A Novel Interface for First Person Shooter Games on Personal Digital Assistant Devices

Chen Wei

January 2008

Dissertation submitted in partial fulfillment of the degree of

Master of Science in Computer Science by

Coursework and Dissertation

Department of Computer Science

University of Cape Town

Abstract

This dissertation explores Novel Interfaces for First Person Shooting (FPS) games on Personal Digital Assistant (PDA) devices. The new approach uses intelligent gesture recognition combined with the optimal implementation of basic game functions (i.e., jump, shoot, walk forward) to improve the interaction in FPS games on PDAs (FPS-PDA). The final prototype, *InteractionPro*, was built for the Expert Evaluation of the new interaction, namely *Gesture Interaction*. This prototype is a mini 3D FPS game engine built specifically for PDA devices, which compares both the newly designed and existing interaction systems. It was developed for the Dell x51v PDA with the help of the .Net Compact Framework 2.0 and Mobile DirectX, using C# as the programming language.

The main aim of this study is to enhance the playability of games on current standard PDA devices. The newly designed interface more effectively leverages current well-established devices, which solves the problem of rapidly and accurately executing a large number of gaming commands. The outcomes of this research are beneficial for interface design of mobile applications.

Content

1 Introduction

Figure 1. New Interaction System for an FPS game on the Dell x51v

1.1 Introduction

With the help of the FPS Game Engine's fast development, 3D FPS games have become very popular on desktops. Successful games, such as the DOOM series and Quake series, not only stimulate the development of the gaming software market, but also greatly help the development of all kinds of computer hardware. Regrettably, these kinds of games, with the exception of unofficial games which have been ported to PDAs (Quake3CE by Rioux, DoomGL by Ryssen), do not exist on PDAs.

Despite the PDA's hardware limitations, interaction in playing PDA games is very difficult to work with for most game developers, designers and players. Due to the nature of PDA input, executing commands in gaming is severely restricted by the stylus pen and its limited set of buttons. Other problems, such as user fatigue due to holding the PDA, difficulties in coordination when performing multiple commands simultaneously and restricted camera movement, were found in Chapter 3.3. These problems present an interesting challenge not just to FPS players, but also serve as an extreme test case for new interaction techniques for controlling and interacting with

complex information on a PDA screen.

Since the power of mobile hardware has received sufficient attention of late [1, 2, 3], PDA devices such as the Dell x51v are now capable of rendering simple three-dimensional (3D) environments. Based on this, a 3D FPS game engine prototype, *InteractionPro*, written in C# was created from scratch and built to study this new interface design for 3D PDA gaming, which we term *Gesture Interaction*. This prototype affords a comprehensive investigation of *Gesture Interaction* through the help of the .Net compact framework 2.0 and Mobile DirectX. It is also the first C# FPS game on a mobile device, showing both the efficiency and quality of this novel combination.

This research area, which focuses on the interaction design for a complex interface especially on mobile devices, is quite new. Documents and references about it are correspondingly scarce. All the tests, analyses, designs, programs and evaluations are original and implemented following a user-centred methodology.

1.2 FPS games and Mobile Device Hardware

According to iResearch 2005 China Mobile Game Research Report[1], the number of global mobile game users was 290 million in 2005 and is expected to reach 1.03 billion in 2008. Various games for mobile devices have been built to meet the requirements of this huge mobile game market. Most of these are 2d games with simple interaction. 3D games are rarely found on mobile devices.

Slow hardware performance is one of the significant reasons hindering the development of mobile 3D games. The 3D processing ability, especially on cell phones, is still very limited although there have been quite a few mobile devices with integrated 3D accelerator chipsets such as the Nokia N-Gage. The PDA, Dell x51v which is currently the most powerful 3D processing mobile device in the market, is still not able

[1] 2005 China Mobile Game Research Report
http://english.iresearch.com.cn/reports/MVAS/detailreports.asp?id=7476

to run Quake3CE smoothly on the Windows Mobile 5 (WM5) platform. The Quake and Doom series are the only FPS games currently available on PDA devices. Further, these were ported from desktop Personal Computers (PC) and not directly designed for mobile devices. This illustrates the fact that there has been very little original development effort for implementing FPS games specifically for PDA devices.

1.3 Current interaction styles and problems

Interaction problems often occur when a game is designed for one interface and later ported to a device with a different one. These problems become even more serious for FPS games such as DoomGL and Quake3CE, which require more complicated interaction. Both DoomGL and Quake3CE have very similar interaction styles; even the most basic interaction events such as 'Looking and Moving' and 'Firing' are the same. In these games, users can control the Avatars viewpoint by sliding the stylus pen over the screen. The avatar in the scene is moved by pressing the directional buttons on the bottom of the PDA interface. 'Firing' is triggered by pressing the Return button which is surrounded by the directional buttons. The other four Hotkeys on the PDA are normally assigned for the advance functions such as 'Menu', 'Select', 'Map', etc. The number of available buttons on a PDA that can be linked to user commands is far less than most FPS games require. For instance, in Quake3 Arena, there are a total of twenty-two commands for 'Moving' and 'Shooting', eleven of which are commonly used. Unfortunately, there are only nine buttons available on normal PDA devices (four directional buttons, one Return button and four hotkeys). Each button is usually dedicated to a single command; therefore, only nine commands are implementable on the PDA device while playing FPS games. This limitation severely affects the FPS-PDA playability.

Besides the aforementioned problems, users find it inconvenient, inefficient and tiring to play FPS-PDA using the current interaction style. Details of current interaction problems will be described later in this paper.

1.4 Significance of Research

Computer games, especially FPS games on desktop PCs have lead to the development of computer hardware because of the demand for better game performance and image quality. Many FPS games have become the standard test applications for testing the configuration performance of PC hardware, such as Quake III, Doom3, etc. This has, however, not happened on mobile devices.

The main aim of this study is to take advantage of current well-established devices and input systems to execute large numbers of commands so as to enhance the playability of PDA games. It may help the design of mobile device interaction that requires multiple functions to be applied simultaneously, such as 2D mobile games and other applications.

We hope that this kind of research will stimulate the game market for PDA devices, which may lead to the stimulation of FPS game engines, the development of PDA hardware, software and applications, similar to the development of desktop PCs in response to FPS games.

1.5 Scope and Objectives

This research is intended to develop new interaction methods for FPS games on PDAs. It focuses on the interaction of FPS games rather than other game types, for the following reasons:

1. FPS games typically involve complex interaction that demands a high degree of co-ordination between both the keyboard and mouse input systems.

2. FPS games have had a great impact on the development of both PC hardware and software.

Research by H. Korhonen and E. Koivisto [4] shows that during the heuristic evaluation of mobile games, situations where playability heuristics violate the usability standards happen quite often. Although this study focuses on the investigation of the interaction

system, both the usability and playability [4, 5, 6] of the new interaction system were considered so as to minimize problems. The mechanics of how an immersive and rapid game (an FPS in this case) can best be played on a standard mobile device is at the heart of this research.

Other elements in the FPS game engine such as 3D rendering and collision detection, were essential for building the prototypes, though they in themselves were not the object of this research.

1.6 Research Questions

PDA interaction, especially in gaming, can hardly compare to the desktop PC because of the structural differences between the desktop PC and PDA devices. The only comparable interaction interface of a FPS-PDA is that of DoomGL, which resembles the style of button pressing. Based on this, the research questions are as follows:

1. How does the newly designed *Gesture Interaction* system compare with the traditional button pressing interaction style in playing FPS-PDA?

2. What is the subjective feeling most players have towards the new interaction style?

1.7 Methodology

This research was implemented drawing from Jones and Marsden's [7] three outcomes, which are:

- Understanding users
- Developing prototype designs
- Evaluation

To understand the users, background research on traditional interaction practices of FPS games on desktop PC's was conducted. An application named *InteractionLog* was built to help this investigation. The primary goal of this was to examine user behaviour

whilst playing FPS games and develop a new interaction style that would support play on PDAs.

In this research, a computer-based low-fidelity prototype and a fully-functional prototype [8] were implemented as the best combination to evaluate the usability of the new interaction design in a low cost approach. The low-fidelity prototype is a Flash application while the fully-functional one is a real PDA 3D application, namely, *InteractionPro*.

The evaluation of the proposed interaction style is a new research area. Traditional heuristic evaluation [9, 10, 11, 12], which is used for formal software applications cannot be applied to electronic games, because electronic games have different design considerations and usability issues. Interaction design in mobile games is very different from a normal application interface design [13, 14].

Korhonen and Kovisto's [4] guidelines for heuristic evaluation of "Game Usability" were chosen for application to this research. They are:

GU6: "Navigation is consistent, logical and minimalist"

GU8: "Game controls are convenient and flexible"

This choice was made because the experiment was intended to evaluate the interaction system solely, instead of the entire game. Two other guidelines were also used to conduct the evaluation:

I. The level of challenge and entertainment that the user obtains from the new interaction

II. The level of the user's overall satisfaction with the interaction.

These criteria were added since the ultimate goal of playing games is to be challenged and have fun, which means the process of achieving tasks must not be too straight forward [4].

These guidelines were assessed through both objective experimental results and the

user's subjective experience. Objective data that logged all the tester's movements whilst running the application was recorded into an XML file. The data recorded includes the time the user spent on the different stages, the number of times each button was pressed, how long each key was held and the mouse movement rate. The application, *InteractionAnalysor*, was built to help investigate the objective data. The user's subjective results were evaluated by heuristic evaluation.

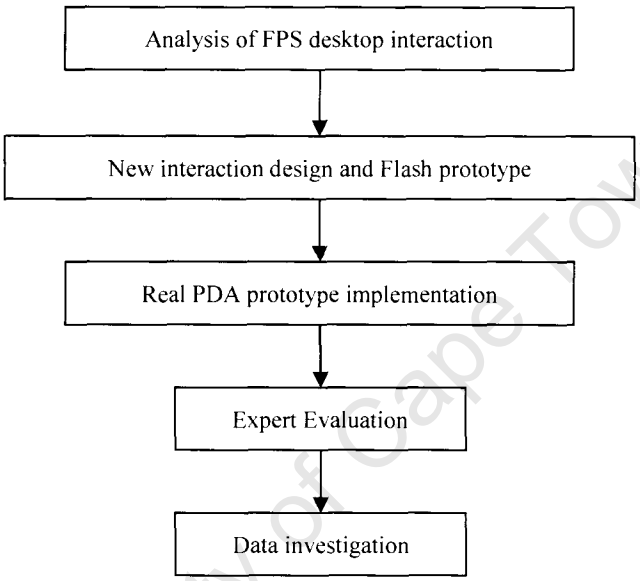


Figure 2. the five stages of the research.

1.8 Overview of the Dissertation

Subsequent chapters review related work and describe the interaction design methodology used and results obtained.

Chapter 2 describes the background and the current situation of game development on PDA devices.

Chapter 3 depicts the investigation of how users play FPS games on desktop PCs and the new interaction system design.

Chapter 4 depicts *Gesture Interaction* on the fully-functional PDA prototype.

Chapter 5 introduces the implementation of *InteractionPro*.

Chapter 6 presents the evaluation of *Gesture Interaction.*

Chapter 7 reports on the conclusion of the experiment and the study.

2 Background study

This chapter describes the history of FPS games on desktop PCs, the development of the PDA's hardware, games and interaction system; and the tools used for implementing the high fidelity prototype. This summary serves to help the reader understand the background of this research topic.

2.1 FPS game history on desktop PCs

Since the 'revolutionary' DOS game Wolfenstein 3D (Wolf3D), which was created by id Software and published by Apogee Software, was released on the 5th of May 1992, FPS games have become wildly popular. This is due to the strong sense of presence [15, 16, 17] such games offer the players.

Later on, with the successful release of FPS games such as Doom, Quake III, etc. FPS games began to be converted from 2D to true 3D. Even now, FPS games are developing so fast that they are not only popular on the desktop PCs; most game producers also release their new FPS games for the Xbox or PlayStation platforms, such as Call of Duty 4, Unreal Tournament 3, etc. These games are becoming more and more complicated in all aspects due to powerful hardware support. For example some visual effects, such as high dynamic range (HDR), which a decade ago, were considered to be available only in movie effects are now being rendered realistically in everyday 3D scenes.

Wolfenstein 3D (1992)

Quake III (1999)

Call of Duty 4 (2007)

Figure 3. Screenshots of three classical FPS games.

2.2 PDA development

2.2.1 PDA games

Mobile games have been developing since the release of the first PDAs. PDA devices were initially designed for business use. The games on these devices were supposed to be fun and relaxing for businessmen to play with while on the road. Most of these games have simple tasks and are easy to manipulate. With the development of other handheld game consoles, such as the PlayStation Portable (PSP) by Sony Computer Entertainment 2004, and the Nintendo DS (NDS) by Nintendo 2004, the hardware and games industry on mobile devices has improved significantly. Nowadays players desire more than just simple 2D games on mobile devices. As the power of mobile computers and PDAs has improved, 3D PDA games have started to appear on the market. For example games such as Toy Golf by Fathammer 2004 and GeoRally EX by LonFx-Studios 2005 have been developed. The rendering speed in these games is capable of running at more than fifteen frames per second (FPS) on the Dell 50/51v.

However, the interaction system in these games still remains a problem.

2.2.2 PDA's interaction

The stylus pen is the standard input device on PDA devices nowadays. The stylus together with a graffiti hand-writing system provides a gesture recognition interface and solves the problem of effective text input. With the help of the four hotkeys, four directional buttons and one confirm button, users are able to manipulate the PDA's normal applications smoothly. Although the convenience and efficiency of this input system cannot compare to that of the desktop PC, it provides users with an appropriate way of entering text, surfing the web and taking notes.

Game playing is very different from operating common windows applications. A large number of games have complex interaction which demands a high degree of collaboration between both keyboard and mouse input systems. The initial investigation of this research (Chapter 3) shows that it is very difficult to play these games using only the nine buttons and stylus pen in a traditional button pressing way on PDA devices. For instance, in playing GeoRally Ex on a PDA device, the user cannot make the race car accelerate and turn left at the same time, which violates the intention of 'accelerating while turning'. This is because the user is not able to simultaneously activate two directional buttons.

The problem of interaction in FPS games is much worse than for car racing games, it involves many more commands and more complex user-interface coordination. A new interaction design is proposed in this thesis to solve these problems.

2.3 Mobile Direct3D and C#

The high-fidelity prototype built for the research of this study is compiled for the Visual Studio .Net 2005 (C#) platform. The 3D API it implements, uses Mobile Direct3D, which was released with Windows Mobile 5 by Microsoft in the end of 2005. It is a mobile device implementation of the desktop computer's Direct3D API. It was

developed based on DirectX 8, though it includes some of the characteristics of DirectX 9.

For the purpose of this research, 3D rendering speed is not that crucial for the final prototype, so long as the motion in the 3D scene is sufficiently smooth. It was necessary to implement the prototype quickly to meet the research schedule. C# in the .Net Framework facilitates advance programming than C++ in that it helps developers focus on logic more than resolving basic programming issues such as memory management. Therefore, the Mobile Direct3D and .Net Compact Framework 2.0 in C# was chosen as the development platform due to its convenient and powerful characteristics.

