First-Person Shooter Controls on Touchscreen Devices: 
a Heuristic Evaluation of Three Games on the iPod Touch

Tuomas Hynninen
Today's touchscreen devices have a large amount of computing power which enables them to run high performance software, such as first-person shooter games rendered in 3D. Devices such as the iPod Touch that were originally built for music and other media consumption also provide a wide variety of features. For example, the iPod Touch can be used to surf the Internet, play quality video and audio and run complex video games.

This thesis concentrates on first-person shooters games on the iPod Touch and provides a detailed breakdown of how developers have used the device's touch interface in their games. Related research, especially regarding first-person shooters and touch input, is explored and discussed. Completely new heuristics were developed in order to analyze the properties and effectiveness of the controls.

In total, three iPod Touch games and nine control modes were evaluated. The results showed that the effectiveness of the controls was lacking. Target acquisition and tracking proved to be especially problematic. The results followed conclusions and observations made in related research. Suggestions for future work include investigating touchscreen hardware improvements and exploring different FPS game designs for mobile platforms.

Keywords and terms: first-person shooter, FPS, touch input, iPhone, iPod Touch, haptic user interface, mobile user interface, mobile games, multi-touch input, tactile feedback, 3D
# Table of Contents

1. Introduction........................................................................................................... 1
2. First-person shooter games.................................................................................. 2
   2.1 History............................................................................................................. 2
   2.2 Evolution of control schemes.......................................................................... 3
   2.3 Gameplay.......................................................................................................... 4
   2.4 Free and fixed movement................................................................................ 5
   2.5 Game perspective............................................................................................ 5
   2.6 Traversing the 3D world................................................................................ 6
   2.7 Target acquisition........................................................................................... 7
   2.8 Controls on different input devices................................................................. 8
      2.8.1 Keyboard and mouse controls................................................................. 8
      2.8.2 Gamepad controls.................................................................................. 11
   2.9 Summary.......................................................................................................... 13
3. iPod Touch............................................................................................................. 14
   3.1 Introduction..................................................................................................... 14
   3.2 Criteria for device selection........................................................................... 14
   3.3 Technical specifications.................................................................................. 15
   3.4 Touch interface and sensors........................................................................... 16
   3.5 Summary.......................................................................................................... 17
4. Research on Fitts’ law......................................................................................... 18
   4.1 Fitts’ law......................................................................................................... 18
   4.2 Fitts’ law in target acquisition........................................................................ 18
   4.3 Fitts’ law and game controllers....................................................................... 19
   4.4 Applicability of Fitts’ law............................................................................... 19
   4.5 Summary.......................................................................................................... 20
5. Research on touchscreen input.......................................................................... 21
   5.1 Advantages..................................................................................................... 21
   5.2 Disadvantages................................................................................................. 21
   5.3 Pointing and selection strategies..................................................................... 22
   5.4 High precision pointing and selection strategies.......................................... 22
   5.5 Tactile feedback in touchscreen devices...................................................... 24
   5.6 Multi-touch pointing and selection strategies.............................................. 24
   5.7 Unimanual and bimanual interaction............................................................ 26
   5.8 Summary.......................................................................................................... 27
6. Research on game controllers, first-person shooter games and player experience... 28
   6.1 Game controllers and 3D environments....................................................... 28
   6.2 Input devices in first-person shooter games................................................. 31
   6.3 Game controls on mobile devices................................................................. 31
   6.4 Effect of frame rates....................................................................................... 32
   6.5 Effects of screen resolution and frame rates............................................... 34
   6.6 Usability issues............................................................................................... 34
   6.7 Player experience............................................................................................ 35
   6.8 Summary.......................................................................................................... 35
7. Game evaluation................................................................................................ 37
   7.1 Research summary......................................................................................... 37
   7.2 Existing heuristics.......................................................................................... 39
   7.3 Designing heuristics....................................................................................... 40
   7.4 Methodology................................................................................................. 42
1. Introduction

Games on touchscreen devices have come to stay. In 2012 a high-end smartphone contains a gigahertz processor and hundreds of megabytes of RAM. Increased computing power enables these devices to run complex video games. The App Store by Apple [2008] and the Android Market by Google [2008a] have given thousands of game developers an easy access to a large consumer market. As the budgets of games have skyrocketed, many game developers have turned to smaller-budget mobile games. It stands to reason then that the mobile phone games industry is going to grow at a fast rate.

The iPhone by Apple [2007a] has been one of the leading touch input smartphones since its release. The total market share for Q2 2011 for the whole iOS platform in smartphones was 18.2% according to Gartner [2011]. The iPhone's simple and intuitive user interface has earned its fair share of copycats and imitators. As smartphones with touch input interfaces become more widely spread, it is important to study the strengths and weaknesses of touch input in a mobile context.

The iPod Touch by Apple [2007b] is nearly identical to the iPhone. The user can use the device to watch movies, display pictures, surf the Internet and, most importantly for this thesis, play video games. The iPod Touch doesn't have the iPhone's SMS or calling capabilities, but it can play the same video games as the iPhone, thanks to the same hardware and operating system. It is no surprise then that gamers have embraced the iPod Touch. Games from every genre have been converted to the device.

The main theme of this thesis is to study how well suited the iPod Touch's interface is for first-person shooter (FPS) games. FPS games are known for input-heavy controls. The player has to target and hit rapidly moving opponents while simultaneously moving. Combining hard controls with fast gameplay action is a challenge for game designers on any platform. It is then interesting to study how game developers have solved the control design challenge on the iPod Touch.

This study will start with Chapter 2 where basic concepts of FPS games will be explored. In Chapter 3 a brief introduction to the iPod Touch is given. In Chapters 4 to 6 related research on touch screen devices and first-person shooter games is discussed. In Chapter 7 select iPod Touch first-person shooter games are analyzed and discussed. Chapter 8 will close up this thesis with discussion about the results of the evaluation.
2. First-person shooter games

This Chapter introduces the first-person shooter genre and shows how common input devices work in a first-person shooter game. Related terminology and subjects, such as 3D worlds and movement styles, will also be explored.

2.1 History

The history of first-person shooters goes back to the 1970s [Wikipedia, 2012a]. Games like Maze War provided a simple game play experience: players moved through a maze and shot at each other, thus gaining points for winning the game [Colley, 1974]. In the 80s, Battlezone by Atari [1980] provided a more immersive experience through vector graphics. In Battlezone the player fights against tanks, UFOs and missiles in a desolate battlefield while protecting his own tank from enemy fire.

In 1992, id Software released Wolfenstein 3D, a first-person shooter game in which the player controlled a character called William “B.J.” Blazkowicz, whose mission in the game was to escape Castle Wolfenstein [id Software, 1992]. The game provided fast and brutal action never seen before. The player moved through the world of Wolfenstein while shooting Nazis and collecting hidden treasures. While Wolfenstein 3D provided a novel game play experience and exciting graphics, it was not without its critics. The explicit violence got the game banned in several countries. The World War II theme and Nazis led to a swift ban in Germany [Wikipedia, 2012b].

In 1993, id Software released Doom which took the technological advancements of Wolfenstein 3D even further [id Software, 1993]. Now the player was able to move up and down stairs, pop into huge rooms or outdoor environments. Again, the game was marked by fast and brutal action, with gory death animations included. Doom was lauded for its graphics and game play and criticized for its violence.

id Software’s success was not limited to these games. Games such as Doom II [id Software, 1994], Quake [id Software, 1996], Quake 2 [id Software, 1997], Quake 3 [id Software, 1999] and Doom III [id Software, 2004] continued id Software’s lineage of first-person shooters. Competition roused soon after Doom’s success. Duke Nukem 3D by 3D Realms [1996] was another controversial commercial success. Unreal [Epic MegaGames, 1998] brought new 3D graphic innovations [Wikipedia, 2012c]. Impressive outdoor environments and highly detailed indoor environments were the strengths of Unreal game engine used for rendering the game graphics.

The success of 3D games in general marked a new era in the game industry. Soon 3D games needed separate acceleration in the form of 3D cards. The game worlds got more complex and intricate, which affected the game development process. No more were commercial games developed in two months in someone’s garage. Instead, game companies employed tens or hundreds of employees while crafting a single game over a period of two to four years.
After the turn of the millennium it was common place to see 3D graphics in other genres also. The makers of strategy, role-playing and adventure games began utilizing 3D graphics in their works. However, the advances in 3D technology were still being pushed by FPS games. Games such as Half-Life [Valve Corporation, 1998], Medal of Honor [DreamWorks Interactive, 1999], Halo: Combat Evolved [Bungie, 2001], Call of Duty [Infinity Ward, 2003], Far Cry [Crytek, 2004] and Battlefield 1942 [Digital Illusions CE, 2004] took the ever increasing graphical fidelity to new heights.

