Developing Multi-User Applications For iPhone

Designing Engaging Applications for Multiple Users on Next-Generation Mobile Platforms

Nizar Khalife
University of Puerto Rico
khalifenizar@gmail.com

Jasmine Jones
University of Maryland-Baltimore County
jasmin3@umbc.edu

Paul Clay
Reed College
contactclay@gmail.com

Brandon Newendorp
Iowa State University
bnew@iastate.edu

Ken Kopecky
Iowa State University
kennyk@iastate.edu

Dr. Eliot Winer
Iowa State University
ewiner@iastate.edu

Abstract

This project explores Apple iPhone's capacity as an engaging, multi-user application environment. Taking advantage of iPhone's unique operating system, powerful hardware, and various modes of interaction, two multiplayer games for Apple's iPhone were developed, WapIt! and Turret Wars . Next generation mobile devices like iPhone exhibit great potential for diverse new mobile applications, but, until recently, developers have given little consideration to the differences in the way users interact with and perform tasks on mobile and desktop computing devices. As technology advances and mobile computing becomes increasingly popular, focused effort, as advocated by Gorlenko and Merrick [4], must be applied in addressing usability and functionality issues associated with fully mobile wirelessly connected computing devices like iPhone. The games developed in this project demonstrate the process used to develop multi-user applications, including the design of the networking structure, the visual user-interface, and customizable multitouch and accelerometer gestures.

1. Introduction

The purpose of this project was to explore iPhone's capacity as an engaging multi-user application environment. In the past, developers have approached mobile application design as if writing for tiny desktops with novel hardware; little consideration has been given to the differences in the way users interact with mobile and stationary devices. Gorlenko and Merrick advocate focused effort in addressing the usability issues associated with "fully mobile wirelessly connected" (FMWC) computing devices like iPhone in order to better design applications beneficial to users in mobile situations[3]. To this end, this research team presents the design process for two multiplayer iPhone games, *WapIt!* and *Turret Wars*, which address issues related specifically to the user interface design and networking structure.

Figure 1. iPod Touch

The game *WapIt!* was designed as a multiplayer mini-game to be included in a set of other single level games as part of an overarching multi-player environment. Created using Cocoa Touch, the development framework of the iPhone OS, this game uses the basic command and response structure of the popular children's toy BopIt. *WapIt!* asks the player to perform a certain gesture on the multitouch screen or with the accelerometer, a tap (Wap!), swipe, pinch, or shake, then checks to see if the appropriate gesture was entered. If the player follows the command correctly in the given amount of time, then he or she is presented with another gesture, and the cycle repeats faster and faster until the player makes a mistake. The multiplayer version of *WapIt!* is turn-based, and played on multiple iPhones connected to each other over a wireless

Figure 2. Graphics from *WapIt!*

network. On each turn, the game issues a certain number of gesture commands that must be correctly entered by the active player, then "passes" the turn by sending a message to the next player's iPhone. The other players can see what the active player is doing and hear the beat on their iPhones, but cannot enter a gesture until it is their turn. If a player loses, they can still see and hear the game but will remain in inactive mode until the game is over, which occurs when there is only one player left.

Turret Wars was modeled after the classic DOS game, *Dome Wars*, and uses OpenGL ES, an open-source graphics library for embedded systems, to draw the scene. Two turrets, placed on randomly generated terrain, are controlled by opposing players whose goal is to destroy the opposite player's turret. Each turret is aimed and fired using finger gestures on the multitouch screen; an opening pinch motion creates an aiming circle around the turret that can be rotated by sliding the fingers around the arc of the circle. Shots are fired after aiming by tapping the area within the circle; shot strength is proportional to the distance of the player's finger from their turret. Shots will damage the enemy turret and can destroy terrain wherever they hit. The game continues until one of the turrets is destroyed by multiple direct hits. This game is also played over a wireless network with another local player, and works similarly to *WapIt!* In single-player mode, the player competes against the computer.

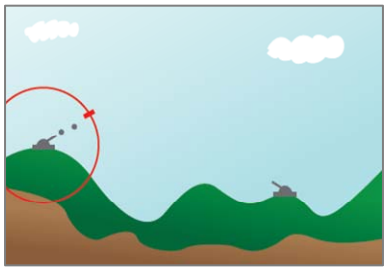

Figure 3. Screen of Turret Wars

2. Application Design Process

The design process used in this project is comparable to the Dartmouth-Thayer Engineering Problem-Solving Cycle, an engineering design framework that uses recursive iterations to help engineers develop a specific solution to a general problem. The Dartmouth-Thayer cycle consists of seven steps which move the problem-solver through a process of identifying the problem, setting the limits of usable solutions, brainstorming a set of solutions, analyzing the set against the limits to select the best solution, then repeating the process with increasing specificity until an appropriate solution is found[8]. The design process for this project has only four steps, and is organized for software development.