2.4 Summary

This chapter shows that FPS games have made a significant impact on the development of the desktop personal computer, which in turn enables these games to be more and more realistic with the support of more powerful computers. Unfortunately, this synergy has not occurred on PDA devices. Executing commands in gaming is severely restricted by the stylus pen and its limited set of buttons. This severely affects the playability and the development of PDA games. A new interaction system design is needed to meet the requirements for playing mobile games. For the purpose of developing and evaluating the new interaction system, the Mobile Direct3D and .Net Compact Framework 2.0 in C# was chosen as the development platform due to its convenient and powerful characteristics.

3 Investigation and Design

This chapter discusses the investigation of user behaviour whilst playing FPS games on desktop PCs and current interaction problems with playing FPS games on PDA devices. After analysing these problems, an initial interaction design was proposed and evaluated using a low-fidelity prototype.

Figure 5. Visualizations of 30 second logs from a Counter Strike player. Avatar moves are shown in green, MouseButton events in red, EventTriggers in blue and WeaponChanges in purple. The upper plot shows assault rifle usage, while the lower one shows sniper rifle usage.

3.1 *InteractionLog* and Behaviour Investigation

3.1.1 Primary Goal of the Investigation

When it comes to the usability evaluation of games on mobile devices, Korhonen and Kovisto [4], recommend that the function keys (the buttons which control the specific game commands) should be consistent and follow standard conventions (e.g., the keys 'W', 'S', 'A' and 'D' are conventionally used for moving 'Forward', 'Backward', 'Left' and 'Right'). However, for FPS games there are no "standard conventions" on a PDA. Therefore, the first goal of this research is the development of such conventions. In order to do this, users behaviours whilst playing FPS games were examined, and then effective interaction conventions were developed which best support those behaviours.

To examine user behaviours, in particular relating to the keyboard and mouse (these are the only input systems in FPS games on Desktop PCs), these questions were formulated:

1. What keys and mouse buttons are most frequently used when playing a game?
2. How often are these keys and buttons used during the game?
3. How are these keys and buttons used together?
4. What is the relationship of the actions executed by these keys and buttons?

The answers to these questions refined the range of the users' behaviours and offered a better understanding of the most important functions in FPS games. These results were very helpful for designing the new interaction conventions.

To answer these questions by manual observation is not easy and can often be inaccurate. Therefore, the *InteractionLog* software was built to help capture the results precisely and objectively.

3.1.2 Introduction of *InteractionLog*

InteractionLog is a .Net windows application which was originally built to help investigate user behavior whilst playing FPS games on desktop PCs. It logs interaction events of the keyboard (key presses) and mouse (button clicks and movement rate) in the background while the player is playing an FPS games. These keys and buttons are the function commands used in FPS games on desktop PCs. All these user interaction events are recorded in an XML file which is intuitively visualized; this visualization can be performed instantaneously while the game is being played or offline, to reduce lag on the game and for future reference. The visualization, as shown in Figure 5, helps with the analysis and understanding of the relationships between different interaction events.

3.1.3 Implementation of *InteractionLog*

Counter-Strike 1.5 was the FPS game chosen for the test. It is one of the most popular FPS games in history, and most FPS game players are familiar with it. Ten volunteer Counter-Strike players with varying degrees of expertise were contacted and had the *InteractionLog* installed on their machines. After one week, their log files were sent back for the further investigation.

Action commands vary from FPS game to game, but basic functions are common across most games. These common interaction actions are divided into four main types according to their characteristics:

a) Avatar movement

b) Camera movement

c) Aiming and Firing

d) Advanced Function Commands (e.g., triggering events, changing weapons, etc.)

These actions are highlighted in different colours in the visualized plot. For instance,

avatar moves are shown in green, camera moves (mouse move rate) in blue, MouseButton events in red, EventTriggers in blue and WeaponChanges in purple.

3.1.4 Work Flow of *InteractionLog*

The Figure 6 illustrates the work flow of *InteractionLog*. This application consists of two modules. The left module is in charge of logging the data from the testing computer, compressing the results and sending them to us. The right module shows the visualization of the XML data result. Based on these two modules, the *InteractionLog* offers an efficient way to help the investigation of the users' behaviors whilst playing FPS games on the desktop PCs.

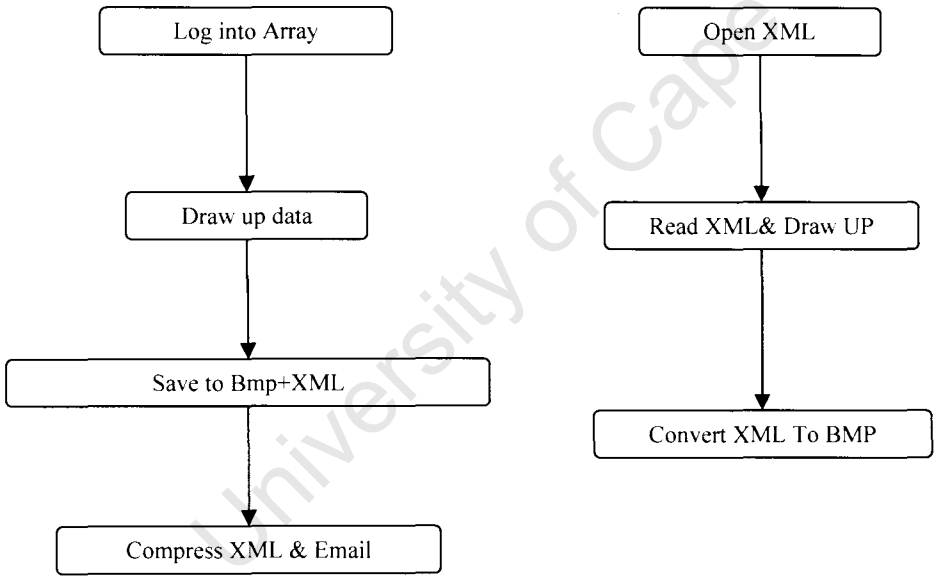


Figure 6. The work flow of *InteractionLog*

3.2 Data Analyses

As can be seen in Figure 5, there are many time sections (episodes) where the red, green and cadet blue segments overlap. This means players often engage in the first three activities simultaneously (moving the avatar, moving the camera and firing). This situation frequently occurs when the player intends to dodge opponent's fire and shoot back. Whilst the choice of weapon did affect the exact pattern of usage (e.g. an assault

rifle is used in a completely different way to a sniper rifle), the simultaneity of the actions remained. Many such episodes can be seen in Figure 5. Clearly, the PDA solution must support moving, firing and camera actions that can be executed simultaneously without interfering with each other.

The two most common activities observed were those of firing and moving. Again, the plot in Figure 5 shows 42 shots fired and 15 separate avatar movements inside the 30 second period. Solutions must make such actions rapidly accessible.

Finally, the advanced function commands, such as weapon changing, happened in isolation, which means they can be supported in a less direct fashion.

3.3 Evaluation of DoomGL Interface

3.3.1 Interface of DoomGL

How is the interface of existing PDA FPS games structured? One of the most popular games in this genre is DoomGL. Looking at the online reviews for this software does not make for encouraging reading:

"*On the other hand, the control scheme is plain awful and this is one of the first handheld titles where customizable controls don't really let you customize much at all. Working across Toshiba, HP and Viewsonic handhelds, none of the controls really feel right. There are too many useful functions left to the virtual keyboard.*"[2]

This is hardly surprising when one looks at the control choices made by the game designers.

[2] game-over.net review
http://www.game-over.net/reviews.php?page=handreviews&id=146

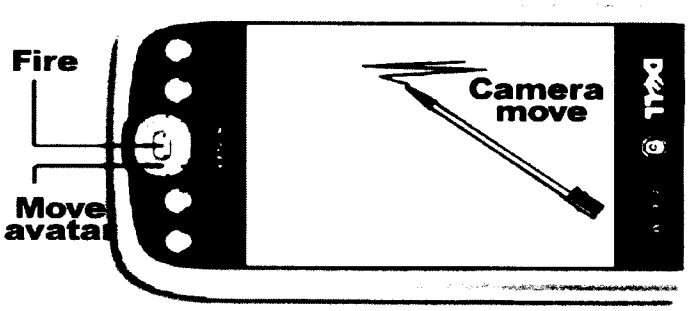


Figure 7. Controls for the PDA version of Doom

As can be seen in Figure 7, the avatar movement is controlled by the D-Pad and the shooting is controlled by the button in the centre of the pad. All camera movement is controlled by the stylus. This will almost certainly lead to problems as these actions (moving avatar, moving camera and shooting) occur at the same time. If one hand is holding the stylus, then the user is required to take their hand off the D-Pad to press the fire button. Moving the avatar and firing are now, effectively, mutually exclusive.

To see if these predictions held, a brief user observation was run afterwards.

3.3.2 Observed Play

In order to gain some insight into how difficult it was to interact with an FPS game on a PDA, seven players were recruited to play DoomGL. All the recruits had played FPS games on the desktop, but none had played it on the PDA. A number of serious problems were observed:

Users did indeed struggle to press the fire button whilst moving the avatar. The form-factor of the device required users to employ a single thumb for moving and firing, meaning that it was not possible to execute both actions simultaneously. This problem is serious since combined 'Dodging and Firing' are frequently necessary.

Camera movement is overly restricted by the fixed size of the PDA screen. Unlike using a mouse on a large desk, the stylus movement range on the PDA screen is fixed and very limited. The fixed movement rate makes it difficult to balance between micro-adjustments (e.g., aiming at the target through tiny adjustments) and big-turns (e.g., to change the camera horizontal angle by 100 degrees).

Most users felt that their left hand was exhausted after playing DoomGL since they were holding the entire weight of the PDA in that hand. And the left thumb had to stabilize and balance the device. In DoomGL, the PDA has to be held in a landscape fashion, with the left hand solely supporting the whole weight of the device from the edge. This is more tiring than holding the PDA in a portrait fashion as it severely affects the stability of holding the PDA and fatigues the hand, wrist and arm.

3.4 Initial Design and Flash Prototype

From the observations, it was clear that better interaction techniques needed to be found. To develop an initial set of techniques, a design workshop with five designers, who all had at least two years worth of experience in designing applications for PDAs, was convened. This led to an initial Flash prototype which was then refined through an iterative process, in which users were observed playing the prototype and interviewed about their experience.

3.4.1 Initial Design

The initial goal was to separate out the avatar movement and firing controls. The approach was to remain with the D-Pad to control avatar movement but move the fire control to tapping on the screen with the stylus. The central button in the D-Pad is now used to activate the 'stroke drawing' functions. Pressing and holding down the button activated a 'gesture recognition' area, wherein gestures can be entered to access the various control functions (e.g., weapon change). One alternative to this design was to use the extra buttons on the PDA to give direct access to these features. However, the gesture system was chosen as (a) it did not rely on specific hardware buttons that may not be present on all devices and (b) it did not limit function availability to the number of buttons on the device.

Firing: The initial idea is that the weapon fires at the point where the stylus is tapped.

This caused problems because the physical stylus obscured the target. To overcome this problem, 'auto' and 'manual' fire modes were created here. When the stylus is tapped a single time and the view ray hits the opponent, 'auto' fire mode is activated; this involves the screen panning to make the tap point the centre of the screen and, once this happens, the weapon will fire until the stylus is lifted from the screen. As an alternative, users could fire a single shot through the center of the screen by double-tapping anywhere on the screen.

Gestures: Whilst the gesture recognition worked well, users wanted more immediate access to gestures. This resulted in removing the gesture mode to allow gestures to be written on the main screen. Also, the gestures themselves were simplified to allow them to be written more quickly. After the user observations, the strokes commands were settled on, as shown in the Table 1.

Table 1.The description of the strokes

Strokes	Commands	Description
	Next Weapon	Draw the stroke from left to right
	Previous Weapon	Draw the stroke from right to left
	Jump	Draw the stroke from bottom to top
	Event Trigger (Such as door opening, map view, etc.)	Draw the stroke from top to bottom and turn to the left at the end

3.4.2 Flash Prototype

Figure 8. Interface of the Flash prototype

A computer-based low-fidelity [8] prototype which presents the new interaction design was built before implementing the real PDA high-fidelity [8] prototype. This prototype was built using Flash on the desktop. The aim of this prototype was to validate the main concept of the new interface through comments. It was also meant to evaluate other major usability issues such as unclear meanings, graphical representation issues and appropriate positioning of interface elements. Other issues like physical handling and operation, comparison with other similar products and performance-related issues were evaluated by the final PDA fully functional high-fidelity prototype.

Due to the functional limitations of Flash, the Flash prototype only simulates the interface and the newly designed interaction method but not the game. It is not fully 3D and 3D effects are simulated by panning. The functions of the Flash prototype are:

1. The motion of the avatar's camera view is simulated by sliding the background picture around in a specific range. The background, the machine gun and the enemy are implemented using pictures from Quake III.

2. The actions for 'Walking Forward/Back' and 'Shifting to the Left/Right' are implemented by zooming and sliding the background picture around.

3. The 'Firing' effect is simulated through the 'Firing Gun' picture and a sound effect. There is no actual bullet, which means no collision detection had to be implemented.

4. The motion, 'Manual Shoot' and 'Auto Aim & Shoot', are implemented. 'Manual Shoot' is triggered by double tapping on the screen. And 'Auto Aim & Shoot' is activated when the user taps the stylus pen on the virtual enemy which is surrounded by a green box in the scene. 'Auto Aim & Shooting' is intended to prevent users from tapping the touch screen excessively. However, the 'Auto Aim & Shooting' does not happen immediately after one has tapped onto the object. There is a short lag (about 0.5 seconds) caused by the centre of the screen panning to the point tapped before shooting starts. This can result in the shot missing the target, if the object moves and is no longer in the position previously tapped during that 0.5 seconds. This strategy is designed to balance the challenge in 'Auto Aim & Shooting' mode.

5. The 'Drawing Stroke' function is also simulated in the Flash prototype.

3.4.3 Evaluation of the Flash Prototype

Seven players who attended the test of DoomGL were used again. They were introduced to this Flash prototype and explored it. According to the interview afterwards, all of the participants were pleased with the main conception of the new interaction design, which was:

(a) Separating the functions 'Firing' and 'Moving' to two hands.

(b) Optimizing the function of 'Aiming and Shooting'.

(c) Triggering the commands through the 'Stroke Drawing'.

However, problems related to the interface were also found during the evaluation. For instance,

(1) Users were not fond of the idea of putting the green frame around the enemy in the scene. This was intended to facilitate recognition of the target and tapping onto it. Unfortunately, users thought the green frame made the game too easy to play.

(2) It was suggested that the motion speed of the 'Camera Panning' which happens in the 'AutoAim&Shooting' mode be increased.

(3) Some of the players preferred to tap on the button-like commands to execute the functions rather than drawing strokes.

These problems were practical, which might result in user's dissatisfaction with the new interaction system. Especially problem c) , which actually shows the uncertainty of user's acceptance of the *Gesture Interaction*. This leads to the further questions which were intended to be answered in the final high-fidelity prototype evaluation:

Is the 'Drawing Strokes' mechanism rapid enough for FPS game-play?