Now, in 2012, the latest games not only push the graphical limits, but the interactivity of the game environment. The latest installment in the Battlefield series, Bad Company 2, includes destructible environments such as houses, trees and bunkers [Digital Illusions CE, 2010]. This new trend in increasing the player's interaction with the environment only serves to advance the ever-rising FPS experience.

FPS games are also appearing in regular cellphones, thanks to the advances made in mobile technology. Games such as N.O.V.A. [Gameloft, 2009] and Rage Mobile [id Software, 2010] have extremely good graphics, especially when compared to the cell phone games released in the late 1990s and the beginning of 2000s. Some of these games even support multiplayer games through 3G or WLAN connections.

Aside from graphical improvements, there have been obvious control related advancements in the FPS genre and these will be observed in the next section.

2.2 Evolution of control schemes

Impressive graphics are the hallmark of FPS games, but it is important to also look at the evolution of controls in FPS games.

In Maze War by Steve Colley [1974], the controls are rather primitive, as expected. The player can move one step at a time and can only turn in 90 degree turns. The player can't move freely, look up or down or manipulate the viewpoint in any other way than turning. The movement is controlled only from the keyboard.

Battlezone by Atari [1980] offers significant improvements over Maze War. In Battlezone, the player is able to roam freely in the world. The viewpoint can be adjusted smoothly in contrast to Maze War's 90 degree jumps. The tank in Battlezone can be turned left or right which will also automatically adjust the viewpoint. The controls in Battlezone consists of two joysticks, which control the tracks of the tank. However, there are no height differences in the game, which means the tank can be only driven on flat ground. There isn't also any other way of manipulating the viewpoint than turning the tank with the joystick controls.

In Wolfenstein 3D by id Software [1992] the player can turn left or right and move forwards and backwards. Additionally, the player can strafe left or right, which means moving the avatar laterally while the viewpoint is not rotated. Mouse controls can be enabled and used for shooting
and turning. It is to be noted though that there are no height differences in the game. The player cannot also aim upwards or downwards or jump.

Doom has the same controls as Wolfenstein 3D. The only difference is that there are height differences in the game, such as elevators and stairs. The addition is only cosmetic though, as the player will automatically scale the stairs instead of demanding input from the player, such as jumping.

One of the first true 3D FPS games, Quake by id Software [1996] also has controls which have become the staple in FPS games. The player can turn right and left, aim upwards and downwards, and jump. Quake is also one of the games where mouse aiming started to make a difference as the games, which came before, didn't really have height levels. Natapov and MacKenzie [2010] note that Quake had a feature called mouse look or free look, which allowed the players to use the mouse to control the camera.

Starsiege: Tribes by Dynamix [1998] contains all the control elements of Quake and adds the concept of air control. In Starsiege: Tribes, the player is equipped with a jet pack, which can be used to control the player's movement while mid-air. The jet pack enables the player to go high up in the air or propel the player forwards, backwards and sideways while mid-air.

The next section will provide a short introduction to typical FPS gameplay.

### 2.3 Gameplay

The main objective in most mainstream FPS games is killing or subduing opponents in one way or another. The game experience can take place in single player mode where the player fights against computer controlled opponents, or in multiplayer mode where the opponents are controlled by humans.

The context of the game can be anything from the game designer's imagination. Usually, the context is some historical war, e.g. World War II or the Vietnam War, and the player relives historical or fictional situations and storylines based around the war (Battlefield, Medal of Honor, Call of Duty). Another popular setting in FPS games is science fiction. In these types of games the player is usually a human character whose objective is to kill and survive an alien onslaught (Doom, Unreal, Half-Life).

The single player experience consists of a storyline where the player achieves objectives and completes levels, moving the story line further bit by bit. The transitions from level to level are usually accompanied by movie scenes, which depict the flow of the story. The player can collect power-ups (items which boost the player's abilities), fight enemies, hide and seek cover from enemy fire, defend allies and move forward to new objectives.

The multiplayer mode is a replica of the single player mode, with the exception of the story line. In the multiplayer mode players fight against each other, and the objectives usually are to kill each other, kill the other team, capture the other teams flag or defend your base. These objectives are
placed on levels which are then rotated automatically, creating an evolving experience for the players.

These are just some examples of the most common gameplay types. There are, of course, games where the emphasis is on something different. In Love the player works together with other players to defeat the AI player [Steenberg, 2010]. In Minecraft the game world is made of LEGO like blocks which can be used to construct any type of building with seemingly limitless size [Mojang, 2010].

Player movement can also be restricted to one or two axes. Pseudo 3D or 2.5D gameplay in this context is used to describe games which are rendered in 3D, but the player movement is restricted to two axes. This thesis concentrates on games which are both rendered in 3D and also allow players to navigate freely on the three axes (X, Y and Z).

A modern FPS game contains a variety of different gameplay elements, and these elements place certain requirements on the game controls. For example, the player should be able to move freely in the environment, point and manipulate objects and control the camera movement. There are, however, FPS games which limit these aspects artificially, as seen in the next section.

2.4 Free and fixed movement

All FPS games do not have the same type of movement system. The most pervasive one is the one seen in Minecraft [Mojang, 2010]. In these games the player is able to move freely in the world without any kind of involvement by the game. This type of a system can be aptly called a free movement system. However, games like Doom Resurrection by id Software [2009] feature gameplay where the player's route is pre-calculated and the player's avatar moves through this route automatically, like a train on a railroad track. This could be called a fixed movement system. It is common to see arcade FPS games, such as Virtua Cop [Sega, 1994], use this kind of a system.

This thesis concentrates on games which implement a free movement system as it is important to analyze both the movement of the player's avatar and the manipulation of the avatar's viewpoint in the context of the game.

2.5 Game perspective

Claypool and Claypool [2009] have defined game perspective as having two properties. First, how the camera is positioned with respect to the avatar. For example, is the camera behind or “inside” the avatar. Second, how objects in the game world change when the position of the camera changes. Furthermore, the authors have divided perspectives in to three classes.

1) First-Person Linear Perspective: the game world is seen through the eyes of the avatar. In this perspective the camera, which views the world, is placed behind the avatar's eyes. The objects in the world appear smaller or larger depending on their distance from the camera.

2) Third-Person Linear Perspective: the camera is placed around the avatar (usually behind) and the size and clustering of the objects change depending on their distance from the camera.
3) Third-Person Isometric Perspective: the camera is placed around the avatar, but the size and clustering of the objects does not change even if the distance from the camera changes.

In this thesis the emphasis is on the First-Person Linear Perspective, as suggested by the title of the thesis. Some FPS games support changing the camera perspective from first to third, but for the sake simplicity and cohesion this thesis concentrates on the first-person perspective.

### 2.6 Traversing the 3D world

As implied by the term, 3D world includes another axis of interactivity or dimension, called the z-axis. For example, in 2D platform jumping games the player can move along the x- and y-axes. This is seen by the player as the character walking left and right or jumping up and down on the screen. An example of 2D movement can be found in Braid by Number None [2008]. If there was the ability to move on the z-axis, this would be seen as the player walking towards or away from the screen.

Figure 1 illustrates the Z axis in relation to the viewer.

![Figure 1. 3D Cartesian Coordinate System.](image)

In FPS games which employ a 3D world, the z-axis is experienced as going forward or backward in the world, i.e. having "depth" in the world. The illusion of actually traveling inside a 3D world is achieved by rendering the objects in relation to the player's perspective. As mentioned in the previous section, the size and clustering of the objects respond to player movement. This means that the experience of the world as a whole is dynamic, rather than static. For example, if the player walks towards a building, the building will get larger and larger in relation to the player's view, until it takes up the whole view. In contrast, objects farther and farther away look smaller and eventually disappear in the vanishing point, as noted by Sidelnikov [Sidelnikov, 2004].

Figure 2 illustrates how objects in a 3D world appear in relation to the viewer.
2.7 Target acquisition

In order to interact with 3D world, the player needs to be able to affect the objects inhabiting it. In first-person shooters, this usually means firing projectiles, manipulating objects such as doors and windows or grabbing items.

Most, if not all, FPS games provide a targeting helper called reticle. The player can use the reticle to line up shots at enemies or to point at objects. The reticle is both horizontally and vertically centered on the screen. Without the reticle it would be challenging to line up a shot as there wouldn't be a visual indicator for targeting. It would be akin to shooting a rifle without using the scope. As we can see, firing a projectile in an FPS game and in real life is achieved by pointing the reticle at a target and initiating fire.

Touching and grabbing objects is implemented by the player rotating the camera so that the reticle is over the object. The object is then highlighted which indicates that the player can perform an action on it. Looser et al. [2005] have called this pan-based target acquisition. Instead of moving the reticle or cursor to the object, the player pans the camera and adjusts it so that the target is in the center of the screen and under the reticle. Touching objects in this way seems difficult, but it is a concession made to simplify the controls. For example, if a person wants to touch a book standing on a shelf in real life, he does not need to turn his head and line his eyes to initiate an action on the object. Instead, he can just reach over and feel for the book and grab it.

