Iowa State University
Digital Repository

Graduate Theses and Dissertations Graduate College

2010

AQUA-G: a universal gesture recognition framework

Jay Roltgen
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/etd

Part of the Psychology Commons

Recommended Citation
Roltgen, Jay, "AQUA-G: a universal gesture recognition framework" (2010). *Graduate Theses and Dissertations*. 11430.
http://lib.dr.iastate.edu/etd/11430

This Thesis is brought to you for free and open access by the Graduate College at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

المنارة للاستشارات

www.manaraa.com

AQUA-G: a universal gesture recognition framework

by

Jay William Roltgen

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-majors: Human Computer Interaction; Computer Engineering

Program of Study Committee:
Stephen Gilbert, Co-major Professor
James Oliver, Co-major Professor
Phillip Jones

Iowa State University

Ames, Iowa

2010

Copyright © Jay William Roltgen, 2010. All rights reserved.

المنارة للاستشارات

www.manaraa.com

TABLE OF CONTENTS

LIST OF TABLES vii

LIST OF FIGURES viii

ACKNOWLEDGEMENTS xi

ABSTRACT xii

CHAPTER 1. INTRODUCTION 1

1.1 Gestures 2

1.2 Overview of related work 3

1.2.1 Gesture-enabled hardware devices 3

1.2.2 Gesture recognition 4

1.2.3 Existing gesture recognition frameworks 5

1.2.4 Software for input processing 13

1.2.5 Interfaces with multiple input devices 14

1.2.6 Collaborative environments 15

1.3 Motivation for AQUA-G 15

CHAPTER 2. AQUA-G 18

2.1 Design goals 18

2.1.1 Performance 19

2.1.2 Programming language independence 20

2.1.3 Platform independence 20

2.1.4 Ease of customization 21

المنارة للاستشارات

www.manaraa.com

2.1.5 Provide a set of unified standard gestures 22

2.1.6 Support for component-centric vs. global gestures 22

2.2 Tools 22

2.3 Key concepts 23

2.3.1 Events 23

2.3.2 Gestures 26

2.3.3 Creating gestures and events dynamically 27

2.3.4 Regions 28

2.4 Event flow 28

2.5 System decomposition 29

2.5.1 EventProcessor 31

2.5.2 InputProtocol 31

2.5.3 InputDeviceConnection 32

2.5.4 GestureServer 33

2.5.5 GestureEngine 33

2.5.6 GlobalGestureLayer 35

2.5.7 Region 36

2.5.8 ClientConnection 37

2.5.9 Utilities 37

2.5.10 Gesture and event creation 39

2.6 Client application interaction 40

2.6.1 Initialization state 40

2.6.2 Running state 41

2.7 Supported input devices 43

2.7.1 Windows and Linux mice 43

2.7.2 HP TouchSmart touchscreen 43

2.7.3 iPad and iPhone 43

المنارة للاستشارات
www.manaraa.com

2.7.4 Wii Remotes 45

2.7.5 Cricket location sensors 46

2.7.6 Sparsh-UI input devices 46

2.8 Supported gestures 47

2.8.1 Drag gesture 47

2.8.2 2D rotate gesture 47

2.8.3 Zoom gesture 48

2.8.4 Flick gesture 48

2.8.5 Double-click gesture 48

2.9 Supported event translators 48

2.9.1 Get handID gesture 49

2.9.2 Kinetic gesture 49

CHAPTER 3. CASE STUDIES 50

3.1 A first application 50

3.2 A user-identification based application 54

3.3 LABET unmanned vehicle controller 57

3.4 The flick gesture 59

3.5 The kinetic gesture 60

3.6 The iPad and iPhone driver 61

CHAPTER 4. EVALUATION 63

4.1 Initial developer survey 63

4.2 User evaluation 66

4.2.1 Method 67

4.2.2 Example student 1 68

4.2.3 Example student 2 69

4.3 Discussion of evaluation results 70

المنارة للاستشارات

www.manaraa.com

CHAPTER 5. CONCLUSION 74
5.1 Primary advantages 74
5.2 Limitations 75
5.3 Future work 77
APPENDIX A. DEVELOPING SOFTWARE USING AQUA-G . . . 79
A.1 Developing a custom event 79
A.2 Developing a custom gesture 85
A.3 Developing an input device driver 88
A.4 Developing a client application 93
APPENDIX B. WAYFINDER 105
B.1 Title and authors 105
B.2 Abstract 105
B.3 Introduction 106
B.4 Related work 107
B.5 Wayfinder 109
B.5.1 Hardware and software 110
B.5.2 Features 111
B.6 Experiment 1 118
B.6.1 Method 118
B.6.2 Performance metrics in the simulated mission 119
B.6.3 Training 121
B.6.4 Results 123
B.7 Experiment 2 - map manipulation 125
B.7.1 Results 126
B.8 Limitations 127
B.8.1 Hardware 127

المنارة للاستشارات
www.manaraa.com

B.8.2 Situational awareness 128

B.9 Discussion 128

B.10 Conclusions and future work 129

B.11 Acknowledgments 130

APPENDIX C. FORMS 131

C.1 Informed Consent 131

C.2 Interview Protocol 131

BIBLIOGRAPHY 136

للاستشارات المنارة

www.manaraa.com

LIST OF TABLES

Table 4.1 Major emergent themes in initial survey. 65

Table 4.2 Student 1's responses to interview questions. 72

Table 4.3 Student 2's responses to interview questions. 73

www.manaraa.com

LIST OF FIGURES

Figure 1.1 Three-layer depiction of MT4j architecture. 7

Figure 1.2 Three-layer depiction of PyMT architecture. 10

Figure 1.3 Three-layer depiction of Sparsh-UI architecture. 11

Figure 1.4 Three-layer depiction of Tisch architecture. 12

Figure 1.5 Comparison of existing gesture recognition systems and frameworks. 16

Figure 2.1 AQUA-G event flow. 29

Figure 2.2 AQUA-G class diagram. 30

Figure 2.3 EventProcessor interface. 31

Figure 2.4 InputProtocol class. 31

Figure 2.5 InputDeviceConnection class. 32

Figure 2.6 GestureServer class. 33

Figure 2.7 GestureEngine class. 34

Figure 2.8 GlobalGestureLayer class. 35

Figure 2.9 Region class. 36

Figure 2.10 Client Connection class. 37

Figure 2.11 Filesystem class. 37

Figure 2.12 AquaSocket class. 38

Figure 2.13 EndianConverter class. 39

Figure 2.14 EventFactory class. 39

www.manaraa.com

Figure 2.15 GestureFactory class. 40
Figure 2.16 The HP TouchSmart. 44
Figure 2.17 The Apple iPad. 44
Figure 2.18 The Nintendo Wii Remote. 45
Figure 2.19 A Cricket sensor. 46

Figure 3.1 A first AQUA-G application. 51
Figure 3.2 A user interacting with the sample application using the HP TouchSmart. 52
Figure 3.3 A user interacting with the sample application using a standard mouse. 52
Figure 3.4 A user interacting with the sample application using a Wii Remote. 53
Figure 3.5 A user interacting with the sample application using an iPad. . . 54
Figure 3.6 A user interacting with the sample application using both an HP TouchSmart and a standard mouse simultaneously. 55
Figure 3.7 Two users playing the AQUA-G Game of Life. 56
Figure 3.8 The LABET unmanned aerial vehicle. 57
Figure 3.9 Architecture for the AQUA-G solution to LABET UAV control. 58
Figure 3.10 The LABET avatar simulated in OpenGL. 58
Figure 3.11 The iPad device driver. 61

Figure 4.1 Method for conducting developer study. 67

Figure 5.1 A comparison of gesture recognition systems 76

Figure B.1 The Vigilant Spirit controliInterface 109
Figure B.2 The Wayfinder application. Visible are vehicles (circles), threats (triangles), waypoints (flags) and control panels (left). 110
Figure B.3 The 25.5” HP TouchSmart computer. 111

www.manaraa.com

Figure B.4 The Stantum SMK 15.4” multitouch device 111

Figure B.5 Wayfinder’s video control panel 113

Figure B.6 Wayfinder’s waypoint panel 114

Figure B.7 Classification pie menus. Threats were classified by type, behavior, size, and severity, all of which were described to participants in a training video. 116

Figure B.8 Threats displayed in Wayfinder. 117

Figure B.9 Multitouch experience among participants. 119

Figure B.10 Wayfinder instructing a participant to place waypoints. Note the small circular waypoint targets with the numbers inscribed. . . . 120

Figure B.11 Situational awareness prompt. 122

Figure B.12 Results of waypoint task. Users were able to set waypoints an average of 6.01 seconds faster using the mouse. 125

Figure B.13 Map manipulation task. The participants manipulated the small black rectangle so that it filled the screen with the arrow pointing up. 125

Figure B.14 Results of the map manipulation task. Participants completed 6.6 more manipulations with the multitouch interface. 126

Figure C.1 Informed Consent Document, Page 1. 132

Figure C.2 Informed Consent Document, Page 2. 133

Figure C.3 Interview Protocol, Page 1. 134

Figure C.4 Interview Protocol, Page 2. 135

المنارة للاستشارات

www.manaraa.com

ACKNOWLEDGEMENTS

I would like to thank all those who have supported me throughout my academic career. My major professor, Dr. Stephen Gilbert, has been a constant source of encouragement and enlightenment, and has always guided me in the right direction.

I would also like to thank my peers at the Virtual Reality Applications Center who have supported, mentored, or worked with me over the past two years, especially Mike Oren, Tony Ross, Rob Evans, Peter Wong, and Tony Milosch. I have developed a great deal of knowledge and experience as a result of working with these incredibly talented individuals.

Finally, I would like to thank my wife Katie, who has been an unfailing source of encouragement and support, even as I worked many long hours to finish this thesis. Her assistance in the final weeks proofreading and reviewing this thesis were invaluable, and I could not have finished this without her love and support.

المنارة للاستشارات

www.manaraa.com

ABSTRACT

In this thesis, I describe a software architecture and implementation which is designed to ease the process of 1) developing gesture-enabled applications and 2) using multiple disparate interaction devices simultaneously to create gestures. Developing gesture-enabled applications from scratch can be a time-consuming process involving obtaining input from novel input devices, processing that input in order to recognize gestures, and connecting this information to the application. Previously, developers have turned to gesture recognition systems to assist them in developing these applications. However, existing systems to date are limited in flexibility and adaptability. I propose AQUA-G, a universal gesture recognition framework that utilizes a unified event architecture to communicate with a limitless variety of input devices. AQUA-G provides abstraction of gesture recognition and allows developers to write custom gestures. Its features have been driven in part by previous architectures and are partially based on a needs assessment with a sample of developers. This research contributes a scalable and reliable software system for gesture-enabled application development, which makes developing and prototyping novel interaction styles more accessible to a larger development community.

المنارة للاستشارات

www.manaraa.com

CHAPTER 1. INTRODUCTION

Gestural interfaces are becoming increasingly popular. With the recent proliferation of touchscreen devices including the iPhone (3), interactive kiosks (51), and gesture-enabled video game controllers such as the Wii Remote (52), users are becoming more familiar with gesture-enabled interfaces. However, interaction styles for these devices are often tied to the hardware platform; users expect certain gestures on the iPhone and certain gestures on the Wii Remote. While this approach is to some extent necessary due to ergonomic constraints, a software framework which would allow for the exploaration of simultaneous use of gestures across multiple devices could offer novel interaction opportunities.

To achieve this end, I address the following problem. Rapidly prototyping a gesture-enabled interactive application is generally not trivial. Building and testing an application involves a great deal of work to communicate with input devices, recognize gestures, and react to gesture-releated events appropriately. The research presented in this theses addresses this problem by providing a universal gesture recognition framework that is capable of communication with a limitless variety of input devices. The goal of this work is to simplify the developer's tasks by providing standard gesture recognition capabilities and to allow testing of applications with a variety of input devices with little development overhead. This should allow developers to experiment with other interesting input methodologies that they may not have considered previously due to a perceived complexity of developing an application which utilizes other types of input.

AQUA-G is an implementation of the described software architecture. As of July 1,

المنارة للاستشارات
www.manaraa.com

2010, AQUA-G communicates with several multi-touch devices, standard mice on both Windows and Linux, the Wii Remote (52), the iPad (2) and iPhone (3), and a Cricket location system (56) which provides user identification on a multi-touch table.

Support is planned to enable interaction with the ZCam 3-D camera (27), haptic devices such as the Phantom (39), multitouch tables which offer user ID recognition through the use of overhead cameras (17), and tangible user interfaces similar to SLAP Widgets (74).

Ideally, AQUA-G will make existing interaction techniques accessible to a wider audience. Its support for multiple simultaneous input devices will allow for systems which utilize multiple methods of input, such as touch-tables augmented with user identification (17), tangible user interfaces, and novel interaction styles which utilize multiple inputs, to be united together under a common architecture.

1.1 Gestures

A question that must be addressed before presenting this research is "What is a gesture?" Generally, we think of interactive gestures as those which are associated with touchscreen-enabled devices. However, the idea of gestural input is not limited to multi-touch screens. Dan Saffer describes gestural input in a much broader sense, in that anything users do in an attempt to perform some action be considered a gesture (64). Typing on a keyboard, moving a mouse, blinking an eye, waving a hand, and seemingly limitless possible interactions can all be described as "gesture-based" interactions.

Generally, it is our goal as software and interface developers to make these gestural interactions as natural and effortless as possible. We want the user interface to "come alive" and interact with users in a way that is easy to learn and easy to understand. In order to do this and create interactions which utilize gestures, it is necessary to create software to recognize input from input devices and process that input to output gestural

المنارة للاستشارات
www.manaraa.com

information. It is precisely this process which this research addresses and attempts to streamline for all future developers wishing to utilize gestural interaction in their applications.

1.2 Overview of related work

Bill Buxton and many others in the human-computer interaction community have researched gestural interfaces (tablet pen-based, finger-based, hand-based) since the early 1980s (12). Furthermore, some researchers have focused on how users learn new gesture-based interaction techniques (6; 1). Still others have researched improving the accessibility of new gesture interaction techniques to new users (9). Over the years, much work has been devoted to creating software architectures and frameworks to allow developers to take advantage of these new styles of input (18; 58; 45; 57; 16). I will examine and comment on existing gesture recognition software frameworks and assess how this architecture, AQUA-G, might be informed by each design.

1.2.1 Gesture-enabled hardware devices

Gesture-based input devices are ever-increasing in popularity and ubiquity. Often as users, we may not even realize the presence of these devices, but they are nevertheless there and present. Gestural interfaces such as automatic sliding doors, automatic paper towel dispensers, and automatic hand dryers can all be considered gestural interfaces under Saffer's definition. With the recent proliferation of touchscreen phones, interactive kiosks, and gesture-enabled gaming controllers, these types of gestural interactions are becoming more and more commonplace.

Bill Buxton provides a concise summary of several touch systems which have been created over the years, going as far back as the 1960s (12) . Perhaps one of the most famous early gestural interaction systems was VIDEOPLACE, developed by Myron Krueger and

المنارة للاستشارات
www.manaraa.com

others, and published in 1985 (36). Krueger's work was ahead of its time and had a great deal of influence in this field. He addressed issues such as whole-body gestural input, co-located and remote collaboration between users, and multi-touch gestures with his VIDEOPLACE system.

In more recent years, researchers produced a system called DiamondTouch (16) in 2001 which is capable of detecting multiple points of touch. The system can detect pressure of each touch as well as identify which user created that touch. In 2005, Jeff Han (24) contributed to a spark of interest in multi-touch interaction by showing how multi-touch devices could be created by anyone with relatively little cost. Since 2005 there has been an explosion of multi-touch and gesture-enabled hardware devices (49; 3; 47; 42; 53) which have led to increasing accessibility of these devices to developers interested in creating gesture-enabled applications.

1.2.2 Gesture recognition

A great deal of research has been performed in the field of gesture recognition, and a full examination of this body of research is outside the scope of this work. The proposed software architecture servers as a framework for gesture recognition, so that gesture recognition algorithms and techniques they may unified under a common architecture and made accessible to a wider developer community.

Still, it is important to recognize the broad scope of gesture recognition. In general, gesture recognition can be classified into two categories: static gesture recognition and dynamic gesture recognition. In this work, I will define static gesture recognition as any form of recognizing gestures which evaluates a sequence of states of input data in order to output some meaningful result at the end of gesture processing. Examples of static gestures would include the Graffiti handwriting recognition feature on many Palm OS devices, Windows pen flicks available on tablet editions of the Windows operating system, and many others.

المنارة للاستشارات
www.manaraa.com

Static gesture recognition involves evaluating a sequence of states and comparing them against some target data using an algorithm called a classifier. There are many methods by which this is done; among the most popular classification methods involve using Hidden Markov Models (66; 73), Support Vector Machines (8; 38; 13), and neural networks (5; 48).

Dynamic gesture recognition, by contrast, involves recognizing gestures which must have some action in real-time, such as drag, scaling, or rotation gestures. Dynamic gestures output the result of processing every time input data is received, as each gesture event corresponds to a single state of the input data. Dynamic gesture recognition is used extensively in multi-touch systems to provide a natural, interactive interface. It is generally less based on classifying input data and attempting to match it to a target expected action (as in most classifiers), and more on evaluating users' actions in real time in order to to augment the user experience.

1.2.3 Existing gesture recognition frameworks

Much work has been proposed which aims to provide a framework for gesture recognition, especially for multi-touch interfaces, but all implementations that precede AQUA-G have been limited by design or by consequence. For example, some systems are limited by programming language (45; 57), in that they can only communicate with applications written in a specific language. Others are platform-dependent (32; 4), or do not provide the ability to write custom gestures and input device drivers (53; 49). These existing systems have informed the design and architecture of AQUA-G.

This section reviews existing gesture recognition systems and describes each system's strengths and weaknesses in an attempt to explore and compare all existing solutions which are similar to Aqua.

In an attempt to unite all of these disparate systems, it is useful to have a common framework for comparing these disparate systems. It is best to think of gesture pro-

المنارة للاستشارات
www.manaraa.com

cessing architectures in terms of components and more specifically, layers. In addition, through research and evaluation of these systems, I have found that all gesture processing software architectures have three main logical components, or layers. Individual software architectures may use the layers in different ways, or may utilize different means of communication between the layers, but each layer is always present in some form. The three layers are:

- The application layer
- The gesture processing layer
- The input device layer

In each figure presented in this section, I will show the three layers as they relate to the existing systems. While the original authors of these systems did not describe their architectures with these specific terms or layers, I have attempted to re-describe their architectures accurately using this three-layer framework for ease of comparison between systems.

Ideally, the three layers would function independently of one another, so applications, gestures, and input device implementations can be changed easily without affecting the other layers. However, fully separating functionality in this way can introduce more complexity into a system. For this reason, these three layers are provided slightly differently in some frameworks, but they are always present, I we will discuss each software architecture in terms of these layers.

We will first examine two systems which are intended for use in the multi-touch application domain: Multi-Touch For Java (MT4j) (45) and PyMT (57). These systems are robust and flexible but are tied to a specific programming language. AQUA-G should be available for any programming language and any platform, but may also be greatly informed by the architecture of these existing systems.

المنارة للاستشارات
www.manaraa.com

MT4j is a Java gesture processing system. As shown in Figure 1.1, MT4j implements the four layers described, though uses different names for each layer and divides the input device layer into two layers: the input hardware layer, and the input hardware abstraction layer.

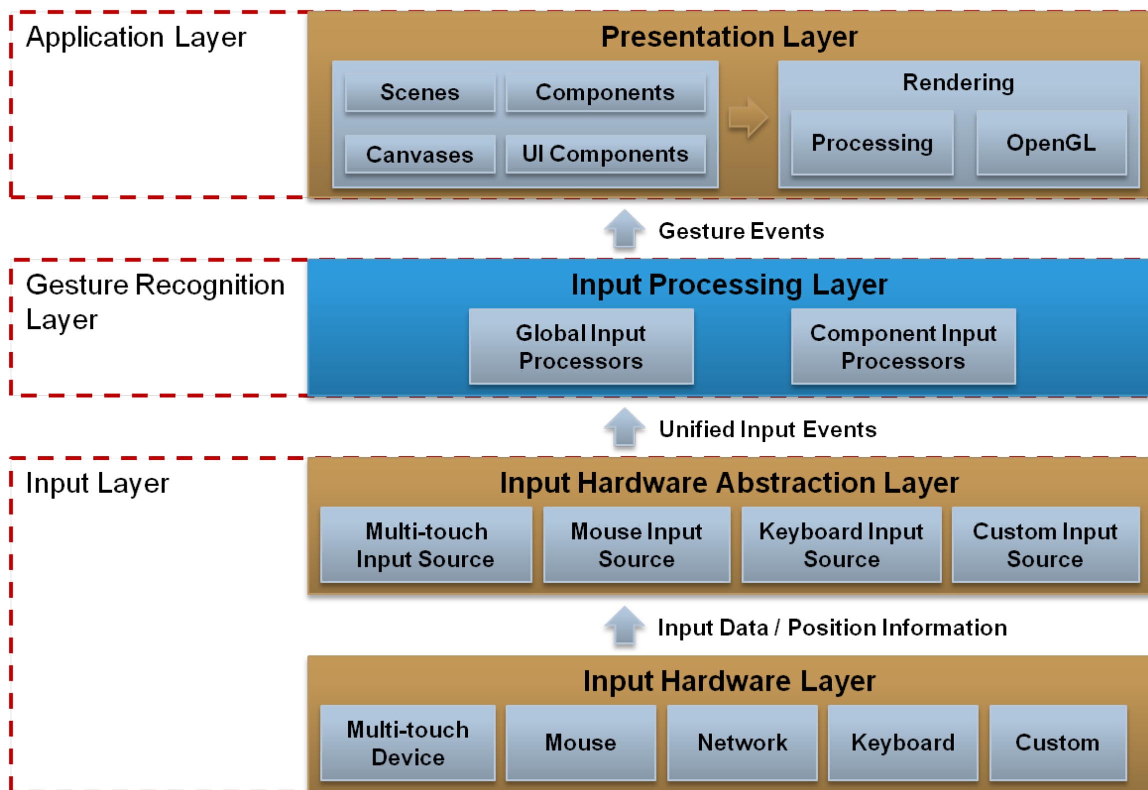

Figure 1.1 Three-layer depiction of MT4j architecture.

In addition to providing an architecture for gesture processing, MT4j also provides a set of multi-touch enabled Java GUI widgets in the presentation layer, which allows developers to quickly and easily develop multi-touch enabled Java applications by simply sub-classing the provided GUI widgets.

This is one of the great strengths of MT4j, because a common problem in multi-touch application development is that most GUI frameworks provide widgets which do

المنارة للاستشارات
www.manaraa.com

not know how to respond to native multi-touch events. For example, the widget may know what to do with mouse events, but cannot interpret a zoom or rotate event unless the input device converts or maps these multi-touch events to standard mouse events. Since MT4j provides a widget framework, the widgets provided are able to handle zoom and rotate events without this conversion or mapping. This provides the benefit of greatly simplifying the development of Java multi-touch enabled applications.

Additionally, MT4j provides a space for global gestures which may not be associated with a particular UI component. For example, gestures such as waving at the display or shaking a controller are hard to associate with any component in the UI, but may be designed to trigger certain actions. As a result, these are considered global gestures instead of component gestures and are handled appropriately.

In addition to this elegant handling of global vs. component gestures, MT4j also provides an input hardware abstraction layer which converts all device inputs into unified input events. This allows MT4j to be compatible with a variety of input devices. MT4j provides input device abstraction for multi-touch devices, mouse devices and keyboard devices. This allows MT4j a great deal of flexibility when it comes to handling input from new input devices, and allows a developer to create new input devices by conforming to MT4j's abstraction layer. This idea of unified events informed the architecture and design process for Aqua, because Aqua aims to be compatible with a variety of input devices, just like MT4j.

Thus, to create a multi-touch application using MT4j, one simply uses the provided widget classes. Gestures can be created and registered using the provided API, which makes it easy to receive multi-touch gesture events on components which extend the provided widgets. MT4j provides the ability to create custom gestures, though it is not clear as of this writing how to do so.

MT4j provides a clean interface for creating multi-touch Java applications, and makes it very easy for developers to create multi-touch Java applications. MT4j has some

المنارة للاستشارات
www.manaraa.com

weaknesses, particularly in that it provides little support for extending the framework to other GUI widget frameworks or creating applications in other languages besides Java. Additionally, as of this writing it is not clear how to create a new custom gesture for use with the framework. AQUA-G will improve on these weaknesses in part through cross-language compatibility and by providing dynamic loading of gestures so that developers do not have to recompile the main source code.

Another gesture recognition system which is tailored for a specific programming language is PyMT (57), tailored for Python applications. Similar to MT4j, PyMT aims to provide a robust and easy-to-use multi-touch gesture recognition platform. However, in PyMT, the three layers discussed above are implemented differently than in most gesture recognition frameworks (See Figure 1.2). This is because in PyMT, gesture processing is performed in each individual UI widget. This means that in order to receive gesture events for a particular gesture, the widget you wish to utilize these events must extend the UI widget that appropriately processes those events. Thus, gesture processing in PyMT is inherently tied to the widget, which makes it easy to understand, but rather inflexible.

Since python supports multiple inheritance, developers can create widgets which extend the functionality of two or more other widgets and use gestures from each event. PyMT allows developers to create multi-touch applications by simply subclassing the provided multi-touch enabled widgets.

PyMT is written in python and provides a graphics library which is built on top of OpenGL, meaning the widgets in PyMT are very responsive and fast. It is a reliable platform for developing python applications, but providing the custom GUI widget library and tying the gesture processing to these widgets means that all of its gesture-processing functionality is tied to the Python programming language.

Sparsh-UI (58) is an open source gesture recognition framework for multi-touch hardware and software. The system strives to move away from the programming language

المنارة للاستشارات
www.manaraa.com

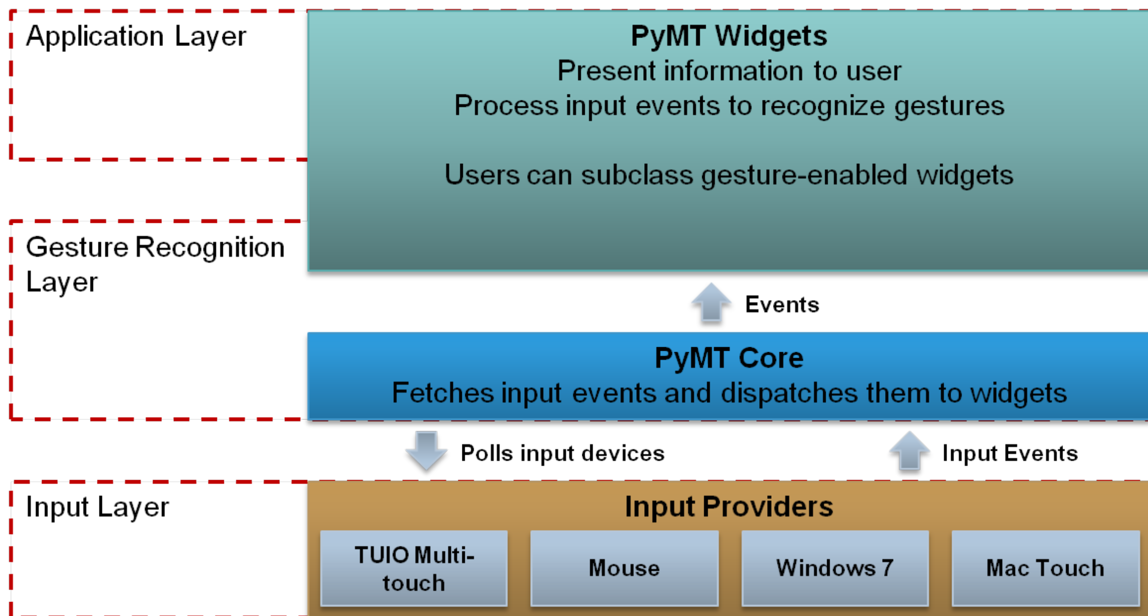

Figure 1.2 Three-layer depiction of PyMT architecture.

dependence exhibited by the two systems described above. However, this comes at a cost: Sparsh-UI does not provide a set of multi-touch enabled widgets. This is because it aims to abstract gesture processing from the particular widget library used, so that the developer can receive gesture events regardless of whatever programming language he or she is using.