Do the players like this mechanism in the real enviroment?

Does the 'Auto Aim and Shoot' mechanism work effectively?

Are the strokes hard to learn?

Is the device able to run the 3D application with so many interaction functions smoothly?

3.5 Summary

With the help of *InteractionLog*, the results from the investigation of the user behaviour on desktop PCs are clear and helpful. Based on these results, function commands common in most FPS games are divided into four categories. Problems with implementing these function commands were found through the observation of user behaviour whilst playing DoomGL on PDA devices. A new interaction system, *Gesture Interaction*, was designed and implemented as a Flash prototype. The evaluation of the Flash prototype shows that the main conception of the new interaction system is accepted by the users. However, there are still some questions that need to be answered by the fully functional prototype.

4 High-Fidelity Prototype

This chapter introduces the high fidelity prototype *InteractionPro*, which has been developed from scratch,. It also presents an overview of the Dell x51v device on which the prototype was implemented. The emphasis of the chapter is on the interaction and stage design required for this prototype.

4.1 PDA Dell x51v Introduction

Figure 9. Photo of the PDA Dell Axim x51v

The Dell Axim x51 family, which are x51 Low-end (x51 Low), x51 Mid-end (x51 Mid) and the x51v high-end (x51v), was released on September 2005 by Dell. Like most standard PDA devices, Dell Axim x51v is equipped with a stylus pen and eleven function buttons. These buttons are:

Four directional buttons on the Direction-Pad (D-Pad) which is in the middle bottom of the device

One 'Return' button which is in the center of the D-pad

Four hotkey buttons located at the bottom, which are specific for: 'Home Page', 'Mail Box', 'Contact List' and 'Calendar'

Two left side buttons which are for 'Wireless On/Off', 'Recorder'

One 'Power Switch' button in the top center of the device

In addition to these, the x51v has the following features according to the device manual:

- 3.7" VGA LCD screen which is around 2.22"x2.96" with a resolution of 480*640,
- Intel XScale Processor running at 624MHz
- 256 Intel StrataFlash ROM with 64MB on-board RAM
- An independent Intel 2700G 3D accelerator & video decoder chipset integrated with 16 MB video RAM
- Microsoft Windows Mobile 5 operating system (WM5) which supports Mobile DirectX and .Net Compact Framework v2

The built in 3D accelerator chipset Intel 2700G ensures that this x51v device is more capable of running 3D applications than other mobile devices which do not have one. To confirm this, a 3D application test between HP iPaq 4700 and Dell Axim x51v was executed. In the mean time, this experiment revealed further details of the x51v device, which greatly helped the further programming on this device.

HP iPaq hx4700 is also a very powerful PDA device in the market. It has the same processor, same size of the RAM and the same resolution of display as the x51v. Its graphic chipset is the ATI Imageon 2300, 2MB memory with no 3D accelerator.

The test application was a sample program from the Managed Direct3D Mobile Samples by Microsoft. A minor modification was made in the sample code so that extra 3D objects could be imported into the sample application. These extra models were a cuboid skybox, a grass plane and a 3D cottage model which is in the 'md3dm' format supported only by Mobile Direct3D. Instead of using the single texture model bundled with the sample application, the new 3D scene contains 213 vertices and 16 textures, which was intended to simulate the same complexity required by the final prototype.

The results of the experiment were assessed by the Frames Per Seconds (FPS) detector which was embedded in the sample code. The higher the score, the faster the 3d scene can be rendered. The following figure shows the result. The PDA on the left is the HP hx4700 and the one on the right is the Dell x51v.

HP hx4700	Dell x51v
4.55	29.25

Figure 10. The test of the 3D performance: HP hx4700 on the left, Dell x51v on the right. The average frames per seconds are shown in the table.

This shows clearly that the 3D capabilities of these two devices are not even on the same level. 3D games are definitely not playable on a device which runs the game at less than 5 frames per second. As the Dell x51v is the only PDA on the market that has a 3D accelerator, it becomes the only choice for the research prototype implementation.

In addition to that, the powerful development platform, Visual Studio .Net 2005 and Mobile Direct3D can help speed up the development of the final prototype.

4.2 Prototype Introduction

The final high-fidelity prototype is a 3D FPS game-like application called *InteractionPro*, which is implemented specifically for the Dell x51v. It was developed in Visual Studio .Net 2005 (C#). .Net Compact Framework v2. It uses Mobile Direct3D as the 3D API.

The aim of this prototype was to evaluate the novel interface design of FPS games on PDA devices. The approach is to compare the results of two different interaction styles by implementing them through a series of stages. These two different interactions modes are:

Interaction A ('Doom' mode). This mode uses a traditional button pressing interaction style.

Interaction B ('Gesture' mode). This mode uses the newly designed interaction style which involves 'Stroke Drawing' and 'Stroke Recognition' and an optimized set of commands.

Once the application launches, users have to choose one of the two interaction styles. The selected style will be used in all stages of the game (a user cannot switch between the styles once the game starts).

There are a total of four stages in this application. These four stages are designed for the purpose of testing different interaction aspects, specifically: 'Moving and Jumping', 'Shooting', 'Moving and Shooting' and 'Function Execution'. Objective data that logged all the tester's movements whilst running the application was recorded into an XML file. The data recorded includes the time the user spent on the different stages, the number of times each button was pressed, how long each key was held and the stylus movement rate which logs the pixel distance of the stylus pen on the touch-screen at intervals of 50ms (pixel difference/50ms). The recorded objective data is used for evaluation and validation.

The mini 3D game engine involved in this application was created from scratch. Unlike traditional game engines, which are trying to make the game attractive and fun, the aim of this mini 3D game engine is to help investigate the new interaction system in a fast and effective way, which includes:

- Fast and effective prototype development according to the research schedule
- Meeting the requirement of the interaction evaluation

Therefore, this mini engine consists only of three components: graphics, logic control and interaction control as showed in Figure 11.

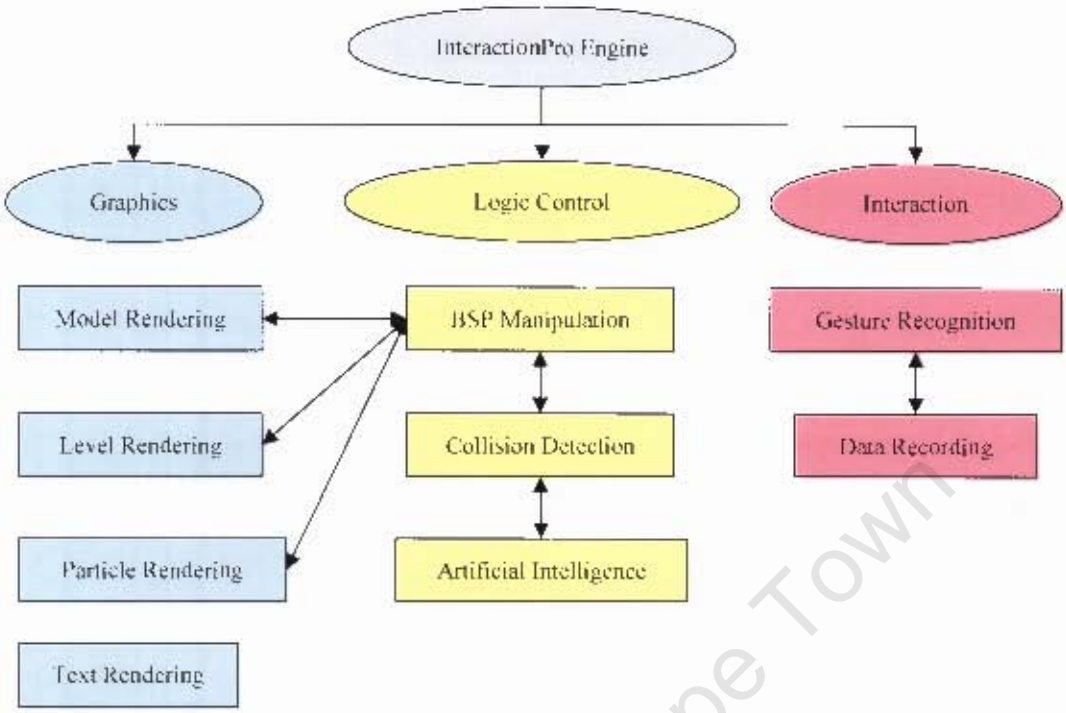


Figure 11. The game engine of *InteractionPro*

As the purpose of this prototype is different from any other FPS games, this mini 3D engine is not based on any existing FPS game engine. To achieve the main tasks of 3D scene generation, rendering and collision detection, the BSP map format, which is used in Quake III, is implemented in this prototype.

Therefore, the most challenging task in the development of this prototype, is the use, for the first time, of C# and Compact .Net framework to implement the rendering of BSP map on a Windows Mobile platform.

The process of implementing the prototype took six months.

4.3 Interaction Design of the PDA prototype

4.3.1 Interaction A

Interaction A ('Doom' mode) basically uses the traditional button pressing style. Figure 12 shows the distribution of the function commands on the PDA device.

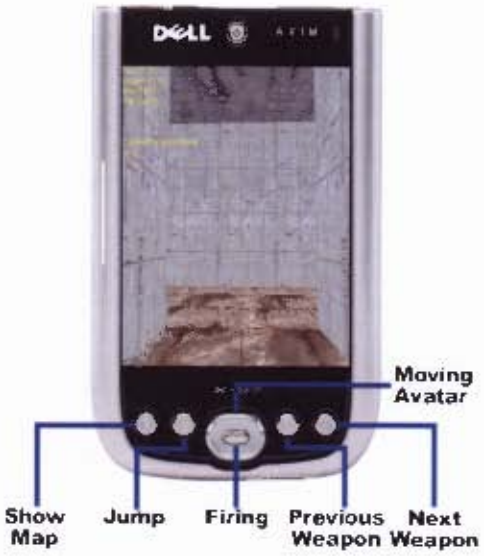


Figure 12. Command distribution of interaction A on the PDA device.

- The avatar's movement is controlled by the Directional Pad (D-Pad) with up and down moving the avatar forwards and backwards, and left and right moving it from side to side.

- The 'Firing' button is located in the middle of the D-Pad. This is because the distance between 'Firing' and 'Moving' buttons should be the shortest as it is the second most frequently used command (as learned from the Chapter 3).

- The 'Jump' button is on the left and next to the D-pad. The direction of the 'Jump' is directly upwards. If the avatar is intended to jump forward, the 'Jump' action needs to be executed in combination with 'Moving Forward'. This characteristic shows that 'Jump' button should also be close to the directional buttons. For most left-handed people, it is easier to press the buttons on the left side to the D-Pad than those on the right side, since, while the PDA device is held by the left hand, the distance between the left thumb to the left side buttons is shorter than for buttons on the right side.

- The 'Map View' button is on the far left.

- The buttons for 'Previous Weapon' and the 'Next Weapon' are on the right of the D-Pad.

- Camera view is controlled by the stylus pen. The avatar's view follows the movement of the stylus pen (e.g., the view turns to the left when the stylus pen slides to

the left on the screen). It leads the way for where the avatar is facing and moving. The range from 'Look Up' to the 'Look Down' is limited to -1.5 to 1.5 radians. This range prevents the confusion of avatars seeming to be upside-down.

4.3.2 Interaction B

Interaction B ('Gesture' mode) uses the newly designed interface for 'stroke drawing'. Figure 13 shows the strokes of the commands.

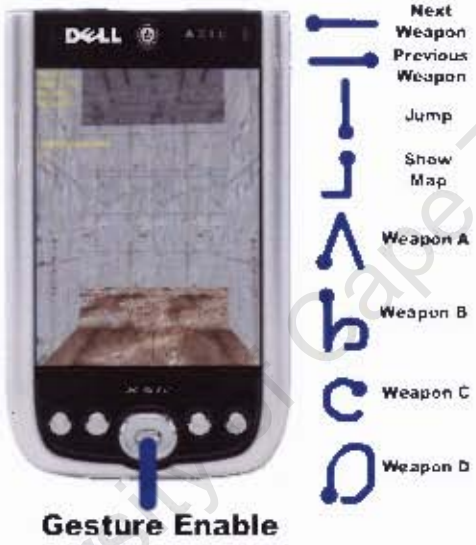


Figure 13. The gesture control for interaction B.
The track of the strokes on the right start from the bold dot.

This interaction style is basically the same as the one developed for the flash prototype. Camera control with the stylus pen is the same as in interaction B. When the 'Gesture Enable' button (the 'Return' button) is pressed and held, strokes can be drawn on the whole screen. The recognition of the stroke and the execution of commands occurs immediately once the stylus pen is removed from the screen in the 'Gesture Enable' mode.

'Auto aiming & firing' and 'Manual Shoot' are implemented as the 'Shooting' mechanism in this interaction style, again in a similar fashion to the mechanism in the Flash prototype.

- 'Manual Shoot' is triggered by double tapping anywhere on the screen. The bullet

is emitted from the center of the screen.

- 'Auto Aim & Shoot' takes effect when the stylus pen is tapped on a drone in the scene and remains in effect as long as the stylus touches the screen. The center of the camera view pans to the tapped position and then starts shooting when the panning is finished.

Only one button, the 'Return' button, is used in the new interaction style. Although the other four buttons can be assigned 'Hotkey' functions and this can greatly enhance the efficiency of this interaction style, they remain null so as to force users to employ the 'Stroke Drawing' method to accomplish the mission, ensuring validity of the investigation of the newly designed interaction system.

4.4 Design of Stages

As has been described above, the aim of this application was to evaluate the novel interface design for FPS games on PDA devices. Therefore, design of the different stages should test all the interaction commands comprehensively.

Previous research in Chapter 3 revealed problems with traditional 'button pressing' interaction, such as failure to execute large number of commands, inability to execute multiple simultaneous commands effectively (e.g., 'Firing' and 'Moving'), etc. New problems that might occur in the novel interaction system were also raised during the investigation of the low-fidelity prototype. Finding the answers to these problems is crucial to the evaluation of the new interaction. Based on understanding the FPS commands features which were derived from the research of the *InteractionLog* and Flash prototype, action commands are divided into the following categories.

- Moving and Looking
- Stationary Shooting
- Moving and Shooting
- Function Execution

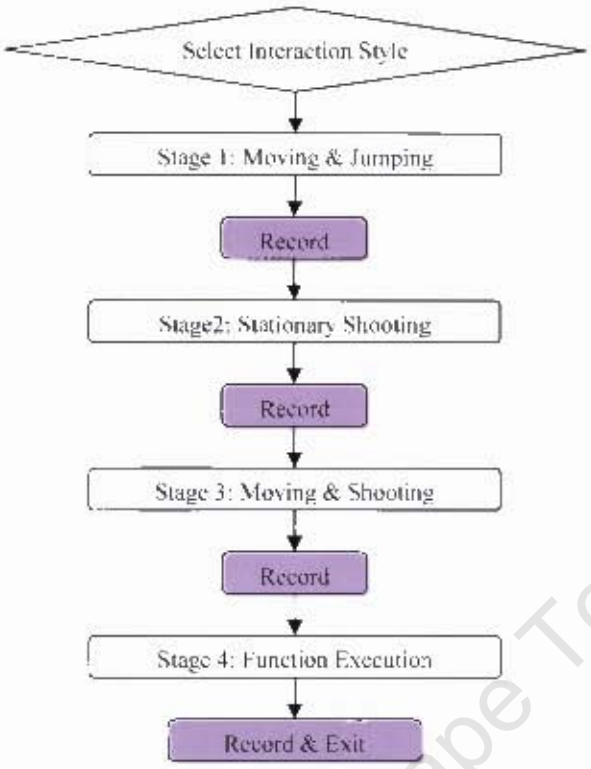


Figure 14. The flow of the different stages.

