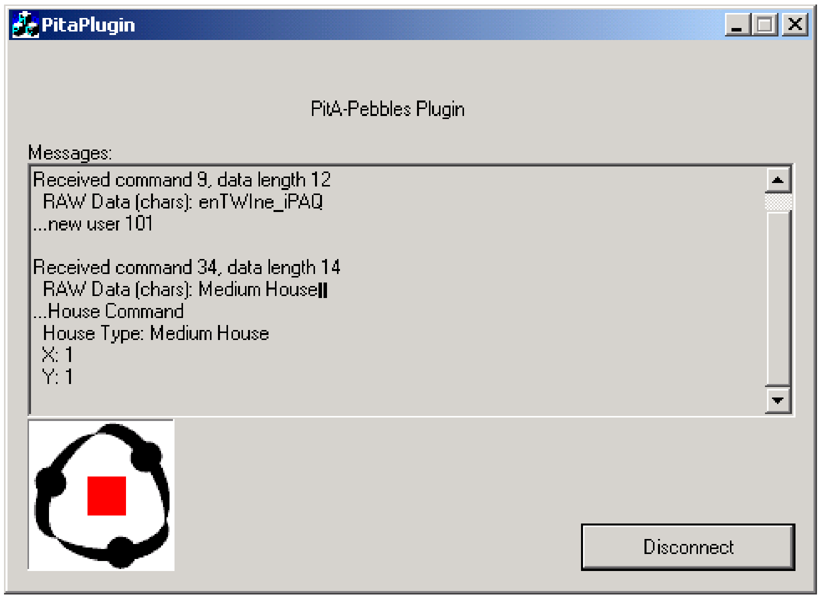
Size of House: Medium House

Small House
Medium House
Large House
Apartment Complex

Grid Location:
X: 1
Y: 1

Create PitA Pebbles plugin

A new plugin was created to investigate receiving messages from Pebbles. This plugin uses MFC for Visual C++ and connects to Pebbles using sockets. It identifies itself as the "PitA" plugin, and receives data from any PitA program running on connected PDAs. It is a simple test program, having only a "connect" button that connects through sockets to Pebbles and upon successful connection changes to allow for disconnecting. It then displays in a text window messages and commands from the PitA PDA users. Although this program is now obsolete due to investigations in connecting Squeak directly to Pebbles, it was an invaluable tool in learning more about the Pebbles software and Windows MFC classes. A screenshot of the program can be seen below.



```
Received command 9, data length 12
  RAW Data (chars): enTWIne_iPAQ
...new user 101

Received command 34, data length 14
  RAW Data (chars): Medium House
...House Command
  House Type: Medium House
  X: 1
  Y: 1
```

Create Pebbles class in Squeak

After the creation of the PitA plugin program, it became apparent that simply having Squeak connect directly to Pebbles would be the easiest way to facilitate communication between Pebbles and simulations running in Squeak. Following protocol used in the original PitA plugin, a class Pebbles was created in Squeak that allows socket communication to take place between Pebbles and the Squeak environment. Currently this class initializes a socket connection to Pebbles and sends Pebbles the appropriate information regarding its plugin, with no capability to send or receive PDA messages. Although this class is in its infancy, now that the basics of socket communication in Squeak have been discovered rapid progress can be made towards getting a simulation to interact with a PDA program.

Future Work Plan

In the near future, work will continue to focus on integrating the PDA into the PitA-Board system. Specifically, by the end of April, I would like to have a simple proof-of-concept done, at the least. This would involve enabling Squeak to send and receive PDA messages via Pebbles, and have the simulations running in Squeak communicate with the Pebbles object in order to process the PDA commands appropriately. Based on the current PDA program, the first proof-of-concept will likely be the ability to add or modify houses at specified grid locations. Once this is complete, the code will expand and gain complexity from this base.

One of the first expansions would be to allow simulations to send information back to the iPAQ. Without this a PDA user would have no feedback, and Pebbles would only be a one-way conduit of information. To create a truly useful tool there must be bi-directional communication. This will likely start as a simple "house added" or "house modified" type of dialog box, but once again, this will only form the base from which further features will be added. These two features will be completed by the end of April, and upon conclusion of these initial investigations, a more feature rich PDA interaction can be designed with the newly attained knowledge.

In the more long term, there are some other features that I would like to have, but am unsure of the feasibility of completing them within the month. Firstly I intend to investigate another interaction mode—the use of the PDA in close conjunction with the PitA-Board. This is the "personal reflection space" I have previously discussed. It would allow information tailored to a specific user to be displayed on the PDA. I intend to investigate this possibility by linking a specific piece tag to the PDA and using it as a "personal query" tool, allowing information to appear on the PDA as opposed to the main reflection space. Preliminary investigations into this will likely begin in mid-to-late April.

I would also like to implement a feature that only displays the active simulation space (i.e. the board projection) as opposed to the entire PC screen to the PDA, as most of the default Pebbles programs do. This would allow the PDA to display and interact with the simulation. It is likely this investigation will begin towards the end of April, and will perhaps entail using Squeak on the PDA itself such that simple morphic commands can be passed.

A more long-term goal is an annotation tool like the Pebbles "Scribble" program. This program allows a user to draw all over the PC's screen in a selected color, and even allows multiple PDA users to participate simultaneously. This seems like an obvious choice for an addition to the PitA-Board system, so that PDA users can diagram on the PitA-Board display. This feature would utilize the previously discussed simulation-screen-capture to display only the simulation space, as opposed to the entire screen as "Scribble" does. I would also like to investigate using morphic in Squeak to accomplish the actual annotation, such that it can be captured by the PitA-Board system. It is unlikely that this investigation will begin in April, however.

With these tasks to be dealt with, and surely more on the way, the PitA-Board PDA project should be making great strides in the near future. Barring large setbacks, a version of the PitA-PDA should be functional by the end of April.