This cost is difficult to mitigate when developing a system which is intended to be cross-language, as providing a set of GUI widgets for every programming language would be a large undertaking. Aqua will share this limitation with Sparsh-UI, at least at first. After Aqua begins to be widely used, it is possible developers will contribute Aqua widget frameworks for individual programming languages.

Sparsh-UI also implements the three-layer system described in the opening paragraph of this section, and is shown in Figure 1.3. Sparsh-UI is cross-platform, is provided in a Java and C++ version, and utilizes TCP/IP sockets to provide communication between

المنارة للاستشارات
www.manaraa.com

the three layers. As a result, Sparsh-UI is compatible with all programming languages that support socket communication. Applications for Sparsh-UI have been written in Java and C++ to date. Sparsh-UI also provides excellent documentation for developers who wish to create a client application or new gesture.

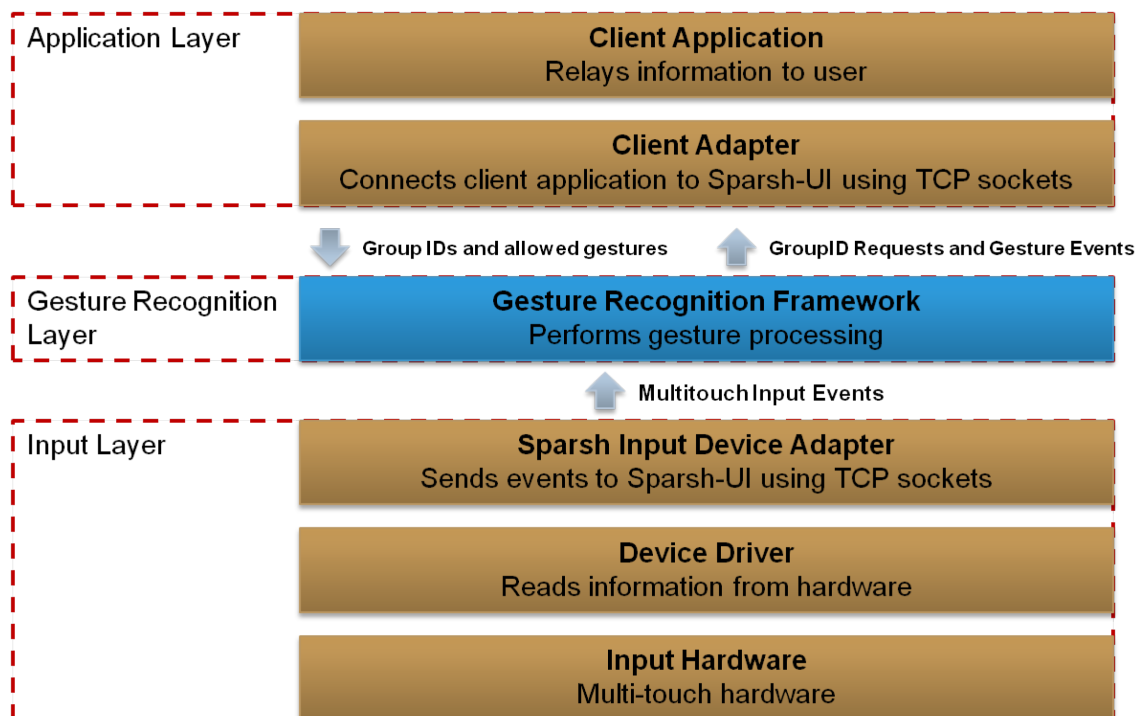

Figure 1.3 Three-layer depiction of Sparsh-UI architecture.

Sparsh-UI is created exclusively for multi-touch applications. The API is not extensible to accepting other types of input, such as may be received from systems that may provide additional information about the touches such as height, width, userID and other custom information. However, Sparsh-UI is a well-developed and mature software system that is an excellent solution for applications which utilize multi-touch input exclusively.

Another multi-touch software architecture which is very similar to Sparsh-UI is Tisch

المنارة للاستشارات
www.manaraa.com

(18). Tisch is a unified multi-touch software architecture which aims to support gesture processing for multi-touch systems. The architecture of Tisch (Figure 1.4) is nearly identical to Sparsh-UI, but it adds a transformation layer which calibrates input device data, and an interpretation layer, which processes gestures. This layer is also present in Sparsh-UI in the Gesture Recognition Framework, but in Tisch this layer behaves slightly differently. In Sparsh-UI, the gesture recognition framework asks the client adapter for the component that the touch point ocurred over each time a new touch point is received. In Tisch, the regions-of-interest are communicated to the interpretation layer by the widget layer upon initialization, so that no communication is necessary until events are generated.

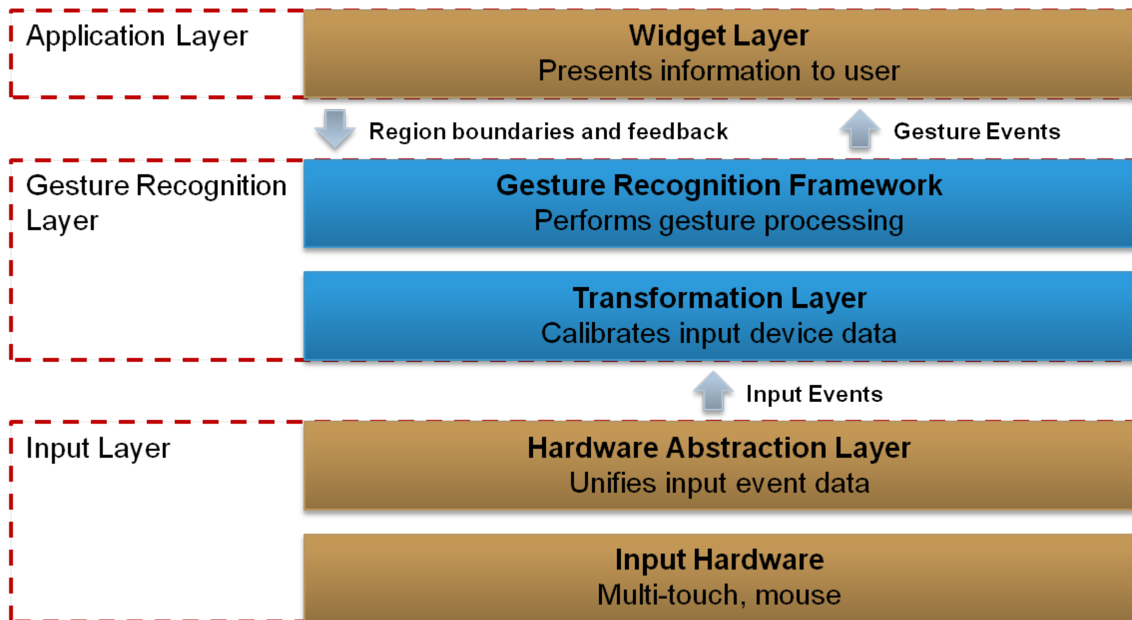

Figure 1.4 Three-layer depiction of Tisch architecture.

This is an important distinction and was a crucial decision in the design of AQUA-G. Since AQUA-G is intended to be extensible to 3-D input devices and client applications, we do not know how developers will choose to utilize the 3-D space. Clearly, input

المنارة للاستشارات
www.manaraa.com

device space coordinates in 3-D space may not map well to coordinates in the client application space. Thus, we have decided that it is best to ask the client application to locate the appropriate region each time a new 3-D coordinate is received. Our model is thus similar to that used by Sparsh-UI.

1.2.4 Software for input processing

The four systems described above, Tisch, Sparsh-UI, PyMT, and MT4j, are perhaps the most robust software architectures which utilize the three-layer model described above. There are many other systems which intend to allow developers to create multi-touch applications by providing interfaces to process input from hardware, but their gesture processing systems are not customizable as in the above systems. Examples of these systems include:

- The NextWindow Two-Touch API (49), which allows developers to write applications for NextWindow touchscreens.
- The N-trig API (53), which allows developers to write applications for N-trig touchscreens.
- The Windows 7 Touch SDK (32), which allows developers to utilize the multi-touch functionality built in to Windows 7. The Windows 7 touch SDK is rather flexible, but does not allow the addition of new gestures, and applications must utilize the Windows API for creating GUI interfaces.
- The iPhone SDK (4), which allows developers to write multi-touch enabled iPhone applications.
- Community Core Vision (54), a cross-platform solution for tracking touch points in optical touch systems.

المنارة للاستشارات
www.manaraa.com

The software architecture presented in this research is highly informed by existing gesture recognition frameworks, of which there are many. A key contribution of this research is to present a framework that is not tied to a specific input device or type, but may be compatible with a variety of input devices which provide dissimilar types of events. The system should allow for a limitless variety of input events, so that it will not become obsolete as new input methodologies are developed. The key advantages to using a system such as this will be the ability to test interactive applications with a variety of input devices with very little development overhead.

1.2.5 Interfaces with multiple input devices

One of the key contributions of the Aqua framework is that it provides support for mulitple input devices and multi-modal gesture recognition. This is a major contribution because of the radiply expanding research area of multi-modal interaction systems. A great deal of user interfaces are beginning to use multiple means of input, such as combining keyboard, mouse, touchscreen, hand tracking, and many other forms of input.

One such application involves using a large touchscreen system and multiple keyboards and mice. Cheng et al (14) describe a system which is capable of recognizing both multi-touch and mouse input and utilizes the Multi-pointer X software to provide multi-touch functionality for the X window system.

Another application of multi-modal interaction is described by Hartmann (25) and involves user interaction with a large multi-touch display that is augmented with multiple keyboard and mice combinations. The researchers go on to describe several intriguing and new interaction styles which utilize these multi-modal user interactions. Dealing with multiple input devices appropriately can be one of the key strengths of a system like Aqua.

Another research area which is related to this idea of multi-modal interaction is the area of tangible user interfaces (28; 74; 72; 34). Tangible UIs usually involve touchscreen

المنارة للاستشارات
www.manaraa.com

interaction combined with tracking of physical objects on top of the screen or table. An example of a tangible UI is detailed by (74) in which researchers augmented a touchscreen user interface with physical UI widgets which users could turn, slide or type with. The widgets noticeably improved user interaction.

An important part of developing effective tangible user interfaces is that the tangible objects must be tracked and their locations and state must be reported to some software application. To date, all of these systems implement their own custom method of doing this reporting. By using a common framework, it would be possible to greatly reduce the development work necessary to create such systems.

1.2.6 Collaborative environments

Much research involves investigating collaboration of users in tabletop environments (29; 70; 30). Often, research in collaborative environments is benefited by a technology which provides user identification. Differing means of doing this user identification are presented by (16), (25), and (17). A unified system might be able to accept touch input from some hardware device and hand positions from the same or another separate device, and output user identification information for each hand and touch point. The ability to switch out a touchscreen quickly and retain the hand-tracking technology makes AQUA-G very powerful in the flexibility it could allow experimenters wishing to evaluate different means of doing this type of user id association.

1.3 Motivation for AQUA-G

As shown, to date, many gesture recognition frameworks are designed for a specific application domain: multi-touch interaction. Other frameworks are only available for use with a specific programming language or platform. For a side-by-side comparison of some of these existing systems, please refer to Figure 1.5. As shown in the figure, all of

المنارة للاستشارات
www.manaraa.com

the existing systems have some limitations.

		Next-Window API	Windows 7 Touch SDK	iPhone SDK	MT4j	PyMT	Tisch	Sparsh-UI	AQUA-G
Platforms	Windows	●	●		●	●	●	●	
	Linux				●	●	●	●	
	Mac OS X			●		●	●	●	
Available app development languages	Java				●		●	●	
	Python					●	●	●	
	C++	●	●				●	●	
	C#	●	●				●	●	
	Other			●					
Input devices	TUIO				●	●	●	●	
	Multitouch	●	●	●	●	●	●	●	
	Mouse				●	●	●		
	Wii Remote						●		
	Windows 7 Touch		●		●	●		●	
	Other								
Features	Provides standard multitouch gestures	●	●	●	●	●	●	●	
	Allows custom gestures				●	●	●	●	
	Allows custom input devices		●		●	●	●	●	
	Provides gesture-aware UI widget set			●	●	●	●		
	Support for non-touch events								
	Dynamically loaded gestures								

Figure 1.5 Comparison of existing gesture recognition systems and frameworks.

These limitations become especially important as gestural interaction expands beyond multi-touch devices. Novel input methodologies utilizing devices such as 3-D cameras (27), Wii Remotes (52), tangible interfaces (74) and other interesting gesture-based interactions, are increasingly used to explore novel interaction styles (37; 25; 74). With the advent of new technology that is allowing more precise sensing of human behavior, gestural interactions can become increasingly complex as we can utilize whole-body

المنارة للاستشارات
www.manaraa.com

interaction.

AQUA-G is intended to provide developers with a way to rapidly develop and prototype gesture-enabled applications. As such, the software architecture should allow developers a great deal of flexibility when it comes to implementing an application.

Furthermore, AQUA-G should provide user interface and interaction researchers with a way to easily test different means of input with their application. Prior to the proposal of this architecture, I wrote an application called Wayfinder to evaluate the differences between multi-touch and mouse interaction for command and control applications. A paper that describes application is given in Appendix A. In order to write this application, I was required to spend time writing additional code to talk to both the multi-touch and mouse device.

Had this application been developed using AQUA-G, I would have not incurred this development overhead, and furthermore, the research could be extended to evaluate other types of input devices with little to no additional software development.

AQUA-G will be of great use to those wishing to perform similar research, and will make developing gesture-enabled applications much easier and more accessible for future software developers.

المنارة للاستشارات
www.manaraa.com

CHAPTER 2. AQUA-G

Developing a software framework for gesture recognition will involve identifying design goals, defining requirements, designing software, and implementing that software. In this chapter, I will begin by identifying several design goals for AQUA-G. I will continue by describing the tools used in creating the software, and discuss key concepts related to the architecture. I will give a system decomposition and and describe the software architecture and components which allow the design goals to be accomplished. Finally, I will describe input devices and gestures which the software framework currently supports.

2.1 Design goals

AQUA-G is intended to be used widely by other software developers. To ensure that it would satisfy the needs of the eventual users of AQUA-G, I developer user-centered design goals which were created based on the results of the initial survey described in chapter 4. These design goals were driven in part by this survey, and based on limitations of previous systems. The importance of flexibility, scalability and performance is paramount when designing a software framework that deals with user input and is intended to be used widely by other software developers. I will describe each of the design goals that I developed for AQUA-G, and also describe how each of those objectives has been satisfied.

المنارة للاستشارات
www.manaraa.com

2.1.1 Performance

Of crucial importance in any software system that deals with user input is performance. As I developed the design for AQUA-G, it was crucial to keep performance in mind when making all major architectural decisions.

Some decisions that were made to ensure good performance were:

- Minimized network communication between components to the extent possible.
- Utilized sorted data structures to store large amounts of data in order to minimize search time.
- Used native platform libraries instead of relying on third-party code to avoid any unnecessary overhead.
- Used C++ for implementation and compiled for each platform, rather than using an interpreted cross-platform language.

For network communication, I chose to utilize TCP sockets. This decision to use TCP sockets rather than UDP sockets was motivated by a desire for stable and reliable communication, and for this decision, I have traded thoroughput for security. As an important example of the benefit of the use of TCP sockets, if an input device driver or client application crashes or disconnects unexpectedly, AQUA-G can observe this through the broken stream connection and clean up appropriately. If UDP sockets were used, AQUA-G would be unable to perform this action without the use of a time-out on the connection, which would put unnecessary constraints upon the input device driver or client application developer.

Furthermore, lost information when dealing with user input can be difficult to detect and handle appropriately. For example, if AQUA-G receives a stream of touch input events, and the touch death (also known as up or release) is lost on the connection, it

المنارة للاستشارات
www.manaraa.com

is difficult for AQUA-G to decide whether the user is holding their finger very still in one place, or whether they have actually released it but the up event was lost. For this reason, the decision was made to utilize TCP sockets and accept the performance loss.

2.1.2 Programming language independence

For AQUA-G, developers should be able to write input device drivers and client applications in any language of their choice. The limitations of doing this have been discussed in chapter 1. However, this decision will allow developers greater flexibility when using the framework and will allow them to choose a language which suits their needs, without having to learn a different gesture recognition framework. Programming language independence was accomplished by utilizing TCP sockets for communication between layers, rather than tying input devices and client application to the framework through method calls. As a result, any language which supports socket communication is compatible with AQUA-G.

2.1.3 Platform independence

Prior to the design and proposal of AQUA-G, potential developers were surveyed about the usefulness of a system like it. This survey is described in chapter 4. The developers expressed a desire that AQUA-G should run on Windows, Linux and Mac OS, so that they could choose a platform which would be best for their specific application. I have chosen to utilize C++ for the implementation, and AQUA-G will need to make use of platform-specific functionality such as filesystem access, threading and TCP sockets. In order to achieve this cross-platform functionality, I had to decide whether to rely on third-party libraries which abstract the OS-specific functionality into wrapper classes, or write the cross-platform functionality myself. I wanted AQUA-G to have as few dependencies as possible, so I chose to write the cross-platform functionality myself. Though this involved more work during implementation, the result was code that I fully

المنارة للاستشارات
www.manaraa.com

understand and am able to test and debug easily. Furthermore, it allows AQUA-G to stand alone without the need to link to additional third-party libraries. The code abstracts the cross-platform functionality under invariant interfaces so that AQUA-G can run without knowledge of its underlying platform on Windows, Linux and Mac OS. Currently, support has been developed for Windows and Linux. Support for Mac OS is left for future work.

2.1.4 Ease of customization

In order to encourage adoption of AQUA-G, it is crucial that developers be able to customize the framework by adding custom input devices, gestures, events, and client applications. This functionality should be readily apparent and allow for easy customization to suit the developers' needs. To help provide this functionality, AQUA-G will dynamically load gestures and events at runtime. This adds complexity to the framework, but provides increased flexibility for developers wishing to customize the framework by creating their own gestures and events. To accomplish this, gestures and events are compiled into shared libraries, which can be compiled for the correct platform easily throught the use of a cross-platform build tool such as SCons (35) or CMake (33). This feature allows developers to compile new gesture and event classes without making any changes to the framework source code, which eases integration of developer-created gestures and events.

Furthermore, good documentation has been provided so that developers can create these new types of gestures and events quickly and easily. I have provided sample projects which allow developers to learn how to create their gestures and events quickly. The sample projects are pre-configured to build the appropriate shared libraries, which AQUA-G can then use at runtime.

المنارة للاستشارات
www.manaraa.com

2.1.5 Provide a set of unified standard gestures

AQUA-G is not simply be an empty framework which allows for customization by developers, or it will not be able to obtain initial acceptance. It should provide a standard set of gestures that developers can code applications to, so that developers do not have to start from scratch when they wish to create a new application. In order to satisfy this design goal, AQUA-G includes basic zoom, rotate, and drag gestures. Support for additional gestures is planned for future work. It is my hope that developers will contribute additional gestures which will be available for use by the entire development community. Since AQUA-G allows extensive customization, unique and creative gestures can be created quickly and easily.

2.1.6 Support for component-centric vs. global gestures

As described in chapter 1, MT4J provides elegant handling of component-centric vs. global gestures. Similarly, AQUA-G should distinguish between component-centric gestures such as "drag," "zoom," and "rotate," and global gestures such as "turn off," "wave," "turn on," etc. The latter types of actions should not need to be associated with a particular region or UI component, and would be considered global gestures. Providing this distinction allows application developers flexibility when creating gestures and defining the desired application behavior.

2.2 Tools

In order to implement AQUA-G using C++ and still provide cross-platform functionality, I utilized several tools to help write the software. The primary tool I used in developing and creating this software is called SCons (35). SCons is a cross-platform build system written in Python which allows reliable, repeatable software builds across multiple platforms.

المنارة للاستشارات
www.manaraa.com

SCons uses a simple user-created configuration file to build software, and I used the syntax described in the SCons documentation to create a configuration file for AQUA-G. The configuration file, named "SConstruct" by requirement, has a simple syntax, and SCons uses this file to build software for the platform it is executed on by invoking the platform's native compiler and linker tools. This allowed me to write a single build file and easily compile the software for multiple platforms, which was a great benefit.

2.3 Key concepts

Before discussing individual modules and components in AQUA-G, it is important to discuss the basic components upon which AQUA-G relies, and the principles of design which allow it stand apart from previous similar systems. The key components in AQUA-G are events and gestures, and the ability to load these events and gestures dynamically makes AQUA-G flexible and makes it easy for developers to customize.

2.3.1 Events

In AQUA-G, Events represent information that is passed between input devices and client applications. Crucial to one of the primary contributions of AQUA-G is the concept that input device events and client application events can and do share a common interface under a "unified event architecture." This allows a great deal of flexibility in the software framework, as detailed in the following scenario.

Assume that a developer wants to build and test an application with which they may evaluate the differences between interaction using the mouse or using a multi-touch device. This test application will allow basic drag and zoom gestures in order to allow users to pan and zoom a top-down view of a map.

Our multi-touch device can provide information such as location and state information of all detected points of touch. However, to provide zoom events, the application

المنارة للاستشارات
www.manaraa.com

requires a zoom gesture which will take as input multiple points, determine the change in distance over time between these points, and output an AQUA-G zoom event.

The developer also wants to use the mouse device. However, the mouse device can provide other unique information such as mouse wheel events. He or she would like to use the mouse wheel to send zoom events to the application. However, these events should not go to zoom gestures for processing, because they are already zoom events! Instead, they should be sent directly to the application! In previous frameworks, this is difficult or impossible. The developer would be forced to provide workarounds by simulating multiple touch points when the mouse wheel was used, or connecting the mouse to the application directly, bypassing the framework entirely.

In AQUA-G, I have solved this problem. It is simple to configure this scenario in AQUA-G. The developer can configure the mouse to send *native* AQUA-G zoom events to AQUA-G when it detects a mouse scroll. The test application informs AQUA-G that it is interested in receiving these zoom events, and configures zoom gestures to process the touch point information received from the multi-touch device. Now, the application receives information from both the mouse and touchscreen and can process it appropriately.

2.3.1.1 A unified event architecture

Developer-created custom events should be able to contain any desired information. However, it is important that we provide a standard event interface to facilitate gesture recognition across multiple devices, so that developers can write input device drivers which are compatible with the standard set of gestures provided by AQUA-G.

A unified event framework dictates that events share a common set of data so that they may be uniformly processed by the basic gestures. The information which all events share was determined through an evaluation of the types of information basic gestures require, as well as what information might be needed for most interactions using AQUA-

المنارة للاستشارات
www.manaraa.com

G.

Here is the structure of the information which is shared by all AQUA-G events:

- Event name (string)
- Event description (string)
- Event type (byte) One of:
  - DOWN, UP, MOVE, HOVER, OTHER
- Event ID (for keeping track of related DOWN, MOVE, UP events)
- Event location (float[3]) (x, y, z)

The Event base class provides these data members, and all developer-created events will subclass this event, facilitating uniform processing of all events. In addition to providing this basic structure, the Event class provides the following methods:

- Event(byte* data)
- byte[] serialize(short* lengthOut)

These two methods are provided to enable AQUA-G to send events over the TCP socket connections. The constructor takes as input a byte array and un-serializes this information into the event's fields. The serialize() method does the reverse by serializing the event's fields into a byte array.

Since developers will create custom events, it is necessary to provide these custom events with the ability to serialize their own custom data representation, be it accelerometer readings, button presses, or some other custom data. In order to do this, the Event class delegates the custom event serialization to the subclasses while still providing a common interface for all events. Each event must override the virtual method "serializeData()," which is used by the serialize method to serialize this custom data.

المنارة للاستشارات
www.manaraa.com

2.3.2 Gestures

Gestures are the core of the AQUA-G framework. They are responsible for processing input events in order to return some meaningful information as the result of the gesture processing. For example, a zoom gesture will take as input touch point events and output zoom events which contain scale and center information. The Gesture base class in AQUA-G provides developers a means of implementing their own custom gestures. AQUA-G is designed to make creating custom gestures as easy as possible for developers, so there is only a single method that developers must implement for event processing. It is defined as:

- bool processEvent(Event * e)

Developers must override this method to implement their own gesture processing. It is also necessary that AQUA-G provide some means of gestures to publish their resulting information, or output events. However, developer-created gestures should not need to know where this information is going. As an example, for global gestures, event translators, and region or component-centric gestures (distinction described in section 2.5), the destination for the resulting output events is not the same.

This could be done by allowing the processEvent method to return a list of gestures, which the caller could then send to the appropriate destination. However, this implementation would limit gestures to only output events when their processEvent method was called by some other code. Some gestures will want to implement timers and asynchronous gesture processing, so providing a means of creating output events at will is absolutely essential.

In order to satisfy this need, the Gesture base class provided by AQUA-G contains a protected method publishEvent(Event * e) which developers can use in their implementations to output events which are a result of gesture processing. The publishEvent method sends the event to the next component in the event flow diagram, which is

المنارة للاستشارات
www.manaraa.com

shown below in Figure 2.1. Each gesture is initialized by the framework with this "next component," and the publishEvent() method will send the event to this target component. Therefore, any gesture may implement threading, timers, or other asynchronous processing and call this method at will to output events.

2.3.3 Creating gestures and events dynamically

In order to ease gesture and event development for AQUA-G, I have chosen to implement dynamic class loading in C++ to allow runtime loading of gesture and event code. This is a departure from previous systems, and is one of the key contributions of this work. In previous systems, to implement custom gesture processing, it was necessary to add a new object to the system and hard-code object creation functionality to incorporate the new gesture. AQUA-G does not share this limitation with other systems. In AQUA-G, gestures and events can stand alone. Developers do not have to have any knowledge of AQUA-G's underlying implementation to create new gestures and events. Furthermore, this gives developers the flexibility of not having to re-compile the AQUA-G source code in order to add a new gesture or event.