Once the application is launched, a dialogue box showing 'select interaction style A?' pops up. This indicates which interaction the tester is going to use in subsequent stages. 'Interaction A' refers to 'Doom' mode and 'B' refers to 'Gesture' mode.

Table 2. The explanation of the Interaction Code.

Interaction Code	Mode	Mechanism
A	Doom	Button Pressing
B	Gesture	Stroke Drawing

This application uses code 'A or B' instead of the name of the interaction style or mechanism throughout all the stages. This is intended to avoid prejudicing the evaluation by preventing the testers from having any preconceptions about which is the newer interface. In particular, for FPS experts, the word 'Gesture' is definitely more modern than the word 'Doom'.

After an interaction style is chosen, the game starts.

Stage 1: Moving and Looking

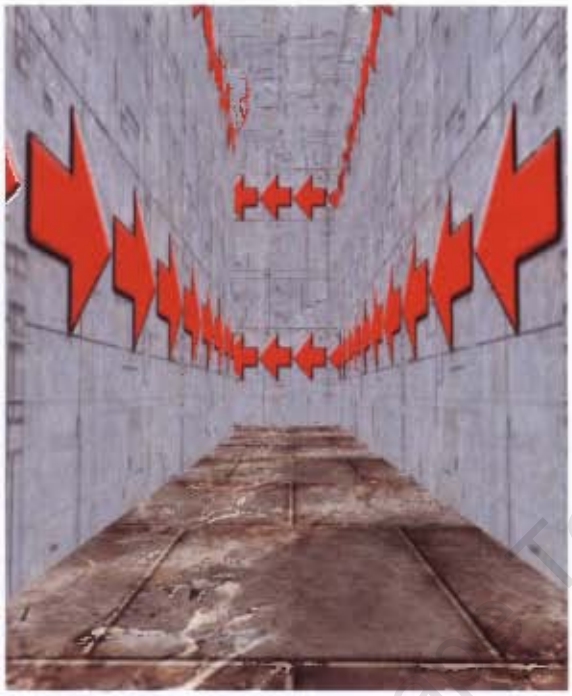


Figure 15. The first-person view in the aisle

In the first stage, the most basic elements in FPS games, 'Moving, Looking and Jumping', are implemented. The aim of this stage is to test how the stylus mechanism handles basic functions.

The stylus mechanism uses the left thumb to control the movement of the avatar by pressing the directional buttons; meanwhile the right hand manages the heading (Camera View Changing) by sliding the stylus pen on the touch screen. When the user intends to make a left turn, he must press the 'Move Forward' button and slide the stylus pen to the left at the same time instead of pressing the 'left' directional button only. This is because the 'Left' direction button is designed for 'Shifting to the left', not for 'Turning to the left'. This mechanism is used in both interaction styles, so it is very important to make sure that this mechanism works efficiently on PDA devices.

Figure 16. The structure of the map in this stage. The start point is on the bottom right and the termination is on the top left, in the wooden section.

The task of this stage is to walk through a long series of corridors by following the red arrows indicated on the wall, jump over two bumps and arrive at the terminus as soon as possible. As figure 16 shows, the corridor includes 90 degree turns (left/right), 180 degree turns (left/right) and 'W' bends. There are also two bumps in the route that require the testers to execute a 'Jump' command. The method of triggering a 'Jump' command is different in each interaction system, and demonstrates the characteristics of the corresponding style. 'Jumping' in interaction A is executed by pressing the 'Return' button, which is located in the middle of the directional buttons. 'Jumping' in interaction B is triggered by drawing a stroke from bottom to top once the 'Return' button is held.

In fact, the 'Jump' command represents the only difference between the two interaction styles in this stage. Therefore, besides the evaluation of 'Moving and Looking', this stage also compares the features and differences of the 'Jumping' implementation in the common game-play context.

The time that testers spent on finishing the route (the arrival at the terminal) was recorded.

Stage 2: Stationary Shooting

Figure 17. Stationary shooting of white cubes

A 'Shooting' action consists of 'Camera View Changing', 'Aiming Adjustment' and 'Firing'. The aim of this stage is to compare these actions, which are very different in the two interaction styles. The best approach for a test that focuses on these basic actions is to keep the avatar stationary. Testers were expected to be very familiar with these actions using the traditional 'Button Pressing' style. In this style, 'Firing' is triggered by pressing the 'Firing' button and 'Aiming' and 'Look Around' are handled by the stylus pen, which is similar to the use of the mouse on desktop PCs. In the new interaction style, these three actions are coordinated only by the stylus pen.

There was a debate in the beginning of designing the shooting mechanism about how to implement the 'shooting' action. There was some suggestion that the action should be implemented using the 'Whacking Gnome' method directly, which can help to focus the study on the basic shooting mechanism, although it is rather simplistic. 'Whacking Gnome' is a popular arcade game. The player uses a hammer to whack on the gnome's heads that pop out randomly from the machine. With regard to FPS games, it means once the user taps on to the target, the target is considered to have been shot. Another suggestion was to create an ammunition system, with Different ammunition having different speeds. The drone is considered to be shot only when it collides with the

ammunition. This means that even if a user actually taps on the target and triggers firing, the ammunition may still miss the target due to the delayed arrival. Users need to predict the direction and the speed of the moving objects so as to collide the ammunition correctly.

Most common FPS games on desktops such as Counter Strike, Doom and Quake have their own implementation of 'shooting'. For instance, in Counter Strike, an entity is divided into several parts with specific weight. The weight of the part will be lost once this part has been shot. When the total weight drops to a certain level, the entity is considered dead. This shows the trend of modern FPS games that 'Shooting' should not be as simple and straightforward as 'Whacking Gnomes'.

Therefore, the second shooting mechanism is chosen in this study. This is because the new designed interaction system is intended to be used in a real game environment, not only for testing its performance on the basics. The result of 'Shooting' test can only be considered effective and valid in a relatively real gaming context.

In this stage, there are ten 'Drones' (white cubes) randomly floating in front of the avatar. Once one of these 'Drones' gets shot, it respawns in another randomly generated position in front of the avatar. Users have to shoot twenty drones to accomplish this stage.

While the 'Gesture' interaction style is selected, the user can choose to use the 'AutoAim&Shooting' or 'Manual Shoot' to destroy the drones. Both of these shooting methods were introduced and explained to subjects before beginning, and they had to practice both of these shooting methods in a practice round of running the application under inspection. When it comes to the formal round, users were free to choose the shooting mode based on their own interests.

The time that testers took to eliminate the required numbers of the drones was recorded.

Stage 3: Moving and Shooting

Figure 18. The first-person view in stage 3.

Situations in which players need to dodge and shoot simultaneously happen very often in most FPS games. The design of this stage is intended to simulate this kind of situation. The actions taken in such situations are very intensive. Good coordination between 'Moving', 'Aiming' and 'Firing' are essential. The main goal of this stage is test the performance of the new interaction system in such intensive situations.

This stage consists of a closed cuboid room, as shown in Figure 19. This room is divided into two sections by a wall with a hole in the middle. There are in total six drones (white cubes) in the larger section. There are four stationary drones at the corners and two drones floating around randomly. The red cube in the plot represents the starting position of the avatar. Once the avatar starts moving, the player is not allowed to stop for more than two seconds; otherwise, the player will be respawned at the starting position. The wall in this room blocks the avatar's direct view of the drones from the starting position. This requires that the user move, changing this stage from a 'Stationary Shooting' stage to a 'Moving and Shooting' stage.

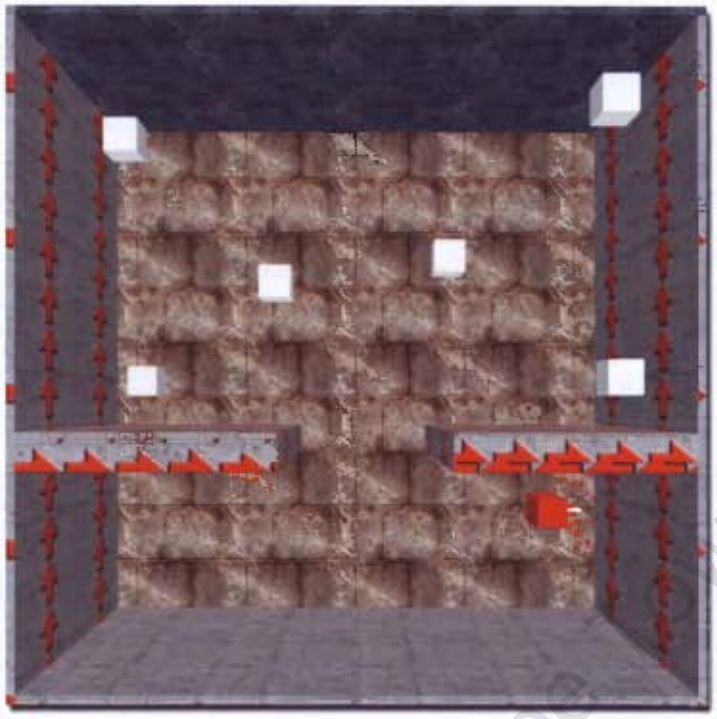


Figure 19. A top-down view of stage 3. The red cube represents the avatar's starting position. The white cubes are the drones which are supposed to be eliminated.

The time that testers took to eliminate all the targets and the number of times the avatar was respawned was recorded.

Stage 4: Function Execution

In addition to the basic 'Shooting' and 'Walking' actions, the interactions of advanced commands such as 'Weapon Changing', 'Events Handling', 'Menu View', 'Map View', etc.. are also part of the commands in FPS games. Although these commands are less used than the basic commands, according to the analysis in the *InteractionLog* section, they are necessary and crucial for enhancing the game usability and playability (e.g., 'Menu' is used to start the game).

In order to prevent the results from being affected by advanced commands, task design of the previous stages focused only on the commands which were to be evaluated. The evaluation of the advanced function commands is different from such basic function commands.

'Gesture' mode can be used to execute many more function commands than the 'Doom' mode, due to the unlimited pattern of strokes. On the other hand, the efficiency of

button pressing is definitely much faster than stroke drawing, due to the extra time required to draw a stroke. Irrespective of the speed or the number of commands the interaction style is capable of, the ultimate purpose of this evaluation in PDA games is to find out whether the players like and accept the new interaction style while playing the game. Therefore, the aim of this stage is to test the performance of executing these functions in a proper FPS game context.

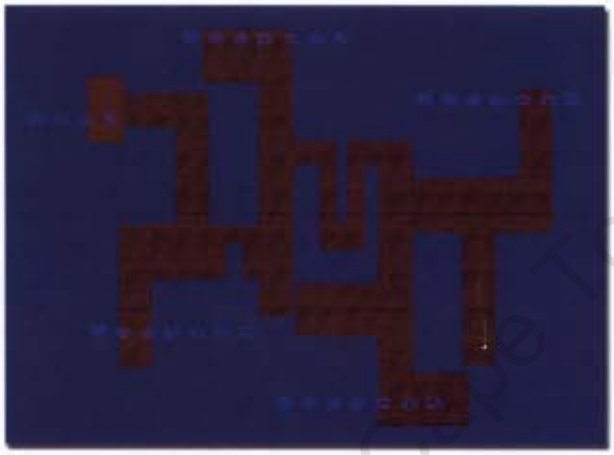


Figure 20. A top-down view of the maze in stage 4. The white spot and green line on the right bottom represent the avatar's current position (starting position) and heading direction.

Based on this, a maze was created for this stage, as shown in Figure 20. Users have to find their way to the 'Exit' and kill all the drones (each must be killed with a specific different weapon). Users can get help information by executing the 'Map View' command to see a top-down view of the maze. The map shows the structure of the maze, the location of the 'Exit', 'Avatar', 'Drones' and the type of the weapon to use for killing the drones.

Since the number of physical buttons on the PDA x51v is very limited, only four advanced function commands are implemented in this stage, which are 'Map View', 'Next Weapon', 'Previous Weapon' and 'Jump'. Besides this, in 'Gesture' mode, users can draw the strokes 'A','B','C' and 'D' in a graffiti style to specify which weapon to use directly instead of pressing the 'Next' or 'Previous' weapon multiple times.

The time that users spent in this stage was recorded. Their subjective feelings towards the interaction systems were captured by a questionnaire (See Appendix A) and explained during an interview at the end.

4.5 Summary

With the help of 3D accelerator chipsets, the PDA Dell x51v is able to render simple 3D scenes smoothly. A high-fidelity prototype was implemented on this device. The prototype allows users to run four stages of an FPS game using two different interaction styles. One of these interaction styles uses conventional button pressing, while the other uses the newly designed gesture style. The function commands from the four categories discussed in Chapter 3 should be tested comprehensively. The prototype was designed with this in mind; the four stages, which are 'Moving and Jumping', 'Stationary Shooting', 'Moving and Shooting' and 'Function Execution', involve performing all these commands in different situations.

5 Implementation

This chapter discusses the implementation of the prototype. The main algorithms and technologies used in this prototype are BSP Trees, Collision Detection, Stroke Recognition and *InteractionAnalysor*. Although these algorithms and technologies are not the objective of this study, they are essential parts of the prototype implementation.

5.1 Binary Space Partitioning (BSP) Tree Introduction

Although PDA hardware has developed rapidly, its performance is still very limited compared to desktop PCs. A proper level in 3D games consists of thousands even millions of arbitrarily oriented polygons [18]. To render all these triangles is infeasible on the current generation of PDA devices. The Binary Space Partitioning (BSP) algorithm is the key to solve this problem in this prototype.

The BSP tree is an efficient and popular algorithm for creating 3D scene. This algorithm was originally proposed by Fuchs et al. [19]. The goal is to solve the problem of hidden surfaces so as to draw a 3D scene more quickly. It recursively partitions the 3D scene into a binary tree hierarchy. The subdivided space information is recorded in the tree nodes. These nodes and leaves can be traversed during real-time rendering to decide whether the polygons in the partition of the traversed nodes or leafs should be drawn.

A reasonably balanced BSP tree can greatly improve rendering efficiency [18]. Nowadays, BSP tree has been used widely in 3D games, not only to speed up rendering, but also for the implementation of effective ray tracing and collision detection.