In some games the mouse can be used to target and manipulate objects. For example, in Daggerfall [Bethesda Softworks, 1996], the player can switch between mouse modes for camera manipulation and object targeting. In camera mode, mouse movement will affect the camera view.
In object targeting mode the camera will stay still and the player can use a targeting cursor to manipulate objects. This kind of mode switching works for slower paced games such as Daggerfall. FPS games are very fast paced so this kind of mode switching is a hindrance in the middle of fast action.

Most games also use some sort of auto acquiring system for grabbing objects. The player only needs to walk over or near the object, and it will automatically be added to the inventory or applied to the player's status. The objects are usually beneficial in nature, such as health regeneration or extra ammunition. **Power-ups** is an often used term for these items.

Pan-based target acquisition is a key element in 3D FPS games. It allows the player to seamlessly look around the game world and interact with objects. Thus the player doesn't need to switch between different modes of targeting. Pan-based target acquisition is examined further in Chapter 4.

### 2.8 Controls on different input devices

As noted in the previous sections, user interaction in FPS games comprises of several components. First, there is the issue of player movement, i.e. how the player navigates a 3D world which has 3 axes. The game developer has to provide the player a way to effectively maneuver these 3 axes. Second, the player must be able to point at objects and affect them somehow. Third, the player must be able to manipulate the view in order to look in different directions.

To reiterate, there is one component relating to the movement of the player object itself, one component regarding target acquisition and one component for camera manipulation. The following sections will explore how different control schemes are used in FPS games for these three aspects.

Only keyboard, mouse and gamepad controls are given an indepth review. There are certainly other valid input devices for FPS games: joysticks, trackballs and motion control devices such as the Nintendo Wii Remote. However, these are not the most pervasive controlling schemes used in FPS games on the PC or the consoles, so for the sake of brevity, they are left out of this review. The term console is used to refer to a gaming device which can be attached to a TV or a monitor, such as the Nintendo Wii.

There is a key difference between the mouse and the gamepad. The mouse is a **position-control device**, whereas the analog stick on a gamepad is a **rate-control device**. The mouse can directly position the camera, while the analog stick can only control the direction and speed of the camera panning. This means that while the analog stick offers fine-grained control, for example, over the camera view, it can be slower than the mouse.

#### 2.8.1 Keyboard and mouse controls

On the personal computer the standard for FPS controls is a combination of the keyboard and the mouse. The keyboard is used to maneuver the player on the X, Y, and Z axes of the world and the mouse is used to control the player's camera. The same observation was also made by Klochek and
MacKenzie [2006]. This control scheme is just a generalization though, as games usually offer the option to re-configure the keyboard and mouse functions.

Keyboards come in several layouts but the most universally used layout is QWERTY. QWERTY indicates a layout in which the letter Q, W, E, R, T and Y appear as the first 6 letters in the top-left letter row of the keyboard. The following key mapping examples assume that the keyboard uses a QWERTY layout. The examples also assume that the player is left-handed, thus keeping his left hand on the left side of the keyboard. Figure 3 presents a standard keyboard with a QWERTY layout.

![Figure 3. A standard keyboard with a QWERTY layout.](image)

Table 1 is an example of a movement key mapping.

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
<th>Axis</th>
<th>Direction on 3D cartesian coordinate system</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>Walk forward</td>
<td>Z</td>
<td>-</td>
</tr>
<tr>
<td>S</td>
<td>Walk backward</td>
<td>Z</td>
<td>+</td>
</tr>
<tr>
<td>A</td>
<td>Strafe left</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>D</td>
<td>Strafe right</td>
<td>X</td>
<td>+</td>
</tr>
<tr>
<td>SPACE</td>
<td>Jump</td>
<td>Y</td>
<td>+</td>
</tr>
<tr>
<td>CTRL</td>
<td>Crouch</td>
<td>Y</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1. Keyboard key mappings for player movement.

These actions can also be combined. For example, pressing both 'W' and 'D' down will cause the player to move on axes Z and X respectively.

Table 2 is an example of how mouse movement affects the camera view.
Many games offer the option to invert these actions. Acquiring targets follows the pan-based method explained earlier. The player pans the camera with the mouse until the target is in sight and under the reticle. The target can be then interacted with.

Table 3 contains key mappings for example object manipulation actions.

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Open door, turn switches, etc.</td>
</tr>
<tr>
<td>G</td>
<td>Throw items</td>
</tr>
</tbody>
</table>

These actions are mapped right next to the movement controls in a W-S-A-D key mapping. This mapping facilitates easier access to object manipulation actions, as the player does not need to reach across the keyboard.

Table 4 contains mouse mappings for example object manipulation actions. Table 4 assumes that the player is using a mouse with at least two buttons.

<table>
<thead>
<tr>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>Fire projectile</td>
</tr>
<tr>
<td>Right</td>
<td>Use current weapon for a melee attack</td>
</tr>
</tbody>
</table>

Most modern FPS games contain more actions than what has been described here. It is interesting to note the large number of keyboard and mouse actions needed for very basic FPS user interaction functionality. This will be highlighted when studying how touch controls work in FPS games. In the next section similar controls are examined for gamepad controllers.
### 2.8.2 Gamepad controls

Current mainstream consoles, such as the Xbox 360 by Microsoft [2005] and the PlayStation 3 (PS3) by Sony [2006], come equipped with a gamepad with dual analog sticks. Cummings [2007] notes that gamepads with analog sticks evolved from directional pad (D-pad) controllers in order to provide players proper 3D control. D-pad control schemes had only eight possible directions, whereas analog sticks can handle a full range of motions.

In the Xbox 360 and PS3 controllers the analog sticks are placed so that they can easily be operated with the thumbs. Both controllers also have a regular D-pad and two analog triggers. The controllers also have an assortment of digital buttons. The PS3’s DualShock controller also has pressure sensitive buttons [Sony, 1997]. Figure 4 shows the PS3’s DualShock controller.

![DualShock 3 controller](image)

**Figure 4.** DualShock 3 controller.

Usually, the camera is controlled by the left analog stick. This leaves the right analog stick for controlling the character. This kind of control setup can be seen in Killzone 2 by Guerrilla Games [2009].

The following examples assume a gamepad with dual analog sticks, such as the DualShock Analog Controller by Sony [1997].

Table 5 presents an example movement mapping for an analog stick.
As mentioned before, the player can use the full range of an analog stick for character movement. This is different from the keyboard based movement shown in section 2.8.1, as the analog stick can be pushed in a 360 degree fashion. Keyboard controls can only be combined to point to 8 different directions.

<table>
<thead>
<tr>
<th>Stick action</th>
<th>Action</th>
<th>Axis</th>
<th>Direction on 3D cartesian coordinate system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push forward</td>
<td>Walk forward</td>
<td>Z</td>
<td>-</td>
</tr>
<tr>
<td>Push backward</td>
<td>Walk backward</td>
<td>Z</td>
<td>+</td>
</tr>
<tr>
<td>Push left</td>
<td>Strafe left</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Push right</td>
<td>Strafe right</td>
<td>X</td>
<td>+</td>
</tr>
<tr>
<td>Press</td>
<td>Crouch</td>
<td>Y</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5. Player movement on an analog stick.

The camera movement functionality is identical to the mouse controlled camera movement shown in section 2.8.1. Table 6 shows how object manipulation actions could be mapped on a gamepad.

<table>
<thead>
<tr>
<th>Stick action</th>
<th>Camera viewpoint</th>
<th>Player view</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push forward</td>
<td>Pans up</td>
<td>Player looks up, e.g. watches the sky</td>
</tr>
<tr>
<td>Push backward</td>
<td>Pans down</td>
<td>Player looks down, e.g. watches the ground</td>
</tr>
<tr>
<td>Push left</td>
<td>Pans left</td>
<td>Player looks left</td>
</tr>
<tr>
<td>Push right</td>
<td>Pans right</td>
<td>Player looks right</td>
</tr>
</tbody>
</table>

Table 6. Camera movement on an analog stick.

<table>
<thead>
<tr>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Use current weapon for a melee attack</td>
</tr>
<tr>
<td>Circle</td>
<td>Open door, turn switches, etc.</td>
</tr>
<tr>
<td>Triangle</td>
<td>Throw items</td>
</tr>
<tr>
<td>Square</td>
<td>Fire projectile</td>
</tr>
</tbody>
</table>

Table 7. Object manipulation actions on a gamepad.

Previous tables have shown an example control mapping for a gamepad. Most FPS games have mapped all the controls in the gamepad for different actions, but it is not necessary to present them here.