As a matter of implementation, run-time class loading in C++ is accomplished through an implementation of the Factory Method design pattern (22). Each gesture or event class must implement a createEvent() or createGesture() method which returns an instance of itself. These methods are exposed through DLL export in Windows or shared object libraries in Linux, and can be used by the GestureFactory and EventFactory in AQUA-G to load custom events and gestures dynamically.

In order to implement this approach, AQUA-G requires that event and gesture filenames match their class names exactly. The factories use the data contained in the "Event name" field of the event to load the appropriate library and call the creator method to create the appropriate event.

المنارة للاستشارات
www.manaraa.com

2.3.4 Regions

The idea of "regions" was introduced by Sparsh-UI (58) and Tisch (18). A region is an area of interest in the client application space. A region could be a UI widget, a polygon in a 3D environment, or some other on-screen area of interest. Crucial to accurate gesture processing is the ability to distinguish between on-screen objects and process events appropriately.

In AQUA-G, I have chosen to follow the implementation of Sparsh-UI for region identification and processing. In this method, event time AQUA-G receives an event from an input device, it will ask the client application for a unique region identifier. Therefore, the client must maintain knowledge of the locations of its UI components, so that when AQUA-G asks which region an event should be sent to, the client can respond with the correct identifier, and AQUA-G can process the event by sending it to the appropriate region for processing by component-specific (also hereafter known as region-specific) gestures.

2.4 Event flow

In AQUA-G, events pass through the system as shown in Figure 2.1. Events pass from the input devices at the bottom of the figure, to the client application at the top of the figure.

Events begin at the input device level, where they are sent to the AQUA-G gesture server by the input device drivers. These events are received by a corresponding InputDeviceConnection object residing inside the AQUA-G framework.

Each client application in AQUA-G has a stack of components associated with it, as shown in Figure 2.1. This stack is made up of a gesture engine, event translators, regions, and global and region-specific gestures.

Once AQUA-G has received events from input devices, the events are passed to all

المنارة للاستشارات
www.manaraa.com

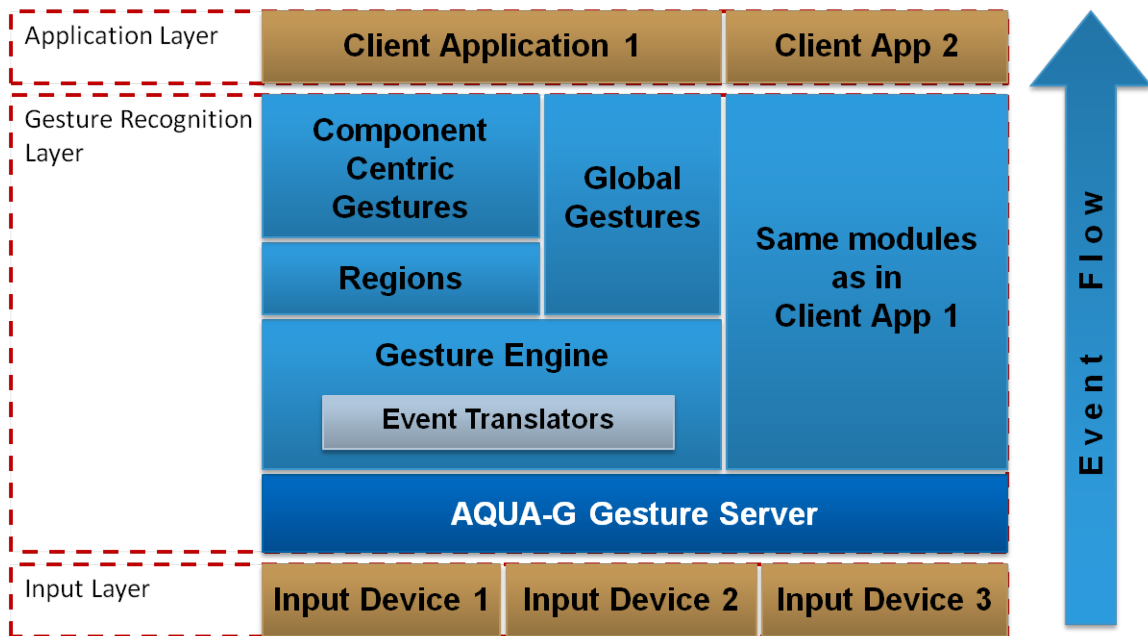

Figure 2.1 AQUA-G event flow.

available GestureEngine objects. From there, they are sent to event translators, and then move on to global gestures, regions, and region gestures. All of these components will be described in the next section, which decomposes the system into individual components.

AQUA-G itself makes up all components in Figure 2.1 between the input devices and client applications. AQUA-G does not impose any restrictions on the maximum number of input devices or client applications. As stated above, each component in Figure 2.1 will be described in the system decomposition section. However, the exact module names may differ slightly from this logical depiction of the architecture. Therefore it will be useful to refer often to this diagram when reading the next section to remember where each component lies in the event flow.

2.5 System decomposition

This section describes each module which is shown in the AQUA-G class diagram, as shown in Figure 2.2. This diagram represents the object-oriented design and implemen-

المنارة للاستشارات
www.manaraa.com

tation of AQUA-G. Each module is responsible for a specific task, and its responsibilities are described in this section. Modules are generally described in the order in which

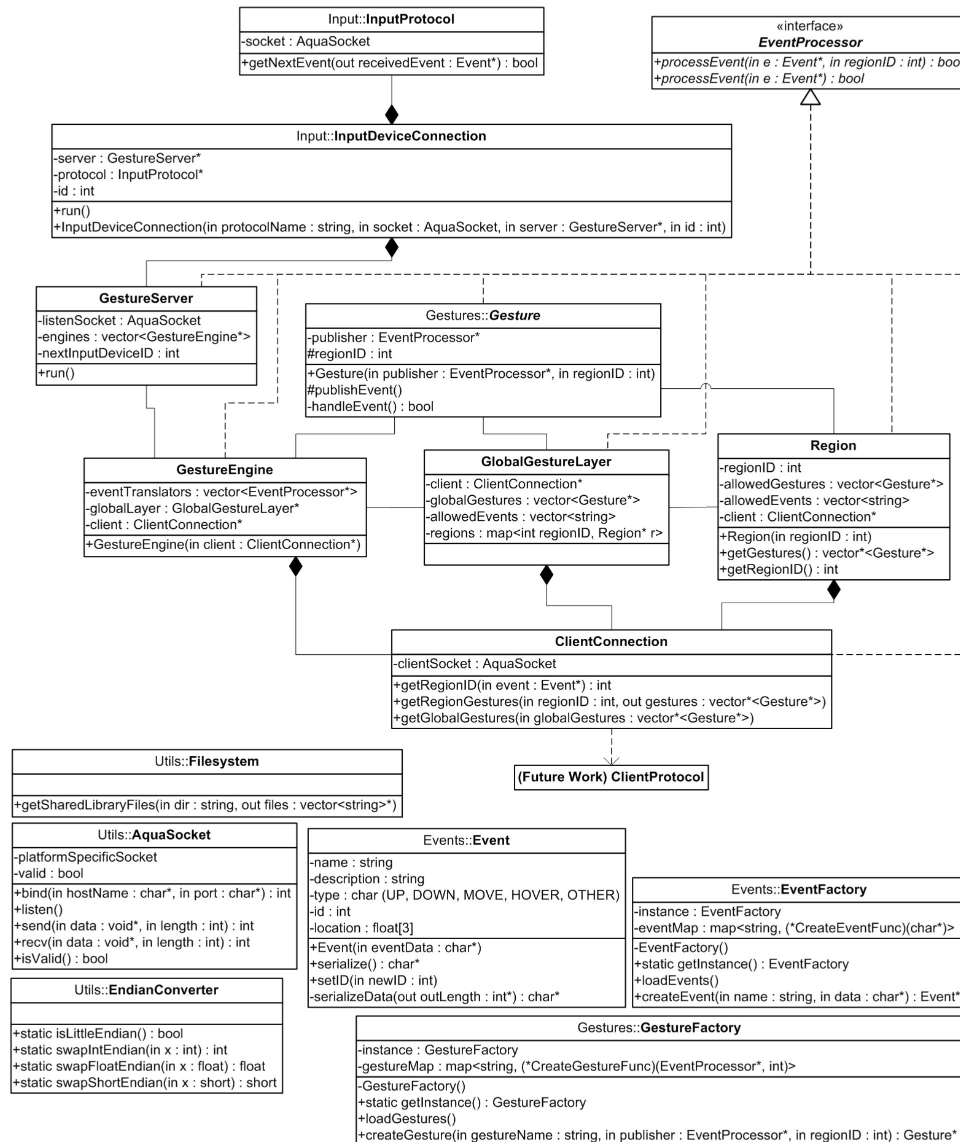

Figure 2.2 AQUA-G class diagram.

events flow through the modules, as shown in Figure 2.1. Thus, modules which appear early in this section receive events sooner than modules which appear later in the section. An exception is this first module, EventProcessor, which provides a standard interface for event processing which is used by many of the classes in the architecture.

www.manaraa.com

2.5.1 EventProcessor

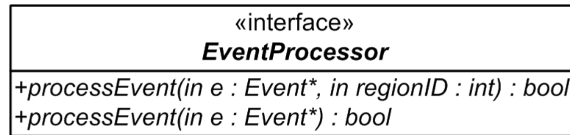

Figure 2.3 EventProcessor interface.

Many classes in AQUA-G have the need to perform processing on Event objects. This common functionality is defined by the EventProcessor interface. Any class that provides this interface shall implement the two processEvent methods provided here. The methods are different only in that one method takes a integer *regionID* parameter in addition to the Event parameter *e*. The *regionID* is provided by the client application and is a logical equivalent to a GUI widget or component in the UI.

2.5.2 InputProtocol

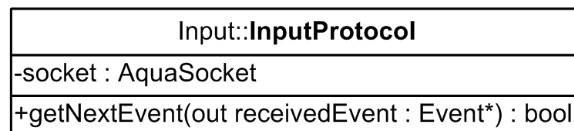

Figure 2.4 InputProtocol class.

The InputProtocol class (Figure 2.4) is responsible for communicating with an input device. In AQUA-G, input devices communicate with the AQUA-G gesture server over a TCP socket connection. The InputProtocol class has a member variable *socket* which is a TCP socket that is connected to the input device driver. It has a single method, *getNextEvent*, which returns the next incoming event from the input device using the pointer passed in as an argument (*receivedEvent* in Figure 2.4).

The AQUA-G InputProtocol specifies the format in which input devices should send information to the AQUA-G gesture server. The input device driver connects to the

المنارة للاستشارات
www.manaraa.com

gesture server using a TCP socket connection. After establishing the connection to the gesture server, it can begin sending events.

When an input device wishes to send an event, it should first send a short integer which contains the number of bytes in the incoming event. The InputProtocol class in the AQUA-G framework reads this short integer and proceeds to read that number of bytes into a buffer. Finally, it constructs the appropriate event using the event name and the data it received and places it in the output parameter *receivedEvent*.

Implementing the input protocol in a separate module allows for easy customization of the protocol or extension of the protocol by class inheritance. For example, implementing a custom input device protocol could be performed by sub-classing the input device protocol and overriding the *getNextEvent* method.

The InputProtocol represents only the socket connection to the input device. Its getNextEvent() method is called repeatedly by an instance of the InputDeviceConnection class, which is responsible for processing events received from an input device.

2.5.3 InputDeviceConnection

Input::**InputDeviceConnection**
-server : GestureServer* -protocol : InputProtocol* -id : int
+run() +InputDeviceConnection(in protocolName : string, in socket : AquaSocket, in server : GestureServer*, in id : int)

Figure 2.5 InputDeviceConnection class.

The InputDeviceConnection class represented in Figure 2.5 maintains information about an input device connection. It contains a reference to the AQUA-G gesture server in the member *server* and a reference to the associated InputProtocol object *protocol*. Furthermore, each InputDeviceConnection has a unique identifier which is stored in *id*.

المنارة للاستشارات

www.manaraa.com

The InputDeviceConnection is constructed when the AQUA-G gesture server accepts an incoming connection from an input device. It has a single method *run* which is called by the GestureServer after it is constructed. This method begins reading events using the getNextEvent() method in the InputProtocol class. Upon receiving an event, it sends it to *server* by calling the *processEvent* method which is implemented in GestureServer because it implements the EventProcessor interface.

2.5.4 GestureServer

GestureServer
-listenSocket : AquaSocket -engines : vector<GestureEngine*> -nextInputDeviceID : int
+run()

Figure 2.6 GestureServer class.

The GestureServer is the core of the AQUA-G framework. It handles incoming connections from input devices and client applications by listening on the member variable *listenSocket*. Incoming connections made to this socket are processed, and the gesture server creates input device or client application connections as necessary. Each client application has its own associated GestureEngine object, and the GestureServer will send all incoming events from the InputDeviceConnections to all available GestureEngines, again by calling the *processEvent* method on the GestureEngine.

2.5.5 GestureEngine

The GestureEngine class is tasked with processing all events targeted for a specific client application, and is the first client application-specific module to receive events from input devices. It maintains a list of all event translators (see Figure 2.7), which are described below. Upon receiving an event, it sends the event to all available translators.

المنارة للاستشارات
www.manaraa.com

GestureEngine

-eventTranslators : vector<EventProcessor*>
-globalLayer : GlobalGestureLayer*
-client : ClientConnection*

+GestureEngine(in client : ClientConnection*)

Figure 2.7 GestureEngine class.

When the translators have finished processing the event, the GestureEngine sends the event to the *globalLayer* for further processing. The GestureEngine also maintains a reference *client* to a ClientConnection object. It uses this object to communicate with the client application. When a GestureEngine is initialized, it asks the client which event translators the client would like to register.

2.5.5.1 Event translators

The event translators mentioned in this section are given a special place in the AQUA-G gesture framework, and represent another new area which has not been previously implemented. In the Tisch architecture (18), the transformation layer is responsible for calibrating incoming data (See Figure 1.4). In AQUA-G, event translators have much more power. Event translators are instances of gestures, but they have additional abilities. They can translate input device information into other, perhaps more relevant information. They publish their resulting events back to the gesture engine, allowing for "layered" translators if desired by the client application. Here are two examples which demonstrate the need for event translators:

- A certain input system may provide touch point information and also track user hands via diffuse illumination (65), Cricket location sensors (56), or with an overhead camera (17). An example of this type of application is described in chapter 3. An event translator may accept as input these two types of information and output touch point data augmented with a hand identifier.

المنارة للاستشارات
www.manaraa.com

- A Wii Remote may provide input events such as accelerometer values. An event translator may filter this information to provide smoother values, or it may interpolate the accelerometers to provide velocity or even attempt to provide position events.

Event translators can also consume events so that they are not processed by the global or region gestures. This allows for appropriate translation of events, since event translators often intend to replace incoming events with the translated events.

2.5.6 GlobalGestureLayer

```
GlobalGestureLayer
-client : ClientConnection*
-globalGestures : vector<Gesture*>
-allowedEvents : vector<string>
-regions : map<int regionID, Region* r>
```

Figure 2.8 GlobalGestureLayer class.

Global gestures are not associated with a particular component in the user interface. For example, gestures which have global actions such as "turn system on" or "mute volume" are generally not associated with a particular UI component, and thus should be processed on a global or application-wide level. Gestures in this class will receive all events from the input devices, and their resulting events are published directly to the client application.

The GlobalGestureLayer (Figure 2.8) is a placeholder for these types of gestures. It maintains a list of all of these gestures in *globalGestures*, and also maintains a list of allowed events which the client is interested in receiving. The client specifies the types of events it wishes to receive during initialization.

المنارة للاستشارات
www.manaraa.com

Finally, the GlobalGestureLayer maintains a list of regions, which were described in section 2.3.4. It is responsible for determining which region events should be sent to as they are received from the input device. After the GlobalGestureLayer receives the region identifier for the event from the client, it calls the appropriate processEvent() method for that particular Region.

2.5.7 Region

```
Region
-regionID : int
-allowedGestures : vector<Gesture*>
-allowedEvents : vector<string>
-client : ClientConnection*
+Region(in regionID : int)
+getGestures() : vector*<Gesture*>
+getRegionID() : int
```

Figure 2.9 Region class.

Each Region object module is responsible for maintaining a single region of interaction, and is a counterpart to its visible representation in the client application. Usually, this representation is a component or widget in a user interface.

When a Region object is first created, it will ask the client application which gestures should be allowed for this region using its reference to the client application. The client application will respond with a list of the available gestures and events for that region. As an example, for a photo organizing application, each region might represent a single photo in the interface, and the the client would respond to this message with "zoom gesture, rotate gesture, drag gesture" or some other list of available gestures it deems appropriate.

المنارة للاستشارات
www.manaraa.com

ClientConnection
-clientSocket : AquaSocket
+getRegionID(in event : Event*) : int +getRegionGestures(in regionID : int, out gestures : vector*<Gesture*>) +getGlobalGestures(in globalGestures : vector*<Gesture*>)

Figure 2.10 Client Connection class.

2.5.8 ClientConnection

The client application communicates with AQUA-G through TCP sockets. This class contains a socket with which it uses to communicate with the client application, and provides methods which the framework can use to sent the client application messages. These messages and the communication that takes place is described in greater detail in section 2.6.

2.5.9 Utilities

Since AQUA-G must run on multiple platforms, platform-specific functionality has been abstracted into wrapper classes which provide the necessary functionality. The utilities package in AQUA-G also provides functionality for endian conversion, which is required for transmitting event data over the TCP sockets.

2.5.9.1 FileSystem

Utils::**Filesystem**
+getSharedLibraryFiles(in dir : string, out files : vector<string>*)

Figure 2.11 Filesystem class.

The FileSystem class is responsible for using platform-specific APIs to find shared libraries in a given directory. Its getSharedLibraryFiles() method returns a list of file

المنارة للاستشارات
www.manaraa.com

names which represent all of the available shared library files in the given directory. This method is used by the gesture and event factories to dynamically load the gestures and events. The method must take into account the different prefixes and extensions which each platform uses - on Windows, the method looks for “¡MyGesture¿.dll” files, on Linux it looks for “lib¡MyGesture¿.so” files, and on Mac OS it looks for “¡MyGesture¿.dylib” files.

2.5.9.2 AquaSocket

Utils::**AquaSocket**
-platformSpecificSocket -valid : bool
+bind(in hostName : char*, in port : char*) : int +listen() +send(in data : void*, in length : int) : int +recv(in data : void*, in length : int) : int +isValid() : bool

Figure 2.12 AquaSocket class.

Since AQUA-G communicates with input devices and applications over sockets, it is necessary to wrap the platform-specific socket APIs so that the framework can make use of sockets in a platform-independent manner. I investigated several other socket-wrapping libraries, but settled on writing my own because of the additional dependencies that AQUA-G would require in order to build using these libraries. The AquaSocket class makes use of preprocessor definitions to compile the correct code during the build process for AQUA-G, and it provides standard socket functionality. For exception handling, the socket will throw a generic exception which can be caught by users of the class.

2.5.9.3 EndianConverter

The EndianConverter class provides simple conversion from little-endian to big-endian byte ordering, and also provides a method which will check if the current platform

المنارة للاستشارات
www.manaraa.com

Utils::**EndianConverter**

+static isLittleEndian() : bool
+static swapIntEndian(in x : int) : int
+static swapFloatEndian(in x : float) : float
+static swapShortEndian(in x : short) : short

Figure 2.13 EndianConverter class.

is a little or big-endian platform. Users of the class can utilize these functions to convert all data to be sent over the network into network-endian (big-endian) byte format, and back to host-endian format when it is received.

2.5.10 Gesture and event creation

AQUA-G dynamically loads events and gestures, and makes use of the factory method design pattern described in (22). Each shared library will expose a creator method, which the framework can find through the use of platform-specific dynamically-linked library code. It then uses this method to create the appropriate class, given a certain class name. The factories shown in Figure 2.14 and Figure 2.15 expose methods which will return a pointer to the instance of the created class, given the class name.

Events::**EventFactory**

-instance : EventFactory
-eventMap : map<string, (*CreateEventFunc)(char*)>

-EventFactory()
+static getInstance() : EventFactory
+loadEvents()
+createEvent(in name : string, in data : char*) : Event*

Figure 2.14 EventFactory class.

المنارة للاستشارات
www.manaraa.com

```
Gestures::GestureFactory
-instance : GestureFactory
-gestureMap : map<string, (*CreateGestureFunc)(EventProcessor*, int)>
-GestureFactory()
+static getInstance() : GestureFactory
+loadGestures()
+createGesture(in gestureName : string, in publisher : EventProcessor*, in regionID : int) : Gesture*
```

Figure 2.15 GestureFactory class.

2.6 Client application interaction

Because AQUA-G will communicate with user interface applications, the interaction with these applications is more complicated than interaction with the input devices, where it is sufficient simply to send events to AQUA-G. The messages that the client will be expected to respond to are described in this section.

2.6.1 Initialization state

Upon initialization, the client must identify itself to AQUA-G as a client application. As soon as a connection has been established, the gesture server will require knowledge of the allowed global gestures and event translators allowed for a particular client application. Thus, it will expect the client application to respond to the following messages.

2.6.1.1 Get global information

This message is sent by AQUA-G when it requests the global gestures that will be associated with this client. The client application must respond by sending an integer which represents the number of gestures it wishes to register, followed by a set null-terminated strings, each of which contains the name of each gesture it wishes to register as a global gesture.

Furthermore, AQUA-G will also need to know which events should be passed globally to the client application. Events specified will be sent directly from the input devices

المنارة للاستشارات
www.manaraa.com

to the client applications. This allows AQUA-G to act as a simple event pipe between input devices and client applications. This functionality can be useful for debugging input devices or working with input device data directly in the client application.

To send events, the client application must send an integer which represents the number of events it wishes to register, followed by a set of null-terminated strings, each of which contains the name of the event it wishes to register as a global allowed event.

2.6.1.2 Get event translators

This message is sent by AQUA-G to request the event translators that will be associated with this particular client. The client application must respond by sending an integer representing the number of gesture names it will send. It will then send a set of null-terminated strings, each of which contains the name of each gesture it wishes to register as an event translator.

2.6.2 Running state

After initialization, the client application should be prepared to respond to any of the following messages which are sent by AQUA-G.

2.6.2.1 Get region identifier

This message is sent when AQUA-G recognizes a new event. Along with this message, AQUA-G sends a location which represents where the event occurred. The client should use this information to find the appropriate region identifier and sent it back to AQUA-G as an integer value.

2.6.2.2 Get region information

This message is sent by AQUA-G when it recognizes a new region – when the client sends a region identifier as a result of the Get Region Identifier message that AQUA-G

المنارة للاستشارات
www.manaraa.com

has not used before. For proper behavior, the newly created region will need to know which gestures should be associated with it. The client shall respond similarly to the Get Global Gestures message, because the region needs to be initialized with the same type of information as the global gesture layer - a set of allowed gestures and events. However, these gestures and events will only apply to this specific region. To respond to this message, the client will respond in exactly the same was as when responding to the Get global gestures message - by sending an integer representing the number of gesture names it will send, followed by a set of null-terminated strings, each containing a name of a gesture it wishes to register with this region. It must also send an integer which represents the number of events it wishes to register as allowed for this region, followed by a set of null-terminated strings, each of which contains the name of the event it wishes to register as an allowed event for this region.

2.6.2.3 Process global event

This message is sent when a global event has occurred. In the examples given previously, this would be something such as a "turn off" or "mute volume" type of event, which has been recognized by a global gesture. In addition to the message itself, AQUA-G sends a byte array containing the event data which the client shall use to identify the event and construct the appropriate event.

2.6.2.4 Process region event

This message is nearly identical to the Process Global Event message, but in addition to the event data, AQUA-G also sends an integer representing the region ID for which this event occurred. Using this information, the client can respond appropriately to the incoming event.

المنارة للاستشارات
www.manaraa.com

2.7 Supported input devices

The AQUA-G framework supports multiple simultaneously connected input devices, and allows developers to create their own input devices by following a standard protocol. Already, support for several devices exists, and support for additional devices is planned for future work. Devices drivers have been written for the following input devices.

2.7.1 Windows and Linux mice

AQUA-G provides a device driver for a standard windows mouse as well as a standard Linux mouse. This allows developers to test their applications using conventional hardware before they acquire other input devices. The mouse provides unified events such as down, move, up, and hover events, and sends an identifier which represents the left, middle, and right mouse buttons.

2.7.2 HP TouchSmart touchscreen

A device driver has been written which takes input from an HP TouchSmart touchscreen using the NextWindow Two-Touch API. Since the driver uses this standard API, it is believed that this device driver will work for any input device which has a NextWindow touchscreen. However, this has not yet been tested.

2.7.3 iPad and iPhone

A developer has contributed a device driver for the iPad and iPhone as a result of the evaluation conducted of AQUA-G. The driver turns these devices into wireless trackpads, where the screen on the device is mapped to the application using a 1-1 correspondence. Thus, any point pressed on the iPad or iPhone will be mapped directly to the application screen. This has both advantages and disadvantages, because a user

المنارة للاستشارات
www.manaraa.com

Figure 2.16 The HP TouchSmart.

Figure 2.17 The Apple iPad.

المنارة للاستشارات
www.manaraa.com

can interact with any on-screen information quickly, but in a less precise manner as a result of the small input surface.

2.7.4 Wii Remotes

Figure 2.18 The Nintendo Wii Remote.

Support is also provided for multiple simultaneously connected Wii Remotes. The Wii Remote support utilizes the C-sharp WiiMoteLib API written by Brian Peek (55), and, as a result, only works on Windows operating systems. The Wii Remote interface to AQUA-G follows the standard Wii interaction paradigm of pointing at the screen, and requires a Wii sensor bar or equivalent sources of infrared light. The driver uses the two infrared light sources, tracked by the Wii Remote's infrared camera, to indicate the place on the screen that the user is pointing. Using this information, the driver allows

المنارة للاستشارات
www.manaraa.com

the user to click on objects using the trigger button located on the bottom of the Wii Remote.

2.7.5 Cricket location sensors

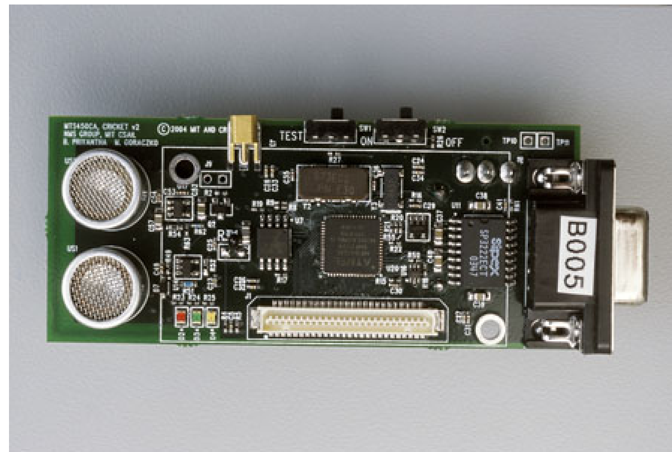

Figure 2.19 A Cricket sensor.