5.2 Implementation of BSP in *InteractionPro*

The BSP module from Quake III was ported to *InteractionPro*. All the maps in the stages were created from the scratch in Q3Radient by Id software. This level design application also helps to compile the model of the map into a BSP tree, which is a format that Quake III can read. Quake III inserts each polygon of the map into the BSP tree. Each polygon divides the space in which it is located into two regions. A different order of inserting the polygons, which implies splitting up the space differently, has an impact on the efficiency of the nodes and children traversal. The basic rule that Quake III follows is to set a really high priority on axial polygons. The axial polygons are the polygons whose normal vectors are aligned with one of the axes. Then, it will find planes that will split up the remaining uninserted polygons the least [20, 21, 22, 23]. For instance, Figure 22 shows the subdivision of the maze map in Stage 4.

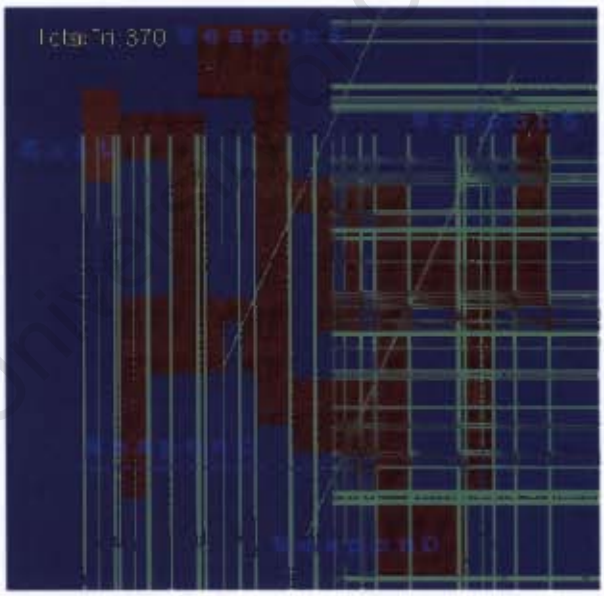


Figure 22. The BSP subdivision of the maze map. Green lines represent BSP splitting planes.

As *InteractionPro* uses Mobile Direct3D and is written in C#, the module for loading and rendering the BSP map was rewritten. Although the Quake III uses OpenGL and was written in C++, the basic method is similar since the structure of the BSP map is the same according to the Unofficial Quake 3 Map Specs by Proudfoot. The structure of the BSP file contains the 17 categories (Lump) of the map data. See Table 3.

Table 3. The structure of the BSP file in *InteractionPro*

Index	Lump Name	Description
0	Entities	Objects such as md3 model file
1	Textures	Texture Files
2	Planes	All the planes in the map
3	Nodes	Nodes of BSP tree
4	Leafs	Leafs of BSP tree
5	Leaf faces	Lists of face indices (one list per leaf)
6	Leaf brushes	Lists of brush indices (one list per leaf)
7	Models	Rigid world geometry
8	Brushes	Polyhedra refer to the solid space
9	Brush sides	Surfaces of the Brush
10	Vertexes	Vertices in respect to the faces
11	Mesh verts	Lists of vertices offsets (one list per mesh)
12	Effects	List of special effects
13	Faces	Surface geometry
14	Light maps	light map information
15	Light vols	Local illumination information
16	Visible Data	Cluster-cluster visibility data

One of the most helpful lumps in the structure is the last: 'Visible Data'. 'Visible Data' indicates which area of the map is visible, given the avatar position. Thus, the prototype only needs to render the visible polygons, which greatly reduces the rendering burden and enhances rendering speed. For example, there are in total 370 triangles in the maze map; Table 4 shows that only 81 triangles on average are actually rendered while the avatar is walking in the maze, which reduces rendering overheads by 78.1%. The Number of triangles drawn varies from region to region.

Table 4. Different regions of the maze (Light Green area) are rendered when the Avatar (Red Spot) is in different places.

		
Start position (58 triangles)	Weapon D area (78 triangles)	V tunnel (60 triangles)
		
Weapon A area (72 triangles)	N tunnel (116 triangles)	Exit (104 triangles)

Effective collision detection is also aided by the BSP tree. According to the Quake 3 BSP Collision Detection by Ostgard, each leaf of the Quake III BSP tree has a set of brushes which are associated with a set of brush planes. These brush planes are actually the walls, ceilings and floors that the collision detection is performed against in *InteractionPro*.

Each node in the BSP tree contains a dividing plane, which recursively splits the space until it results in a leaf with no children. Once it traverses to this node, if it is found that the checked line segment is not in this region (node), it means there is no plane in this region (all of its sub nodes) that would collide with the line segment. Then the line segment being checked can traverse to the sibling of the node. Therefore, instead of the collision detection with all the planes in the scene, the BSP tree decreases the number of the planes which need to be detected dramatically.

5.3 Tracing in Collision Detection

The motion of a moving object is simulated by rendering the object in two different positions between two adjacent frames according to its velocity. The faster the object moves, the larger difference there is between these two positions. If there is a very thin wall (the depth of all the walls in the *InteractionPro* is set to zero) in the way of the moving object, it is difficult to find the exact collision point between the moving object and the wall [24]. Tracking technology is implemented to solve this problem.

The trace function in collision detection takes several arguments, which are the starting point for the line segment, the ending point of the segment, and a bounding box around of the object which moves in the map

The distances between the starting and ending point to the plane are recorded when the trace function checks each leaf it encounters. This can shows whether the line segment is in the front of the plane or behind the plane. If the line segment splits a plane, the closest point of collision along the line to a given precision can be found by recursively dividing the line segment in half where the collision point is located.

5.4 Stroke Recognition in 3D scenes

The Graffiti handwriting system, classified as 'easy' and 'immediate' in terms of usability [25, 26], has been widely used on current PDA devices. *Gesture Interaction* used in *InteractionPro* was developed based on the graffiti recognition system [27, 28].

The basic principle for this recognition system is that each stroke is normalized by being sampled into a unit square with a constant number of interpolations. Then the drawing stroke is compared to each reference stroke, by computing the distance between each differential point on the input stroke to the reference point on the normalized stroke patterns. The reference stroke that is found to be the closest is the match. Finally, a refinement measure is applied to distinguish the difference between multiple similar letters, such as 'L' and 'h' or 'D' and 'P'.

The main difference between these interaction systems is that the newly designed system is intended to work in a 3D environment.

Instead of using Windows GDI, strokes represented in *InteractionPro* are drawn as 3D line segments. Once the drawing mode is enabled, multiple vertices are generated following the track of the stylus movement using the Mobile Direct3D API. From study of the Dell x51v device, it was found that the vertex buffer in video memory is not supported. This indicates that generating vertices during scene rendering will affect the speed seriously, and this was confirmed during debugging of the prototype. Hence, the number of vertices that represent the drawing strokes is limited to 100. Although the length of the drawing strokes is no longer infinite, it is sufficient for the strokes used in the prototype. In the meantime, the cost of the rendering performance is the minimised.

5.5 *InteractionAnalysor*

InteractionAnalysor, See Appendix B, is a .net windows application, which is developed based on the *InteractionLog*. Although it is not part of *InteractionPro*, which is a 3D prototype on the mobile device, it is a module crucial to the implementation of the research experiment, as it is designed specially for visualizing and comparing the results from the *InteractionPro* in a convenient and intuitive way.

Each tester ran the prototype four times in total, two practise rounds and two formal rounds for each interaction mode. The prototype generates four XML files for each stage after one round of running the application, so that each tester generates 16 XML files in total. Thus, there were 160 XML files for the ten testers in the end. Each plot drawn from an XML file contains hundreds of interaction events in a period of more than 40 seconds. Therefore, investigating and comparing the plots one by one is extremely hard and inconvenient.

InteractionAnalysor was developed to solve this problem. Since the comparison between two interaction styles is the point of the investigation. This application automatically tiles two visualized results, which are from 'Doom' mode and 'Gesture'

mode, in a horizontal pattern. These two plots are from the same stage and round. Before clicking on the 'Show' button, the only variables that need to be defined are the name of the tester, the stage number and the type of round (Practise or Formal). In addition to this, the length of the visualized plot can be scaled freely. The 'OverAll' slide bar can move both plots to the desired section simultaneously.

Therefore, details of what the interaction events were happening can be viewed more clearly and vividly. And the comparisons between the two interaction styles can be managed more effectively and objectively.

5.6 Summary

The BSP Tree data structure successfully implemented in the *InteractionPro* is a fundamental technology for this prototype. It not only greatly optimizes the 3D rendering speed, helping the Dell x51v render a scene that contains more than 300 polygons smoothly, but also enables the execution of effective collision detection. A traditional stroke recognition system, which works in a 2D environment, has also been converted to be able to work in the 3D environment. With the help of the *InteractionAnalysor*, comparison between the two interaction styles can be managed and investigated more effectively and objectively.

6 Evaluation

This chapter describes the implementation of the evaluation and discusses the results from the experiment. A case study approach, similar to Heuristic Evaluation, is adopted and the low number of tester's makes it infeasible to perform statistical tests. Nevertheless, the results are sufficient to show that testers are satisfied with the new designed interaction system, which is the ultimate goal of this experiment.

6.1 Background on HCI Evaluation Methodologies

Heuristics [9, 10, 11, 12] are design guidelines for evaluating the usability of interfaces and have been widely used for internet and software development. Their purpose is to make the software interface easier to learn, use and master. During the heuristic evaluation, the user interface is tested by HCI experts. The feedback from the usability heuristics is assessed. Any usability problems confusing users and making interaction with the interface difficult, are recorded. The usability problems found by Nielsen in his experiments, commonly relate to consistency and misleading information, etc.

Experience in game play is different from traditional software. As the design goals for games are usually characterised as "easy to learn, difficult to master" [29], it is necessary to go beyond the basic interface usability, in order to evaluate additional properties of the game, including gameplay, story, and mechanics.

It is for this reason that Heather Desurvire [13] has introduced the Heuristic Evaluation for Playability (HEP). It is a comprehensive set of heuristics for playability, specifically tailored to evaluate video, computer and board games. However, interaction design in mobile games is also different from desktop game interface design. Korhonen and Kovisto [4] proposed guidelines for the heuristic evaluation of mobile games. Figure 33 shows the three groups of guidelines categorised by them, which are the heuristics for evaluating game usability, mobility and gameplay.

No.	Game Usability Heuristics
GU1	Audio-visual representation supports the game
GU2	Screen layout is efficient and visually pleasing
GU3	Device UI and game UI are used for their own purposes
GU4	Indicators are visible
GU5	The player understands the terminology
GU6	Navigation is consistent, logical, and minimalist
GU7	Control keys are consistent and follow standard conventions
GU8	Game controls are convenient and flexible
GU9	The game gives feedback on the player's actions
GU10	The player cannot make irreversible errors
GU11	The player does not have to memorize things unnecessarily
GU12	The game contains help

No.	Gameplay Heuristics
GP1	The game provides clear goals or supports player-created goals
GP2	The player sees the progress in the game and can compare the results
GP3	The players are rewarded and rewards are meaningful
GP4	The player is in control
GP5	Challenge, strategy, and pace are in balance
GP6	The first-time experience is encouraging
GP7	The game story supports the gameplay and is meaningful
GP8	There are no repetitive or boring tasks
GP9	The players can express themselves
GP10	The game supports different playing styles
GP11	The game does not stagnate
GP12	The game is consistent
GP13	The game uses orthogonal unit differentiation*
GP 14	The player does not lose any hard-won possessions

No.	Mobility Heuristics
MO1	The game and play sessions can be started quickly
MO2	The game accommodates with the surroundings
MO3	Interruptions are handled reasonably

Figures 33: Heuristics for Mobile Games

Nevertheless, this research does not focus on the playability of an entire mobile game,

but on the evaluation of the feedback for the new proposed interaction design, especially for FPS games on PDA devices. Thus the two guidelines from Korhonen and Kovisto's research for heuristic evaluation of "Game Usability" were chosen for application to this research. They are:

GU6: "Navigation is consistent, logical and minimalist"

GU8: "Game controls are convenient and flexible"

Furthermore, based on the additional properties for games as mentioned above, two other guidelines were also used to conduct the evaluation:

I. The level of challenge and entertainment that the user obtains from the new interaction

II. The level of the user's overall satisfaction with the interaction.

These criteria were added since the ultimate goal of playing games is to be challenged and have fun, which means the process of achieving tasks must not be too straight forward [4].

6.2 Evaluation Implementation

Figure 23. A tester is running the prototype.

Nielsen and Landauer [10] describe the relationships between the number of the evaluators and the usability problems found in an interface design using heuristic evaluation, as shown in Figure 24.

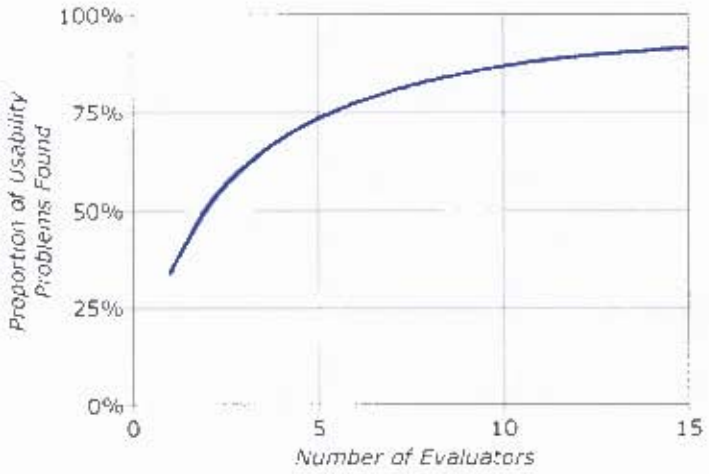


Figure 24. Curve showing the proportion of usability problems in an interface found by heuristic evaluation using various numbers of evaluators. (by Jakob Nielson)

It seems that five evaluators are able to produce sufficient results. Unfortunately, PDA FPS games, with the exception of the transplanted Quake3 and Doom games, do not currently exist. This makes it impossible to find HCI Experts in playing FPS games on Mobile devices. Therefore, experts in Mobile HCI, FPS gamers and two amateurs, with ten evaluators in total, performed the Expert Evaluation. The main goal is to find the degree of acceptance of the new interaction style by the players.

These ten participators were divided into three groups:

-Experts in HCI (4 testers): Those who are familiar with the stylus pen interaction on PDAs.

-FPS gamers (4 testers): Those who are familiar with the interaction of FPS games.

-Amateurs (2 testers): Those who have little or no experience with PDAs and FPS games.

Experiments were run individually. There were seven sessions during each experiment which took around one hour on average.

These sessions were:

1. Introducing the experiment and the interaction styles.

2. Running the application as a practice round using Interaction A (or B, depending on the previous tester's choice) to familiarize the tester with the application and the interaction.

3. Running the application as a formal round using the same interaction mode.

4. Running the application as a practice round using another interaction mode.

5. Running the application as a formal round using another interaction mode.

6. Answering the questionnaire.

7. Interviewing the tester so as to get extra feedback.

The experiment equipments were:

1. A Dell x51v PDA device

2. Two Video Cameras: One video camera was used for recording the tester's hands motions while interacting with the prototype on the PDA device. It is placed above the top of the tester's head and aimed down at the PDA device. The other camera recorded the whole scene of the experiment from the corner of the room.

