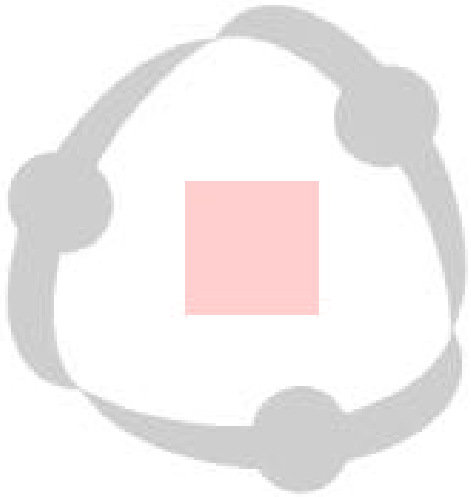
PitA-Board PDA

QUARTERLY REPORT



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

Third 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 embedded 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

Work through the third quarter of 2002 progressed well, however at a somewhat slower pace than accustomed too in previous quarters. The reason for this is the system dependence on the subsystems being modified, and the complexity of these dependencies. Most of the PitA-Board components were modularized in this quarter—removing any explicit dependence on other components (the current server side components are PitaController, PitaServer, VoteServer, and PitaSim). This allows each to be created and destroyed without affecting the overall system, and will allow the system to be more easily integrated into whatever overall EDC architecture is chosen. The difficulty of decoupling components (especially PitaController and PitaSim), however, led to a fairly long debugging process, and is still not entirely completed. The new version of the PitaSockets code, PitaSockets2, was created in this quarter, with the ability to send executable strings and encoded objects. This allows for much more varied and interesting remote interactions, and allows the server (which generally has much more computational power) to create and setup objects prior to sending them to the client. Attempting to use this system on the PDA made problems with the previous Squeak image apparent, and a new VM and image was acquired for the PDA. Throughout this entire process, large-scale changes took place in the PitA-Board system, and in the very last part of the quarter, the mess created within the Squeak image during the development phase was cleaned up.

PROGRESS SPECIFICS

- Create voting functionality
- Create version 2 of PitaSockets
- Investigate architectures
- Modularization of components
- Install functionality into PitaSockets2
- Port PitaSockets2 to PDA
- Load Squeak 3.2 onto PDA
- Voting system modified

Create voting functionality

One of the features common in the older editions of the EDC yet not present in newer iterations was an ability to log votes. In order to add this ability, the simulation was made to keep track of multiple votingIssue objects, each of which contained its name, a description of the issue and a vote count. The object also keeps track of who has voted, such that each person can only vote once. Currently the only object with the ability to vote is the PitaClient object on the PDA, but this is easily expandable.

Create version 2 of PitaSockets

As features were added to the PitaClient throughout July, the ByteStreams communication became more and more inconvenient. A new version of the PitaServer and PitaClient was created that utilized StringSockets for communication. These have the ability to send simple strings, executable code, or encoded objects. This allows for greater freedom in communication, and allows the PitaClient to operate without the need to load all necessary code beforehand. At the current time, these new abilities are only used for voting. PitaClient2 has no knowledge of the voting system or any of the supporting code. PitaServer2 sets up a voting dialog, complete with issues and vote buttons, and sends the entire object to the PitaClient, which then displays it. Through this received object, the PitaClient interacts properly with the voting system.

Investigate architectures

With the continued expansion of the EDC and PitA-Board code, the communication between objects is becoming very confused. With this in mind, a new discussion has cropped up in the PitA-Board group about new communication architectures. The architecture I favor is a bus architecture, where every device has access to every message produced, and can act accordingly. This has the advantage of being very modular, but the disadvantage that messages cannot be intercepted and reinterpreted before devices receive the first message.

Modularization of components

All of the proposed EDC architectures have one thing in common: all components are modularized and operate independently. However, many of the other objects in the EDC have interdependencies. Work began in late July to modularize components, starting with the voting system. Through this endeavor the voting system should become dissociated with the simulation, and may be used even when a simulation is not present in the system. This is the eventual goal for all objects in the PitA-Board system.

In August the effort to better modularize the PitA-Board system continued with the creation of a modularized version of the PitaServer system. The new version runs independently of the simulation and watches the network for PitaClient connections. Upon finding a connection, PitaServer creates a PitaSClient, representing the server's view of the client, and properly sets it up with its connection, name, and listeners. It assigns listeners to the PitaSClient by searching Squeak for all instances of a simulation, and assigning active simulations to the

client's listeners. This situation is currently temporary, allowing communication while the final architecture of the EDC is decided upon. The PitaServer then releases the PitaSClient and has no more interaction with it. All PDA communication is performed through the PitaSClient.