In the haptics lab at the Virtual Reality Applications Center, Iowa State University, we have an implementation of a Cricket location system which, utilizing Cricket sensors (Figure 2.19) strapped to users' hands, can provide hand positions when multiple users are interacting with a 60" multi-touch table. I have written an AQUA-G device driver for this system, which allows us to write applications which utilize user identification. An example of this type of application is described in the case studies chapter later on in this work.

2.7.6 Sparsh-UI input devices

Since many of the devices we use in our lab use Sparsh-UI for gesture recognition, AQUA-G provides backwards compatibility in the sense that it will accept connections from Sparsh-UI devices. A custom protocol has been written which allows these de-

للاستشارات المنارة
www.manaraa.com

vices to communicate with the AQUA-G framework without any changes to their device drivers.

Support for more input devices is planned for future work, and we expect that developers will continue to contribute device drivers for many new and perhaps as of yet unknown types of input devices in the future.

2.8 Supported gestures

AQUA-G comes with a standard set of dynamic gestures which allow for the development of gesture-enabled applications. It is my hope that developers will continue to contribute gestures for AQUA-G, adding to an ever-expanding list of supported gestures. The current gestures that come with AQUA-G are described in this section.

2.8.1 Drag gesture

The drag gesture allows for multi-point movement of objects. Therefore, users can drag objects with multiple fingers or mice at the same time. The drag gesture calculates the drag amount based on the centroid of these points, and outputs a drag event which indicates where on the screen the desired component should be moved to.

2.8.2 2D rotate gesture

The 2D rotate gesture allows for multi-point rotation of objects. The gesture calculates the rotation based in the change in the angle of points about the centroid of those points. Upon determining the amount the user has rotated multiple points, the gesture creates a rotate event and outputs this to the application.

As of July 1, 2010, only 2D rotation is supported. Using three or more points to rotate objects in 3D space is a complex operation, and we are still determining the best way to handle these types of gestures.

المنارة للاستشارات

www.manaraa.com

2.8.3 Zoom gesture

The zoom gesture allows for multi-point scaling of objects, where the scale is calculated based on the relative change in distance between the points on the object. The scale is calculated using this relative change and is sent to the applications as a zoom event.

Because multiple gestures can be combined together, if the drag, rotate, and zoom gestures are all processed at the same time, applications can allow for rotation and scaling about any point of the component, not just the center.

2.8.4 Flick gesture

This gesture behaves similarly to mouse gestures in Firefox (44) or pen flicks in tablet editions of Windows (43). It simply detects quick user "flicks" in four directions: up, down, left, and right. The gesture will send a flick event to applications if a flick is detected.

2.8.5 Double-click gesture

This gesture detects when users have tapped or clicked twice rapidly. This allows developers to utilize double-clicks in their applications without having to perform the double-click detection themselves. The gesture sends double-click events to applications; along with the amount of time that elapsed between the two clicks. This allows applications to customize timing for individual clicks if they find their application is not responding appropriately.

2.9 Supported event translators

AQUA-G also provides a few basic gestures which are intended to be used as event translators. In AQUA-G, any gesture can serve as a standard gesture or an event

المنارة للاستشارات
www.manaraa.com

translator; so the term "gesture" in AQUA-G also extends to event processing algorithms or translators which are implemented as gestures, such as the following two gestures.

2.9.1 Get handID gesture

This gesture associates hand position with touch points. It can be utilized by applications which require user identification, meaning they need to know which user is associated with each individual interaction. The gesture is intended to be used as an event translator. It consumes hand positions and unified touch events and outputs HandIDTouch events which contain a hand identifier of the hand which was closest to that point at the time the point was detected.

2.9.2 Kinetic gesture

In some multi-touch applications, developers have attempted to simulate physical properties in interfaces. For example, objects in an interface, when moved quickly and released, will continue to move and then slow down gradually, as if by the force of friction. This can have a pleasing aesthetic effect, and I wanted developers using AQUA-G to have access to this effect. Therefore, this gesture takes as input unified events, and, upon receiving an UP event, will continue to simulate additional events which are in the direction of motion of the previous MOVE events that were received. The simulated events will slow down over time and, when the speed of the object has fallen below a predefined threshold, the gesture will send an UP event to the application.

المنارة للاستشارات
www.manaraa.com

CHAPTER 3. CASE STUDIES

In this chapter, I will present several case studies of AQUA-G. The case studies describe example applications, device drivers, and gestures which have been developed using the AQUA-G framework, and demonstrate its power and flexibility for developing varied types of software applications. Three of the projects described in this chapter were created by the author, and three others were created by other developers using the AQUA-G framework.

3.1 A first application

In order to test and debug AQUA-G, it was necessary to develop a simple application which would allow for testing of basic gestures and input devices. To achieve this end, I developed an application which allows users to drag, scale and rotate several colored blocks on the screen. The application is analogous to the rather widespread photo-organizing demonstration available on many multi-touch devices. An image of this application in use is shown in Figure 3.1.

The application was written in Java and utilizes AQUA-G unified events and the Drag, 2D Rotate, and Zoom gestures provided by AQUA-G and described in section 2.8. The sample application currently allows the following input devices to be connected simultaneously.

- Windows mouse
- HP TouchSmart touchscreen

المنارة للاستشارات
www.manaraa.com

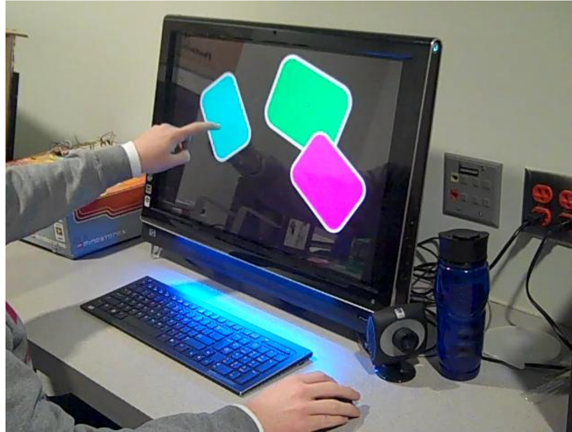

Figure 3.1 A first AQUA-G application.

- Wii remotes
- iPad
- Sparsh-UI input devices

Using the HP TouchSmart, users can drag, scale and rotate the blocks on the screen using two fingers to pinch, stretch, or rotate the blocks using the gestures described above (Figure 3.2). The HP TouchSmart API only provides two simultaneous points of touch, so even though the gestures provided by AQUA-G are capable of recognizing and processing more than two fingers, users can only use two fingers to interact with the application.

Users may also interact with the application using a standard mouse. With the mouse, users can drag objects around on-screen and also use the mouse wheel to scale the objects (Figure 3.3). Moving the mouse wheel up or down makes the objects bigger or smaller; similar to using two fingers to scale the objects using the TouchSmart. However, by utilizing AQUA-G, the mouse is able to send native AQUA-G zoom events to AQUA-G. These zoom events are passed directly to the application rather than being sent through the gestures for processing.

المنارة للاستشارات
www.manaraa.com

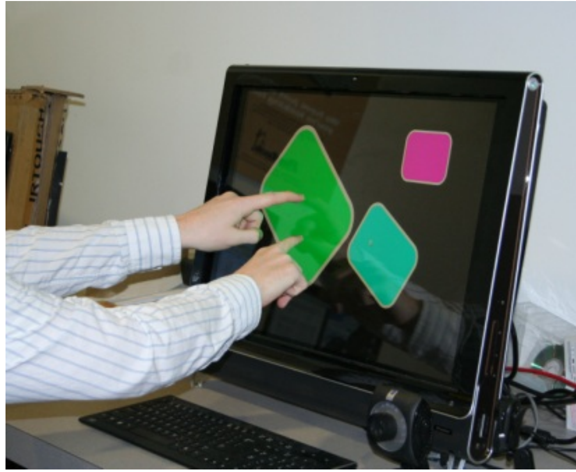

Figure 3.2 A user interacting with the sample application using the HP TouchSmart.

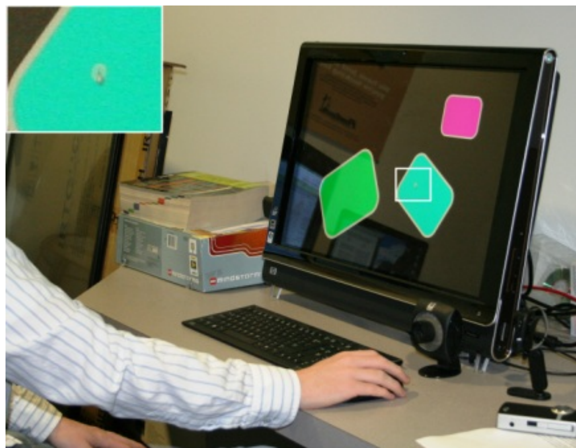

Figure 3.3 A user interacting with the sample application using a standard mouse.

المنارة للاستشارات
www.manaraa.com

Users may also interact with the sample application using a Wii Remote (Figure 3.4). The Wii Remote allows users to drag objects around on the screen by pressing and holding the B button, located on the bottom of the remote. It also allows a user to scale objects by pushing the Wii Remote towards the screen and by pulling the Wii Remote away from the screen.

In order to accomplish this, the Wii Remote utilizes an infrared camera to track the distance between the two infrared points of light, which are emitted by a Wii sensor bar. By pushing the Wii Remote closer to the screen, the camera moves closer to the screen, and as a result, the camera detects that the distance between the two points of light has increased. This relative change in the distance between the two points is sent to AQUA-G as a native zoom event, similar to the mouse driver implementation.

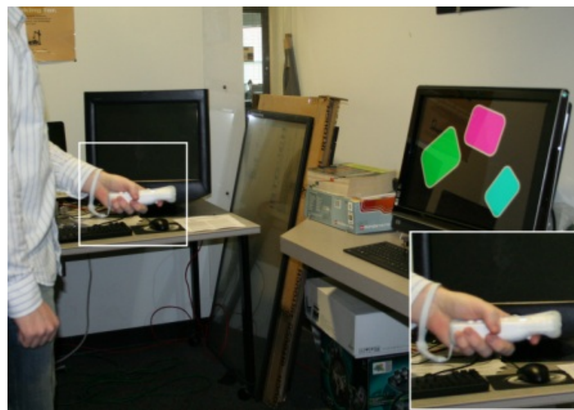

Figure 3.4 A user interacting with the sample application using a Wii Remote.

Finally, the application allows users to interact using an iPad or iPhone (Figure 3.5). The device driver turns the iPad or iPhone into a touch-based input device which operates in the same space as the display using a 1-1 correspondence. To clarify, when the user touches in the middle of the iPad, a touch point is shown in the middle of the sample application's screen. Similarly, if a user were to touch near the upper left corner, the touch would be displayed near the upper left corner of the application's screen. In

المنارة للاستشارات

www.manaraa.com

this way, users can interact with the on-screen objects in the same manner as they would using the HP TouchSmart.

Figure 3.5 A user interacting with the sample application using an iPad.

Since the sample application supports multiple simultaneous input devices, users can interact with the application using multiple devices at the same time. In Figure 3.6, a user is simultaneously using the HP TouchSmart and the mouse to interact with the application. I found that interacting with both the touchscreen and mouse simultaneously afforded precise and engaging interaction, which I did not necessarily expect. Prior research has shown that multi-touch interaction can offer advantages for manipulating objects when it is necessary to simultaneously zoom, drag, and rotate them (46). However, it can often afford less precise interaction than a standard mouse (21; 61). I anticipate that AQUA-G will allow us to experiment with these and similar unexplored types of interaction for other applications.

3.2 A user-identification based application

In addition to the sample application, I have developed another application which utilizes the GetHandIDGesture provided by AQUA-G to recognize individual users gathered around a 60" multi-touch table. The application is written in Java and is an im-

المنارة للاستشارات
www.manaraa.com

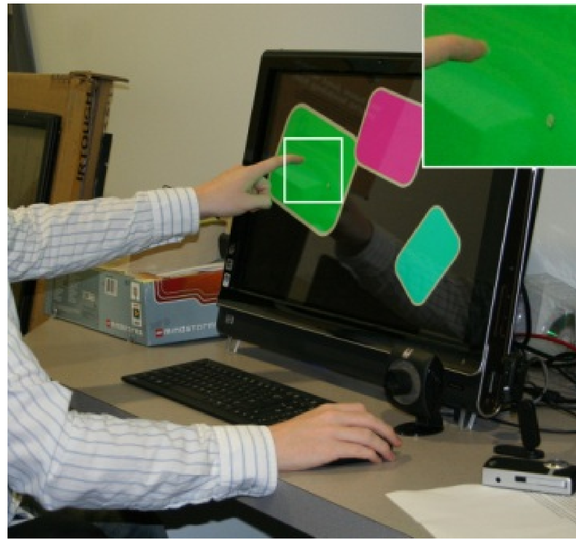

Figure 3.6 A user interacting with the sample application using both an HP TouchSmart and a standard mouse simultaneously.

plementation of Conway's Game of Life, as described by Gardner in a 1970 issue of Scientific American (23). The game is played on a square grid, and is played as follows. First, a player enables or gives life to a set of squares in the grid, called life forms. Once this setup has been performed, the user's task is complete, and the user simply watches their creation evolve. The game evolves iteratively based upon the following rules which govern the calculation of the next iteration. In the next iteration:

1. any alive cell with less than 2 alive neighbors dies from loneliness.

2. any alive cell with more than 3 alive neighbors dies from overcrowding.

3. any alive cell with 2 or 3 alive neighbors remains alive in the next iteration.

4. any dead cell with exactly 3 neighbors becomes alive in the next iteration.

These simple rules evoke surprisingly interesting and engaging behavior of the simulated life forms.

In this application, users touch the screen to create alive cells. The game differs slightly from the standard version of the game described above, in that it allows for user-

المنارة للاستشارات
www.manaraa.com

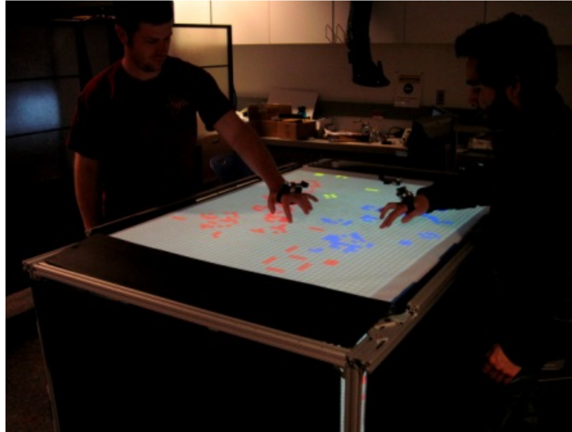

Figure 3.7 Two users playing the AQUA-G Game of Life.

controlled creation of new alive cells during the evolution of the system. Furthermore, this application has been augmented with user identification, which allows each user to create uniquely colored life forms. One user's alive life forms are able to assimilate other alive life forms if they own a majority of the life form's neighbors. This evokes a somewhat competitive environment as users attempt to spread their own life forms throughout the playing surface and try to repel other users' life forms. An example of two users playing the game is shown in Figure 3.7.

To provide user identification, the application uses the Cricket user identification system developed by Ramanahally (59) for the Virtual Reality Applications Center. The Cricket sensors are visible in Figure 3.7 and provide hand positions to AQUA-G, and touch points are provided by the multi-touch sensitive table. AQUA-G processes these inputs in the GetHandIDGesture and reports touch points augmented with a user identification field to the application.

This game demonstrates the power of AQUA-G to utilize varying types of input from devices other than standard pointing devices. Furthermore, it showcases the collaborative (or competitive) nature interaction such as this can provide, and the interaction paradigm demonstrated extends easily to important applications such as virtual assem-

المنارة للاستشارات
www.manaraa.com

bly, collaborative design, mission planning, and others. Using AQUA-G allowed this application to be developed rapidly using the same method and similar code as the first example application described in the previous section.

3.3 LABET unmanned vehicle controller

This unique application of AQUA-G was contributed by an external developer utilizing AQUA-G to provide an event handling framework for controlling an unmanned aerial vehicle (UAV) named LABET (Low Altitude Balloon Experiments in Technology). A picture of the LABET vehicle is shown in Figure 3.8.

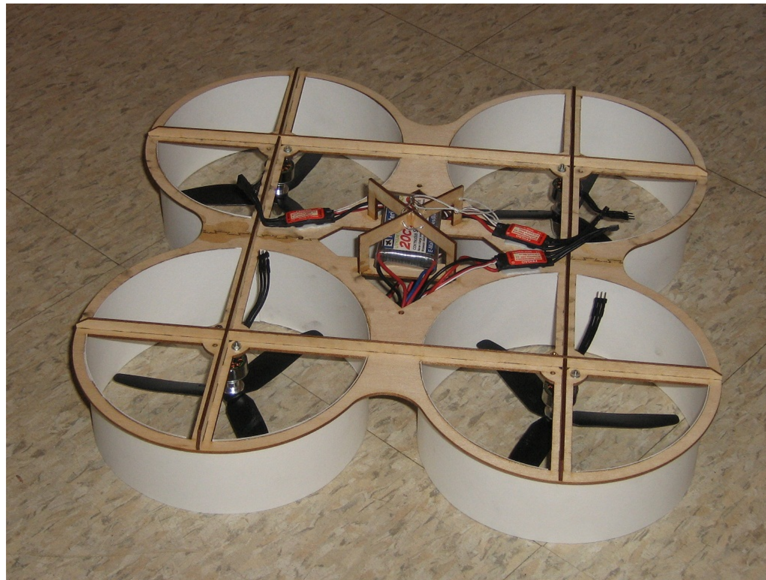

Figure 3.8 The LABET unmanned aerial vehicle.

The architecture involves three input devices: a joystick, mouse, and the UAV, and three client applications: two GUI applications for controlling the UAV, and the UAV itself. This application extends the use of AQUA-G to an application which I did not originally expect, and demonstrates the power and flexibility of the framework to be used in widely varied applications. A diagram of the architecture of this application is given in Figure 3.9.

المنارة للاستشارات
www.manaraa.com

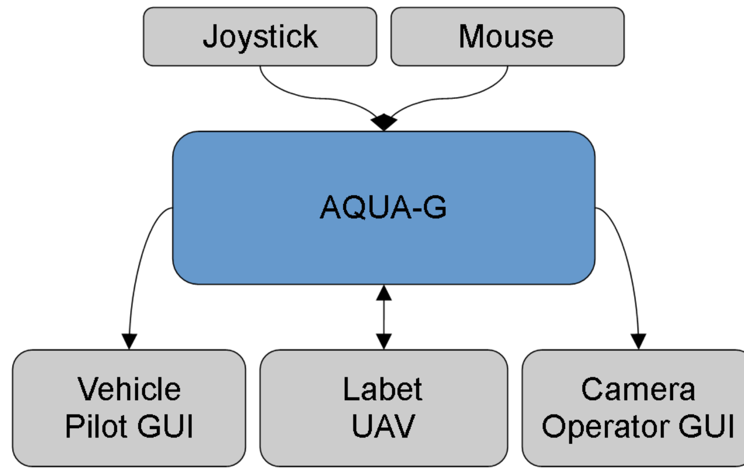

Figure 3.9 Architecture for the AQUA-G solution to LABET UAV control.

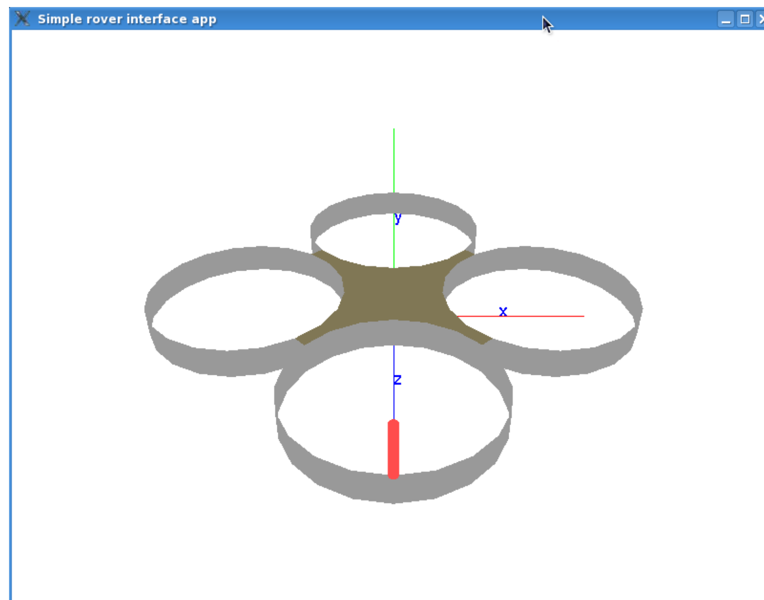

Figure 3.10 The LABET avatar simulated in OpenGL.

www.manaraa.com

As shown in the architecture diagram, there are three applications for this AQUA-G solution: two GUIs and the vehicle itself. These applications subscribe for global events from the joystick and the mouse controllers, and utilize the events to update user feedback (GUI applications) or control the vehicle (LABET UAV). Furthermore, the two GUI applications also subscribe for global events from the LABET UAV, which, in this architecture, functions as *both* an input device and client application.

This AQUA-G case study is still under development, and will be completed after the publication of this thesis. So far, work has been performed to develop a joystick device driver and simple client application which, using OpenGL, allows users to control a virtual LABET avatar using the joystick. This avatar is shown in Figure 3.10.

3.4 The flick gesture

As part of the evaluation of AQUA-G, a developer created a gesture which I have have called the Flick gesture. The gesture recognizes flick gestures similar to Pen Flicks in tablet editions of the Windows operating systems, or Firefox mouse gestures which are available as a Firefox add-in. The gesture uses time to detect rapid click, drag, and release motions in four directions (up, down, left, and right). The generated "Flick events" can be interpreted and used as command shortcuts by applications wishing to incorporate this functionality.

As described in Appendix A, I have created a website (60) which contains tutorials and example code for developers wishing to develop software using AQUA-G. The developer who created this gesture did so by modifying the sample gesture and event projects which are provided on the AQUA-G website.

المنارة للاستشارات
www.manaraa.com

3.5 The kinetic gesture

In some multi-touch demonstrations, developers have attempted to recreate physical properties of tangible, real-world objects. They have attempted to make objects on-screen behave as objects in the real world do when moved on a table. These objects, if moved quickly and released, will continue to move even after the user has released them. The objects will come to a stop as if suppressed by the force of friction. I wanted developers using AQUA-G to be able to utilize this visually appealing effect.

To achieve this, I set out to create an AQUA-G gesture which would provide this feature. The result has been named the Kinetic Gesture, and it is intended to be used as an event translator. If the gesture detects that the user has moved an object quickly, when the object is released, the gesture will simulate additional events which appear to continue moving the object. The speed of the simulated events will decrease over time, as if by friction.

In order to implement this gesture, it was first necessary to decide how to determine if the object should continue to move after the user has released contact with the object. I have implemented this detection based upon the acceleration of the point of interaction with the object. I assume that if the object is accelerating in the same direction it is traveling, then it should continue moving after the user releases it, and if it is decelerating, it should not continue to move.

Therefore, the kinetic gesture determines that a point of interaction should continue moving based on the location of the five most recent events received for that point of interaction. Upon receiving an up event for that point, it calculates the average velocity and acceleration of these last five events.

If the acceleration is in the same direction as the velocity of the object, the gesture assumes that the object should continue moving, and starts a new thread. This new thread uses the average velocity which was calculated by the gesture and continues to

المنارة للاستشارات
www.manaraa.com

send move events after the real up event was received. To simulate the force of friction, the thread applies a constant deceleration of the object, calculates the resulting position and new velocity of the object, and sends these simulated events. Once the velocity has fallen below a threshold, the gesture sends an up event to the client application.

This gesture, similar to the flick gesture, was created by modifying the sample gesture project provided on the AQUA-G webpage.

3.6 The iPad and iPhone driver

As part of the evaluation of AQUA-G, a developer created an AQUA-G device driver for both the iPad and the iPhone. The device driver allows a user to utilize an iPad or iPhone to connect to AQUA-G. The developer completed this device driver quickly by reusing existing code to communicate with a standard mouse. An example of a user using this device driver is shown in Figure 3.11.

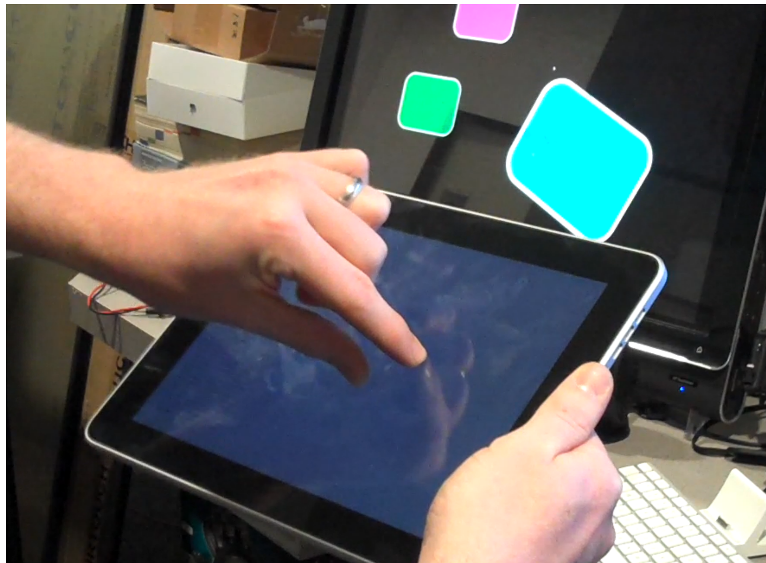

Figure 3.11 The iPad device driver.

Since the iPad or iPhone functions as an input device for AQUA-G, but not a client application, it cannot easily duplicate the screen that the client application shows. Cur-

www.manaraa.com

rently, the device driver works by mapping the input device screen directly to the client application screen. Thus, a user touch on the device is scaled and mapped to the larger screen. However, since the iPad and iPhone do not provide the pressure of the touch, it is impossible to know exactly where a user is about to touch until after they have done so. I have observed that this leads users to perform a "guess-and-check" style of interaction where they first tap to determine where their finger is on the larger screen and establish a frame of reference, then move their finger and tap again. A user will repeat this process until the object they find is selected.

This is not an ideal method of interaction because users, before interacting with the application, must spend time determining where their fingers are on the screen. Matejka et al describe means of using multi-touch displays to emulate a mouse (40), and it would be interesting to apply their techniques in order to allow the iPhone or iPad to function more like a wireless trackpad. This would provide interesting opportunities for exploring different means of interaction with this device.

This case study shows the capabilities of AQUA-G to communicate with devices over a wireless connection. Using a multi-touch input device which demands very high throughput over a network connection limits the performance of the device and the application. Thus, it will be necessary in future work to evaluate potential areas of improvement for performance when connecting input devices over a network connection.