Often games on consoles include some sort of targeting helper. The helper is usually in the form of an auto aim, which can be used to automatically target enemies without manually pointing at them. The auto aim can assist the player slightly by automatically centering on the target when the
pointer is near the target. The auto aim can also be triggered by pressing a preset button - in this mode the player's reticle is automatically centered on the nearest target without any need for manual aiming. This control scheme can be seen, for example, in Grand Theft Auto 4 [Rockstar Games, 2008]. This kind of assistance is rarely present in FPS games which are on the PC platform, mostly because the mouse offers superior targeting efficiency.

2.9 Summary

It is crucial to note how many different inputs and variables the player has to keep in mind consciously, or subconsciously, while playing a first-person shooter game. This sets the tone for the whole thesis, as it is interesting to see how complex controls translate to a device with limited input options.

The next Chapter will contain an assessment of this thesis's main platform, the Apple iPod Touch.
3. iPod Touch
In this Chapter the iPod Touch by Apple [2007b] will be discussed. The touch interface and the technical specifications will be given extra attention.

3.1 Introduction
The first generation of iPod Touch devices were introduced in 2007. As of 2012, the devices have gone through four generations of improvements [Wikipedia, 2012d]. The iPod Touch and the iPhone have nearly identical hardware. The iPod Touch doesn't have cellular or GPS components which are required for making phone calls and device tracking.

The iPod Touch can be used to, for example, surf the web, play music, display videos and run games. It runs the same operating system as the iPhone, so functionally it is almost identical to the iPhone.

The device is operated by touch input and gesture recognition. For example, flicking or dragging on the device's home screen will cause the home screen to change to another screen. Touching an application icon will start the corresponding application. Sliding a bar user interface element will unlock the device.

Users can buy applications and games from the App Store by Apple [2008] and install them on their iPod Touch. As of July 2011, the App Store has over 425 000 applications in 20 categories, for example games, news and business [Apple, 2011]. Easy access and low prices provide a cornucopia of games for users.

As of 2012, the Apple homepage for the iPod Touch clearly markets it as mainly as an entertainment device [Apple, 2007b]. The same page also mentions that there over 100 000 games available for the device. This makes the iPod Touch an ideal device for studying FPS games on touchscreen devices.

3.2 Criteria for device selection
As of 2012, there are many high-powered touchscreen devices available, especially smartphones with Google's Android operating system [2008b]. For example, the Galaxy Nexus [Google, 2011], which was developed jointly by Google and Samsung, is an Android mobile phone with a 1.2 GHz processor and 1GB of RAM. Clearly, the Galaxy Nexus is also capable of running high-end games, such as first-person shooters.

A critical difference between the iPod Touch and the Galaxy Nexus is the size of the application markets. The App Store, as of April 2012, has over 110 000 applications in the games category [148Apps, 2012]. The Android Market, as of May 2012, has almost 60 000 application in the same category. As one can see, the App Store has almost twice the amount of games available. Thus, the
App Store provides a more fertile ground for a game researcher as it holds more potential candidates for game studies.

To further compound this issue, Apple has a longer vetting process for applications, meaning that in theory the quality of the applications should be higher. The developer guidelines for the App Store state that, after submitting the game it, will be subjected to a review [Apple, 2012]. Of course, this process cannot guarantee high quality, but it will at least prevent very poorly developed applications from getting published. The Android Market guidelines state that after submission it will take a couple of hours for the game to appear in the store [Google, 2012]. This difference in quality control means that, in theory, the App Store should have more polished games available.

Mobile devices with Windows Mobile, Windows Phone or Blackberry OS operating systems could also be viable options for a first-person shooter game study in the future. Unfortunately, the Symbian platform is no longer actively supported by Nokia, and it seems developers are abandoning the platform as a result [Wikipedia, 2012g]. The market share of users for Windows Mobile, Windows Phone and Blackberry OS devices is, as of 2012, relatively small. It can be expected that the Windows Phone platform will be a strong competitor in the future, as both Microsoft and Nokia are backing the platform [Wikipedia, 2012h].

3.3 Technical specifications

As mentioned previously, there are four generations of the iPod Touch line. The model used in this thesis is from the first generation, with model number MC086KS. Figure 5 shows a first generation model with the display turned on and the operating system's home screen selected.

Figure 5. A first generation iPod Touch.
Table 8 and Table 9 present some physical features and technical features of a first generation iPod Touch model [Wikipedia, 2012d]. The device contains many more components, but they are left out here for the sake of brevity.

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Display dimensions</th>
<th>Weight</th>
<th>Materials</th>
</tr>
</thead>
<tbody>
<tr>
<td>11cm in height</td>
<td>7,5cm in height</td>
<td>120g</td>
<td>Stainless steel back and aluminum bezel, glossy glass covered LCD</td>
</tr>
<tr>
<td>6,28cm in width</td>
<td>5cm in width</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0,8cm in depth</td>
<td>8,9cm diagonal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8. Physical features of the iPod Touch.

<table>
<thead>
<tr>
<th>Processor</th>
<th>620 MHz 32-bit RISC ARM 1176JZ(F)-S v1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics processor</td>
<td>103 MHz PowerVX MBX Lite 3D GPU</td>
</tr>
<tr>
<td>Memory</td>
<td>128 MB LPDDR DRAM</td>
</tr>
<tr>
<td>Storage</td>
<td>8, 16 or 32 GB</td>
</tr>
<tr>
<td>Connectivity</td>
<td>Wi-Fi, USB 2.0</td>
</tr>
</tbody>
</table>

Table 9. Technical features of the iPod Touch.

The iPhone and the iPod Touch both have a component called accelerometer [Johnson, 2007]. It can be used to detect when the user changes the position of an iPhone or an iPod Touch. The information provided by the accelerometer to the operating system can be used to drive changes in the user interface. For example, if the browser software, Safari, is opened in portrait mode and the user flips the device on its side, Safari will automatically change its user interface into landscape mode. The same action can be used to control player movement in a game.

Looking at these features, it is clear that the iPod Touch is a very powerful hand-held device. In comparison, the PlayStation 2 has a 299MHz processor and 32 MB of RDRAM [Wikipedia, 2012e]. The iPod Touch's technical capabilities enable game developers to build high-end gaming experiences, such as first-person shooters rendered in 3D.

### 3.4 Touch interface and sensors

The iPhone and the iPod Touch have capacitive touchscreens which can be operated with multiple fingers for multi-touch sensing [Apple, 2007a]. The screen also has three sensors. One sensor detects changes in light which is then used for adjusting the screen's brightness. The proximity of the user's face is also measured, so that when the user is resting his face on the device, the operating system can turn off touchscreen functionality. The orientation of the device is tracked by the accelerometer, which is briefly discussed in the next section.

Blindmann [2011] discusses how capacitive touchscreens work. The screen has an insulator which is coated with a transparent conductor. The human body also works as a conductor, so touching the screen causes a distortion in the local electrostatic field, which can be then measured
as a change in capacitance. In essence a touchscreen requires bare skin to operate, as the sensors do not work through gloves because they insulate electrical conductivity. Capacitive styli can also be used for operating a touchscreen.

3.5 Summary
The reasoning why the iPod Touch was chosen as the main platform for this thesis should be clear: it has enough computing power to run 3D rendered games, it has a multi-touch screen and a huge amount of games is available for the platform. This Chapter also presented a brief summary of the technical capabilities of the iPod Touch and explored how the capacitive multi-touch screen works. Research on Fitts' law will be discussed in the next Chapter.
4. Research on Fitts' law

Fitts' law is a frequently used metric in measuring the pointing performance of different input devices. A good starting point for understanding Fitts' law is studying the original thesis by Paul M. Fitts.

4.1 Fitts' law

Fitts' law refers to a formula developed by Paul M. Fitts [1954] where he measures the throughput of the human motor system. In essence, by incorporating Fitts' Law into an experiment, one can measure the throughput of an input device in point and select tasks. The throughput, or index of performance, is then measured as $x \text{ bits/second}$. For example, in Fitts' first experiment, the subjects were asked to tap two metal plates with a light weight stylus and a heavier stylus. The metal plates were rectangular in shape and fixed in width, and also placed within a fixed distance of each other. The metal plates were surrounded with error areas which the user had to avoid tapping. By having these restrictions, the movement tolerance and amplitude were controlled. Fitts notes that the rate of performance for the light weight stylus was from 10.3 bits/second to 11.5 bits/second, and the heavier stylus' performance was also relatively stable.

Index of Difficulty (ID) is also related to Fitts' law. The Index of Difficulty can be used to describe the movement difficulty which is related to the distance a limb moves and the size of the limb's target.

Fitts notes that the information capacity for different limbs might be different. For example, the arm, which was used in Fitts' experiments, might have a lower information capacity than the hand. Fitts speculates that multiple fingers in coordination would output a higher level of performance.