6.3 Evaluation of Results

6.3.1 Results Introduction

Four categories of data were collected from the experiments:

1. The *Interactionlogs* containing all the objective data of the interaction events during the experimentation.

2. The questionnaires representing the subjective feelings towards the new interaction from the players.

3. A memo of the interview with the testers after the experiments.

4. Video tapes of the experiments

The information on interaction events was compiled to an XML file and imported into a database. The *InteractionAnalysor* application was then used to visualize the results.

Table 5. Table of the Objective Results (Time spent on all the stages)

Stage	Interaction	Respawns	Total Time(s)	Average of Stylus Samples	Average of Stylus Rate(pd/50ms)
1	DOOM(A)		34.52	365.125	3.65
	GESTURE(B)		42.45	349.5	3.49
2	DOOM(A)		32.21	445.125	4.45
	GESTURE(B)		42.12	446.25	4.46
3	DOOM(A)	2.25	44.42	478.25	4.78
	GESTURE(B)	1	37.41	366.875	3.66
4	DOOM(A)		126.31	808.5	8.08
	GESTURE(B)		187.62	996.75	9.96

Table 6. Table of the Objective Results (Key Pressing and Stylus Tapping times in all stages)

Stage	Interaction	StylusTap	KUp	KDown	KLeft	KRight	KReturn
1	DOOM(A)	18.875	14.375	0.25	1.625	1.75	0
	GESTURE(B)	33.75	16.5	0.125	1	2.25	5.375
2	DOOM(A)	3.25	0.25	0	0	0	18.63
	GESTURE(B)	36.125	0	0	0	0	0
3	DOOM(A)	26.75	28.875	13.375	9.75	7	23
	GESTURE(B)	36.5	19.5	10.875	6.125	3.375	0.875
4	DOOM(A)	65.375	40.125	3.375	2.75	5.875	6.875
	GESTURE(B)	135.13	46.875	4.75	6.75	6.375	38.25

6.3.2 Stage Results

Stage1: Moving and Jumping

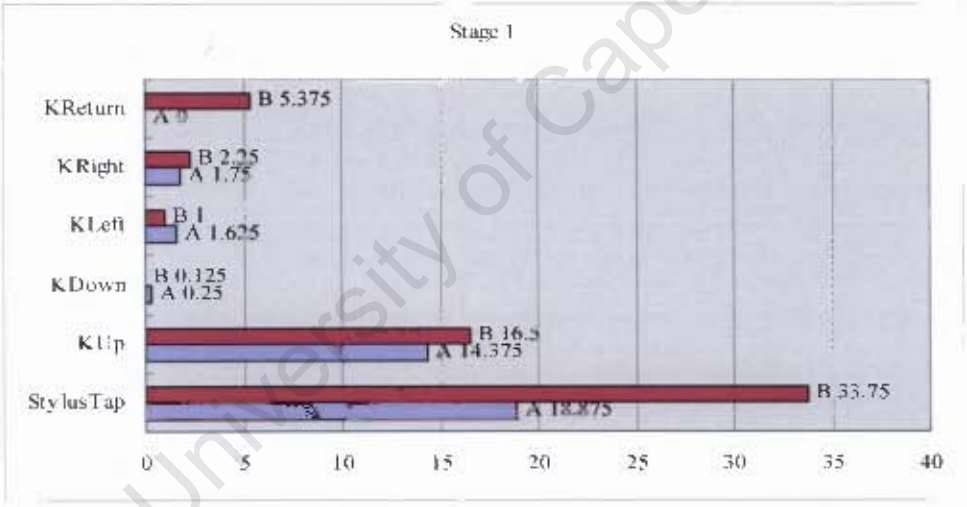


Figure 25. The times for key pressing and stylus tapping in Stage 1

As expected, Figure 25 shows that the stylus pen was tapped using Interaction B more than using interaction A. This is because the 'Jumping' command in Interaction B is executed by drawing strokes, which takes longer and requires more tapping than pressing the buttons. Therefore, it is understandable that in 'Gesture' mode, testers took on average eight seconds more than in 'Doom' mode, as shown in Table 5. However, the main concern in the evaluation of games is the user's subjective feeling.

The results of the questionnaire (see Figure 26) show that the slight delay did not affect

the sense of presence[3] for the experts triggering the 'Jump' command using 'Gesture' mode compare to 'Doom' mode in this stage. However, both scores in this question are fairly low indicating testers are not pleased with triggering 'Jumping' in this stage. This shows that most testers have difficulties in executing the 'Jump' command smoothly and effectively.

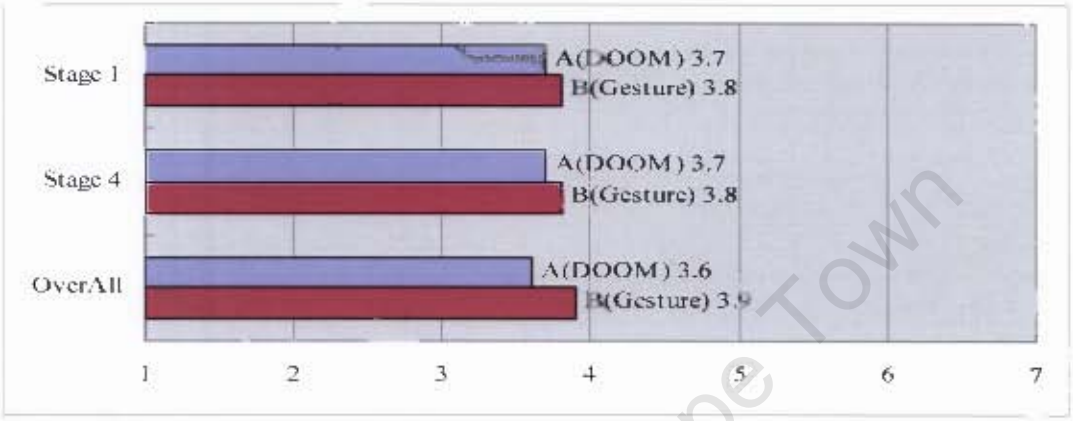


Figure 26. The score of satisfaction with 'Jumping' in all stages

As 'Jumping' is frequently used in combination with other movement commands such as walking and looking. This highly related combination resulted in testers spending a long time practicing it.

In Interaction 'A' mode, the tester executes the 'Jump' by pressing the button left of the D-Pad. Once the avatar is in the air, the tester has to move the left thumb from the 'Jump' button to the D-Pad immediately to press the 'Forward' button before the avatar falls back to the ground. Some of the testers withdrew their stylus pen from the PDA's screen so as to use their index finger to press 'Move Forward' on the D-pad. They re-engage the stylus pen after the 'Jump Forward' motion is finished. During the interview after the experiments, they explained that this way of 'Jumping Forward' is a habit. This habit is hard to change in a short period. This mechanism of 'Jumping' is uncomfortable for the users in interaction 'A'.

In Interaction 'B', 'Jumping' is executed by holding the middle button ('Return') down, then drawing a straight upwards stroke anywhere on the screen. The 'Return' button

[3] Presence [15, 16, 17] is the term used to represent a level of player's immersion in a virtual reality system.

must be released once the avatar is in the air, so as to press the 'Forward' button to jump forward instead of jumping straight up. Testers were not familiar with this new mechanism. Some of them became irritated after failing to execute the 'Jump' command after several attempts, which is confirmed by the recorded video clips.

Figure 26 shows that there is little difference between 'Gesture' and 'Doom' mode, but clearly that the users were not happy with 'Jumping' in either mode.

The main reason behind the users dislike of the 'Jumping' command in 'Gesture' mode is because of the lack of practice and unfamiliarity with the way it is performed. This is inferred from the following evidence:

- 'Jump' is the only command that needs 'stroke drawing' in stage 1,
- user's average score is '5.3' on the scale of 1 to 7 from the questionnaire: 'Please rate your sense of being satisfied at drawing strokes in stage 1', showing that drawing strokes is not the barrier to triggering the 'Jump' command.

Some of the testers blamed the failure of executing the command on the stroke recognition system. However the system inspector's observation on these testers interaction from the video clips and system's logs showed that the stroke recognition system was indeed working correctly. What happened was that the testers were failing to execute the command in the correct way. Some of them released the middle button before finishing strokes; some of them removed the stylus pen from the screen and released the middle button at the same time. All of these testers had trouble coordinating the rhythm of using the middle button and stylus pen together to accomplish the 'Jumping' action. Stroke recognition is expected not to be a problem once they practice enough.

Stage2: Stationary Shooting

All the testers accomplished this stage easily. Although the new shooting style takes longer than using the 'DOOM' game style as is shown in Table 5, it offers much more fun and convenience. As the 'Manual Shooting' in 'Gesture' mode is triggered by

double tapping the stylus pen and the 'Return' button only need to be pressed once to trigger the 'Firing' in 'Doom' mode, the fact that there are double the number of taps as compared to button presses is reasonable in this stage. Results from the questionnaire and interview confirm this.

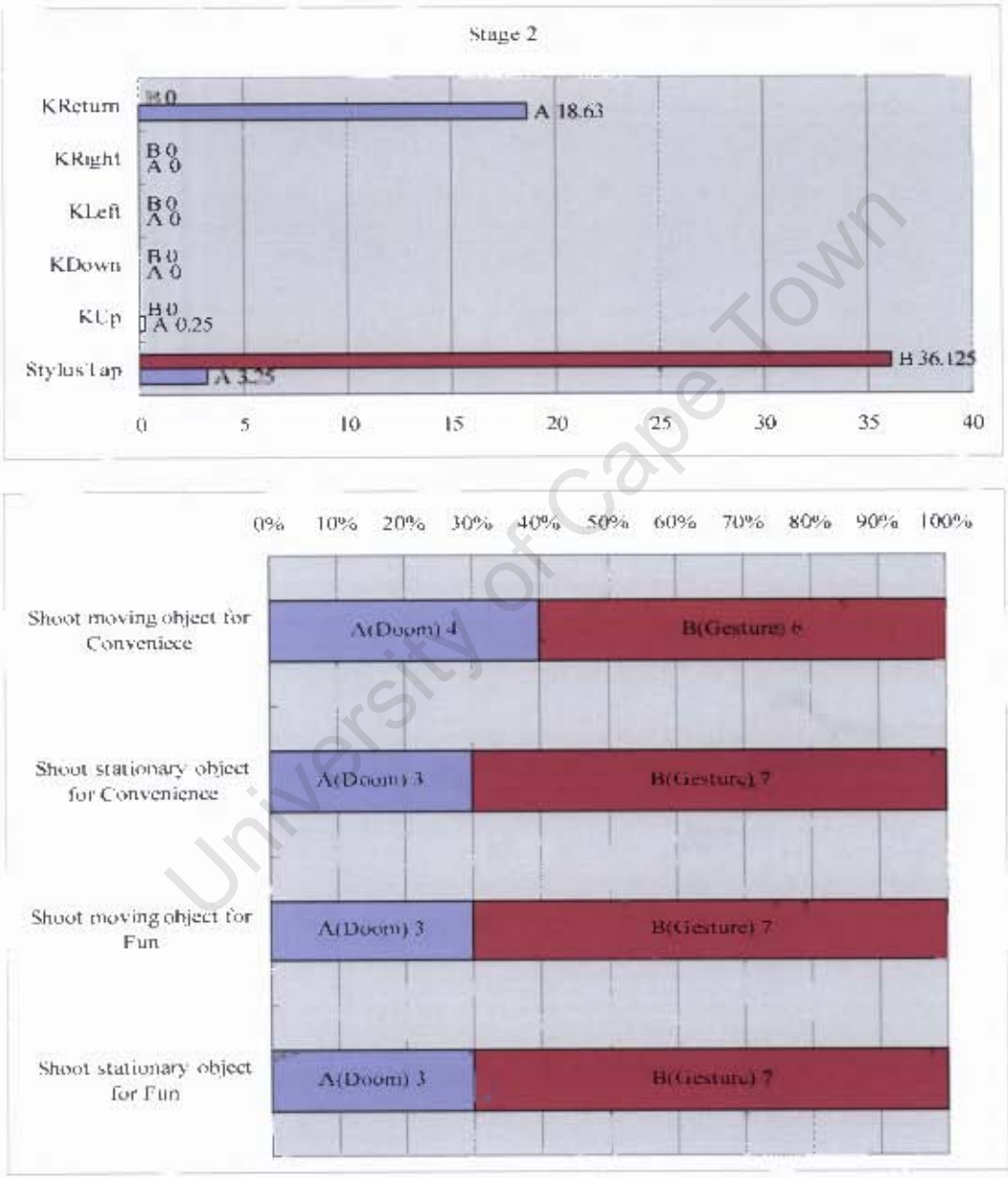


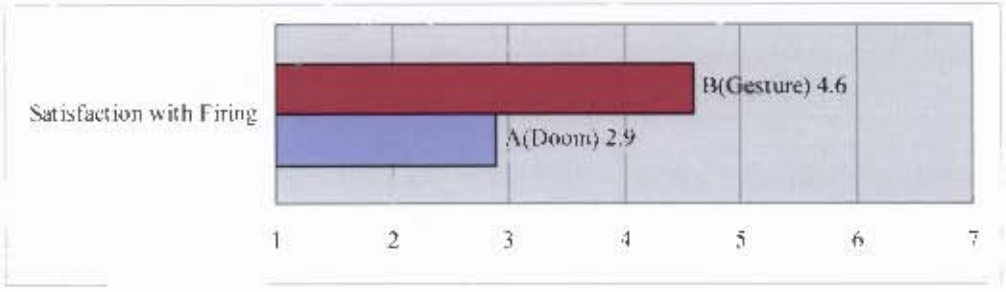
Figure 27. The scores in Stage 2

The testers experienced the new shooting style 'Auto Aiming & Shooting' in this stage. Some of them expressed how much they enjoyed this new shooting mechanism. The average score for satisfaction with this shooting method is '4.4'.

Stage3: Moving & Shooting

This is the most challenging stage in the entire application. All the amateurs failed to accomplish this stage. Even the experienced FPS experts had trouble in finishing this stage. However, the new interaction style shows better performance results than the original interaction style. Table 5 indicates that using the 'Gesture' mode enables users to finish the stage in less time with less mouse (stylus) movement than the 'Doom' mode.

The most difficult task in this stage is the execution of the 'Shooting' command. Using this function in 'Doom' mode is especially inefficient, since users have to release the 'Moving' buttons first and then press the 'Return' button (all of these buttons are controlled by the left thumb), which means the avatar is only able to shoot in a stationary situation. This impedes performing the actions of dodging and counter attack and severely impacts gameplay [4]. The new interaction design converts the execution of the 'Firing' command from the user's left hand to his right hand, which solves the problem of implementing the most commonly used commands, 'Moving' and 'Firing', simultaneously. The coordination of these functions has clearly been improved. Using the new interaction system costs much less button pressing times than using the Doom mode. With the separation of the 'Firing' and 'Moving' command to two hands, a player can easily simulate the 'strafe + rotate' action in PC game combat, which is a major game play move that allows continuous rotation about a fixed centre (the enemy under attack). This can be implemented so as to simulate the use of the keyboard and mouse, provided the player holds onto one PDA direction button while micro-adjusting the stylus pen in the other direction. Testers were satisfied with these new changes, as shown in Figure 28.