المنارة للاستشارات

www.manaraa.com

CHAPTER 4. EVALUATION

AQUA-G was designed to allow other software developers to utilize it to create gesture-enabled applications. Therefore, in order to effectively evaluate AQUA-G, it was necessary to involve outside developers in the evaluation. In order to satisfy this goal, I involved outside developers in two ways: first, by proposing AQUA-G and asking for developer feedback, and second, by requesting developers to use the framework and participate in an interview upon completion of their experience. These two methods of evaluation have provided valuable feedback and validation of the usefulness of AQUA-G for developing gesture-enabled applications.

4.1 Initial developer survey

Before embarking on the task of designing and implementing a system like AQUA-G, I felt that it would be beneficial to request opinions and advice from other software developers who might benefit from such a system.

To evaluate the potential benefits of AQUA-G, I conducted a survey of developers to establish the merit of creating such a system. The survey was administered via an email request and was sent to a sample of developers, all male, between the ages of 20 and 50. Of those surveyed, nine responses were collected. Of these nine responses, three of them were from developers in academia (grad students), and six of them were from developers working in industry. Most developers were alumni of Iowa State University, and all had prior experience developing end-user applications.

المنارة للاستشارات
www.manaraa.com

The developers were provided a PDF document with a description of the AQUA-G concept and features, and were asked to answer questions about the proposed design. The questions in the survey revolved around initial reactions, perceived advantages and disadvantages, and other miscellaneous queries. During evaluation of developer feedback, several major themes emerged. Among these themes were the need for cross-platform compatibility, ease of use, and others. Table 4.1 shows some comments from developers categorized into their respective themes. Furthermore, the number in parenthesis represents the number of developers that made a comment related to this theme. What follows is a summary of their comments in response to the questions asked. This evaluation provided insights into the design of AQUA-G and also affirmed that the system would be valuable to develop.

The initial proposal was generally well-received by the developers who responded to the survey. When asked about their initial reactions, developers commented that "it sounds like it will have some very good features that would be useful in a research or development environment," and that "anything that abstracts away the input from several devices is a great idea." Developers did express some initial concerns with the work as well, commenting that "I'd want to hear more about how the system actually will be implemented," and "Does the input device / gesture events adhere to some standard?" These initial concerns generally expressed a desire to obtain more information.

When asked about the potential advantages of such a system, two developers expressed their opinion that the biggest advantage would be in saved time for developers. Others commented that advantages would be "flexibility to plug-n-play with any device given the proper input interpretation," "multiple input devices with dynamically loading gestures," and "modular generic gesture recognition modules."

Developers were also asked what they thought might be the primary limitations of such a system. They expressed that it would be difficult to convince people to use AQUA-G if alternatives exist. Another concern expressed was that of performance. One

المنارة للاستشارات
www.manaraa.com

Table 4.1 Major emergent themes in initial survey.

Theme	Selected Quotations
Cross-platform importance (6)	"Cross-platform compatibility is extremely important - I'd need it to run on all three major OSes to consider using it" "It is not critical that it be cross-platform but it is a plus. We use both Linux and Windows OS." "You need Win7 and OSX. As soon as you do that, you may as well support Linux."
Developer experience (5)	"It could be difficult to convince people to write drivers for the system when another alternative system may already exist." "If it's any more difficult to use than a regular event driven architecture most people won't see the benefits." "[Consider the] developer interface."
Ease of use (5)	"It sounds like it would make it very easy for developers to add support for receiving gestures from a variety of input devices." "This would clearly make creating applications easier." "Ease of use and configurability of the system would be key criteria."
Performance (3)	"More generic systems are prone to being less performant, and that can really matter when you're dealing with user input." "As the number of gestures supported grows (specifically static gestures), it may have a significant impact on the responsiveness of the system (since multiple inputs from multiple devices are supported)."
Other themes (2)	Flexibility, saved time for developers, dynamically loaded gestures, developer (open source) community, choice of programming language

www.manaraa.com

developer noted that "most generic systems are prone to being less performant, and that can really matter when you're dealing with user input." It was therefore crucial to take performance issues into account when developing AQUA-G.

Developers wrote that they would be excited to try out the system when it was completed. They generally expressed a desire for the system to be cross-platform, and only three noted that they had heard of systems similar to this, those systems being VPRN (68), VR Juggler (7), and Sparsh-UI (58).

4.2 User evaluation

Evaluation of a software framework is crucial in order to determine its usability and usefulness. AQUA-G was no exception, especially since it is to be used by a wide community of developers. Effectively evaluating whether a software framework will be useful should involve asking developers to use the framework and report back on their experiences. I evaluated AQUA-G in this way, and interviewed the developers about their experiences with AQUA-G. Though the sample size of developers was relatively small, the qualitative feedback they provided regarding AQUA-G was essential in determining some of the future work for the framework.

I requested developers to use AQUA-G to develop gestures, applications, and device drivers, and participate in an interview about their experiences. I successfully recruited six other software developers. Five of these students were graduate students in technology-related disciplines, and one was a faculty member in the Dept of Electrical and Computer Engineering. All participants were male, and all had no prior experience with AQUA-G, though one undergraduate student had previously developed software for Sparsh-UI.

Four participants developed gestures for AQUA-G - three worked on a double-click gesture independently, and one worked on a flick gesture. One participant completed an

المنارة للاستشارات
www.manaraa.com

iPhone/iPad device driver, and another completed both a joystick device driver and an application for the LABET UAV case study described in chapter 3.

4.2.1 Method

The participants were tasked with developing additional components for AQUA-G. Each participant was given a short verbal introduction to AQUA-G, a description of the task he would complete, and a verbal walk-through of the AQUA-G website and available documentation and sample projects. After completing this introduction, participants completed their respective development tasks.

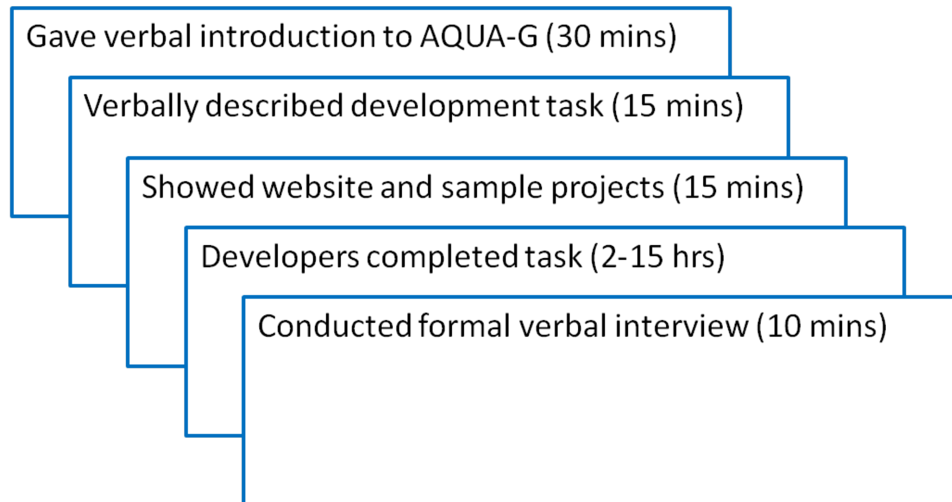

Figure 4.1 Method for conducting developer study.

The participants were instructed that they could ask questions throughout their development process in order to clarify questions or concerns they might have about using AQUA-G. I answered these questions and helped explain the architecture in greater detail, and explained the method for developing code for use AQUA-G, but did not review or assist the participants in writing code to develop their component. As a result, I did not influence the implementation or design of the participants' solutions, or help them write the code to interface their component with AQUA-G.

المنارة للاستشارات
www.manaraa.com

Upon completion of their development task, the participants were interviewed in person about their experiences. Both interviews were recorded for later review and analysis. The interview protocol used may be found in Appendix C, and it contains questions which inquire about the developers' perceived advantages and disadvantages of AQUA-G, how they would rate their experience using it, what could be improved about the development process, and more.

I have selected two example tasks and interviews to present as examples of the evaluation conducted. This will demonstrate in greater detail the study method and the types of questions the participants were asked during the interview.

4.2.2 Example student 1

4.2.2.1 Task

I tasked a computer engineering undergraduate student with developing an iPad and iPhone device driver, and the result of his assignment was discussed as a case study in Chapter 3. The student was provided with AQUA-G documentation and sample code and told to write a device driver for the iPad and iPhone.

4.2.2.2 Difficulties encountered

The student reported that a lot of the time he spent working on the device driver for AQUA-G was working on iPhone or iPad-specific issues, rather than AQUA-G issues. He reported that the biggest problems he encountered were not knowing the specifics of how AQUA-G worked. He didn't know what the device identifier (device ID) was for, or that the array of 3 floats in the touch point represented the x, y, and z of the touch point. The other problems he encountered concerned learning how to develop for the iPhone / iPad. The difficulties this student had could be partially mitigated by improving the documentation and sample projects on the website.

المنارة للاستشارات
www.manaraa.com

4.2.2.3 Results

The student succeeded in developing both an iPad and iPhone device driver for AQUA-G which connects over a wireless connection to the AQUA-G gesture server. The student estimated that he spent approximately 15 hours developing the device driver, which is described as a case study in Chapter 3. The student's abbreviated questions and responses to the interview questions are given in Table 4.2. During the interview, the student noted that there is currently no way to send feedback-related information to input devices to turn on lights, enable vibration, or perform other feedback actions. This will be discussed as future work.

4.2.3 Example student 2

4.2.3.1 Task

I tasked a mechanical engineering student with developing a flick gesture which is also described in a case study in Chapter 3. The student was provided with AQUA-G documentation available on the website as well as sample gesture code.

4.2.3.2 Difficulties encountered

Due to the student's background in mechanical engineering, he was not as experienced with C++ and general programming knowledge as was the first student. This student encountered difficulty setting up the project and build process correctly, especially using Visual Studio on Windows.

Additionally, the student expressed that he had difficulty determining the flow of the events/gestures and figuring out where his code needed to go. He mentioned that this was clarified during a discussion he and I had about the architecture of AQUA-G. This perceived difficulty suggests that the AQUA-G website should contain more documentation regarding the architecture on the wiki page on the website.

المنارة للاستشارات
www.manaraa.com

4.2.3.3 Results

The student estimated that he was able to complete his development task in about 10 hours. The fact that the student was able to learn about AQUA-G having never heard of anything like it, read the documentation, and complete a gesture and event so quickly, is a testament to the student's skill as well as to the design of AQUA-G.

As a result of the design of AQUA-G, the student didn't have to touch the original AQUA-G source code and instead was able to focus on only the code necessary to recognize the gesture. However, the student mentioned that he had in fact used existing gesture code in the AQUA-G source code, which was not part of the sample project he was provided. He noted that the sample projects were too simple, and seeing a more complicated gesture which involved data processing was more helpful. The student's responses to the interview questions are given in Table 4.3.

This will be taken into account for future work and improvement of the framework. The student and I discussed ways which more complicated sample projects could be provided to better show how to write gesture-processing code.

4.3 Discussion of evaluation results

All participants completed their development tasks and created contributions to the AQUA-G framework. The iPad and iPhone device driver, flick gesture, double-click gesture, and joystick driver are now part of the AQUA-G project and can be used with AQUA-G.

All participants expressed positive feedback about their experience. They stated that developing a device driver or application would not be a difficult task for the average developer. The participants most likely based this opinion on the fact that they were all able to complete their tasks in less than 15 hours, demonstrating that developing a component for AQUA-G does not involve an inordinate amount of time. Considering the

www.manaraa.com

fact that this was the participants' first experience with AQUA-G, and that development time for developing additional components would likely decrease, these results show that I was successful in creating a framework which the participants found easy to use and develop for.

Furthermore, because these participants developed their components using AQUA-G, after finishing development of the individual component, each could easily create a fully functional application which utilized their module with very little additional development time. In comparison, asking these participants to write all of this code from scratch, including a device driver, application, and gestures, would take significantly more time.

Still, the participants noted some things could be improved about AQUA-G. They provided valuable feedback, some of which will be included as future work for AQUA-G. Among the most influential and unique ideas the participants came up with for improvement of AQUA-G were:

- Provide a means of sending data back to input devices to trigger lights, vibration, or other input device-specific feedback.

- Provide more complex sample projects in addition to the simple skeleton projects already available - the ones provided are too simple to be able to learn from.

The interviews allowed the participants to voice their opinions and reflections on their experience developing using the framework, and the feedback they provided was and will continue to be essential in determining areas of improvement for AQUA-G.

المنارة للاستشارات
www.manaraa.com

Table 4.2 Student 1's responses to interview questions.

Question	Answer
What do you think are the biggest advantages of using AQUA-G?	The fact that it can natively handle multiple devices.
What do you think are the primary limitations of AQUA-G?	It not being able to send any time of data back to the device. It seems like that's coming up more and more, with vibration stuff.
How would you rate your experience writing a device driver for AQUA-G? Was it positive or negative and why?	Positive because it was pretty quick; I didn't have too many problems with it; the total development time was small.
Could anything be improved about the development experience?	Making the whole idea of AQUA more known to people who don't know about it; the wiki has a couple things like the tutorials to help. Something that confused me was there is the touch point, there is a float array and it doesn't really show that the first two are for x, y and z, and that was a little confusing, so maybe just go through the simple things.
Was the network protocol and event structure documentation clear to you?	Yes - well, I don't know about the documentation, but after reading the sample code it was pretty clear.
Do you think creating a device driver is a difficult task for the average developer? Maybe rate on a 0-10 score?	2 on a 0-10 scale; it's pretty easy after seeing the sample code. I think if you were doing it from scratch, just reading the documentation, it would be pretty hard. There's a lot of different drivers now, so it's pretty easy to look at these kind of things.
What can be done to encourage other people to contribute to the project?	This is always the battle with open source projects - you can modify the license to say that if you modify it, you have to post it. Have some kind of feedback or a really good community around it - that helps a lot because people want to contribute to a big community. Make it easy to contribute.
What do you think other developers will like best or least about it?	It's really easy to work with it; the amount of sample code and documentation makes it easy. Least - maybe it doesn't work well with their existing systems.
What can be done to convince other developers to use AQUA-G?	Other open source projects have a good community; so we need a good community behind it. As soon as you get the community, I think a lot of development will happen on its own.

المنارة للاستشارات
www.manaraa.com

Table 4.3 Student 2's responses to interview questions.

Question	Answer
What do you think are the biggest advantages of using AQUA-G?	Convenience, once you have it finished, you can put it on a lot of different systems; because of the intent, it's very flexible.
What do you think are the primary limitations of AQUA-G?	I don't really know - a unique solution might be a lot more tuned to the system though.
How would you rate your experience writing a device driver for AQUA-G? Was it positive or negative and why?	It was cool - it was really interesting. It's very neat to be part of something that you can see where it's going and where it's going to be used. I've done programming, but I haven't done a lot of larger scale things, but this was interesting because it was a large scale. It was really useful to take existing gestures and just use them as a template. The sample code was helpful, especially the source files (the AQUA-G gestures and not the sample projects).
Could anything be improved about the development experience?	The way that the tutorials are set up are geared to people who have a lot more knowledge of programming than I do; maybe I just have less than everyone else, but I think if the tutorials were structured around making a more complicated event or complicated gesture, so for me it was a little hard to see what's supposed to happen where, because there was really no data processing that had to happen in the sample.
Do you think creating a device driver is a difficult task for the average developer?	I would not think so. I would think that the trick would be refining the logic so that you get the results you intended.
How was creating the event? Was it difficult?	Relatively speaking, I think it would be easy; when I was doing it I was still learning about AQUA-G. Going back and editing it later, it was a lot easier.
What can be done to encourage other people to contribute to the project?	I guess a lot of that would be application specific. If it was able to be something that was able to be used with not just multi-touch type things. I don't know how many people would use it if they're not going to use that sort of device.
What do you think other developers will like best or least about it?	If you were making a lot of events and gestures; I think events are very similar - there's very few things that are unique to each event. Creating unique events might get tedious when it's so similar to others. Gestures have the same template to follow. If someone sat down and wanted to write 100 gestures, they might get tired of sitting down and writing the same thing over and over again.

المنارة للاستشارات
www.manaraa.com

CHAPTER 5. CONCLUSION

This research proposed to solve the following problem. Developing gesture-enabled applications is far from trivial. The time investment required to communicate with input devices, recognize gestures, and provide user feedback is significant. A framework which decreases this time investment could encourage other developers to try out new and novel input devices and use gestures in their applications. The resulting increase in exploration of gesture-enabled input methodologies would advance research in the field of more natural human-computer interaction.

In this thesis I have presented AQUA-G, a software architecture which enables rapid development of gesture-enabled applications. I have described the need for a framework such as this which provides cross-platform, cross-language support. I have described the software architecture of AQUA-G, which has been designed to satisfy this need. Furthermore, as validation that the framework is useful, I have presented case studies of developers using AQUA-G, and described a brief qualitative evaluation of the software framework.

I will conclude this thesis by reiterating the primary advantages and limitations of AQUA-G, and discussing future work for the software framework.

5.1 Primary advantages

AQUA-G is a cross-platform, cross-language solution for the gesture-enabled application development problem. It provides dynamic loading of gesture and event code,

المنارة للاستشارات
www.manaraa.com

which provides added flexibility and simplicity for developers wishing to customize the framework for their own applications. It communicates with a variety of input devices, and allows for these input devices to be connect simultaneously to produce engaging interaction paradigms. As a result, it is easy for developers to try out applications with different input devices in order to evaluate the effectiveness of using different means of input for their application. Finally, the framework is open source and allows anyone to make contributions, which will encourage the future development of AQUA-G.

5.2 Limitations

I would like to acknowledge limitations of AQUA-G that will require future work to improve or eliminate.

One of the developers in the user study noted that there was no way to send information back to input devices for the input devices to provide feedback. This is especially important for devices than can provide haptic feedback, such as standard haptic devices, Wii Remotes, or other similar input devices. The framework was designed to allow data to flow towards the client application, but not the other way around. This helps to avoid circular references and possible infinite loops. As such, no accommodation was made to allow the application to send information back to the input device.

A current workaround for this limitation would be to write a program for an input device which functions as both an input device driver and client application. This two-way exchange of data would allow for client applications to send information back to input devices. This workaround has been implemented in the LABET case study described in Chapter 3 for the unmanned vehicle because it needs to both send and receive information.

The other limitation that has been discovered is that input devices connecting over a network connection through TCP sockets do not have the necessary throughput to

المنارة للاستشارات
www.manaraa.com

produce desirable performance. This is not a problem for devices which do not send a large amount of data over the network connection, nor is it a problem for devices which are connected on the same machine as AQUA-G. However, for devices which are sending touch points or other large amounts of data, a performance decrement is visually apparent.

One potential solution for this limitation will be to include a high-bandwidth UDP socket connection to AQUA-G. This socket would accept connections from input devices and/or client applications requiring higher bandwidth.

		Next-Window API	Windows 7 Touch SDK	iPhone SDK	MT4j	PyMT	Tisch	Sparsh-UI	AQUA-G
Platforms	Windows	●	●		●	●	●	●	●
	Linux				●	●	●	●	●
	Mac OS X			●		●	●	●	●
Available app development languages	Java				●		●	●	●
	Python					●	●	●	●
	C++	●	●				●	●	●
	C#	●	●				●	●	●
	Other			●					●
Input devices	TUIO				●	●	●	●	●
	Multitouch	●	●	●	●	●	●	●	●
	Mouse				●	●	●		●
	Wii Remote						●		●
	Windows 7 Touch		●		●	●		●	●
	Other								●
Features	Provides standard multitouch gestures	●	●	●	●	●	●	●	●
	Allows custom gestures				●	●	●	●	●
	Allows custom input devices		●		●	●	●	●	●
	Provides gesture-aware UI widget set			●	●	●	●		
	Support for non-touch events								●
	Dynamically loaded gestures								●

Figure 5.1 A comparison of gesture recognition systems

المنارة للاستشارات

www.manaraa.com

5.3 Future work

Many improvements are available for AQUA-G, and a great deal of work is needed to advertise AQUA-G to begin creating the community of developers that will advance this research. As described by a developer in the user study, the website must be improved to include documentation which describes the architecture and design of AQUA-G. This will help potential developers understand how the framework operates and where to begin writing code which will be compatible with AQUA-G.

A framework modification should be investigated to allow client applications to send information back to input devices. This will encourage the development of software which communicates with devices providing haptic feedback, including Wii Remotes and other haptic devices.

Improvements should be investigated to allow high-throughput devices to be connected to AQUA-G over the network with less performance loss. This will have the additional benefit of encouraging the development of device drivers and gestures which utilize camera information or other large amounts of data for gesture processing. A potential solution as described above would be to give AQUA-G an additional socket utilizing higher-throughput UDP or other datagram-oriented protocol, which would allow for higher data rates.

Other future work could involve adding provisions to allow customization of gesture parameters by client applications. For example, the kinetic gesture utilizes a coefficient of friction to slow down the components over time, as if by the force of friction. AQUA-G does not currently allow for this time of runtime customization. In the future, AQUA-G could allow client applications to supply this coefficient of friction if they desire to represent different types of surfaces which have different coefficients of friction. This could also be easily extended to customize other gestures which might require user-supplied parameters.

المنارة للاستشارات
www.manaraa.com

Finally, future work should be undertaken to provide gesture-aware UI widget sets for different UI frameworks. Figure 5.1 reiterates the comparison of existing systems to AQUA-G presented in the first chapter. AQUA-G satisfies all of the limitations of existing systems except for providing gesture-aware widget sets for different programming languages. Therefore, this should be an area of immediate exploration.

This concludes this thesis. I have high expectations for AQUA-G, and hope that developers will begin utilizing AQUA-G to contribute device drivers, gesture recognizers, and client applications which will be accessible to a wider development audience, furthering research in gesture-enabled application development.

The AQUA-G website provides a demonstration video, several wiki tutorials, sample projects, and executable downloads, and it also hosts the source code repository. It is maintained through Google Code and is accessible on `http://code.google.com` (60).

المنارة للاستشارات
www.manaraa.com

APPENDIX A. DEVELOPING SOFTWARE USING AQUA-G

In this chapter, I will describe how to develop software using AQUA-G. Much of this information can also be found on the website (60) which was developed as part of this work, and it is provided there to help developers understand how to write software for AQUA-G.

AQUA-G is easily customizable. It allows for developers to write new input device drivers, client applications, gestures, and events quickly and easily. Each section in this chapter will describe the process of creating a custom module for use with AQUA-G.

A.1 Developing a custom event

Occasionally, developers may find it necessary to write a custom event for an application that uses AQUA-G. This is often necessary if they have a new input device that AQUA-G does not recognize, such as a game controller. In addition to writing the input device driver for AQUA-G, he or she may also want to create some custom events, such as button presses, controller accelerometer readings, etc.

Writing new events in AQUA-G is straightforward. In AQUA-G, events are dynamically loaded by the gesture server, so there is no need to download or look at the source code for the gesture server.

Since the events are dynamically loaded by the gesture server, they need to be compiled as shared libraries. This means that for Windows, events will be compiled into a

www.manaraa.com

dynamic link library (*.dll file) and on Linux, they will be compiled into a shared object library (*.so file).

There are some requirements that developers must fulfill to create a custom AQUA-G event:

1. The custom event must subclass the Event class defined in Event.h.

2. The custom event class must declare a char buffer to hold its serialized data.

3. The event class name must exactly match the library name. Developers should define a class "MyEvent" which will get compiled to "MyEvent.dll" or "MyEvent.so"

4. The event .cpp file must export the functions required for dynamic loading. The function is named "createEvent" and is described below.

5. Events in AQUA-G are serialized over the network. Developers must implement a constructor which takes as input a single byte array containing the event's data, and also implement the serializeData method, which returns a byte array containing the same data. This will allow AQUA-G to send the event from the input device to the gesture server and application.

6. The compiled library must be placed where the AQUA-G gesture server can find it. Developers can simply copy the .dll or .so file into (AQUA-exe-home)/events.

In order for the event to be dynamically loaded, you must export the function createEvent, which is used by the operating system to dynamically load the class. Throughout this section, I will use the UnifiedZoomEvent class provided by AQUA-G as an example. Here is an example of the createEvent() function for Windows and Linux:

```
#ifdef _WIN32
extern "C" {
    __declspec (dllexport) Event* createEvent(char* data) {
```

المنارة للاستشارات
www.manaraa.com

```
        return new UnifiedZoomEvent(data);
    }
}
#else
extern "C" {
    Event* createEvent(char* data) {
        return new UnifiedZoomEvent(data);
    }
}
#endif
```

Note that developers will need to change the return statement of each of these functions to match the desired event name (instead of UnifiedZoomEvent, above).

The event class also has to declare a data buffer to hold the data which will be serialized by the serializeData method. Developers will want to declare a member variable "dataBuffer" whose length is the same as the number of bytes in the custom data. This is done in the zoom event class as follows:

```
#define UNIFIEDZOOMEVENT_DATA_LENGTH 16
#include <string>
#include "Event.h"

class UnifiedZoomEvent : public Event {

// Attributes
private:
    float _zoomScale;
    float _zoomCenter[3];
```

المنارة للاستشارات
www.manaraa.com

```
    char  _dataBuffer[UNIFIEDZOOMEVENT_DATA_LENGTH];
    ...
```

Finally, the event will be serialized over the network, so developers need to implement a constructor which takes a byte array containing the event data, as well as the protected serializeData method which returns a byte array containing the same data. AQUA-G serializes events by calling (CustomEvent).serialize(). The serialize method is implemented in the Event superclass, and it delegates the custom data serialization to serializeData(), which is implemented in the custom event class.

Here is a the serializeData method for the UnifiedZoomEvent class:

```
/**
 * Constructs a char array with this event's data.  Data:
 *  - 4 bytes : zoom scale (float)
 *  - 4 bytes : zoom center x-coord
 *  - 4 bytes : zoom center y-coord
 *  - 4 bytes : zoom center z-coord
 */
char* UnifiedZoomEvent::serializeData(short& outLength) {

    outLength = UNIFIEDZOOMEVENT_DATA_LENGTH;

    float tempScale, tempX, tempY, tempZ;

    tempScale = _zoomScale;
    // zoom center
    tempX = _zoomCenter[0];
    tempY = _zoomCenter[1];
```

المنارة للاستشارات
www.manaraa.com

```
    tempZ = _zoomCenter[2];

    if (EndianConverter::isLittleEndian()) {
        tempScale = EndianConverter::swapFloatEndian(tempScale);
        tempX = EndianConverter::swapFloatEndian(tempX);
        tempY = EndianConverter::swapFloatEndian(tempY);
        tempZ = EndianConverter::swapFloatEndian(tempZ);
    }

    memcpy(_dataBuffer + 0, &tempScale, 4);
    memcpy(_dataBuffer + 4, &tempX, 4);
    memcpy(_dataBuffer + 8, &tempY, 4);
    memcpy(_dataBuffer + 12, &tempZ, 4);

    return _dataBuffer;
}
```

Please examine how this method works. All the method needs to serialize is this class's data members. The members of the Event superclass are handled automatically. First, the method sets the parameter outLength to the number of bytes in the returned char buffer. In the example, I use the length which I previously defined in a header file. Then, the method serializes the event's custom data. First, it copies the event's custom data into temporary variables. The data must be returned in network endian or big-endian form, so the method changes the endianness if necessary. Developers may use the EndianConverter utility class provided in sample projects to perform this operation, as shown in the above code. Finally, copy each data member into our data buffer, and return it.