Looser et al. [2005] note that Fitts' law is a standard empirical tool for assessing pointing techniques. However, they criticize that conducting experiments which are solely based on Fitts' law can be tedious for the subjects involved, because the experiments contain an inordinate amount of repetitive tasks. This could be avoided by better experiment design, i.e. changing the style or context of the pointing experiments.

A search for "Fitts' law" in the digital library of the Association of Computing Machinery yields hundreds of results, so a culling of the result set is in order. In the next sections research on Fitts' law in related literature will be explored.

4.2 Fitts' law in target acquisition

Looser et al. [2005] studied Fitts' law in the context of pan-based target acquisition in first-person shooter games. Pan-based target acquisition means panning of the camera so that the target reticle is in the center of the screen.
The authors conducted an experiment where the two targeting styles were measured by Fitts' Law. In the traditional targeting experiment the subjects were asked to target a green target inside a window using a mouse. When the green target was clicked, it would change position randomly between seven fixed positions on the x-axis, but not on the y-axis which was fixed half-way down the window.

The pan-based targeting experiment involved a 3D FPS game environment. The subjects were asked to shoot aliens in the game environment. When the alien was destroyed, it would change position just like in the traditional targeting experiment, i.e. changing position on the x-axis and staying fixed on the y-axis.

The results showed that the traditional targeting method yielded a faster mean time for target selection than the pan-based targeting method. The authors note, however, that the divergence is due to the targeting methods having different Index of Difficulty values. The results show that the traditional targeting method has a throughput of 5.5 bits/second, while the throughput for the pan-based targeting method was 5.3 bits/second. The difference here is quite small, as also noted by the authors.

The authors conclude that using Fitts' law to model 3D pan-based target acquisition results in accurate and comparable results.

4.3 Fitts' law and game controllers

Fitts' law was present in the research by Natapov et al. [2009] where the authors evaluated the pointing accuracy of different video game controllers. The authors conducted an experiment where a Nintendo WiiMote, a Nintendo Classic Controller and a Logitech cordless optical mouse were compared in a point-and-select task on a large television.

The point-and-select task consisted of a home rectangle from where the subjects would point to a round target. After hitting the target or missing it, the home rectangle and the round target would reset and change places, and the subject would do the targeting again.

The results showed that, as expected, the mouse had the highest average throughput of 3.78 bits/second. The WiiMote had a throughput of 2.59 bits/second, significantly lower than the mouse. The throughput for the Classic Controller was 1.48 bits/second, 255% percent drop compared to the throughput of the mouse.

The authors conclude by recommending a WiiMote like device for interactive home entertainment equipment, even though the error rate for the WiiMote was the highest in the study.

4.4 Applicability of Fitts' law

Zhai [2002] points out that Fitts' law should only be used in its complete form, along with error rates, to ensure accurate results. Combining or leaving out dimensions might yield arbitrary results which in turn lead to misleading conclusions. Zhai also notes that Fitts' law is only used to model pointing tasks and other models should be used for more complex tasks.
Albinsson and Zhai [2003] note that modelling Fitts' law in their high precision touch input research did not produce a good fit. The authors note that the movements involved in their experiments were complex and would not fit well with a typical experiment involving Fitts' law. By fitting the data with Fitts' linear regression law, the authors present much lower regression values than normal, thus indicating a bad match with Fitts' law.

4.5 Summary

Using Fitts' law in a controlled study to measure pointing performance is an effective way to gauge of different pointing devices. However, the effectiveness of Fitts' law seems to be quarantined to simple pointing tasks, i.e. moving a cursor from point A to point B. Thus, the evaluation of first-person shooter game controls cannot be solely based on Fitts' law.

In the next Chapter research on touchscreen input will be presented.
5. Research on touchscreen input

Touchscreen devices allow users to naturally access a user interface. No external input mechanisms are needed when touching: pressing an icon or dragging items comes with natural ease. The field of touchscreen research relatively new, but also quite large. In this Chapter the touchscreen research related to this thesis is explored.

5.1 Advantages

Touchscreen devices offer many advantages. If the device's operating system supports it, the whole device can be operated through the touchscreen. For example, the iPod Touch has only four regular buttons: a home screen button, a power on/off button and two volume control buttons. The rest of the device can be controlled through the touchscreen. This allows the manufacturers to use the device's surface for the display instead of sacrificing display space for a control mechanism. Naturally, more information can be shown on a bigger display. In a phone with a regular keypad, the control area can take up almost half of the phone's surface area, whereas on the iPod Touch the display is if the surface area.

The touchscreen can also be used to take advantage of gesture recognition. For example, flicking or dragging on the iPod Touch's home screen will cause the screen to change into another screen. Using gestures can be an effective and intuitive way to command the user interface.

Albinsson and Zhai [2003] note that touchscreens are easy for novice users to use because they do not have to pay attention to the control mechanism. This means that the users can operate the device without diverting their gaze back to the user interface.

5.2 Disadvantages

Albinsson and Zhai [2003] note that using fingers to manipulate the user interface will cause some of the display to become visually blocked. This is called occlusion and it is a widely studied topic in touchscreen research.

Touchscreens also completely lack haptic feedback. Writing on a regular keyboard will constantly give haptic feedback on where the user's fingers are, whereas on a touchscreen the user has to rely on visual or aural feedback to figure out whether his actions are registered. Hoggan et al. [2008] also note the lack of haptic feedback on touchscreen devices.

Finger-based input is also prone to invalid selections when the target is smaller than the finger. This problem is especially prevalent when a number of small targets is clustered together. Zooming has been presented as a way to solve clustering problems with touchscreen devices. For example, on the iPod Touch it is possible to zoom on the native browser program to access smaller objects on a web page, such as links or small images. Albinsson and Zhai [2003] note that while zooming allows for higher precision, it also makes the pointing task more complex.
5.3 Pointing and selection strategies

Potter et al. [1988] studied three different touchscreen strategies and how they performed in a controlled experiment. The motivation of the authors for this study was to present solutions for the lack of precision in touchscreen devices, and the high error rate which is caused by imprecise fingertip movement. Potter et al. approached these problems by implementing a control strategy where the user would drag the cursor on the screen while getting constant feedback of its position. The authors appropriately nicknamed this technique "finger mouse".

The first approach was called **Land-On**. In Land-On, the purpose is to use the first finger contact with the screen as the selection. If the subject's finger doesn't hit a target, nothing is selected. The subject has to lift his finger and try again. Dragging the finger has no functionality. The targeting cursor in Land-On was positioned directly under the finger.

**First-Contact** was the researchers' second approach. As the name implies, in this technique the selection is made based on the first contact of the finger. The cursor can be dragged if the finger touches blank space first. The first target, which is touched by the finger when dragging, is then selected.

The third and final technique was called **Take-Off**. The key difference in Take-Off is that the selection is made when the finger is lifted from the screen, thus allowing for fine-grained accuracy. The researchers also modified the cursor location so that it was positioned half an inch above the finger. Also, the target was highlighted when the cursor was on top of it. The targeting cursor was also stabilized so that subtle finger movements wouldn't interfere with targeting.

The results of the study show that, of the three techniques, the Take-Off had the lowest error rates for target selection. Errors were reported for hitting the wrong target or hitting blank space. However, the First-Contact technique was significantly faster than Take-Off. Some subjects did not like Take-Off's differently positioned cursor, as they expected the cursor to be directly under the finger. The authors conclude by noting that simple and direct techniques, such as Land-On, are more efficient for targeting tasks where the targets are bigger, but when the target size gets smaller, the need for more accurate strategies, like Take-Off, arises.

5.4 High precision pointing and selection strategies

Albinsson and Zhai [2003] present multiple high precision touchscreen strategies in their study. Their first technique, **Cross-Lever**, gives the user two intersected virtual rubber-band lines which can be used to control an activation area. Even though the technique allows users to pin point targets which are one pixel in size, the authors concede that the technique is too time consuming to use. Cross-Lever requires one to visualize how the activation area moves when the rubber-band line is moved, and then physically move one line at a time to move the activation area.

Their next attempt, **Virtual Keys**, is a control mechanism with arrow keys. It is identical to the arrow keys found on a regular keyboard. The arrows are placed on the side and it can be used to
move the target crosshair. In this approach the user constantly has to watch the virtual arrows on the side in order to realign his fingers. Albinsson and Zhai argue that the Virtual Keys approach would work better with a tactile feedback mechanism.

Cross-Keys is the authors' third attempt. In this design the virtual arrows devised in Virtual Keys are placed directly on the display with the targeting reticle in the center of the arrows. The arrows can be moved by tapping handles on the side. Now the controls are directly on the screen and the user can directly manipulate them without gazing at the sides. However, the handles can be displaced if the user moves the arrows too close to the edge of the screen. This causes the handle system to break.