The PitaController and PitaSim systems had become tightly coupled over the previous months, and in order to conform this system to the new modularized design, these components needed to be decoupled. This was successful in September, and all interdependence was removed. This also has a number of interesting repercussions. Firstly, the PitA-Board coordinate space is now unrelated to the simulation coordinate space. In previous iterations, the simulation would ask the PitA-Board what its dimensions were and conform to them, however they now have no knowledge of each other's sizes. This is a vital part of the modularization, and will allow more interesting interactions—dragging and zooming with the PitA-Board, for instance. Although visualization is still handled by the simulation, this will also lend itself more readily to a decoupled visualization system in the future. This decoupling also allows the PitaController to be restarted without restarting the simulation and loosing all the information contained therein. This was theoretically possible in the old system, but only with an intimate knowledge of both the Squeak and the PitA-Board systems, and great patience.

Install functionality into PitaSockets2

Until this point, PitaSockets2 was simply a demonstration piece, with no useful functionality to speak of. Starting in August, actual functionality was added to the system, including the reimplementation of the wand utility. This, in addition to the previously implemented voting system, allowed the testing of nearly all modes of PitaSockets2 communication. Wand requests were made using a simple <CMD><DATA> string, identical to the communication utilized in the previous version of PitaSockets. The user was informed of a wand grant/denial through an object send. This displayed a dialog box in the PitaClient identical to the one in previous iterations, except that the actual *object* was sent in this example, whereas previously the PitaClient required pre-programmed knowledge of the dialog box. The voting system, unchanged from previous iterations, was sent via a communicating-object that was setup on the server by the votingServer system. A communicating-object is identical to an object, except that it is configured with the socket connection when it is recompiled on the client such that it can communicate to the server. With these modes programmed, it became important to test them on an actual PDA.

Port PitaSockets2 to PDA

Upon porting the new version of the PitaClient to the PDA image a number of problems became apparent. Firstly, the PDA image did not contain any StringSocket code, requiring a treasure hunt to file in the correct classes and their dependencies. Once this was completed, the PitaClient code was loaded onto the PDA image and tested on the PC. The wand system—which tests simple strings and non-communicating objects—worked perfectly. However problems were discovered with the voting system—which tests communicating objects. The Squeak system is unable to recompile classes that aren't present in its own image into objects. This means that the code for the VoteDialog and VoteIssues used would have to be preprogrammed into the PDA image—exactly what was hoped could be avoided. This is unfortunate, yet not a total loss, as objects based on items the PDA *does* know about can still be easily constructed on the more powerful PC—such as the wand request box.

Load Squeak 3.2 onto PDA

After a rather poor experience initially porting PitaSockets2 to the PDA system, it was realized that the PDA actually had enough memory to load a full-blown 3.2 image. A new Squeak VM for the ARM was downloaded and a fully functional 3.2 image was loaded onto the iPAQ. This solved many of the problems with treasure hunting for interdependent classes within Squeak, and allowed objects to file into the system correctly. Although loading an ~12MB image onto the PDA is certainly not a realistic final system scenario, it creates a valuable testing and proof-

of-concept platform without the time-consuming difficulty of creating a custom image of a more appropriate size.

Voting system modified

With the success of loading Squeak onto the iPAQ, the voting system was tested and modified to attain a partially functional system on the PDA. Although it is not a complete system yet (interaction between the client and server is minimal at best), the required complexity of the voting interaction makes it one of the most useful for defining client/server interactions and interfaces. At the moment, both client and server maintain a voteIssue object and it is an interesting topic to think about whether this is a correct model of interaction and these two objects need to somehow stay synchronous, or if there should be some central object which both poll and synchronize to. Complex models of interaction such as this are why the voting system is a primary focus at this time.

Future Work Plan

In the immediate future, work on the PDA interaction system will continue. This will likely happen primarily through the voting system, due to the complexity if its interactions. With the newly discovered fact that the PDA must contain the source code for any objects it receives, the usefulness of the send object capability must be examined more closely. It is my suspicion that this ability will still prove useful, and remain a vital part of the interactions. The executable string functionality has not been used to its full capabilities either, and through iterating over the design of the voting system a clear and easily understandable interaction can be created.

With the discovery that the PDA must contain the source code for any objects it receives, a system for the PDA to automatically download and file in required code will be investigated. It is unavoidable that the PDA will have to have at least the code for a PitaClient object, however through having the PDA download most code on demand, version conflicts will be minimal. I suspect that this will be a fairly easy task to accomplish once the compiler commands which actually execute a code file-in can be discovered.

If these tasks go well and a fairly clear model of interaction between the PDA and PC emerges, code on a serious application can commence. This would likely happen in conjunction with a serious push for a new EDC demonstration for L^{3}D's DLC space. The creation of a serious application would likely result in a new set of problems and weaknesses within the PitaSocket code, that would need to be debugged and updated.

Tasks remaining on the to-do list, but currently on a "back-burner" are the administrator PDA and annotation using a PDA. The former would allow the facilitator of an EDC session to access all the features he/she would normally use on the PC from the PDA. This would allow the facilitator to be more mobile and dynamic in the interactions. Annotation using the PDA would allow users to draw directly on the simulation space as in the "Scribble" demo by the Carnegie Mellon Pebbles project. This feature is awaiting a semi-final version of the PitaSockets code to emerge to base its code on.