المنارة للاستشارات
www.manaraa.com

Along with the serializeData method, developers must also implement a constructor which un-does this serialization. Here is the constructor for UnifiedZoomEvent:

```
UnifiedZoomEvent::UnifiedZoomEvent(char *data) : Event(data) {
    int i;
    int dataPos = (_name.length() + _description.length() + 2 + 17);

    memcpy(&_zoomScale, &data[dataPos], 4);
    dataPos += 4;
    memcpy(_zoomCenter, &data[dataPos], 12);

    // Handle endianness.
    if (EndianConverter::isLittleEndian()) {
        _zoomScale = EndianConverter::swapFloatEndian(_zoomScale);
        for (int i = 0; i < 3; i++) {
            _zoomCenter[i] =
                EndianConverter::swapFloatEndian(_zoomCenter[i]);
        }
    }
}
```

Note how this constructor works. The class UnifiedZoomEvent first calls the superclass constructor Event(data), which un-serializes the parameter data into the Event class's fields. Then, the constructor must un-serialize its custom data.

When events are sent over the network, the data pointer contains all of the data for the event. This is why the variable dataPos is defined in the above code - since the data parameter points to the start of the data buffer, and some of this data is the Event class's data, developers can use dataPos to represent the first data item of the custom

www.manaraa.com

data. This declaration and initialization should be the same for all events.

Then, the method copies the data members into the custom fields zoomScale and zoomCenter, and handles the endianness appropriately. Notice the "action and reaction" style of the constructor and the serializeData method. One puts the data into a byte array, and the other pulls the data back out.

Once developers have finished implementing the event and can compile it successfully, it is ready to use. The event can be used by input devices, gestures, and the application itself. In order to use the event with custom gestures, developers must place the compiled library where the AQUA-G gesture server can find it. They must place the .dll or .so file into (AQUA-exe-home)/events. The next time a developer runs AQUA-G, the event will be dynamically loaded.

A.2 Developing a custom gesture

Allowing developers the ability to write their own custom gestures was one of the key factors in developing the design and architecture for AQUA-G.

Since gestures are dynamically loaded through the use of shared libraries, when a developer creates a gesture, it must be compiled to a .dll file on Windows or a .so file on Linux. The AQUA-G gesture server recognizes these files and loads the gesture code at runtime. This means that developers don't have to look at a single line of AQUA-G code to write their own gesture.

To write a custom gesture, developers must satisfy the following requirements:

- The gesture must subclass the Gesture class defined in Gesture.h.
- The gesture class name must exactly match that of the compiled library. Developers must define a class such as "MyGesture" which gets compiled to "MyGesture.dll" or "MyGesture.so"

المنارة للاستشارات
www.manaraa.com

- The gesture class must export a function "createGesture" which is used by AQUA-G to load the class dynamically.
- Developers must override the method "handleEvent" in the Gesture class. This is where the gesture can process incoming events.
- Developers must place the compiled library into a place where Aqua can find it, in (AQUA-exe-home)/gestures. They will find all of the included gestures in this directory.

In this section, I will describe and analyze the simple HelloWorldGesture which is provided in AQUA-G. Here is the header file for this class:

```
#ifndef _HELLOWORLDGESTURE_H_
#define _HELLOWORLDGESTURE_H_
#include "../events/Event.h"
#include "Gesture.h"

class HelloWorldGesture : public Gesture {
// Methods
public:
    HelloWorldGesture(EventProcessor* publisher, int regionID = -1) :
        Gesture(publisher, regionID) {};
    virtual bool handleEvent(Event* event);
private:
    void printHello();
};
#endif
```

المنارة للاستشارات
www.manaraa.com

Notice how simple this class definition is. It declares the HelloWorldGesture class, which subclasses the Gesture class. It then declares a constructor which calls the Gesture superclass constructor. Developers should provide a constructor with this same signature in their gesture. The EventProcessor object is the EventProcessor that the gesture will send events that it generates to, using the sendEvent method provided by the Gesture class. This is described below.

Additionally, developers need to export functions which are used by AQUA-G to load the gesture class. These can be placed in the header file or the cpp file. Here are these functions for the HelloWorldGesture:

```
#ifdef _WIN32
extern "C" {
     __declspec (dllexport) Gesture* createGesture(EventProcessor*
            publisher, int regionID) {
        return new HelloWorldGesture(publisher, regionID);
    }
}
#else
extern "C" {
    Gesture* createGesture(EventProcessor* publisher, int regionID) {
        return new HelloWorldGesture(publisher, regionID);
    }
}
#endif
```

Developers will have to modify the return statement so that it returns an instance of the new gesture, but otherwise these methods should remain the same.

Here is the implementation cpp file for HelloWorldGesture:

www.manaraa.com

```
#include <stdio.h>
#include "HelloWorldGesture.h"

bool HelloWorldGesture::handleEvent(Event* event) {
    printHello();
    publishEvent(event)
    return false;
}

void HelloWorldGesture::printHello() {
    printf("Hello, world! My region ID is: %d\n", _regionID);
}
```

The cpp file implements only one public method, handleEvent. This is where all of the gesture processing code belongs. HandleEvent is called by AQUA-G whenever a new event is available for this gesture to process. In this simple gesture, I simply print "Hello, World" along with this gesture's regionID to the console, and publish the same event I received. However, more complex behavior can easily be defined. For example, the UnifiedZoomGesture handles events, calculates relative change in scale over time, creates instances of UnifiiedZoomEvent, and publishes these new events. AQUA-G then sends these events to the application.

A.3 Developing an input device driver

AQUA-G is designed to communicate with a limitless variety of input devices. In order for it to do this, it communicates with input devices using a custom serialization protocol over a TCP socket. Writing a new input device driver is made easier, however, by using the Event classes already provided with AQUA-G. These classes already

www.manaraa.com

have built-in serialization functionality which developers may use in their custom device drivers.

As stated above, AQUA-G communicates with input devices via TCP/IP sockets. Therefore, in order to communicate with AQUA-G, an input device driver must create a TCP socket. Then, the input device driver must send a single byte with value 0x02, which will identify it as an input device. Here is some example code which which performs this using Windows sockets:

```
#include <windows.h>
#include <winsock2.h>
#include <ws2tcpip.h>

SOCKET sock;
WSADATA data;
struct addrinfo *result = NULL;
struct addrinfo *ptr    = NULL;
struct addrinfo  hints;
int iResult;
char deviceType = 2;

iResult = WSAStartup(MAKEWORD(2, 2), &data);
if (iResult != 0) {
    printf("Error starting WSA\n");
    return 1;
}

ZeroMemory(&hints, sizeof(hints));
```

المنارة للاستشارات
www.manaraa.com

```
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP;

 // Resolve the server address and port
iResult = getaddrinfo("127.0.0.1", SERVER_PORT, &hints, &result);
if (iResult != 0) {
    printf("getaddrinfo failed: %ld\n", WSAGetLastError());
    WSACleanup();
    return 1;
}

sock = INVALID_SOCKET;

// Attempt to connect to the first address returned by addrinfo
ptr = result;
sock = socket(ptr->ai_family, ptr->ai_socktype, ptr->ai_protocol);

if (sock == INVALID_SOCKET) {
    printf("Error at socket(): %ld\n", WSAGetLastError());
    freeaddrinfo(result);
    WSACleanup();
    return 1;
}

iResult = connect(sock, ptr->ai_addr, (int)ptr->ai_addrlen);
if (iResult == SOCKET_ERROR) {
```

المنارة للاستشارات
www.manaraa.com

```
        printf("Error connecting to socket.\n");
        closesocket(sock);
        sock = INVALID_SOCKET;
        return 1;
    }

    freeaddrinfo(result);
    if (sock == INVALID_SOCKET) {
        printf("Unable to connect to server!\n");
        WSACleanup();
        return 1;
    }

    // Send our device type - 2, input device.
    iResult = send(sock, &deviceType, 1, 0);
    if (iResult == SOCKET_ERROR) {
        printf("Send failed: %d\n", WSAGetLastError());
        closesocket(sock);
        WSACleanup();
        return 1;
    }
```

Once this initialization is complete, it is quite easy to send events to AQUA-G. In the Event class, a single method is defined:

```
char* serialize(int& outLength)
```

This method returns a pointer to an array of bytes which contains the event data, as well as an integer "outLength" which contains the length of this array.

www.manaraa.com

To send the event to AQUA-G, the developer simply sends a short int containing this length, followed by the data. Below is example code which demonstrates how to do this.

```
int sendEvent(Event* e) {
    char outLength[2];
    short iOutLength, tempLength;
    int iResult;

    char* eventData = e->serialize(iOutLength);
    tempLength = iOutLength;
    if (EndianConverter::isLittleEndian()) {
        tempLength = EndianConverter::swapShortEndian(tempLength);
    }
    // Send the length of the event
    memcpy(outLength, &tempLength, 2);
    iResult = send(sock, outLength, 2, 0);
    if (iResult == SOCKET_ERROR) {
        printf("Send length failed: %d\n", WSAGetLastError());
        closesocket(sock);
        WSACleanup();
        exit(0);
        return -1;
    }

    // Send the event itself.
    iResult = send(sock, eventData, iOutLength, 0);
```

المنارة للاستشارات
www.manaraa.com

```
    if (iResult == SOCKET_ERROR) {
        printf("Send data failed: %d\n", WSAGetLastError());
        closesocket(sock);
        WSACleanup();
        exit(0);
        return -1;
    }

    delete[] eventData;
    return 0;
}
```

In order to develop an input device driver, it will be beneficial to review example code. A good place to start is by looking at the code for the Windows mouse input device driver provided by AQUA-G. Other example device drivers may be found by browsing the source code.

A.4 Developing a client application

AQUA-G communicates with client applications using a standard protocol over TCP sockets. Writing a new client application involves writing code which conforms to this custom protocol. Fortunately, much of this work has been done for developers wishing to develop Java and C++ applications. The implementation of said protocol will be outlined in this section, and the sample implementation is given in Java code, though it should be relatively easy to port this code to other languages.

In order to communicate with AQUA-G, a client application must create a TCP socket and connect to a predefined port. Then, it must send a single byte with value 0x03 on that socket. This alerts AQUA-G to the fact that the incoming connection

المنارة للاستشارات
www.manaraa.com

should be treated as a client application and should receive events appropriately. The code for initialization the connection is given below. The Java code sets up a socket with data input and output streams, and begins handling events from AQUA-G.

After initialization, the client application must begin reading data from the socket, which is coming from AQUA-G. AQUA-G will send the following messages to the client application, which it will be expected to process and handle appropriately. The message type is sent as a single byte, which it reads from the socket and uses to determine the appropriate action. The message types are as follows:

- Get Region ID (0)
- Get Global Info (1)
- Get Region Info (2)
- Process Global Event (3)
- Process Region Event (4)
- Get Event Translators (5)

Here is the sample code which connects to the AQUA-G gesture server:

```
public class AquaClient {


    enum MessageType {
        REGION_ID,
        GLOBAL_INFO,
        REGION_INFO,
        PROCESS_GLOBAL_EVENT,
        PROCESS_REGION_EVENT,
                TRANSLATORS
```

www.manaraa.com

```
    }

    private Socket _socket;
    private DataInputStream _input;
    private DataOutputStream _output;

    public static void main(String[] args) throws
            UnknownHostException, IOException {
        AquaClient ac = new AquaClient();
        ac.connect();
        while(true) ac.handleRequest();
    }

    /**
     * Connects to Aqua.
     * @throws IOException
     * @throws UnknownHostException
     */
    private void connect() throws UnknownHostException, IOException {
        _socket = new Socket("localhost", 3007);
        _input = new DataInputStream(_socket.getInputStream());
        _output = new DataOutputStream(_socket.getOutputStream());
        _output.write(1);
    }
}
```

Here is the code which handles the messages from Aqua. It is a simple switch statement

www.manaraa.com

which checks the message type which comes in from Aqua.

```
private void handleRequest() throws IOException {
    MessageType type = null;
    try {
        type = MessageType.values()[_input.read()];
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Message not recognized...\n");
        return;
    }

    switch(type) {
    case REGION_ID:
        handleGetRegionID();
        break;
    case GLOBAL_INFO:
        handleGetGlobalInfo();
        break;
    case PROCESS_GLOBAL_EVENT:
        processGlobalEvent();
        break;
    case PROCESS_REGION_EVENT:
        processRegionEvent();
        break;
    case REGION_INFO:
        handleGetRegionInfo();
        break;
```

المنارة للاستشارات
www.manaraa.com

```
        case TRANSLATORS:
            handleGetTranslators();
            break;
        default:
            System.out.println("Error - message not recognized: " + type);
            break;
    }
}
```

Notice that the code above simply checks the message type and hands off the processing of each message to appropriate handler functions. This is generally a good idea, and encouraged in implementation of a custom client application. I will describe in detail each handler function below.

The first handler method is handleGetRegionID. This function is called when AQUA-G recognizes an event that should be passed to a component in a client application's user interface. After the message type is sent, AQUA-G sends three floating-point numbers which represent the x, y, and z location of the new event. X and Y values will be normalized between 0 and 1, where (0,0) represents the upper left corner of the screen, and (1,1) represents the lower right corner. Z values may not be normalized as they differ depending on input device, but the convention is to represent it in centimeters from the surface of the display, increasing as distance increases.

The client application code must return a unique identifier for the component which this event occurred over. To do this, developers must keep track of the locations of all of the GUI components or have some means of determining which component lies where. Java provides this through the getComponentAt() method, and other GUI frameworks should be able to perform similar actions. It is up to the developer to maintain unique identifiers for each component.

www.manaraa.com

Here is the code for handling the getRegionID message:

```
/**
 * Handles the getRegionID message.
 */
private void handleGetRegionID() throws IOException {
    float[] location = new float[3];
    location[0] = _input.readFloat();
    location[1] = _input.readFloat();
    location[2] = _input.readFloat();

    if (location[0] < 0.5) {
        _output.writeInt(1);
    } else {
        _output.writeInt(0);
    }
}
```

This code maintains two groups, one for the left half of the screen, and one for the right half of the screen. The group ID is written back to AQUA-G as an integer value.

AQUA-G needs to know which global gestures should be allowed for the client application. Global gestures are those which are not associated with a particular UI component in the application. These gestures are generally things such as a “shake,” “wave,” or similar gesture, though technically any gesture can be classified as a global gesture. These gestures will receive all of the events from the input devices and process them accordingly.

Furthermore, developers may specify global allowed events, which will always be sent to the client application. They can use this to specify a certain type of event which should

المنارة للاستشارات
www.manaraa.com

be passed directly to the application without being passed to individual widgets. For example, developers may want to receive all touch events regardless of the component they occurred over if they want to log touch point information in an application.

In order to determine these global gestures, AQUA-G sends the client application the "Get Global Info" message. The application must implement the following protocol to send the allowed global gestures back to AQUA-G:

- Send an integer value representing the number of allowed global gestures.
- Send each gesture name as a NULL-TERMINATED string.
- Send an integer value representing the number of allowed global events.
- Send each event name as as NULL-TERMINATED string.

The application can also write 0 as the number of gestures or events and simply not write any strings, if it does not wish to receive gestures.

Here is sample code which performs this action:

```
/**
 * Handles the getGlobalInfo message.
 */
private void handleGetGlobalInfo() throws IOException {
    _output.writeInt(1);

    // Sent gestures as null-terminated strings.
    _output.writeBytes("HelloWorldGesture\0");

    // Send the events
    _output.writeInt(1);
```

www.manaraa.com

```
    _output.writeBytes(UnifiedEvent\0);
}
```

Similarly, AQUA-G will require knowledge of the event translators allowed for the client application. Event translators are gestures which generally consume some events and translate them into other events. An example of an event translator is a gesture which converts Wii Remote accelerometer values into velocity or position values. Another example could be a gesture which associates touch point information with hand location information provided by two separate input systems. The code for returning this information is as follows:

```
/**
 * Handles the getTranslators message.
 */
private void handleGetTranslators() throws IOException {
    // Send # of translators.
    _output.writeInt(1);
    // Send translator names.
    _output.writeBytes(UserIDHandGesture\0);
}
```

Now that the application has set up the global gestures and event translators, and knows how to return a unique identifier for each region in the UI, it is almost ready to begin processing events. After AQUA-G handles an incoming input device events by asking the application for the region ID, it will need to know the allowed gestures for that region. For example, allowed gestures for a photo in a photo organizing application might be zoom, rotate, and drag.

AQUA-G will request region information by sending the application a GET REGION INFO message. Then, it will send the application the unique region ID which it is

المنارة للاستشارات
www.manaraa.com

requesting information for. Here is some example code which will process this message appropriately. In this sample code, if the region ID is 1, the application will return a HelloWorld Gesture and a DragGesture; if the regionID is anything else, no gestures are allowed. When processing this message, the application should send AQUA-G a single integer representing the number of allowed gestures, follow by a null-terminated string containing the name of each gesture. Then, it may also tell AQUA-G which events should be allowed from this region. For example, it may choose to receive a WaveEvent or ShakeEvent on a particular region. These types of events would usually come from global gestures, but this code demonstrates that an application can allow them on a region-level only if it so desires.

```
/**
 * Handles the getRegionInfo message.
 */
private void handleGetRegionInfo() throws IOException {
    int regionID = _input.readInt();
    if (regionID == 1) {
        _output.writeInt(2);
        _output.writeBytes("HelloWorldGesture\0");
        _output.writeBytes(UnifiedDragGesture\0);
        _output.writeInt(1);
        _output.writeBytes(WaveEvent\0);
    } else {
        _output.writeInt(0);
        _output.writeInt(0);
    }
}
```

www.manaraa.com

Now, the application has handled all messages for which action is required. The only two messages left are the PROCESS REGION EVENT and PROCESS GLOBAL EVENT messages. These two messages are sent when AQUA-G sends the application an event. Here is code which will process these two types of messages. In sending the PROCESS REGION EVENT message, AQUA-G will send the application an integer value, which is the unique Region ID which that event should be passed to. In the PROCESS GLOBAL EVENT message, no regionID is sent.

After receiving the regionID if necessary, the application needs to read the event from the socket. AQUA-G will first send a short integer representing the number of bytes contained in the event. The application should read this first, then continue by reading the specified number of bytes from the socket and placing them into an array.

In practice, event un-serialization is typically handled by the event constructor. The first data in the event array is the name of the event. An application can read the name of the event by reading data from the array that it received until it reaches a null character. It can then compare this name and create the appropriate event, as shown in the sample code below.

```
/**
 * Handles the processGlobalEvent message.
 */
private void processGlobalEvent() throws IOException {
    //System.out.println("Got global event.");
    short length = _input.readShort();
    byte[] data = new byte[length];

    _input.read(data, 0, length);
    String name = "";
```

المنارة للاستشارات

www.manaraa.com

```
    int index = 0;
    while (data[index] != '\0') {
        name += (char)data[index++];
    }

    Event e = null;

    if (name.equals("UnifiedEvent")) {
        e = new UnifiedEvent(data);
        System.out.println("loc: " + e.getLocation()[0] + ", " +
                e.getLocation()[1] + ", " + e.getLocation()[2]);
    }
}

/**
 * Handles the processRegionEvent message.
 */
private void processRegionEvent() throws IOException {
    int regionID = _input.readInt();

    short length = _input.readShort();
    byte[] data = new byte[length];

    _input.read(data, 0, length);
    String name = "";
    int index = 0;
    while (data[index] != '\0') {
```

المنارة للاستشارات
www.manaraa.com

```
        name += (char)data[index++];
    }

    Event e = null;
    if (name.equals("UnifiedDragEvent")) {
        e = new UnifiedDragEvent(data);
    } else if (name.equals("UnifiedZoomEvent")) {
        e = new UnifiedZoomEvent(data);
    } else if (name.equals("Unified2DRotateEvent")) {
        e = new Unified2DRotateEvent(data);
    }

    for (AquaPhoto p : _photos) {
        if (p.getID() == regionID) {
            p.processEvent(e);
        }
    }
}
```

This concludes the section on writing client applications, input devices, and gestures for AQUA-G. Information for those wishing to obtain more information is provided on the website which was developed as part of this work, `http://aqua-gesture-framework.googlecode.com`

المنارة للاستشارات
www.manaraa.com

APPENDIX B. WAYFINDER

This appendix contains the paper "Wayfinder: Evaluating Multitouch Interaction in Supervisory Control of Unmanned Vehicles" (61) which was published in the Proceedings of the World Conference on Innovative Virtual Reality, 2010.

B.1 Title and authors

Wayfinder: Evaluating Multitouch Interaction in Supervisory Control of Unmanned Vehicles.

- Jay Roltgen, Department of Psychology, Virtual Reality Applications Center, Iowa State University, Ames, Iowa, 50010, jroltgen@iastate.edu
- Stephen Gilbert, Department of Psychology, Virtual Reality Applications Center, Iowa State University, Ames, Iowa, 50010, gilbert@iastate.edu

B.2 Abstract

In this paper we investigate whether the use of a multitouch interface allows users of a supervisory control system to perform tasks more effectively than possible with a mouse-based interface. Supervisory control interfaces are an active field of research, but so far have generally utilized mouse-based interaction. Additionally, most such interfaces require a skilled operator due to their intrinsic complexity. We present an interface for controlling multiple unmanned ground vehicles that is conducive to multitouch as well

www.manaraa.com

as mouse-based interaction, which allows us to evaluate novice users performance in several areas. Results suggest that a multitouch interface can be used as effectively as a mouse-based interface for certain tasks which are relevant in a supervisory control environment.

B.3 Introduction

Previous research has been devoted to developing effective supervisory control interfaces for unmanned aerial vehicles (UAVs) (62; 63). A supervisory control interface is an interface which allows an operator to coordinate a variety of processes in a complex system such as a nuclear power plant, a factory, or in this case, a fleet of UAVs. The operator does not control them directly, e.g. flying the UAV, but instead specifies goals or destinations to be reached. A common problem in this field is the desire to represent information to operators in such a way that they can perform tasks effectively and make limited errors.

Research is currently being performed at Wright Patterson Air Force Base (WPAFB) to investigate various supervisory control interfaces. Researchers there have developed an application called "Vigilant Spirit"(VS) (62) to serve as a framework for researching these interfaces as they apply to various real-world scenarios. VS provides UAV operators with supervisory control of multiple simulated UAVs.

Currently, VS utilizes a dual-monitor, mouse-and-keyboard environment. We have created an interface loosely based on VS that is more conducive to multitouch interaction, so that we may explore the potential benefits of multitouch interaction in supervisory control interfaces.

www.manaraa.com

B.4 Related work

Multitouch technology has received a great deal of interest in recent years. An advance in sensing technology and the popularization of do-it-yourself multitouch has made this technology available to a greater population than ever before. Several technologies have been made available to researchers as well as consumers, such as the iPhone (3), the Microsoft Surface (11), and others (11).

Recently, touch-enabled devices have also made their way into the PC market with the introduction of the HP TouchSmart (26) and Dell Latitude (15). It is our expectation that these multitouch-enabled PCs will continue to proliferate in the near future.

Other products such as the DiamondTouch (16), the Microsoft Surface (11), and the iPhone (3) have evolved into reliable sensing systems, and OEM vendors such as NextWindow (49) and N-Trig (53) are providing reliable multitouch sensing technologies to hardware manufacturers.

In addition to advances in the consumer market, much research has been devoted to multitouch sensing technology. Jeff Han is partially responsible for this recent spark of interest, with his paper detailing low-cost do-it-yourself multitouch sensing (24).

While a great deal of this effort is aimed at improving multitouch sensing technology and enabling end-users, additional research has been conducted to evaluate the benefits of multitouch in several different application domains.

Recently, multitouch interfaces have received attention in command-and-control applications (67; 69). One such example is COMET (67), where researchers seek to utilize a multitouch interface to enhance face-to-face collaboration of military officers planning missions on a virtual table. This work is primarily intended to evaluate the potential benefits of multitouch and digital interaction in this type of environment. The researchers are particularly interested in the abilities of the digital interface to save and record mission planning sessions, features that were not available with older technology

المنارة للاستشارات
www.manaraa.com

used for this type of planning work.

Other research has been performed to investigate various supervisory control interfaces (62; 63) which aims to determine what types of tasks and interfaces can have an effect on operator mental workload. Our research takes a similar approach, however this prior research in supervisory control interfaces has been exclusively targeted for mouse-based interfaces.

A great deal of work has been done in the area of remote robot tele-operation and control (20; 50), which exhibits a great deal of influence on this research. Some have even already begun to use multitouch interfaces as effective means for operating remote robots (41; 31), which may lead to more widespread adoption of multitouch interfaces in these types of direct control situations. This research area primarily involves direct control of vehicles, and we intend to build on this work as it may apply to more supervisory means of control.

Finally, advances in performance of touch-enabled hardware have facilitated research to determine if multitouch interfaces offer significant performance gains over similar mouse-based interfaces (46; 21). This research generally shows that multitouch can offer particular advantages for manipulating objects, but is perhaps less precise than standard mouse-based interaction. One of the goals of our research is to verify these results and show that they hold true in a supervisory control environment, and provide a realistic use case of multitouch technology. The results of this research will directly apply to current research in supervisory control interfaces.

It is our aim to bridge the gap which remains between research in multitouch interaction and research in supervisory control interfaces, and explore the extent to which a multitouch interface can be effective in this environment. To accomplish this goal, we have created the software application "Wayfinder".

المنارة للاستشارات
www.manaraa.com

B.5 Wayfinder

The Wayfinder application has been developed as a research platform with which to conduct studies on supervisory control interfaces that might apply to similar interfaces such as Vigilant Spirit. Typical screenshots of Vigilant Spirit and Wayfinder are given in Figure B.1 and Figure 2, respectively. Wayfinder has been designed such that it has similar features to the Vigilant Spirit application, which was developed by our fellow researchers at WPAFB. This is to ensure that the results of this research may apply to current supervisory control interfaces, and especially to Vigilant Spirit. We have chosen so substitute unmanned ground vehicles (UGVs), or rovers, for UAVs. This choice was motivated by our desire to make the application extensible enough to be used with both real and virtual vehicles, and the greater availability of unmanned ground vehicles in our research lab for future research involving real vehicles. Wayfinder is capable of communicating with virtual simulated rovers, as in this experiment, and it provides an invariant software interface for the vehicles which will allow us to use it for real vehicles in the future as well.