2.1 Define the Goal

The goal of this project was to conduct research: to discover and add to the existing knowledge of development on next-generation mobile devices like iPhone and to develop applications that took advantage of as many of iPhone's features as possible. The secondary goal of this project was to make an application that was interesting for the user to experience, and fun for the developer to write.

2.2. Synthesize the Idea

The next step is to determine the form the application will take. Questions like, "What do we want this application to do?", "Why will a person use this application?" should be asked here. Next-generation mobile devices like iPhone provide an extremely flexible platform on which to build an engaging application, however, special care must be taken to conform to the heuristics of usability for mobile devices [3]. To preserve the simplicity and intuitiveness typical of iPhone, preset conventions and methods are built into Cocoa Touch that govern the way developers can access iPhone's functions and hardware. Yet, even for developers familiar with Cocoa, the desktop version of the mobile framework, the transition from desktop to mobile application development presents unexpected challenges. Knowing the type of application that will be developed and the conditions or environments it will ideally used in enables the developer to plan the best design for the interface and underlying processes in order to maximize given resources [7].

For this project, the applications took the form of games since games could incorporate all or most of the novel functions of iPhone more easily than other types of applications. The team was also more likely to invest the amount of time and energy needed to create and perfect a truly engaging application if developing something fun. The games would ideally be used for short-term entertainment purposes when the user is idle or wants a fun way to pass the time.

2.3. Identify Constraints

Challenges in the development of mobile application development stem from three sources: the ideals of the developer, the expectations of the user, and the capabilities of the hardware [6]. If either the ideal of the developer is not realized or the expectations of the user are not met in a given application, then it is deemed a failure. The same

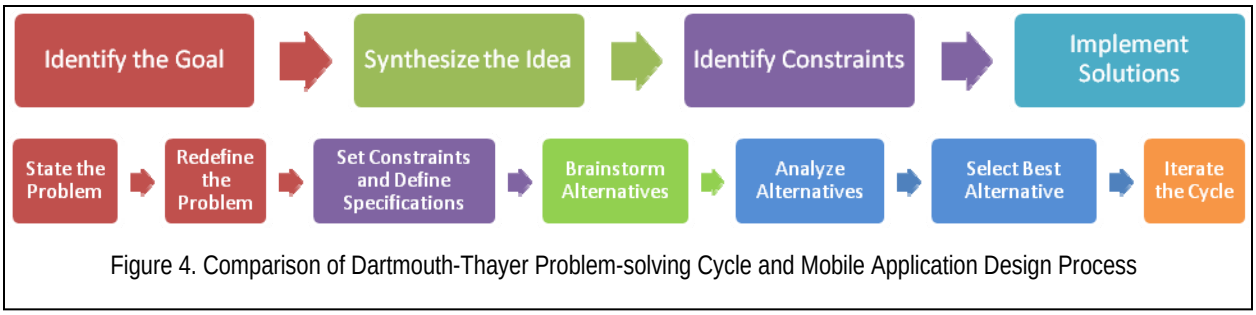

Figure 4. Comparison of Dartmouth-Thayer Problem-solving Cycle and Mobile Application Design Process

holds true for applications that make both the developer and user happy in theory, but are poorly designed for use on mobile devices. A developer must identify the constraints associated with a certain application to determine if it is feasible to implement. In the Dartmouth-Thayer cycle mentioned above, setting general application constraints is intended to eliminate the bias of the engineer from the design process, which is why the limits are set in place before solution are proposed. In the software design process of this project, however, the limits set are designed to make sure the preferences of all parties, including that of the developer, are considered in development of the application. Specifications included standard functionality perimeters as well as the more qualitative desires of the users and developers. To narrow the field of alternatives, the team weighted each constraint based on its importance in the scope of the project. Some constraints, like "has research value," were deemed absolute limits, so game ideas that did not fit these particular criteria could not be chosen at all. Overall, the games in this project were designed to be as simple and streamlined as possible, fulfilling all necessary program functions in an entertaining way while also keeping in mind the limitations of the platform on which the games would be implemented.