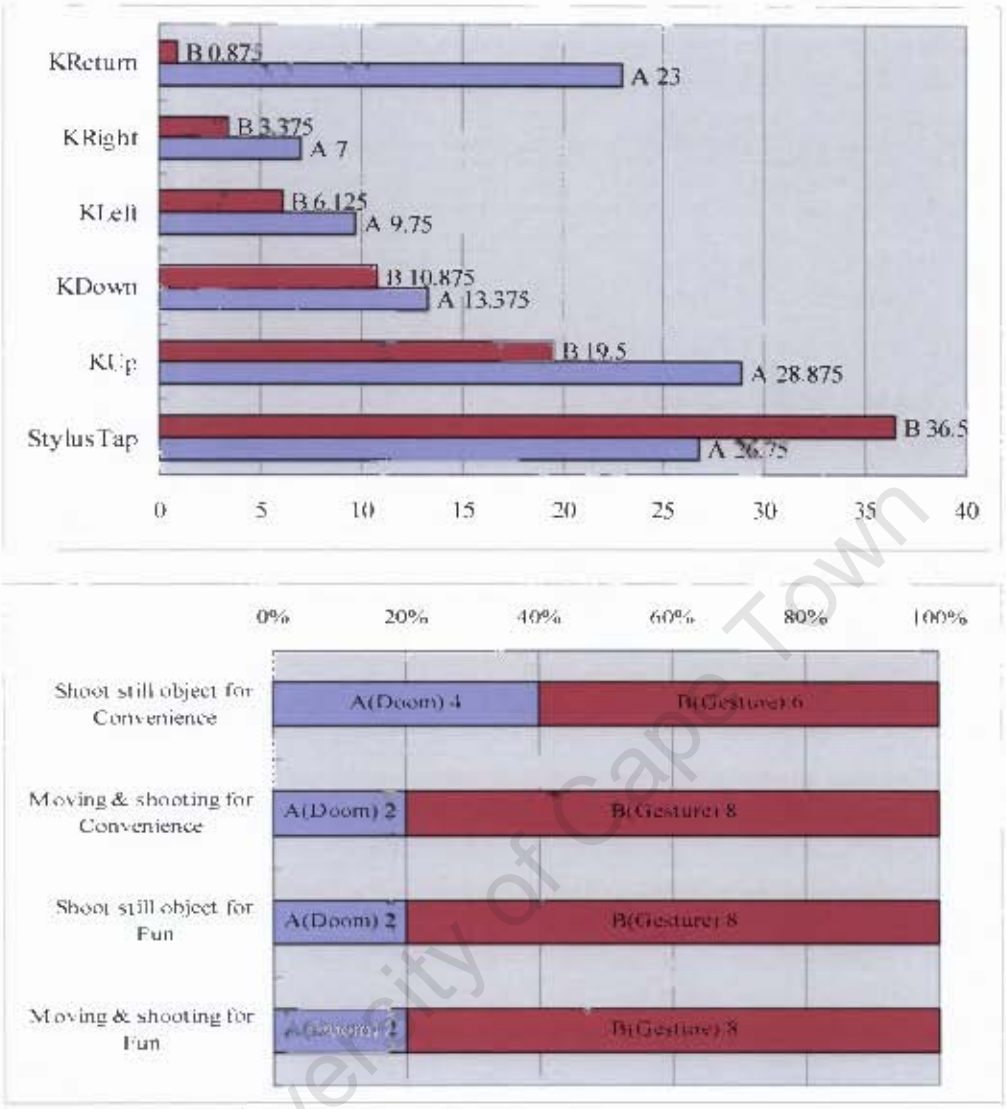


Figure 28. The scores in Stage 3

Only a few testers tried the 'Auto Aiming & shooting' mechanism in this stage. This is because their habit of performing this shooting mechanism in an intense task has not formed yet; and in intense game situations, they forget about this function. However, testers who used this method rated it above average.

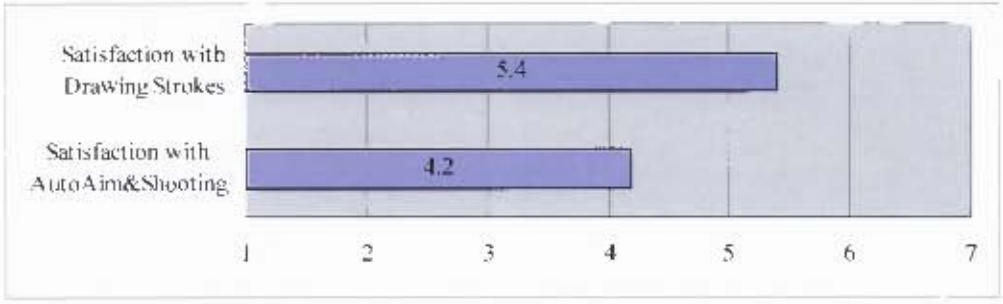


Figure 29. The scores of 'AutoAim&Shooting'

Stage4: Function Execution

Again, the new interaction mode took more time than the 'Doom' mode as shown in Table 5. However, users are satisfied with it.

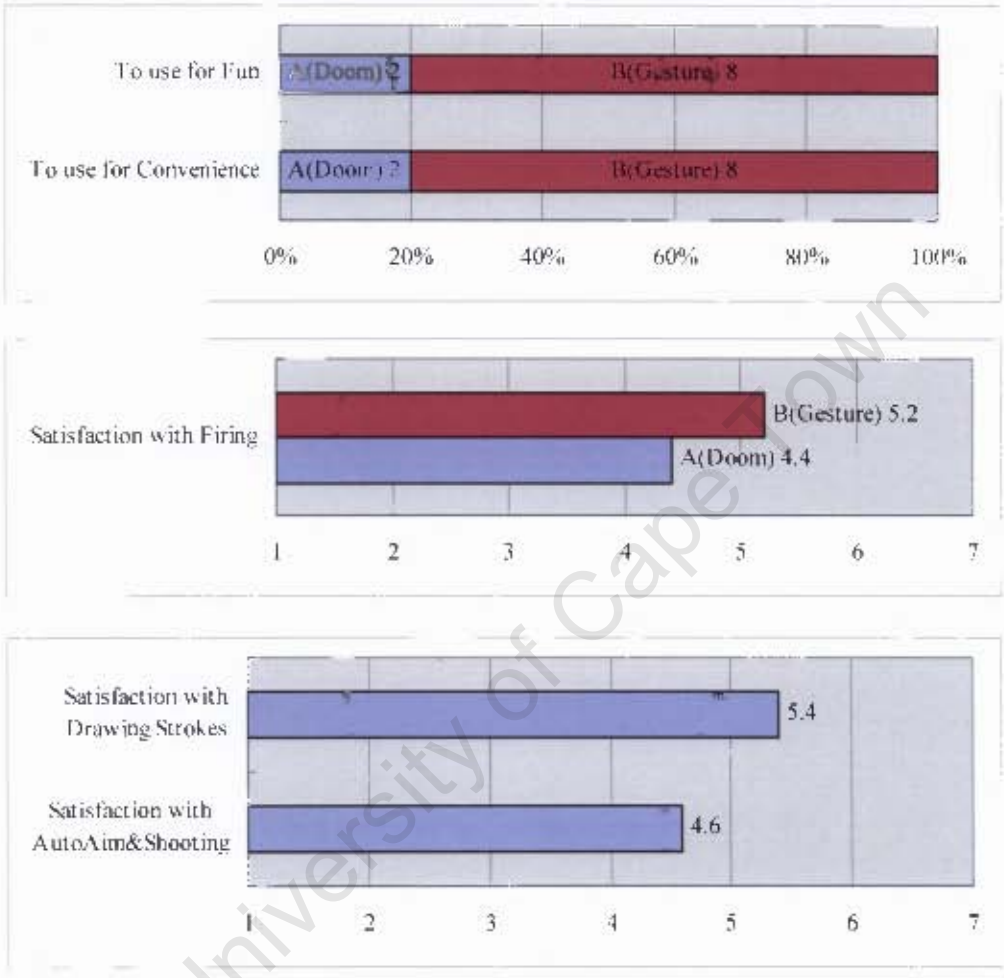


Figure 30. The scores in Stage 4

In addition to the unlimited commands, the entertainment and convenience offered by the new interaction system is satisfactory. Another advantage of the new interaction design is shown in Figure 31. In interaction 'A' (Doom), which the upper diagram shows, the tester had to press the other four hotkeys around 40 times during this stage. These four hotkeys are controlled by the left thumb. Therefore, the left thumb has to take care of nine buttons in total, which is double the number in 'Gesture' mode. This unbalanced command assignment confused the testers significantly. Users sometimes triggered undesired commands by mistake in 'Doom' mode.

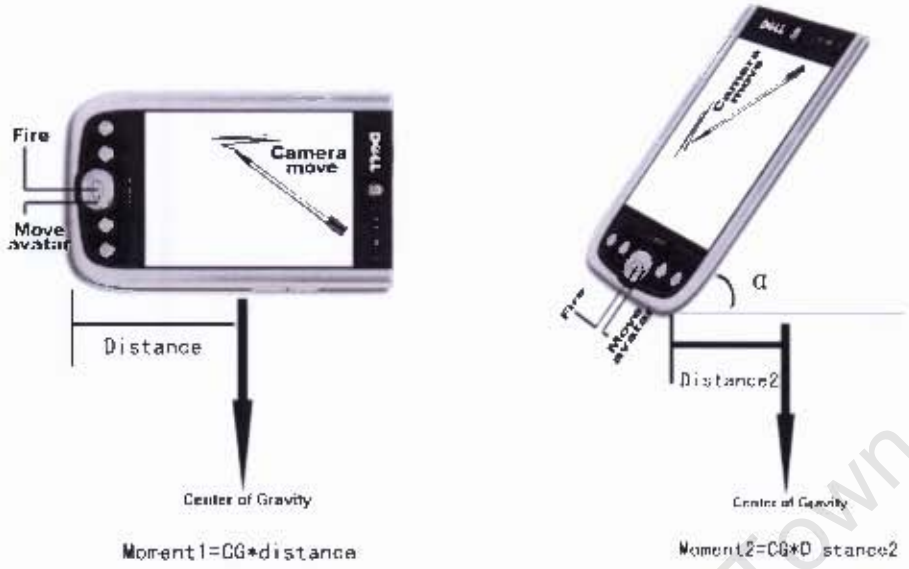


Figure 34: The force exerted on the user's hand is much greater in the landscape-oriented Doom than in our portrait-oriented system.

A further problem with this is that it affects how comfortably the PDA can be held, which is showed in Figure 34. The left palm supports the weight of the device, with the left thumb controlling the balance and grasping of the device. The less movement the left thumb has to make between different buttons, the more stable and comfortable holding the device becomes. With the stylus, movement rate benefits from the 'Gesture' mode and users did not feel the fatigue from holding the PDA device at all, even though they had to hold it for longer.

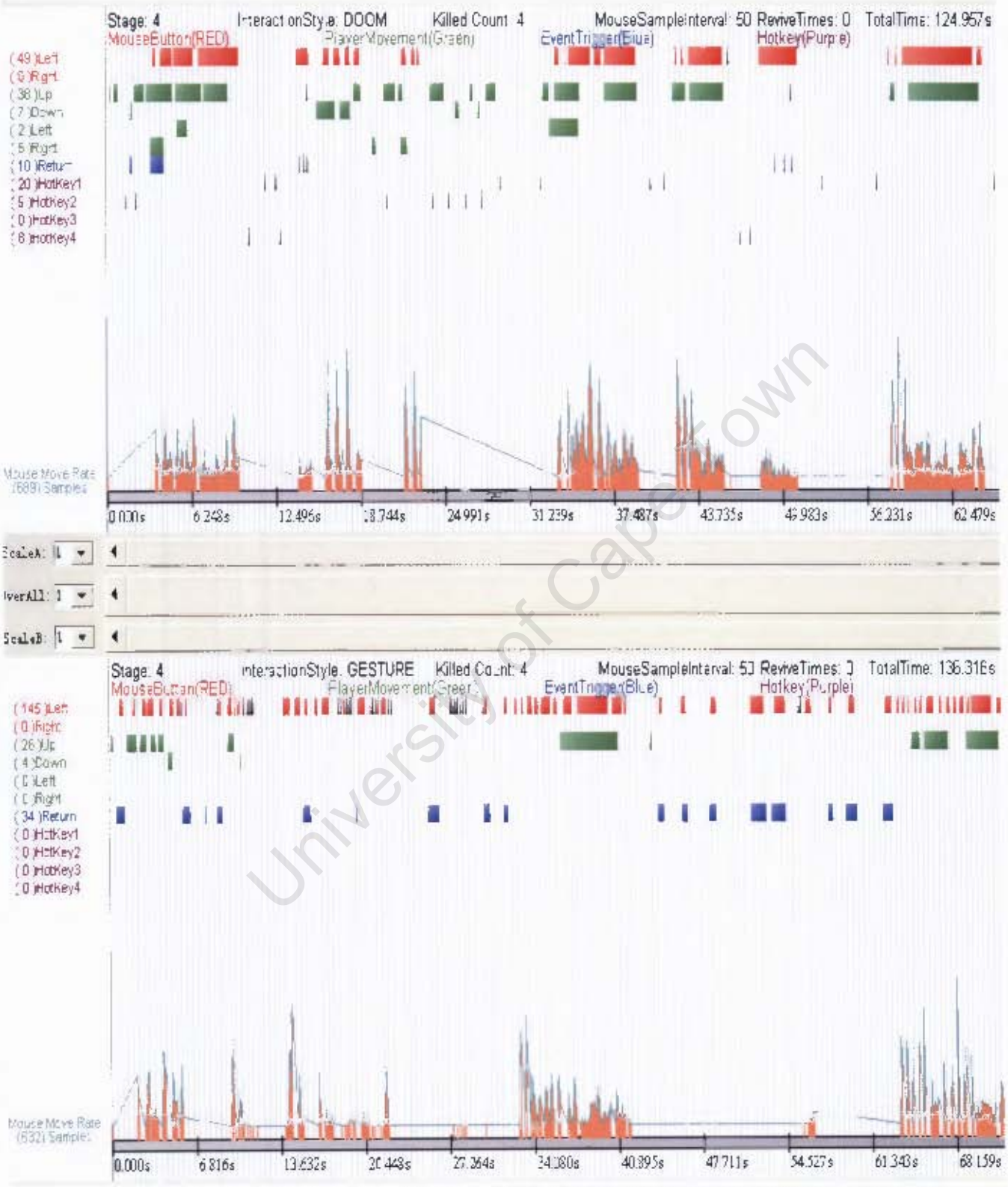
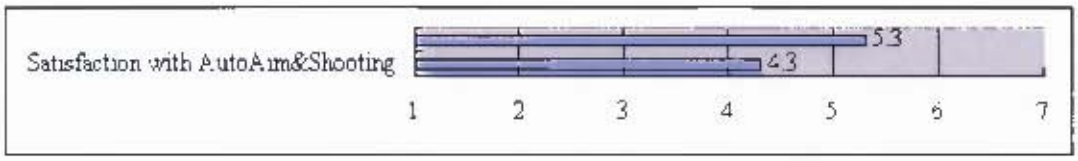


Figure 31. The 'Doom' mode (Upper) used eight keys while the 'Gesture' mode (Lower) just used three buttons in Stage 4.