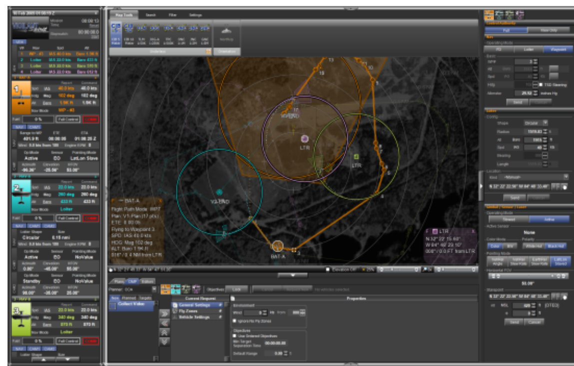

Figure B.1 The Vigilant Spirit controlInterface

www.manaraa.com

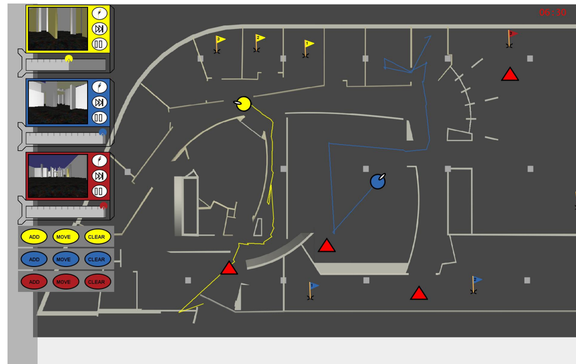

Figure B.2 The Wayfinder application. Visible are vehicles (circles), threats (triangles), waypoints (flags) and control panels (left).

B.5.1 Hardware and software

For multitouch input and gesture recognition, we have effectively utilized the Sparsh-UI gesture recognition framework (58). Sparsh-UI was developed by researchers at Iowa State University in 2008, and it provides a cross-platform gesture recognition system compatible with several input devices. Several other gesture recognition systems are available, but they do not provide the flexibility we desired.

Alternatives to Sparsh-UI gesture recognition include Tisch (18) and Multitouch for Java (45). We chose to use Sparsh-UI because it provides the functionality that we require in order to recognize and process multitouch input, and it is flexible enough to accommodate multitouch input from several types of multitouch-enabled hardware devices.

We decided to purchase and use a 25.5 HP TouchSmart (Figure B.3) device for this study, because it offered the screen real-estate necessary as well as reliable sensing. Due to certain multitouch-sensing limitations of the HP TouchSmart, we also used a second device, the 15.4 Stantum SMK multitouch device (Figure B.4). We chose to conduct two separate experiments with these two devices to more exhaustively evaluate the potential benefits of multitouch hardware.

Sparsh-UI was previously compatible with the Stantum SMK device; however, it was

المنارة للاستشارات
www.manaraa.com

not compatible with the HP TouchSmart. We chose to write a driver for the TouchSmart so that it too would be compatible with Sparsh-UI, allowing us to utilize both input devices as necessary.

Figure B.3 The 25.5” HP TouchSmart computer.

Figure B.4 The Stantum SMK 15.4” multitouch device

B.5.2 Features

Wayfinder provides many features that are common in most supervisory control interfaces. Its purpose is to enable an operator to monitor several UGVs simultaneously,

المنارة للاستشارات
www.manaraa.com

visualizing intelligence and threat information for him or her without overtaxing his or her mental capacity. It provides a top-down map which occupies most of the screen, as shown in Figure B.2. This top-down map functions as the main interaction space for the application. Vehicles appear on the map at their current positions, and users can interact with the vehicles in several ways, which are described in this section.

Additionally, Wayfinder allows users to drag, zoom, and pan this top-down map to view different areas of the map. This allows them to obtain an overall view of the map or zoom in for a more detailed view quickly and easily. In Wayfinder, we chose not to allow users the capability of rotating the top-down map to view it from different angles. This decision was motivated by a desire to maintain control over the tasks in the experiment, which might have varied in difficulty depending on the angle from which the operator viewed the map.

B.5.2.1 Simulated vehicles

In Wayfinder, the operator has supervisory control of three semi-autonomous vehicles (rovers). To instruct the vehicles to travel to intended destinations, he or she may specify navigational waypoints (see Setting Waypoints, below). Wayfinder can support multiple rovers, allowing as many as screen real estate and operator mental capacity will allow.

In this research, the operator controls three simulated "virtual" vehicles within a 3D model of a building rather than actual rovers. Though Wayfinder fully supports interaction with real vehicles, we have chosen to utilize simulated vehicles out of a desire to minimize hardware technical difficulties, video lag, and other variables which might confound our results.

These "virtual" vehicles are simulated with a sister application, which handles navigation and simulated video feeds. For simulating video feeds, we wrote an OpenSceneGraph (10) application which provides the video feed back to Wayfinder. All communication between this application and Wayfinder is performed via TCP/UDP Sockets.

www.manaraa.com

In addition, it is designed to conform to Wayfinders vehicle interface communication standard, meaning that it would be very easy to replace the entire application with code running on a real vehicle.

B.5.2.2 Video reviewing

Wayfinder allows users to view live video feed from each rover with the video control panels (Figure B.5). Each video panel is colored to match the rover that it is associated with. As described above, for this experiment, the video feed is provided by the OpenSceneGraph application which simulates rovers exploring a virtual 3D model of a building.

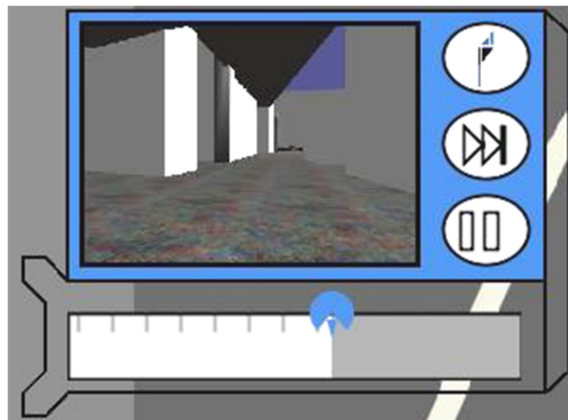

Figure B.5 Wayfinder's video control panel

The video reviewing functionality in Wayfinder is very complex and feature-rich so that it may reflect the needs of Air Force UAV operators. The participant may use the timeline shown beneath each video to replay and review older video. This is done by either clicking or touching the playhead shown on the timeline and dragging it back and forth. Additionally, the user can click or tap and drag the timeline itself to review older video if, for example, the playhead has reached the edge of the timelines boundary box. This feature allows the user to view older video in the event that a threat was detected earlier in the mission.

www.manaraa.com

If the user is reviewing old video, a transparent rover icon will be displayed on the screen to show the location of the rover at that point in time. This transparent rover is very useful to the participants who are reviewing video looking for threats, because it allows them to place the transparent rover on the map where it would have had a good view of the threat.

B.5.2.3 Setting waypoints

In Wayfinder, operators do not control the vehicles directly, but instead set intended destinations, or waypoints, by using the waypoint control panels (Figure B.6). Each waypoint control panel is colored to match both the vehicle and video panel that it is associated with.

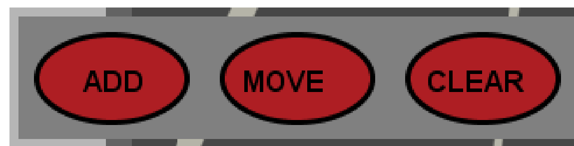

Figure B.6 Wayfinder's waypoint panel

These waypoints can be compared to a bread-crumb trail in which the rover will try to visit all of the waypoints sequentially. Since the rovers are semi-autonomous, they plot the quickest route to their destination automatically, and are able to avoid walls and obstacles that may be in their path.

We allow users to set waypoints with the multitouch interface by touching and holding one of the buttons on the waypoint control panel with one hand, then tapping locations on the top-down map to add or move waypoints with the other hand. For example, in order to add a waypoint for the red rover, a user would tap and hold the "add" button with the left hand. With this button held down, they may tap the map with the other hand. Waypoints will appear on the map where the user tapped.

المنارة للاستشارات
www.manaraa.com

This interaction style was motivated by our wish to have participants utilize both hands when interacting with the application. We observed in a pilot study that many users did not use both hands if they were not forced to. We conducted this pilot study with 5 participants, 3 male, 2 female, and observed their behavior in an attempt to improve the interface for the larger study. We observed during this pilot study that one of the male participants kept his right hand in his lap during the entire duration of the experiment. Thus, in an attempt to get our users working with both of their hands simultaneously in a bi-manual interaction style, we chose to require participants to set waypoints using both hands. We observed that users picked this style of interaction up very quickly, though it may not have seemed natural at first.

Similarly, to move waypoints, the user can tap and hold the "move" button and, with the other hand, drag the desired waypoint to a different a location. To clear all of the waypoints for a particular rover, the user must press and hold the clear button for 2 seconds.

With the mouse interface, the user only has one point of interaction with the interface, so we needed to change the interaction style. For the mouse-based interaction, we settled on a "modal" style of interaction. To enter a 'waypoint add mode" the user simply clicks on the "add" button. The button is now "down" similar to if the user was holding it down, as in the multitouch approach. Once in this mode, the user can click anywhere on the map to add waypoints at that point. Once they are finished adding waypoints, they simply click the "add" button again to finish adding waypoints, and the rover will begin moving.

B.5.2.4 Classifying threats

In our scenario, rovers will detect "threats" in their environment as they move around the map. Threats are represented by red triangles in the interface (Figure B.2). Though the rover is capable of detecting these threats, the task of recognizing and classifying

المنارة للاستشارات
www.manaraa.com

the threats falls to the user, as it often does in real-world scenarios as described by researchers at WPAFB. For this task, we chose to implement a "pie menu" interface for classification (Figure B.7).

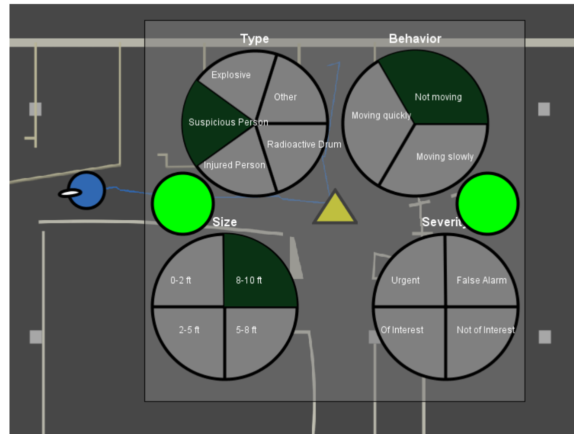

Figure B.7 Classification pie menus. Threats were classified by type, behavior, size, and severity, all of which were described to participants in a training video.

There are four categories by which we are asking users to classify threats, and they are:

- Type (Explosive, Suspicious Person, Injured Person, Radioactive Drum, Other)
- Behavior (Not moving, Moving slowly, Moving quickly)
- Size (0-2 ft, 2-5 ft, 5-8 ft, 8-10 ft)
- Severity (Urgent, Of Interest, Not of Interest, False Alarm).

The threats the users were asked to classify are shown in Figure B.8, and are as follows. The person in Figure B.8 was modified to either have a red or blue shirt, or was lying horizontally on the ground to show injury.

- Explosive Device — Type: Explosive, Behavior: Not moving, Size: 0-2 ft, Severity: Urgent

www.manaraa.com

- Person wearing RED — Type: Suspicious Person, Behavior: Not moving, Size: 5-8 ft, Severity: Of Interest
- Person wearing BLUE — Type: Suspicious Person, Behavior: Not moving, Size: 5-8 ft, Severity: Not Of Interest)
- Injured Person — Type: Injured Person, Behavior: Not moving, Size: 5-8 ft, Severity: Urgent
- Radioactive Canister — Type: Radioactive Drum, Behavior: Not moving, Size: 2-5 ft, Severity: Urgent
- Table / Chair — Type: Other, Behavior: Not moving, Size 2-5 ft, Severity: False Alarm

In order to bring up the classification menu, the user must click or tap on the red threat triangle of a particular threat (Figure B.2). When the classification menu appears, the user must select one element from each of the four categories and tap or click both of the circular buttons on either side of the menu to confirm the classification (Figure B.7).

Figure B.8 Threats displayed in Wayfinder.

المنارة للاستشارات

www.manaraa.com

B.6 Experiment 1

The question we are addressing with this research is whether a multitouch interface is more or less effective than a mouse-based interface for interaction in a supervisory control setting. Using the Wayfinder application, we have designed several tasks and measures by which we will evaluate the answer to this research question (see "Tasks/Performance Metrics," below). We propose that the multitouch interface may offer unique advantages over a similar mouse-based interface, and may also have unique limitations. This hypothesis is partially based on prior research involving the evaluation of multitouch interfaces which has demonstrated that that multitouch interfaces can often be better for complex manipulation tasks, but worse for precise tasks (46).

B.6.1 Method

The study was conducted using a within-participants design with 27 participants, where each participant was asked to use both the mouse and the multitouch interface to accomplish the tasks set forth by the experimenters.

Participants were trained with the interface as described in the "Training" section below, after which they completed two 8-minute missions, one with each interface. To mitigate the learning effect of a within-participants design, we alternated the order in which participants used the two interfaces.

After completing the first mission, the operator was given time to practice with the other interface and completed another 8-minute mission with the second set of threats and waypoints.

After completing both missions, the first 16 participants were asked to complete the second experiment described below. Then, participants were asked to fill out a short written survey and were dismissed.

The participants were all college students participating in the study in order to obtain

المنارة للاستشارات
www.manaraa.com

class credit for their psychology classes, but none of the participants were acquainted with the experimenters. The participants were varied in gender, age and relative experience with multitouch technology. 11 participants were male, 18 female, ranging in age between 18 and 24 years old. Participants were asked to rate their experience with multitouch technology (including the iPhone) on a 5-point Likert scale, and their responses are given in Figure B.9.

We observed that most participants had some experience with multitouch technology, and three owned a device with multitouch functionality.

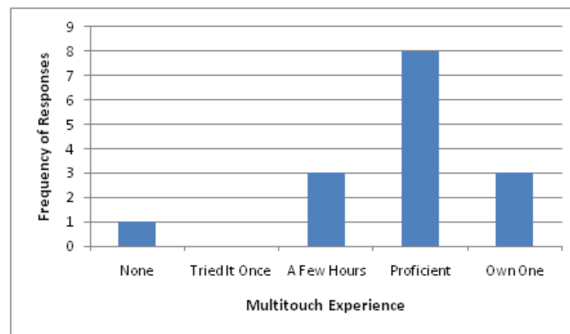

Figure B.9 Multitouch experience among participants.

B.6.2 Performance metrics in the simulated mission

In order to evaluate the effectiveness of a multitouch interface, we have designed several tasks which are based on real-world scenarios described by fellow researchers at WPAFB. The tasks were encapsulated in an 8-minute mission, and each participant completed two missions, one for the mouse interface, and one for the multitouch interface.

Participants were trained using both interfaces, as described below in the "training" section. For this experiment, participants used the HP TouchSmart for both multitouch interaction and as a monitor for mouse interaction. This allowed us to control for the size, brightness, and position of the display.

المنارة للاستشارات
www.manaraa.com

Tasks were presented to the user automatically by the Wayfinder application. The application would display text at the top of the screen instructing the users what to do, as in Figure B.10. When the application displayed a task, participants were instructed to complete the task as quickly and accurately as they could.

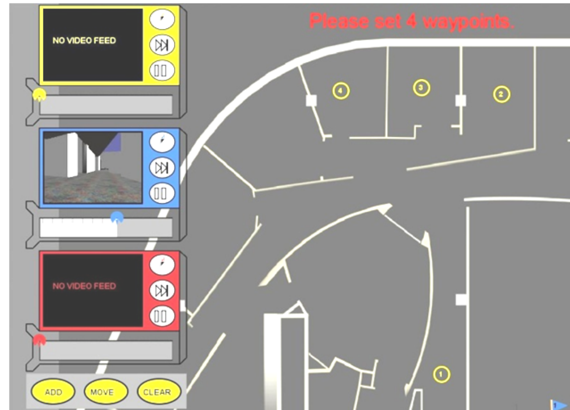

Figure B.10 Wayfinder instructing a participant to place waypoints. Note the small circular waypoint targets with the numbers inscribed.

B.6.2.1 Time taken to set waypoints

At predetermined times throughout the mission, the application would ask the operator to set four waypoints for a rover. We observed the time that it took the user to set all four waypoints, from the time the text was displayed until the participant finished the task. To show the users where to place each waypoint, Wayfinder displayed small circular targets with numbers inscribed to communicate the intended order of the waypoints (Figure B.10).

B.6.2.2 Time taken to classify threats

At predetermined times throughout the mission, the application would ask the operator to classify a particular threat displayed on the map. The operator would have to then use the video control panels and the map to review older video, and use the

المنارة للاستشارات
www.manaraa.com

classification feature to classify the threat they were assigned. We measured the time that it took the operator to classify and confirm the classification.

B.6.2.3 Situation awareness

Additionally, we will also measure whether a multitouch interface has an effect on the operators Level 1 situation awareness, and will use the "freeze technique" as described by M. Endsley in (19). Our implementation of this technique involves blanking the screen at random times during the experiment and asking participants questions about their environment to test their level of situation awareness.

The authors are aware that evaluation of Level 1 SA has some limitations, and that higher levels of situational awareness are also crucial in supervisory control environments.

However, our measure of situational awareness is this research is strictly introductory and will serve as a jumping-off point for future research. We will discuss the primary limitations of Endsleys technique and discuss our motivations for using it in greater detail in the "Limitations" section.

In this experiment, we evaluated Level 1 situational awareness as follows: three times during the mission, we blanked the screen as described by Endsley in (19), displayed the entire map of the building, and asking the operator to estimate the position of each rover on the map (See Figure B.11). The operator dragged three icons, one representing each rover, to his/her best estimate of each rovers position immediately before the screen was blanked. We measured the average distance between the users perceived position of each rover and the rovers actual position and reported it as a measure of Level 1 SA.

B.6.3 Training

During the design of this experiment, we were particularly concerned with the amount and type of training users would receive. We assumed that our participants would have varying degrees of experience with multitouch technology, which would potentially give

المنارة للاستشارات
www.manaraa.com

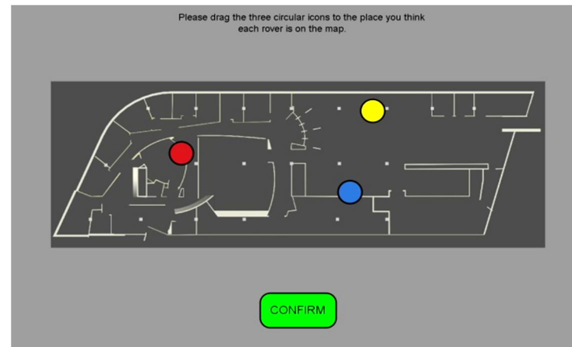

Figure B.11 Situational awareness prompt.

some participants a relative advantage when using the multitouch interface. We trained the participants such that this effect was mitigated, and at the same time, we ensured that users would receive a consistent training experience. Finally, we gave them all enough information and experience to accomplish the goal in an effective and efficient manner.

To help us accomplish this goal, we created a 6-minute training video for participants to watch, which helped us ensure a consistent training experience.

The video trained the users in the different features of the application by demonstrating how to use a particular feature. Each feature was shown using both the mouse and the multitouch interface, so that participants could observe the appropriate behavior to trigger the action they intended. Furthermore, the video also instructed the participants in the manner in which they should classify the threats that appeared in the map, as described in the "Classification" section above. The video also showed participants images of the threats they would be asked to identify, as shown in Figure B.8.

While the training video demonstrated the particular interaction techniques that would be necessary to interact with Wayfinder, it is difficult to tell whether the participant was paying full attention, whether they understood all aspects of the video, and whether they would be able to successfully apply the knowledge they have gained. To help mitigate these effects, we also allowed participants to ask questions immediately

المنارة للاستشارات
www.manaraa.com

following the training video and answered these questions as completely as possible.

After training was completed, the operator was allowed to practice using the first interface that was assigned to them, either mouse or multitouch. To minimize the limitations mentioned above, the operator was "trained to criterion," meaning that they practiced using the interface and performing tasks until the experimenter could verify that they were capable of using the interface effectively to accomplish tasks without assistance.

A common problem that we addressed during training was the relative lack of experience with a multitouch interface when compared to experience with a mouse interface. Participants unanimously have more experience using a mouse than they do using a multitouch screen, specifically the multitouch devices we employed.

Due to this difference in experience, participants generally received longer instruction/practice time with the multitouch interface than they did with the mouse interface. As such, the practice period for multitouch training lasted as little as four minutes or as long as ten minutes in some cases, whereas the mouse training generally lasted between two and five minutes.

B.6.4 Results

Results show that the multitouch interface performs comparably to the mouse interface in classifying threats and in levels of SA obtained when using the interface.

For assessing Level 1 situation awareness, we measured the average distance between each participants estimate of the location of each rover and the actual location of each rover. Results are reported units of the map width, where 1 unit is approximately equal to the width of the map. This was done because we did not have accurate measures of absolute distance. The average difference between estimated and actual positions for those using the mouse interface was 0.114 units, with a standard deviation of 0.051 units. The average difference between estimated and actual positions for those using

المنارة للاستشارات

www.manaraa.com

the multitouch interface was 0.130 units, with a standard deviation of 0.054 units. Analyzing these results with a paired-samples t-test yielded P=0.2067, so we are unable to claim that there was a difference between the two interaction styles, however note that multitouch performed similarly to the mouse-based interface for this task.

For classifying threats, we also observed similar results for both the multitouch and mouse-based interfaces. We observed the average time it took for a user to complete a classification task. When using the mouse interface, the average time to complete a classification task was 24.771 seconds, with a standard deviation of 10.325 seconds. When using the multitouch interface, the average time was 24.933 seconds, with a standard deviation of 10.519 seconds. It is interesting to note here that the standard deviation of scores for this task was relatively high when compared with the mean score for this task, implying that there was a great deal of variability between participants for this task.

For setting waypoints, we observed that the mouse interface performed better than the multitouch interface. We observed a mean task completion time of 13.877 seconds for the mouse interface, with a standard deviation of 5.342 seconds. For the multitouch interface, the mean completion time was 19.887 seconds (6.01 seconds slower than the mouse interface) with a standard deviation of 7.583 seconds. These results are illustrated in Figure B.12. Analyzing this data using a paired-samples t-test yielded P=0.0017, and we can conclude that for setting waypoints, the mouse interface performed better than the multitouch interface.

We observed that many participants struggled when using the touchscreen interface to set waypoints. Unfortunately, the HP Touchsmart produced sensing inaccuracies when using multiple fingers to set waypoints, and users generally found it difficult to overcome these sensing inaccuracies when performing precise actions such as setting waypoints. We believe that other, more precise multitouch hardware would perform relatively better than these results show, and is included for future investigation.

المنارة للاستشارات
www.manaraa.com

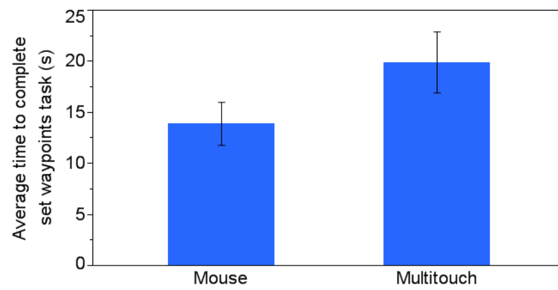

Figure B.12 Results of waypoint task. Users were able to set waypoints an average of 6.01 seconds faster using the mouse.

B.7 Experiment 2 - map manipulation

In addition to our first experiment, we also measured the ability of a user to manipulate the map to view a specific area of the map. To measure this, we asked the user to drag, scale, and rotate a black rectangle such that it filled the screen (see Figure B.13). Orientation was indicated by a red arrow, and participants were instructed that this red arrow should point "up when they were finished. This part of the experiment was conducted independently with the first 16 participants from experiment 1, as described above.

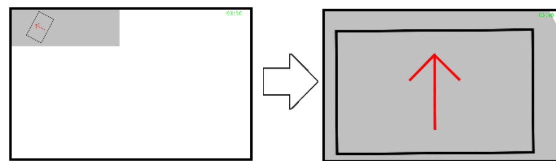

Figure B.13 Map manipulation task. The participants manipulated the small black rectangle so that it filled the screen with the arrow pointing up.

Of these 16 participants, 5 were male, 11 were female, and were in the same age range and experience as in Experiment 1.

With the mouse interface, participants could move the rectangle by pressing the left mouse button and dragging, scale by using the mouse wheel, and rotate by right-clicking

المنارة للاستشارات
www.manaraa.com

and dragging the mouse right to left. With the multitouch screen, participants could manipulate the map by dragging, stretching, pinching and rotating with 2 fingers.

For this task, we used the Stantum SMK 15.4 multitouch device. Participants used both a mouse and multitouch interface for this task, and training was performed in the same manner as for the missions. To analyze the effects, we measured the number of the described manipulation tasks the participant could complete in a two-minute time period.

B.7.1 Results

Results of this experiment show that the use of a multitouch interface allows a user to better manipulate the map to show a region of interest (Figure B.14). Data were analyzed with a paired-sample t-test. On average, participants completed 6.6 more manipulation tasks with the multitouch interface than they did with the mouse interface in the two-minute time period (p ¡ 0.0001). Error bars in Figures 12 and 13 represent a 95 percent confidence interval for the mean.

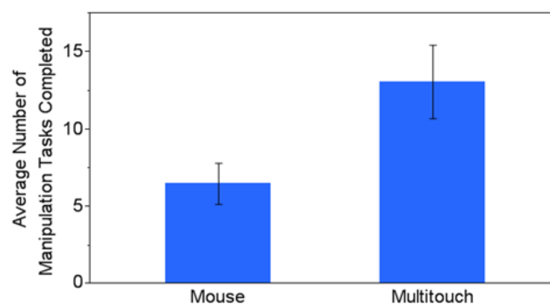

Figure B.14 Results of the map manipulation task. Participants completed 6.6 more manipulations with the multitouch interface.

Finally, participants were asked to rate their preferences for each interface on a continuous scale from 0-100, where 0 was "preferred mouse" and 100 was "preferred multitouch."

المنارة للاستشارات
www.manaraa.com

Results show that participants slightly preferred the multitouch interface for manipulating the map with an average response of 77.5, organizing information (62.4), and classifying threats (69.0) (p ¡ 0.01). However, participants preferred the mouse interface for setting waypoints with a response of 36.4 (p ¡ 0.01).

B.8 Limitations

The authors would like to express an acknowledgement of some limitations of this research, primarily the decision to use two different hardware devices and the choice to evaluate only Level 1 situational awareness.

B.8.1 Hardware

Initially, we did not intend to use more than a single input device in order to maintain consistency; however, we were unable to find a commercial input device which satisfied both of our requirements, which were:

- Must be large enough to display detail and allow the user a broad view of the environment.
- Must have accurate sensing capabilities, and preferably the ability to sense multiple fingers reliably.

We decided to purchase and use a 25.5" HP TouchSmart device for this experiment, because it offered the screen real-estate necessary. However, the device did not satisfy our second requirement as well as we thought, and presented significant sensing issues (wherein the device cannot distinguish between multiple possible finger positions). This made it difficult if not impossible to perform a 2-finger rotate gesture, which we required for evaluating the ability of the user to manipulate the map.