The fourth attempt, 2D Lever, is a simplification of the Cross-Lever technique. The 2D Lever has a handle, a pivot point and at the tip, a crosshair. The 2D Lever is first inserted near a target, then the handle can be used to manipulate the crosshair at the pivot point. When the tip of the lever is placed on the target, the user can press the surrounding activation circle to select the target. The authors report that while the 2D Lever is faster than the Cross-Lever, it still doesn't reach the efficacy of Cross-Keys.

The fifth, and last, attempt is called the Precision-Handle. The Precision-Handle is a simplified version of the 2D Lever. The pivot and the inverted movement features were removed from the 2D Lever, whilst keeping the precision targeting.

Albinsson and Zhai compared their two of their approaches, Cross-Keys and Precision-Handle, against two other well-known high precision targeting techniques, ZoomPointing and Take-Off. A user study was conducted for comparing these techniques.

ZoomPointing performed well, as expected, and was especially good in the selection of smaller targets. Take-Off wasn't as good as the new techniques when it was used to select small targets. The authors note that the error rate was high for Take-Off, and also its speed was lacking when compared to Cross-Keys and Precision-Handle. However, the results show that Take-Off was faster than the other techniques in the selection of larger targets, and the authors note that the subjects liked using Take-Off.

Cross-Keys excelled in the selection of small targets while having a low error rate. Occlusion proved to be a problem with Cross-Keys, as the subjects reported having difficulty seeing the crosshair or the handles while using it. Precision-Handle's performance was satisfactory in speed and accuracy for small and large targets.

Finally, Albinsson and Zhai note that none of these techniques is clearly better or worse for selection tasks. The authors state that it is better to use a certain technique for tasks it is better suited for, rather than trying to force one technique to cover all possible selection tasks.
5.5 Tactile feedback in touchscreen devices

Hoggan et al. [2008] conducted a study comparing text entry between a physical keyboard, a standard touchscreen and a touchscreen with tactile feedback. The motivation behind the study was to find out the effects of implementing artificial tactile feedback in a touchscreen device.

The experiments of the study were conducted in a lab and a mobile environment. The results of the study and the observations made by the authors indicate that tactile feedback can significantly improve fingertip interaction with a touchscreen device. For example, in a mobile environment, the device with tactile feedback enabled had noticeably better accuracy scores, sometimes up to 74% better than the device without tactile feedback. Text entry speed also suffered, as users with the tactile feedback enabled device could enter phrases much faster (up to six times faster) than the users with the standard touchscreen device.

The authors conclude that touchscreen manufacturers should include tactile feedback in new devices, as there were only positive effects from it in their controlled study.

5.6 Multi-touch pointing and selection strategies

Benko et al. [2006] present five multi-touch selection strategies, called Dual Finger Selections, in their study. The purpose of these strategies is to enable users to select very small targets. The authors note that the lack of precision and the fact that fingers can block smaller targets are major improvements points in target selection techniques for touchscreen devices.

The authors introduce key concepts related to their techniques. The finger, which makes the first contact with the screen, is called the primary finger. The authors note that the primary finger is the finger subjects point with, and is usually the index finger of the dominant hand. The secondary finger is the finger on the helper hand, and it can be any of the fingers. Benko et al. observed that the subjects also used the index finger on the helper hand. In Dual Finger Selections, the secondary finger handles controls which assist the primary finger.

Dual Finger Offset is a simple technique where the cursor is slightly offset next to the primary finger when the secondary finger is placed on the screen.

In Dual Finger Midpoint the cursor is placed between the primary and the secondary finger. Thus, when the fingers are in movement, the cursor moves automatically and stays between the fingers while maintaining the same speed as the fingers. If only one finger is moving, the cursor will move at half the speed of the finger. Benko et al. note this gives the subjects fine-grained control over the cursor. The downsides of this technique are the inability to access user interface icons in corners and the lack of precision with targets sized 2 pixels or less. The authors acknowledge that the inability to reach corners is a major disadvantage as small user interface icons often reside in the corners of the screen.

Dual Finger Stretch provides the ability to zoom and select in the user interface. The authors note that this technique is similar to ZoomPointing in the work of Albinsson and Zhai [2003], but
comes with some notable differences. First, the act of zooming and selecting is concurrent in Dual Finger Stretch, thus making the interaction smoother. Second, the manipulated area in Dual Finger Stretch zooms into every direction. Benko et al. note that this ability eliminates the need to "capture" the target in a rectangle, like in ZoomPointing. In Dual Finger Stretch, the subject can place their finger directly on the target and zoom in, which shortens the amount the primary finger needs to move.

*Dual Finger X-Menu* is designed to give users the ability to control the cursor movement speed and offset. As the name implies, this technique is driven by a user controlled menu. The menu appears when the secondary finger is placed on the screen. It has six functions. In normal mode, the cursor will move at the same speed as the primary finger. In slow 4X and 10X modes, the movement of the cursor is slowed by a factor of 4 or 10. In freeze mode, the cursor is completely frozen and movement of the primary finger with regards to the cursor is ignored. By using the snap function, the cursor offset is removed and it will reset to the current location of the primary finger. The last function is the magnify option. When it is selected, the current area under and near the primary finger is zoomed in and highlighted inside the Dual Finger X-Menu.

*Dual Finger Slider* is a combination of the Dual Finger X-Menu and Dual Finger Midpoint techniques. In this technique the Dual Finger X-Menu modes, which change the cursor's speed, are controlled by the positions of the primary and secondary fingers. Moving the secondary finger closer to the primary finger will trigger consecutively slower modes, until reaching the freeze mode. If the fingers are moved in opposite directions from each other, then the snap mode is engaged and the cursor offset is removed.

All of these techniques were also accompanied by a pressing technique called *SimPress*. The authors developed SimPress (Simulated Pressure) to simulate a pressure-sensitive device because their touch input platform does not recognize pressure. In essence, the finger's contact area is used to simulate pressure. When just the edge of the fingertip is used, it is used to indicate the user is hovering or tracking. Pressing with the full fingertip will registered as dragging. In order to register a click, the user needs to rock the finger gently, essentially moving from the tracking state to the dragging state in succession.

A controlled study was conducted to measure the effectiveness between Dual Finger Offset, Dual Finger Stretch, Dual Finger X-Menu and Dual Finger Slider. The results show that the error rates for Dual Finger Offset were the highest, while Dual Finger Stretch had the lowest error rates. Also, Dual Finger Stretch had the fastest mean performance time, and it was most the most preferred technique in the subjective evaluation. The authors note that these results follow the observations made in the work by Albinsson and Zhai [2003], where a similar technique, ZoomPointing, was strong in the controlled study.

However, Benko et al. note that they were pleased with Dual Finger X-Menu's and Dual Finger Slider's low error rates, which were comparable with Dual Finger Stretch's error rates. Also, the
authors note, the speed difference between these techniques was not huge, as the Dual Finger Stretch technique's mean performance time was one second faster on average.

In conclusion, the authors note that their results show that Dual Finger Selections provide increased precision and accuracy for smaller target selection tasks, and application designers could fully use these techniques in different contexts. It is though highly unlikely that any of these approaches is a good fit for a fast paced FPS game. If anything, the techniques clearly illustrate that even getting precision pointing and selection is hard in touchscreen devices. What happens when precise pointing tasks are combined with tasks which require the user to make decisions under a time pressure?

5.7 Unimanual and bimanual interaction

Moscovich and Hughes [2008] study multi-touch interaction using one and two hands. Also the authors present the results and analysis of two experiments regarding multi-touch interaction.

Again, the authors start by noting the same issues as observed in Section 5.2: occlusion is a major problem and precise selection with fingers is difficult. Specifically relating to multi-touch, the authors ponder the relationship between the control structure of the input device and the structure of the task. This means figuring out how to design the mapping of a multi-touch interface for a specific task. As an example the authors use Etch-A-Sketch, a mechanical drawing toy. Etch-A-Sketch has a stylus which is controlled by turning two knobs, one for moving horizontally and one for moving vertically. This is an example of an indirect mapping, where the task of drawing is accomplished by manipulating the stylus by knobs, rather than directly controlling the stylus by hand.

Moscovich and Hughes note that using one finger on each hand is basically bimanual (two-handed) interaction. Thus, research related to bimanual interaction can be used to study multi-touch tasks involving two hands. Key differences between the kinematics (motion of points) of bimanual and unimanual (one-handed) finger-interaction are also made. The authors state that the fingers of the same hand inherit the hand's motion, meaning the movement of the fingers is related to the hand's frame of reference. On the other hand, the movement of two hands need to be controlled relative to each other, or relative to a global reference frame. Moscovich and Hughes note that while unimanual finger-interaction may be more easily coordinated, the movement and motion of separate hands may be more easily uncoupled.