Overall Results



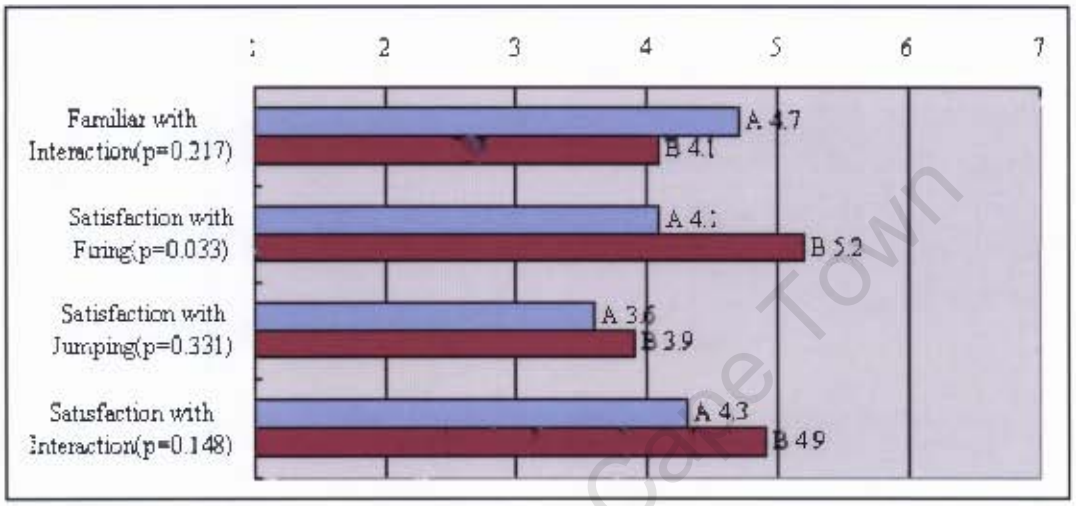


Figure 32. The over all results

A summary of the results are shown in Figure 32. Overall, there was no significant difference (p=0.148) between preference scores for each system. The only significant result (p=0.033) was for the new firing mechanism we developed, which players rated as increasing playability.

So, overall the experiment results show that the new interface is certainly no worse than existing systems and, in the case of firing, is capable of making FPS games on PDA devices more practical and enjoyable than current implementations. Although the user spends more time using 'Gesture Interaction' than current interaction styles, the new style solves major problems including function limitation, user fatigue due to holding the PDA, difficulties in coordination when performing multiple commands simultaneously and restricted camera movement.

Table 5 and 6 show that in 'Gesture' mode, both the number of stylus taps and its average movement rate are more than in 'Doom' mode. These results are reasonable because:

Unlike the 'Doom' mode the stylus pen only controls the change of camera view. In 'Gesture' mode, it is also in charge of the execution of commands ('Firing' and advanced function commands) by tapping and stroke drawing.

The 'Gesture' mode requires a bit more time in finishing the stage than the 'Doom' mode (with the exception of the Stage 3).

But the column 'Average of Stylus Samples' in the Table 5 shows that the 'Doom' mode recorded more stylus events than the 'Gesture' mode, with the exception of Stage 4. This leads to the hypothesis that in 'Gesture' mode, the stylus pen actually spent less time on the touch screen than in 'Doom' mode, based on the fact that:

a stylus event would only be recorded as a sample when it had been left on the touch screen for more than 50 million seconds,

that with using 'Doom' mode it takes, in general, less time to finish the stages than the 'Gesture' mode.

This assumption was confirmed with the help of *InteractionAnalysor*. As Figure 31 shows clearly, the density of the stylus (mouse) samples recorded in 'Doom' mode is higher than the one in 'Gesture' mode. Therefore, using 'Doom' mode is more intense than using the 'Gesture' mode in running this prototype.

6.4 Summary

Three groups of testers participated in the evaluation experiment. These included experts in mobile HCI, FPS gamers and amateurs. Four categories of the experimental results helped in evaluating the new interaction system in an objective and effective way. The results show that although playing the FPS games using the new interaction style is a bit slower than using the button pressing interaction style in some situations, users prefer the new interaction system. This is because it is more fun, convenient and efficient. There are, however, still some problems, such as 'Jumping', which remain to be solved.

7 Conclusion

This research designed a new interaction paradigm for FPS games on PDA devices following a user-centered methodology. First of all, an interaction logging system, called *InteractionLog*, was build to help investigate user behaviour whilst playing FPS games on desktop PCs. Based on the understanding of these behaviours and the interaction problems found from an experiment into actual FPS game play on PDA devices, a new interaction system was proposed to solve these problems. Both a low-fidelity prototype and a high-fidelity prototype were built to evaluate the new interaction design.

The results of the study show that the new interface is capable of making FPS games on PDA devices more practical and enjoyable than current interaction styles or interfaces. Although the user spends more time using *Gesture Interaction* than current interaction styles, the new style solves major problems, including function limitations, user fatigue due to holding the PDA, difficulties in coordination when performing multiple commands simultaneously and restricted camera movement.

A *Gesture Interaction* approach has the following advantages:

Distribution of crucial commands to both hands creates good performance which is especially obvious in intense battle situations.

The new interaction system is easy to learn. All the testers picked it up quickly. None of them needed to look at the instruction sheet after running the application for a short time.

Players prefer the new interaction system more than the traditional one, irrespective of whether it is slower or not. We believe that the more players get used to the new interaction style, the more they will enjoy using it.

There is still work to be done, however, and it is clear that some functions, such as 'Jumping', still need to be improved. We believe that a sensible balance between

gesture recognition and hardware buttons should support this and provide even greater satisfaction to game players.

In section 4, we stated that there were, as yet, no standard conventions for the controls on PDA-based FPS games. Although we do not believe our research to be mature enough to be adopted as a standard, it does point the way to what that standard might look like – it will almost certainly incorporate some form of gesture recognition and automatic firing system. We hope that other researchers will be able to build on our work to create a canonical set of gestures and a firing system that strikes an engaging balance between automation and user controls.

References

[1] W. Apsman, C. Woodward, M. Hakkarainen, P. Honkamaa and J. Hyakka. 2004. Augmented reality with large 3D models on a PDA: implementation, performance and use experiences. In Proceedings of SIGGRAPH'04, Los Angeles, USA. Pp.344-351.

[2] G. Marsden and N. Tip. 2005. Navigation control for mobile virtual environments. In Proceedings of MobileHCI'05, Salzburg, Austria. Pp.279-282.

[3] M. Webber, T. Pfieffer and B. Jung. 2005. Pr@senZ - P@CE: mobile interaction with virtual reality. In Proceedings of MobileHCI'05, Salzburg, Austria. Pp.351-352.

[4] H. Korhonen and E. Koivisto. 2006. Playability heuristics for mobile games will be held through the expert evaluation. In Proceedings of MobileHCI'06, Helsinki, Finland. Pp.9-16.

[5] M. Federoff. 2002. Heuristics and usability guidelines for the creation and evaluation of fun in video games. Thesis at the Graduate School of Indiana University.

[6] S. Laitinen. 2006. Do usability expert evaluation and test provide novel and useful data for game development. Journal of Usability Studies. Issue 2, Vol. 1, February. Pp.64-75.

[7] M. Jones and G. Marsden. 2006. *Moible Interaction Design*. John Wiley & Sons.

[8] Y. Lim, A. P, S. Periyasami and S.Aneja. 2006. Comparative analysis of high-and low-fidelity prototypes for more valid usability evaluations of mobile devices. In Proceedings of NordiCHI'06, Oslo, Norway. Pp.291-300.

[9] J. Nielsen and R. Mark (Eds.). 1994. *Heuristic Evaluation. In Usability Inspection Methods*. John Wiley & Sons.

[10] J. Nielsen and T. Landauer. 1993. A mathematical model of the finding of usability problems. In Proceedings of INTERCHI'93, Amsterdam, Netherlands. Pp.206-213.

[11] J. Nielsen and R. Molich. 1990. Heuristic evaluation of user interfaces. In Proceedings of ACM CHI'90, Seattle, USA. Pp.249-256.

[12] J. Nielsen. 1992. Finding usability problems through heuristic evaluation. In Proceedings of ACM CHI'92, Monterey, Canada. Pp.373-380.

[13] H. Desurvire, M. Caplan and J. Toth. 2004. Using heuristics to improve the playability of games. In Proceedings of CHI'04, Vienna, Austria. In the collection of Abstracts.

[14] R. Pagulayan, K. Keeker, D. Wixon, R. Romero and T. Fuller. 2003. User-centered design in games. In J.Jacko and A. Sears (Eds.), *Handbook for Human-Computer Interaction in Interactive Systems*. NJ: Erlbaum. Pp.883-906.

[15] S. Mel. 2003. A note on presence terminology. Note at the University College London,

[16] W. IJsselsteijn, J. Freeman and H. Ridder. 2001. Editorial: presence: where are we? Cyberpsychology and Behavior. Issue 2, Vol. 4. Pp.179-182.

[17] B. Witmer and M. Singer.1998. Measuring presence in virtual environments: a presence questionnaire. Presence: Teleoperators and Virtual Environments. Issue 3, Vol. 7, June. Pp.225-240.

[18] A. Winter. 1999. An investigation into real-time 3D polygon rendering using BSP trees. Thesis at University of Wales Swansea.

[19] H. Fuchs, Z. Kedem and B. Naylor. 1980. On visible surface generation by a priori tree structures. ACM Computer Graphics. Pp.124-133.

[20] Y. Wang, H. Bao and Q. Peng. 1998. Accelerated walkthroughs of virtual environments based on visibility preprocessing and simplification. Computer Graphics Forum. Issue 3, Vol. 17. Pp.187-194.

[21] S. Teller and C. S' equin. 1991. Visibility Preprocessing for Interactive Walkthroughs. In Proceedings of SIGGRAPH '91, Los Angeles, USA. Pp.61- 69.

[22] S. Teller. 1992. Visibility computations in densely occluded polyhedral environments. Thesis at University of California at Berkeley.

[23] H. Zhang, D. Manocha, T. Hudson and K. Hoff. 1997. Visibility culling using hierarchical occlusion maps. In Proceedings of SIGGRAPH'97, Los Angeles, USA. Pp.77-88.

[24] D. Badouel. 1990. An efficient ray-polygon intersection. *Graphic Gems*. Academic Press. Pp.390-393.

[25] I. MacKenzie and S. Zhang. 1997. The immediate usability of graffiti. In Proceedings of GRAPHIC INTERFACE '97, San Francisco, USA. Pp.129-137.

[26] C. Blickenstorfer. 1996. Handwriting recognition is alive and well. Pen Computing Magazine, August. Pp.30-33.

[27] Palm Computing. 1995. Suddenly Newton understands everything you write. Pen Computing Magazine, January. P.9.

[28] C. Blickenstorfer. 1995. Graffiti:Wow! Pen Computing Magazine, January. Pp.30-31.

[29] T. Malone. 1995. Heuristics for designing enjoyable user interfaces: Lessons from computer games. In John C. Thomas and M. L. Schneider (Editors), Human Factors in Computing Systems, Norwood, NJ: Ablex Publishing Corporation.

Appendix A

Questionnaire

Mark the interaction style you first tested:

Interaction A (Button Pressing)

Interaction B (Stroke Drawing)

Name:

Age:

Email:

1. Please rate your *sense of being familiar* with using the Interactions, on a scale of 1 to 7, where 7 represents you are *comfortable* with this interaction style.

I had a sense of being familiar with the using the Interaction A:

1	2	3	4	5	6	7
Not at all						Master

I had a sense of being familiar with the using the Interaction B:

1	2	3	4	5	6	7
Not at all						Master

2. Compare the *convenience and efficiency* between the 'Interaction Style A' and 'Interaction Style B'; mark the one you prefer of each question in the following form.

Questions	Interaction A	Interaction B
Which style you prefer to shoot at a still object?		
Which style you prefer to shoot at a moving object?		
Which style you prefer to act stationary shooting?		
Which style you prefer to act moving shooting?		
Which style you prefer to use in the maze?		

3. Compare the *Challenge and Entertainment* between the 'Interaction Style A' and 'Interaction Style B'; mark the one you prefer of each question in the following form.

Questions	Interaction A	Interaction B
Which style you prefer to shoot at a still object?		
Which style you prefer to shoot at a moving object?		
Which style you prefer to act stationary shooting?		
Which style you prefer to act moving shooting?		
Which style you prefer to use in the maze?		

4. Please rate your *sense of being satisfied* at executing *'Firing'* command in the stages, on a scale of 1 to 7, where 7 represents you are most *pleased* at triggering *firing*.

I had a sense of being satisfied at 'Firing' using the Interaction A:

	1	2	3	4	5	6	7
Stage2	Not at all						Very Happy
Stage3							
Stage4							
Overall							

*I had a sense of being **satisfied** at **'Firing'** using the Interaction **B**:*

	1	2	3	4	5	6	7
Stage2	Not at all						Very Happy
Stage3							
Stage4							
Overall							

5. Please rate your *sense of being **satisfied*** at executing the ***'Jumping'*** command in the stages, on a scale of 1 to 7, where 7 represents you are most ***pleased*** with jumping

*I had a sense of being **satisfied** at **'Jumping'** using the Interaction **A**:*

	1	2	3	4	5	6	7
Stage1	Not at all						Very Happy
Stage4							
Overall							

*I had a sense of being **satisfied** at **'Jumping'** using the Interaction **B**:*

	1	2	3	4	5	6	7
Stage1	Not at all						Very Happy
Stage4							
Overall							

6. Please rate your *sense of being **satisfied*** at executing the ***'Auto Aim'*** command in the stages, on a scale of 1 to 7, where 7 represents you are most ***pleased*** with using ***Auto Aim.***

	1	2	3	4	5	6	7
Stage2	Not at all						Very Happy
Stage3							
Stage4							
Overall							

7. Please rate your *sense of being **satisfied*** at ***drawing strokes*** in the stages, on a scale of 1 to 7, where 7 represents you are most ***pleased*** with ***drawing strokes***:

	1	2	3	4	5	6	7
Stage1	Not at all						Very Happy
Stage3							
Stage4							
Overall							

8. Please rate your *sense of being **satisfied*** at using the interactions in completing all the stages, on a scale of 1 to 7, where 7 represents you are most ***pleased*** at using this interaction style.

*I had a sense of being **satisfied** at the using the Interaction **A**:*

1	2	3	4	5	6	7
Not at all						Very Happy

*I had a sense of being **satisfied** at the using the Interaction **B**:*

1	2	3	4	5	6	7
Not at all						Very Happy

Appendix B

Interaction Analysor which compares two interaction modes in an intuitive way. It shows activities of the interaction events in a certain period of time.