المنارة للاستشارات
www.manaraa.com

Therefore, we used a second device in addition to the HP TouchSmart, one that produced greater input precision. We chose to use a 15.4" Stantum SMK multi-touch device. While the Stantum device is significantly smaller than the HP TouchSmart, it offered us much greater precision. The use of two devices required us to conduct and analyze two experiments, while our preference would have been to integrate them into a single experiment. However, the experiments were run and analyzed independently, and the results are still valid within each experiment.

B.8.2 Situational awareness

Our evaluation of Level 1 situational awareness has some limitations; it simply evaluates a participants perception of the details and elements of the environment, and does not evaluate his or her comprehension or understanding of these elements. Our decision to evaluate Level 1 SA was based on the introductory nature of this research, especially as it explores a new application for multitouch interfaces. This research is intended to serve as a jumping-off point for further investigation in the application of multitouch interfaces in supervisory control settings. We acknowledge that further work is needed to evaluate whether multitouch interfaces have an effect on higher levels of SA, and that this evaluation is needed if multitouch interfaces are to become more widely accepted in supervisory control environments.

B.9 Discussion

Results show that a multitouch interface can be an effective interface for manipulating a map of a building to view different parts of the building. Multitouch interaction allows users to perform three operations (zoom, drag, rotate) in a single motion, and the results show a conclusive advantage for multitouch over mouse interaction.

We also found that a multitouch interface performs similarly to a mouse-based inter-

المنارة للاستشارات
www.manaraa.com

face for classifying threats and maintaining situation awareness in supervisory control interfaces. As a result, developers of supervisory control interfaces should not be concerned of a loss of Level 1 situation awareness by moving to a new, perhaps less familiar, multitouch interface.

We found that the mouse interface performed better for setting waypoints for rovers than the multitouch interface. However, users were frustrated by known hardware imprecision with the HP TouchSmart when using the multitouch interface. We found that users spent a great deal of time having to reset waypoints that they had already set because the touchscreen was simply not precise enough.

Although we suspected that users would have more difficulty with precise tasks on the multitouch screen, we believe that with a more precise touchscreen device, some of these difficulties could be mitigated.

B.10 Conclusions and future work

We have shown that multitouch can be used as an effective interface in a supervisory control environment, and have shown its advantages and potential disadvantages over a mouse-based input device. We also expect that touchscreen hardware improvements could lead to more consistent advantages for the multitouch input device.

Future work will involve evaluating a multitouch interface for longer missions to evaluate strain on users, as the 8-minute missions described in this research were not long enough to evaluate user strain and fatigue. These issues may have a significant effect on the feasibility of implementing a multitouch interface for mission-critical supervisory control interfaces.

Finally, developers of these interfaces will need to implement new and effective interface designs that are customized for a multitouch interface. Multitouch gestures could provide additional features that extend the basic functionality of the Wayfinder inter-

المنارة للاستشارات
www.manaraa.com

face, and make multitouch interaction a realistic interface for current supervisory control interfaces.

B.11 Acknowledgments

We especially thank those involved in the development of the Wayfinder application, namely Tony Milosch and Mike Oren, fellow researchers at Wright Patterson Air Force Base for providing perspective and guidance, and those who participated in the research study. This research was conducted with support from the AFOSR.

المنارة للاستشارات
www.manaraa.com

APPENDIX C. FORMS

This appendix contains forms and documents which were used in the user study.

C.1 Informed Consent

This section contains a copy of the informed consent document which was given to participants in the developer study prior to their participation. Please refer to Figures C.1 and C.2.

C.2 Interview Protocol

The interview protocol was developed and submitted before the name of the software framework was changed to AQUA-G. Therefore, all questions reference the system called "OmniGest." During the interview, the word Omnigest was replaced by AQUA-G. The interview protocol is shown in figures C.3 and C.4.

www.manaraa.com

ISU IRB # 1 08-279
Approved Date: 31 March 2010
Expiration Date: 22 July 2010

Informed Consent Document

Title of Study: Study of Multi-Touch Interfaces

Investigators: Stephen Gilbert, Ph.D Mike Oren, M.S Jay Roltgen

This is a research study. Please take your time in deciding if you would like to participate. Please feel free to ask questions at any time.

INTRODUCTION
The purpose of this study is to evaluate the difficulty of software development tasks which will enhance or expand an existing software framework. In order to participate in this study, you must have some software development experience and be at least 18 years old.

DESCRIPTION OF PROCEDURES
The study involves development of a component of a software system and a formal interview following the completion of said component. You will be given a development task and documentation for the existing software framework. Instructions will be given for the task you must complete.

You will complete the development task on your own time and are allowed to ask questions at any time during the development process via email or in person. Your questions will be answered as completely as possible. This is expected to take no more than twenty hours.

Upon completion of your development task, you will be asked to participate in a formal interview, which will be conducted in person. The interview questions will revolve around your experience while developing your component, the difficulties you encountered, and your opinions and/or advice for improving the software system.

RISKS
There are no known participation risks of this study other than those associated with normal computer usage.

PARTICIPANT RIGHTS
Your participation in this study is completely voluntary and you may refuse to participate. If you agree to participate, you have the right to leave the study at any time without penalty or loss of benefits.

CONFIDENTIALITY
Records identifying participants will be kept confidential to the extent permitted by applicable laws and regulations and will not be made publicly available. We are required by Federal and University policy to keep a copy of the informed consent for three years after the close of the

Figure C.1 Informed Consent Document, Page 1.

المنارة للاستشارات
www.manaraa.com

ISU IRB # 1 08-279
Approved Date: 31 March 2010
Expiration Date: 22 July 2010

study. However, no records are kept that allow us to associate your name, which is on your informed consent form, with your behavior. This signed document will be kept in a locked laboratory. Your body and voice will be recorded via video camera which will be destroyed on 10 May 2011.

The data collected in this research may be used for educational or scientific purpose and may be presented at scientific meetings or published in professional journals. No participant will be identified in any use or publication of the data.

QUESTIONS OR PROBLEMS

You are encouraged to ask questions at any time during the study.

- For questions, please first contact Jay Roltgen via email at jroltgen@iastate.edu. You may also request in-person assistance via this email.
- If you have questions that do not directly relate to your development task, please contact Stephen Gilbert, Ph.D., at gilbert@iastate.edu.
- If you have any questions about the rights of research subjects or research-related injury, please contact the IRB Administrator, (515) 294-4566, IRB@iastate.edu, or Director, (515) 294-3115, Office for Responsible Research, Iowa State University, Ames, Iowa 50011.

SUBJECT SIGNATURE

Your signature indicates that you voluntarily agree to participate in this study, that the study has been explained to you, that you have been given the time to read the document and that your questions have been satisfactorily answered. You will receive a copy of the signed and dated written informed consent prior to your participation in the study.

____

Subject's name (printed)

______________________________ ____________

Subject's signature *Date*

INVESTIGATOR STATEMENT

I certify that the participant has been given adequate time to read and learn about the study and all of their questions have been answered. It is my opinion that the participant understands the purpose, risks, benefits and the procedures that will be followed in this study and has voluntarily agreed to participate.

______________________________ ____________

Signature of person obtaining informed consent *Date*

Figure C.2 Informed Consent Document, Page 2.

المنارة للاستشارات
www.manaraa.com

OmniGest: **The Universal Gesture Recognition Framework**
PI: Stephen Gilbert, Ph.D.
Interviewer: Jay Roltgen

Interview Protocol: Software Developers

Initial Reactions

1. What is your initial reaction to this system?
2. What do you think are the biggest advantages of a system like this?
    a. Why is this an advantage?
3. What do you think are the primary limitations?
    a. Are these limitations important?
    b. How do you think these limitations can best be overcome?
4. If you were deciding to use this system in one of your applications, what criteria would you use to evaluate it and decide to use it or not?
    a. What do you think would be the single biggest factor in this decision?
    b. Do you think that it is better for the system to have good documentation or a good interface, and why?
5. What would you change about the system as described?
    a. Would you be willing to contribute to this project if it was open source?
    b. What would convince you to contribute to the project?
6. How important is it to you that this system be cross-platform?
    a. Which is the most important OS that it be able to run on, for you?
    b. What do you think will be the hardest part about keeping this system cross-platform?
        i. How can these difficulties be overcome or mitigated?

Creating a device driver, gesture, or client application
Note: In each blank, substitute the phrase: "device driver," "gesture," "client application," or "event" as necessary for which task this developer completed.

1. How would you rate your experience writing a _____ for OmniGest?
    a. Was it positive or negative and why?
2. What was your favorite part about the experience, and what did you like best about using OmniGest for this purpose?
3. What do you think could be improved about the experience?
4. Was the network protocol and event structure documentation clear to you?
5. Do you think creating a _____ is a difficult task for the average developer?
    a. What, in your opinion, could be done to lower the learning curve?
6. Do you think that using OmniGest makes this process of obtaining gestural input easier?
    a. What is the biggest advantage of this ease of use?
    b. Could anything be improved to increase this ease of use?
7. How did you like the method of creating new events?
    a. Was it easy to use?
    b. What would you change about this method if you could?
    c. What did you most like about it?

Figure C.3 Interview Protocol, Page 1.

المنارة للاستشارات
www.manaraa.com

Creating an OmniGest application

1. Do you feel that you had to "tailor" your application for OmniGest by coding in an unnatural way?
   a. How much of a disadvantage is this in terms of time spent learning the system?
   b. Do you think that this tailoring is reasonable for your application, in that the benefits outweigh the drawbacks?
2. If you did this over, would you continue to use OmniGest, or would you create your own solution?
   a. What could be changed about OmniGest to convince you to use it?
      i. Is the developer interface a significant obstacle?
      ii. Is the lack of documentation a significant obstacle?
   b. What did you like most about creating your application to use OmniGest?

The future

1. Would you be willing to share the device driver, gesture, or application that you wrote for OmniGest as open source and part of the OmniGest project?
   a. Why or why not?
   b. What can the OmniGest developers do to make sharing your work as easy as possible with the community at large?
2. Do you think OmniGest is a valuable addition to your application?
3. Finally, what do you think other developers will like best about OmniGest?
4. What do you like the other developers will like least about OmniGest?
5. What do you think can be done to convince other developers to use OmniGest?
6. Do you have any other comments?

Figure C.4 Interview Protocol, Page 2.

المنارة للاستشارات
www.manaraa.com

BIBLIOGRAPHY

[1] Appert, C., and Zhai, S. (2009). Using strokes as command shortcuts: cognitive benefits and toolkit support. In *Proceedings of the 27th international conference on Human factors in computing systems* (pp. 2289-2298). Boston, MA, USA: ACM. doi:10.1145/1518701.1519052

[2] Apple, Inc. (2010). *iPad.* Retrieved 2 June, 2010 from http://www.apple.com/ipad/

[3] Apple, Inc. (2010). *iPhone.* Retrieved June 2, 2010, from http://www.apple.com/iphone

[4] Apple, Inc. (2010). *iPhone Development Guide.* Retrieved from http://developer.apple.com/iphone/library/documentation/Xcode/Conceptual/iphone_development/000-Introduction/introduction.html

[5] Bailador, G., Roggen, D., Tröster, G., and Triviño, G. (2007). Real time gesture recognition using continuous time recurrent neural networks. In *Proceedings of the ICST 2nd international conference on Body area networks* (pp. 1-8). Florence, Italy: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). Retrieved from http://portal.acm.org/citation.cfm?id=1460247

المنارة للاستشارات
www.manaraa.com

[6] Bau, O., and Mackay, W. E. (2008). OctoPocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on User interface software and technology* (pp. 37-46). Monterey, CA, USA: ACM. doi:10.1145/1449715.1449724

[7] Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., and Cruz-Neira, C. (2008). VR juggler: a virtual platform for virtual reality application development. In *ACM SIGGRAPH ASIA 2008 courses* (pp. 1-8). Singapore: ACM. doi:10.1145/1508044.1508086

[8] Bonansea, L. (2009). 3D Hand gesture recognition using a ZCam and an SVM-SMO classifier. Ames, IA, USA: Iowa State University. Retrieved from http://gradworks.umi.com/14/68/1468148.html

[9] Bragdon, A., Zeleznik, R., Williamson, B., Miller, T., and LaViola, J. (2009). GestureBar: improving the approachability of gesture-based interfaces. In *Proceedings of the 27th international conference on Human factors in computing systems* (pp. 2269-2278). Boston, MA, USA: ACM. doi:10.1145/1518701.1519050

[10] Burns, D., and Osfield, R. (2004). Open Scene Graph A: Introduction, B: Examples and Applications. In *Proceedings of the IEEE Virtual Reality 2004* (p. 265). IEEE Computer Society. doi:10.1109/VR.2004.57

[11] Buxton, B. (2007). Multi-touch systems that I have known and loved. Retrieved June 2, 2010, from http://www.billbuxton.com/multitouchOverview.html

[12] Buxton, W., Fiume, E., Hill, R., Lee, A., and Woo, C. (1983). Continuous hand-gesture driven input. In *Proceedings of Graphics interface* (Vol. 83, pp. 191195).

المنارة للاستشارات
www.manaraa.com

[13] Chen, Y. T., and Tseng, K. T. (2007). Multiple-angle hand gesture recognition by fusing svm classifiers. In *IEEE International Conference on Automation Science and Engineering, 2007.* CASE 2007 (pp. 527530). Scottsdale, AZ, USA. doi:10.1109/COASE.2007.4341729

[14] Cheng, K., Itzstein, B., Sztajer, P., and Rittenbruch, M. (2009). A unified multi-touch and multi-pointer software architecture for supporting collocated work on the desktop (Technical Report No. ATP-2247). NICTA. Retrieved from `http://www.cs.usyd.edu.au/~kcheng/2247_A_unified_2.pdf`

[15] Dell. (2010). Dell Latitude XT2 Tablet PC Touch Screen Laptop Details. Retrieved June 2, 2010, from `http://www.dell.com/tablet?s=biz&cs=555`

[16] Dietz, P., and Leigh, D. (2001). DiamondTouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology* (pp. 219-226). Orlando, Florida: ACM. doi:10.1145/502348.502389

[17] Dohse, K. C., Dohse, T., Still, J. D., and Parkhurst, D. J. (2008). Enhancing multi-user interaction with multi-touch tabletop displays using hand tracking. In *Advances in Computer-Human Interaction, 2008 First International Conference on* (pp. 297-302). doi:10.1109/ACHI.2008.11

[18] Echtler, F., and Klinker, G. (2008). A multitouch software architecture. In *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges* (pp. 463-466). Lund, Sweden: ACM. doi:10.1145/1463160.1463220

[19] Endsley, M. R. (1995). Measurement of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society,* 37, 65-84. doi:10.1518/001872095779049499

المنارة للاستشارات

www.manaraa.com

[20] Fong, T., and Thorpe, C. (2001). Vehicle Teleoperation Interfaces. *Autonomous Robots,* 11(1), 9-18. doi:10.1023/A:1011295826834

[21] Forlines, C., Wigdor, D., Shen, C., and Balakrishnan, R. (2007). Direct-touch vs. mouse input for tabletop displays. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 647-656). San Jose, California, USA: ACM. doi:10.1145/1240624.1240726

[22] Gamma, E. (1995). *Design patterns.* Addison-Wesley.

[23] Gardner, M. (1970). Mathematical games: The fantastic combinations of John Conway's new solitaire game 'Life'. *Scientific American,* 223(4), 120123.

[24] Han, J. Y. (2005). Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th annual ACM symposium on User interface software and technology* (pp. 115-118). Seattle, WA, USA: ACM. doi:10.1145/1095034.1095054

[25] Hartmann, B., Morris, M. R., Benko, H., and Wilson, A. D. (2009). Augmenting interactive tables with mice and keyboards. In Proceedings of the 22nd annual ACM symposium on User interface software and technology (pp. 149-152). Victoria, BC, Canada: ACM. doi:10.1145/1622176.1622204

[26] Hewlett-Packard Development Company, L.P. (2010). *HP TouchSmart.* Retrieved June 2, 2010, from `http://www.hp.com/united-states/campaigns/touchsmart/`

[27] Iddan, G. J., and Yahav, G. (2001). Three-dimensional imaging in the studio and elsewhere. In *Three-Dimensional Image Capture and Applications IV* (Vol. 4298, pp. 48-55). San Jose, CA, USA: SPIE. doi:10.1117/12.424913

المنارة للاستشارات
www.manaraa.com

[28] Ishii, H., and Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 234-241). Atlanta, GA, USA: ACM. doi:10.1145/258549.258715

[29] Izadi, S., Agarwal, A., Criminisi, A., Winn, J., Blake, A., and Fitzgibbon, A. (2007). C-Slate: A Multi-Touch and Object Recognition System for Remote Collaboration using Horizontal Surfaces. *In Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07* (pp. 3-10). Newport, RI, USA. doi:10.1109/TABLETOP.2007.34

[30] Jacob, R. J., Girouard, A., Hirshfield, L. M., Horn, M. S., Shaer, O., Solovey, E. T., and Zigelbaum, J. (2008). Reality-based interaction: a framework for post-WIMP interfaces. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (pp. 201-210). Florence, Italy: ACM. doi:10.1145/1357054.1357089

[31] Kato, J., Sakamoto, D., Inami, M., and Igarashi, T. (2009). Multi-touch interface for controlling multiple mobile robots. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems* (pp. 3443-3448). Boston, MA, USA: ACM. doi:10.1145/1520340.1520500

[32] Kiriaty, Y. (2009). Multitouch capabilities in Windows 7. *MSDN Magazine,* 2009(August). Retrieved from `http://msdn.microsoft.com/en-us/magazine/ee336016.aspx`

[33] Kitware, Inc. (2009). *CMake.* Retrieved June 2, 2010, from `http://www.cmake.org/`

المنارة للاستشارات
www.manaraa.com

[34] Klemmer, S. R., Li, J., Lin, J., and Landay, J. A. (2004). Papier-Mache: toolkit support for tangible input. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 399-406). Vienna, Austria: ACM. doi:10.1145/985692.985743

[35] Knight, S. (2002). SCons design and implementation. Presented at the Tenth International Python Conference. Retrieved from http://www.python.org/workshops/2002-02/papers/16/index.htm

[36] Krueger, M. W., Gionfriddo, T., and Hinrichsen, K. (1985). VIDEOPLACE – an artificial reality. *SIGCHI Bull.,* 16(4), 35-40. doi:10.1145/1165385.317463

[37] Lee, J. C. (2008). Hacking the Nintendo Wii Remote. *IEEE Pervasive Computing*, 7(3), 39-45. doi:10.1109/MPRV.2008.53

[38] Liu, Y., Gan, Z., and Sun, Y. (2008). Static Hand Gesture Recognition and its Application based on Support Vector Machines. In Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, ACIS International Conference on (Vol. 0, pp. 517-521). Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/SNPD.2008.144

[39] Massie, T. H., and Salisbury, J. K. (1994). The phantom haptic interface: A device for probing virtual objects. In *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems* (Vol. 55, pp. 295300). Retrieved from http://www.sensable.com/documents/documents/ASME94.pdf

المنارة للاستشارات

www.manaraa.com

[40] Matejka, J., Grossman, T., Lo, J., and Fitzmaurice, G. (2009). The design and evaluation of multi-finger mouse emulation techniques. In *Proceedings of the 27th international conference on Human factors in computing systems* (pp. 1073-1082). Boston, MA, USA: ACM. doi:10.1145/1518701.1518865

[41] Micire, M., Drury, J. L., Keyes, B., and Yanco, H. A. (2009). Multi-touch interaction for robot control. In *Proceedings of the 13th international conference on Intelligent user interfaces* (pp. 425-428). Sanibel Island, Florida, USA: ACM. doi:10.1145/1502650.1502712

[42] Microsoft. (2010). *Microsoft Surface.* Retrieved June 2, 2010, from http://www.microsoft.com/surface/en/us/default.aspx

[43] Microsoft. *What are pen flicks?* Retrieved July 2, 1010, from http://windows.microsoft.com/en-US/windows-vista/What-are-pen-flicks.

[44] *Mouse Gestures Redox :: Home.* Retrieved July 2, 2010, from http://www.mousegestures.org/.

[45] *MT4j - Multitouch For Java.* (n.d.). Retrieved November 18, 2009, from http://www.mt4j.org/mediawiki/index.php/Main_Page

[46] Muller, L. Y. L. (2008). *Multi-touch displays: design, applications and performance evaluation.* Universiteit van Amsterdam. Retrieved from http://www.science.uva.nl/research/scs/papers/archive/Muller2008a.pdf

[47] MultiTouch Ltd. (2009). *MultiTouch Cell.* Retrieved June 2, 2010, from http://multitouch.fi/products/cell/

المنارة للاستشارات
www.manaraa.com

[48] Murakami, K., and Taguchi, H. (1991). Gesture recognition using recurrent neural networks. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology* (pp. 237-242). New Orleans, Louisiana, United States: ACM. doi:10.1145/108844.108900

[49] NextWindow. (n.d.). *NextWindow - Home.* Retrieved June 2, 2010, from http://www.nextwindow.com/

[50] Nguyen, L., Bualat, M., Edwards, L., Flueckiger, L., Neveu, C., Schwehr, K., Wagner, M., et al. (2001). Virtual realityiInterfaces forvVisualization and control of remote vehicles. *Autonomous Robots,* 11(1), 59-68. doi:10.1023/A:1011208212722

[51] Nicholas, D., Huntington, P., and Williams, P. (2001). Establishing metrics for the evaluation of touch screen kiosks. *Journal of Information Science,* 27(2), 61-71. doi:10.1177/016555150102700201

[52] Nintendo. (2010). *Wii controllers.* Retrieved June 2, 2010, from http://www.nintendo.com/wii/console/controllers

[53] N-trig. (n.d.). *About us.* Retrieved June 2, 2010, from http://www.n-trig.com/Content.aspx?Page=AboutUs

[54] NUI Group Community. (n.d.). *Community Core Vision.* Retrieved June 2, 2010, from http://ccv.nuigroup.com/

[55] Peek, B. (2009). *Managed Library for Nintendo's Wiimote.* Retrieved June 2, 2010, from http://wiimotelib.codeplex.com/

المنارة للاستشارات
www.manaraa.com

[56] Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. (2000). The Cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking* (pp. 32-43). Boston, MA, USA: ACM. doi:10.1145/345910.345917

[57] *PyMT : Open source library for multitouch development.* (n.d.). Retrieved June 2, 2010, from http://pymt.txzone.net/

[58] Ramanahally, P., Gilbert, S., Niedzielski, T., Velazquez, D., and Anagnost, C. (2009). Sparsh UI: A Multi-Touch Framework for Collaboration and Modular Gesture Recognition. In *ASME-AFM 2009 World Conference on Innovative Virtual Reality* (pp. 137-142). doi:10.1115/WINVR2009-740

[59] Ramanahally, P. (2010). Cricket based user identification for multi-touch table. Ames, IA: Iowa State University. *In Press.*

[60] Roltgen, J. (2010). *Aqua-gesture-framework.* Retrieved June 2, 2010, from http://code.google.com/p/aqua-gesture-framework/

[61] Roltgen, J., and Gilbert, S. (2010). Wayfinder: Evaluating Multitouch Interaction in Supervisory Control of Unmanned Vehicles. In *Proceedings of the ASME 2010 World Conference on Innovative Virtual Reality.*

[62] Rowe, A., Kristen, L., and Davis, J. (2009). Vigilant Spirit Control Station: A Research Testbed for Multi-UAS Supervisory Control Interfaces. in *Proceedings of the International Symposium of Aviation Psychology.* Retrieved from http://www.vrac.iastate.edu/~jroltgen/VS.pdf

www.manaraa.com

[63] Squire, P., Trafton, G., and Parasuraman, R. (2006). Human control of multiple unmanned vehicles: effects of interface type on execution and task switching times. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction* (pp. 26-32). Salt Lake City, Utah, USA: ACM. doi:10.1145/1121241.1121248

[64] Saffer, D. (2008). Designing Gestural Interfaces: Touchscreens and Interactive Devices (1st ed.). O'Reilly Media.

[65] Schöning, J., Brandl, P., Daiber, F., Echtler, F., Hilliges, O., Hook, J., Löchtefeld, M., et al. (2008). *Multi-Touch Surfaces: A Technical Guide* (No. TUM-10833). University of Münster. Retrieved from `http://ifgi.uni-muenster.de/~j_scho09/pubs/bymultitouch.pdf`

[66] Starner, T., and Pentland, A. (1995). Real-time American Sign Language recognition from video using hidden Markov models. In *Proceedings of the International Symposium on Computer Vision* (p. 265). Coral Gables, FL, USA: IEEE Computer Society. doi:10.1109/ISCV.1995.477012

[67] Szymanski, R., Goldin, M., Palmer, N., Beckinger, R., Gilday, J., and Chase, T. (2008). Command and Control in a Multitouch Environment. Presented at the Army Science Conference. Retrieved from `http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA503423`

[68] Taylor, R. M., Hudson, T. C., Seeger, A., Weber, H., Juliano, J., and Helser, A. T. (2001). VRPN: a device-independent, network-transparent VR peripheral system. In *Proceedings of the ACM symposium on Virtual reality software and technology* (pp. 55-61). Baniff, Alberta, Canada: ACM. doi:10.1145/505008.505019

المنارة للاستشارات

www.manaraa.com

[69] Tse, E., Shen, C., Greenberg, S., and Forlines, C. (2006). Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. In *Proceedings of the working conference on Advanced visual interfaces* (pp. 336-343). Venezia, Italy: ACM. doi:10.1145/1133265.1133336

[70] Tuddenham, P., and Robinson, P. (2007). Distributed tabletops: Supporting remote and mixed-presence tabletop collaboration. In *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07.* (pp. 1926). Newpot, RI. doi:10.1109/TABLETOP.2007.15

[71] *TUIO.* (n.d.). Retrieved June 2, 2010, from `http://www.tuio.org/`

[72] Ullmer, B., and Ishii, H. (1997). The metaDESK: models and prototypes for tangible user interfaces. In *Proceedings of the 10th annual ACM symposium on User interface software and technology* (pp. 223-232). Banff, Alberta, Canada: ACM. doi:10.1145/263407.263551 technology, ACM (1997), 223-232.

[73] Wilson, A., and Bobick, A. (1999). Parametric hidden Markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 21(9), 884-900. doi:10.1109/34.790429

[74] Weiss, M., Wagner, J., Jansen, Y., Jennings, R., Khoshabeh, R., Hollan, J. D., and Borchers, J. (2009). SLAP widgets: bridging the gap between virtual and physical controls on tabletops. In *Proceedings of the 27th international conference on Human factors in computing systems* (pp. 481-490). Boston, MA, USA: ACM. doi:10.1145/1518701.1518779

المنارة للاستشارات
www.manaraa.com

[75] Wobbrock, J. O., Morris, M. R., and Wilson, A. D. (2009). User-defined gestures for surface computing. In *Proceedings of the 27th international conference on Human factors in computing systems* (pp. 1083-1092). Boston, MA, USA: ACM. doi:10.1145/1518701.1518866

[76] Wu, M., and Balakrishnan, R. (2003). Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of the 16th annual ACM symposium on User interface software and technology* (pp. 193-202). Vancouver, Canada: ACM. doi:10.1145/964696.964718

للاستشارات المنارة
www.manaraa.com