With regards to multi-touch input mappings, the authors note that specific mappings need to be done for both unimanual and bimanual interactions in order to achieve full effectiveness. The results and the analysis of the experiments show that unimanual interaction excels in visual rotation tasks. For example, transporting, rotating and stretching an object is an effortless task for one-handed manipulation. This can be seen in effect in the iPod Touch where zooming is done by stretching the fingers. The authors state that bimanual interaction is good for object manipulation tasks where the
resulting actions are clearly correlated with the manipulation of the control points and the motion of the fingers. The authors warn that if this correlation is missing, degraded performance can be expected.

Moscovich and Hughes observe another indicator for multi-touch mapping design from the experiments. Tasks, which have two separate control points, are better suited for bimanual interaction. For example, the previously mentioned Etch-A-Sketch drawing task falls into this category. The authors list example tasks for two-handed interaction: window manipulation, marquee selection, image cropping, control of separate objects. The authors note that these tasks are also *perceptually compatible* with bimanual control.

In conclusion, Moscovich and Hughes argue that performance differences between unimanual and bimanual multi-touch interaction are small if the task in question has no clear separation of control points, yet is still compatible with both one- and two-handed control.

### 5.8 Summary

It is clear that touchscreen devices offer increased usability especially when the target size is larger than the user's finger. As shown in previous sections, having smaller targets will instantly make pointing and selection tasks much harder. The researchers explored different approaches for effective pointing and selection strategies. Some of these were suitable for tasks like high precision selection and others for fast pointing and selection. None of these strategies was clearly superior for all tasks. It is then interesting to see how game developers have approached the previously mentioned challenges. As discussed in Chapter 2, first-person shooter gameplay contains multiple concurrent interaction tasks, not just static pointing and selection tasks.

It is interesting to note that using multi-touch input in a device is not always a net positive. As shown earlier, some tasks are clearly suited better for one-handed input rather than two-handed input. Performance degradation can be expected if the control points for two-handed input are not designed appropriately.

In the next Chapter research on game controllers and first-person shooter games will be explored.
6. Research on game controllers, first-person shooter games and player experience

Games can be played with various input devices, but the most common are either a combination of the keyboard and the mouse or the gamepad. These were introduced in Chapter 2 and here the research regarding these devices is explored. The discussed research can be then used as a base for creating heuristics for touch input controls in FPS games. Frame rate and usability research in first-person shooter games is also explored. Research on player experience will also be discussed.

6.1 Game controllers and 3D environments

Klochek and MazKenzie [2006] explore the differences between input devices in a 3D environment. The authors present five new performance metrics and utilize two tasks when measuring input device performance.

Klochek and MazKenzie note that when using a keyboard the granularity of the movement speed is binary, meaning that the player either is moving or has stopped. This means that pressing forward or backward will cause the player to go full speed immediately. On the consoles, however, the movement is more fine-grained due to the analog sticks (as discussed in Chapter 2).

A key distinction between types of weapons in FPS games is made. A weapon can be a scan-hit or hit scan weapon, meaning that the velocity of the weapon is infinite. This means that the projectile fired by the weapon will reach its destination instantly.

A projectile weapon, however, will fire projectiles which will take time to reach the target. If the target is moving, the player needs to use a technique called leading to anticipate where the target will be. For example, if the target is moving to the right, the player needs to target to the right of the target, not at the target. The authors call this target the mental proxy of the real target. The amount the player needs to lead is dependent on the velocity of the projectile and the target.

Klochek and MazKenzie introduce two tasks which can be used to measure target tracking. The first task is called Constant Velocity. It can be used to measure targets which do not accelerate. This means that the targets will appear to move at the same speed. The authors note that this task immediately shows differences between a zero-order (e.g., mouse) and first-order (e.g., joystick) device. A first-order device can track the target's movement without further adjustments from the user. A zero-order device needs to be constantly adjusted to match the movement. The zero-order and first-order device differences are similar to the position-control and rate-control issues mentioned in Chapter 2. The mouse, being a position-control device, can directly position the camera, so the user needs to constantly re-position the camera while tracking a target. However with a rate-control device, like a gamepad with analog sticks, the user can subtly change the rate of the camera's movement eventually matching the movement of the target. The authors state that if the Constant Velocity task is to be used in an experiment, the targets should be distributed so that
they can be used to measure the device's tracking capabilities in different directions. The speed of
the tracking target should also be manageable by the device, but not too slow, because then target
tracking becomes too easy.

The second task is called Variable Velocity. In this task, the targets oscillate on a specified axis,
while also gaining acceleration in speed and direction. This means that both zero-order and first-
order devices need to be constantly re-calibrated in order to track the target because the target's
speed is not constant.

As noted before, Klochek and MazKenzie also developed five new performance metrics. The
authors also present two existing metrics from the literature. The authors state that these metrics
were developed to find out why different input devices differ in target tracking. The accuracy of the
track is the main measurement. Klochek and MazKenzie note that tracking accuracy contains not
only the time spent tracking the target correctly, but also measuring the error in acceleration and
velocity. Solid smoothness and consistency values are also good indicators of accurate tracking.
Measuring the amount of time to reacquire a target after losing track is also important, as it will
give information about the responsiveness of the input device.

The first performance metric the authors present is Time on Target (TOT). Time on Target was
originally developed by D.C. Ellson [1947]. In this metric, every time the cursor is inside the
target's radius (in a single sample), it is counted as "in". The resulting TOT metric is then calculated
by summing the "in" count, and then diving it by the total number of samples.

The second performance metric is called Mean Target Distance (MTD). The authors note that
this metric can be found from Poulton's [1974] work. This metric reveals the average distance
between the user's reticle and the target. The authors state that even though this metric is correlated
with Time on Target, it is more useful in describing how close the user was to the target.

The third performance metric is Mean Time-to-Reacquire (MTR). This metric is designed to
measure the mean time it takes the subject to reacquire the target after losing the track.

The fourth performance metric is Mean Speed Variance (MSV). The authors note that if the
value for Mean Target Distance is zero (perfect tracking), it means that the position of the target and
the reticle are identical. From this follows that the perceived velocity of the target, which is being
tracked, should also be zero for a perfect track. This means that the target is at the same point in the
screen at all times.

The fifth performance metric is called Mean Acceleration Variation (MAV). Klochek and
MazKenzie note that this metric is similar to Mean Speed Variance, because for a perfect track, the
perceived velocity must be zero, and if the perceived velocity is zero, then the perceived
acceleration must also be zero.

The sixth performance metric is called Target Leading Analysis (TLA). Klochek and
MazKenzie state that the subjects might experience lag, which is induced by the characteristics of
the input device, while tracking a target. The Target Leading Analysis metric can be used to
measure this effect. The authors note that calculating this metric in screen-space is difficult because the projected motion of the target can vary depending on how the subject adjusts the view. Thus, the calculations should be done in world-space, where all related variables may remain constant.

The seventh and final metric presented by the authors is called Percentage View Moving (PVM). The authors state that, while tracking a moving target, the view will also be moving constantly. A still view indicates an error in tracking which should be reported. Measuring view change is done by setting a threshold which, when the user moves the input device, should be crossed. Movements below or at this threshold should be reported as errors.

An experiment between an Xbox gamepad and a regular mouse using the previously mentioned tasks and metrics was conducted. As expected, the mouse provided a better accuracy. The authors postulated that because mean acceleration (MAV) for the gamepad was lower than in the mouse, it would indicate that the gamepad would be at least as effective as the mouse. The results, however, proved otherwise, as the authors noted. The subjects were not able to quickly adjust their tracking with the gamepad, whereas with the mouse, the high acceleration rate proved to be beneficial in tracking. The MSV values were the same for both of the devices, meaning that both input devices were good at tracking the velocity of the target.

The subjects were also slower to reacquire a target (MTR) when using a gamepad. Klochek and MazKenzie note that the MTR value was significantly lower for the mouse. The PVM value for the mouse was lower than the gamepad, which would indicate a smoother track and better TOT and MTD scores for the gamepad. But again, the results showed something else. The subjects, when using a mouse, would stop and re-evaluate their position, then use the high acceleration of the mouse to quickly reposition the view.

Following these results, the authors make some key observations. Moving the camera (the view) in a 3D first-person shooter game is a position-based task. The authors note that, while using a gamepad, the subjects need to subconsciously calculate the velocity and acceleration of the analog stick's movement into a position. In addition, the movement needs to be done rapidly and accurately. Klochek and MazKenzie note that, judging by the results, this mental task proved to be hard for the subjects. The authors state that these differences between the gamepad and the mouse are due to the fact that the gamepad is a first-order device. These observations directly link to the points mentioned in Section 2.8., where it was suggested that while the mouse doesn't have fine-grained control like the gamepad, it offers superior targeting accuracy especially in fast moving environments.