4. Implement Solution

The next step was the development and implementation of the game idea, which is detailed in the next section. Initially, the game idea that passed the most constraints and all of the absolute limits was a set of educational mini-games. However, the team decided that those types of games did not adhere to one important absolute specification that had been previously overlooked—the games were not much fun to design, and there would be very little room for creative expression. After going through the design process, the team finally chose to develop *WapIt!* and *Turret Wars*, two "just for fun" games, as the applications for the project.

3. Application Development

3.1. Networking

The networking code that supports the multiplayer aspect of this project is based on a Server/Client design pattern. It sets aside a centralized location for storing user data, simplifying communications between connected iPhones, and can use any arbitrary available port on the network, decreasing the likelihood of signals between iPhones become lost or corrupted. To play a networked game, players who chose the "multiplayer" option are presented with the option to "Create New Game," which designates that iPhone as a host, or to "Join Game," which makes the device a client. iPhone then uses Cocoa Touch's implementation of Bonjour, a zero-configuration networking system for automatic discovery of network clients, to enable host iPhones to publish a game service on the local network or client iPhones to discover local servers. Once a client discovers the servers on its network, a table appears displaying a list of current servers, allowing the user to choose which host they would like to play with. This option gives greater flexibility to users who might want to play with a group of friends instead of random players in the area.

The tools the team used and created in making these multiplayer games form a foundation for future developers wanting to design multi-user applications, and, unlike previous networking implementations, allow applications to handle connections from multiple clients. The code allows developers to publish and discover networked applications using iPhone and UDP sockets, as well as general UDP applications and services, and can be extended to suit the particulars of any type of application. To customize the networking capability of specific applications, developers can add more data objects for transfer in messages or can communicate using different message types. Developers can also customize how an iPhone discovers other iPhones running a networked application, how a host device handles devices that wish to connect as clients, and how applications handle server connections.

3.2. Gesture Input Design

The user input aspect of both WapIt! and Turret Wars was developed with simplicity as the principal goal. The multitouch gestures (tap, swipe, and pinch) used for these games are all easily performed within iPhone's small screen space; each gesture was also designed to use up to three fingers, though iPhone's touchscreen can handle more, in order to ensure that a user's hand would not obscure the view. The gestures for WapIt! were chosen knowing that users may already be familiar with them from using other touch-based applications on iPhone, which would make the game simpler to learn and play. The circular brushing gesture used to control aiming in Turret Wars was inspired by the original mouse pointer motion of Dome Wars, and allows the user to easily choose the angle of the firing direction.

The code written for gesture recognition in the games is centralized and customizable. It provides a strong foundation for developing recognition for even more complex gestures. The multitouch code can supply information about individual positions, velocities, and angular relationships of up to three finger touches, which can be manipulated to detect new, unique gestures for future applications.

The accelerometer gesture for WapIt! required extensive user-testing at every stage of its creation. Initially, a twisting motion was to be incorporated into the game, however, it was converted to a simpler "shake"; developing an algorithm that could distinguish the motion while account for the multiple variances in the way individuals hold and move iPhone proved to be too complex and time-intensive for the purposes of this project. The algorithms were built using visual and contextual information from an accelerometer graph application from Apple's developer website that displayed the three-dimensional velocity of the device. The final "shake" gesture allows a larger range of motion and greater sensitivity than the twisting gesture to account for slight differences in individual hand movement, but still requires players to mimic the animation on screen to continue game play.

3.3. Visual Interface

The design of the visual interfaces of mobile devices must be optimized for mobile interaction, taking into consideration the small screen space and divided attention of the user [1]. The iPhone OS addresses a main problem of screen clutter, caused by multiple application windows open in a small space, by setting the "one window per application" rule, which also mandates that only one application

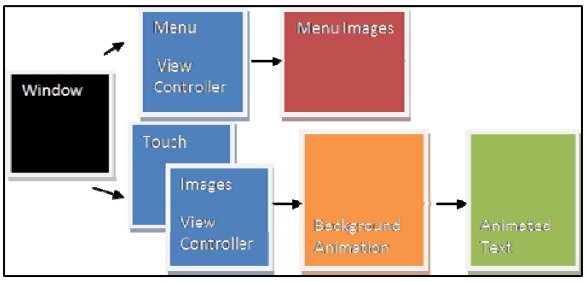

Figure 5. Diagram of Screen Input/Output Hierarchy

window can be displayed at a time. Though this convention may initially seem limiting, Cocoa Touch compensates by allows the developer to define a view hierarchy, composed of layers of interactive display content called views, anchored by one window. High-resolution graphics and animations supported by the iPhone OS, when coupled with the dynamic view setup, expand opportunities for the development of visually-appealing user interfaces.

In *WapIt*! the limits of this setup were explored. Multiple layers of views and view controllers, Cocoa Touch objects used to organize and manipulate views of various types and sizes, were used to build the colorfully animated window shown during gameplay. Using the built-in frameworks of Cocoa Touch allows developers to create vibrant graphics and effects for any type of application, greatly improving user's enjoyment of the experience [1].

Using OpenGL ES for *Turret Wars* allowed us to create a fully animated game that ran within a single view. The initial graphics of the game were written for desktop implementation and included computationally expensive effects, like randomly generated terrain composed entirely of particles. To reduce the load on iPhone's processor, the terrain was changed to a solid object that only rendered exploding particles upon detection of a collision with another object, a change that greatly improved performance on iPhone.

The methods used in developing the visual interfaces in this project can be used to greatly enhance existing and future mobile applications. On devices like iPhone, developers with applications that need to display multiple sets of information can feel restricted by the one-window rule. Using the animations and subview layering approach presented in *WapIt!*, applications can move easily and seamlessly through multiple views, displaying and hiding them as necessary, which give developers the flexibility to include many different sets of information in the application if needed and to choose the best way to layout each set. Developers can also use OpenGL ES to create graphical versions of dynamic information that might otherwise be hard

for the user to decipher, such as a program that indicates the desirability of detected wireless networks based on signal strength, number of current users connected, and speed.

4. Future Work

The immediate future of mobile application development is context-aware mobile computing, in which applications can observe and process "the set of environmental states and setting that either determines an application's behavior or in which an application event occurs and is interesting to the user" [2]. Current, commonly-used devices are already enabled to send and receive information context-aware applications would need: GPS, Bluetooth, RFID (radio frequency identification), wireless internet, and supporting hardware, all enable a device to send and receive messages about its current state, its user's current state, and its environment. In addition, Apple's iPhone has sparked a revolution among manufacturers of multifunction mobile devices, inspiring revamped incarnations of existing phones and PDAs that incorporate similar technology and interface designs to improve usability and increase consumer adoption [6].

Once FMWC devices achieve the necessary level of pervasiveness needed to support a context-aware infrastructure, the types of applications and uses for this technology will be unimaginable. Professionals could build smart buildings like the "Intelligent Hospital" proposed by Cambridge University [5]; engineers would be able to access real-time information about the entire inventory of parts moving through the building process of a factory; travelers can receive updates, reminders, and sightseeing tips based on their location, travel itinerary, and current company. In order for these scenarios to become reality, applications development must be greatly improved. The goal of this paper was to present a design process with example applications to assist future developers in overcoming current design and usability challenges with mobile application development. Hopefully, using the processes, techniques, and open-source code outlined, creating novel applications with functionality beyond what is currently available will be a feasible goal for all iPhone developers.

5. Acknowledgements

Special thanks to Ken Kopecky and Brandon Newendorp, the graduate student mentors of this project for all their help. Thanks to Dr. Eliot Winer and Dr. Stephen Gilbert, the team's faculty mentors, and to Iowa State University's Human-Computer Interaction Graduate Program, Virtual Reality Application Center, and Program for Women in Science and Engineering for making this research project possible. This research was performed as part of an undergraduate research internship funded by a grant from the National Science Foundation IIS-0552522).

6. References

[1] S. Brewster, Overcoming the Lack of Screen Space on Mobile Computers. Personal and Ubiquitous Computing, n.6, 188-205, 2002.

[2] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, Nov 2000.

[3] G.H. Forman and J. Zahorjan, "The Challenges of Mobile Computing," *Computer*, v. 27, n. 4, 38-47, 1994.

[4] L. Gorlenko , R. Merrick, No wires attached: Usability challenges in the connected mobile world, IBM Systems Journal, v.42 n.4, 639-651, October 2003.

[5] S. Mitchell , M. D. Spiteri , J. Bates , G. Coulouris, Context-aware multimedia computing in the intelligent hospital, Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system, September 17-20, 2000, Kolding, Denmark.

[6] S. Sarker, J. D. Wells, Understanding mobile handheld device use and adoption, Communications of the ACM, v.46 n.12, December 2003.

[7] P. Tarasewich, Wireless devices for mobile commerce: user interface design and usability, Mobile commerce: technology, theory, and applications, Idea Group Publishing, Hershey, PA, 2003.

[8] What is Engineering Problem-Solving? Dartmouth Project For Teaching Engineering Problem Solving. <http://engineering.dartmouth.edu/teps/default_what.is.eps.html>