The authors conclude by noting that the performance differences found in the study were related to the first-order nature of the gamepad. The authors also suggest that future studies on first-person games and gaming devices should not only concentrate on view related tasks, like the ones presented in the study, but also research translational motion (movement on a linear axis) on input devices.
6.2 Input devices in first-person shooter games

Isokoski and Martin [2007] conducted an experiment where the efficiency of different input devices in first-person shooter games was measured. The experiment was set inside a 3D world where the subjects would shoot spawning objects moving on random trajectories. The subjects had two consecutive sessions where they would shoot targets.

The input devices used in their study were a combination of keyboard and wheelmouse, the wheelmouse, a trackmouse and an XBOX 360 gamepad. To further clarify, a wheelmouse is a mouse with a scrolling wheel. As an example, the wheel can be used to scroll a web page. A trackmouse is a mouse with a track ball. The difference between a regular mouse and a trackmouse is that the movement of the cursor is handled by the trackball in a trackmouse. Thus, in order to move a cursor, a trackmouse doesn’t need to be moved at all. The trackball is usually thumb-operated.

The results of the experiment show that the combination of keyboard and mouse, and the wheelmouse, had the best hit count of the input devices. The trackmouse was the third and the gamepad last in the hit count comparison. Isokoski and Martin note that there were no statistically significant differences with the amount of missed shots between the input devices. However, the authors note that the combination of keyboard and mouse had a clear increase in missed shots in the second session.

Isokoski and Martin conclude by noting that mouse aiming is clearly more effective, being almost twice as effective as aiming with a gamepad. However, it is also suggested that further work is needed to understand how constant learning affects the efficiency of input devices in first-person shooter games.

6.3 Game controls on mobile devices

Zaman et al. [2010] present a study where the authors compare touchscreen game controls with physical game controls. The authors postulate that the lack of haptic feedback will show significant differences between the touchscreen and the physical device. The game that was tested was Assassin's Creed: Altair's Chronicles by Ubisoft [2008], which is available for Nintendo DS, PC and iOS platforms. Zaman et al. also note that the difference (in game play) between the Nintendo DS and iOS versions is negligible. That the game is cross-platform, and plays the same on different platforms, makes it an excellent candidate for a comparison study.

The game control performance is studied by conducting an experiment where the subjects play the same level in the aforementioned game. The game devices in question are the iPhone 3G and the Nintendo DS Lite. Both the death count and the completion time per trial are tracked. Learning effects, which were mentioned in the previous section, are mitigated by counterbalancing the order in which the devices are played. The authors also conducted a post-experiment questionnaire for gathering qualitative feedback.
The quantitative results show that the iPhone's virtual controls, indeed, have worse performance. Both the death count and the completion time values were worse for the iPhone. The results showed, though, that as the trials progressed the iPhone's control performance improved faster than the performance of the Nintendo DS Lite. The authors note that the initial performance for the iPhone was severely worse than for the Nintendo DS Lite.

The post-experiment questionnaire revealed that all the subjects preferred playing the game on the Nintendo DS Lite.

Zaman et al. conclude by noting that, initially, touchscreen devices are much worse for gameplay performance. The authors state that even though the performance with touchscreen devices will improve quickly with practice, physical controls still offer superior performance. The results reinforce their conclusions and confirm their original hypothesis that the lack of haptic feedback for virtual controls in a touchscreen device will result in degraded gameplay performance.

It is to be noted that, even though the game in the authors' study is not a first-person shooter game, the results of the study are very interesting regarding this thesis. As first-person shooter games are quite complex, it can be postulated that the results of this study could have been even more lopsided, had the studied game been a first-person shooter game.

6.4 Effect of frame rates
Claypool and Claypool [2007] examine the effect of frame rates on player performance in first-person shooter games. Frame rate means the amount of times a game scene is rendered per second, thus forming the metric frames per second. The authors define player performance as a measurement of the number of times the player kills enemies and/or the number of times the player dies. Claypool and Claypool note that lower frame rates might be acceptable for watching, for example, streaming video, but the high interactivity of FPS games demands increased frame rates. A low frame rate affects the game play, because it will diminish the amount of the information the player can see graphically on the screen. Thus, the movement of objects in the game world will be seen as choppy. A high frame rate will provide smooth object movement, whereas with a low frame rate objects will be seen as jerking and jumping from one position to another.

Claypool and Claypool observe three game phases for FPS games. First, the Setup phase, where players can select maps, avatars, teams etc. It is to be noted that the Setup phase can be complex or simple depending on the parameters the game offers for the players.

Second, the Play phase, where the actual game play happens. In FPS games, this means shooting, moving around and achieving objectives.

Third, the Transition phase, where game information is loaded and unloaded. For example, when entering a new level, the game will display a loading screen while the level resources are loaded in the background.
Claypool and Claypool note that for the Setup and Transition phase the requirements for frame rates are lower, as they do not require much player interaction. However, as is natural, the frame rates should be high for the Play phase. In the Play phase, the authors note, player actions are most sensitive to drops and variations in frame rates.

As also observed in Chapter 2, the authors note that the game play in the Play phase in FPS games mostly consists of shooting and moving. Claypool and Claypool also define two properties for actions in the Play phase. Deadline is the time required to complete an action. An example of a deadline is the time it takes a player to place an enemy in the reticle and shoot. Precision is the degree of accuracy required to complete an action successfully. As an example the authors use the sniper rifle in Battlefield 1942 [Digital Illusions CE, 2004]. The precision for this weapon is the size of the enemy.

Claypool and Claypool note that shooting in FPS games in general is a high precision activity with tight deadlines. For example, the area of effect for a sniper rifle is small (for an enemy which is at a distance), meaning it requires high precision. If an enemy moves fast, then the deadline for shooting that enemy is tight, because the time to acquire an target and to execute a shot is short. Combining these two means that firing a sniper rifle at a fast moving enemy, which is at a distance, requires the player to execute with high precision on a tight deadline. Conversely, shooting a weapon with a large area of effect (bazooka, mortar) at a still opponent does not require high precision shooting or actions done under a tight deadline.

According to the authors, movement in FPS games requires high precision, but not necessarily tight deadlines. This is explained by the time it takes to move to another location, which can take many seconds. Claypool and Claypool note that precision and deadline requirements for movement can vary, as running through a twisty path is different than slowly walking in a straight line.

A controlled study was organized to test two hypotheses by the authors. First, the higher the precision requirement for a given game action, the bigger the impact of frame rate on player performance. Second, the tighter the deadline requirement for a given game action, the bigger the impact of frame rate on player performance.

The results of the study show that, indeed, frame rates play a big part in player performance. Low frame rates (below 7 frames per second) were deemed unplayable, as the players couldn't target opponents properly. The authors note that the performance benefit of 60 frames per second compared to three frames per second is over seven fold. However, degraded frame rates did not impact player movement performance as much.

In conclusion, the authors state that higher frame rates play a much bigger role in FPS games than was anticipated. Low frame rates decimated the player performance and enjoyment of the game, leading to the conclusion that high frame rates should be an important goal for each game designer.
6.5 Effects of screen resolution and frame rates

Claypool and Claypool [2009] study the effect of both frame rates and resolution on game actions, namely shooting and navigation. Resolution means the amount of pixels that are displayed on the screen. For example, a 800 x 600 resolution means 800 (width) times 600 (height) pixels can be displayed. Thus, a higher pixel resolution means that more information can be presented graphically to the user. The drawback is that the rendering a higher pixel resolution requires more from the rendering hardware, which can then cause variations in the frame rate. In addition, the perspectives, which were defined in section 2.5, are also included in the analysis. A controlled study was conducted to measure the effects.

As the present study touches on some of the same ideas as discussed in Claypool and Claypool's [2007] previous work, only new findings are reiterated here. Here, the authors observe that frame rate has a bigger impact on user performance than resolution. Also, users performed better when using the third-person isometric perspective for shooting and navigating. Claypool and Claypool note that this can be attributed to the fact that, in this perspective, users are able to see more of the game world, thus enabling users to rapidly acquire targets or objects.

In conclusion, the authors note that the results and observations in this study follow their previous studies. Low frame rates can significantly damage both the player experience and performance in an FPS game, and conversely, higher frame rates will improve both the player experience and performance.

6.6 Usability issues

Pinelle et al. [2008a] study what usability issues rise in different game genres. The authors examined reviews of 18 PC games from each of the six major game genres. In total 108 game reviews were analyzed. The genres were role playing, sports, shooters, action, strategy, and adventure.

The authors present different usability profiles for each genre based on the problems found in the game reviews. Interestingly, consistency and controls rose as top issues among the subjects in the game tests. The authors note that input mappings were not seen as problematic, but rather the sensitivity of the game controls. Consistency issues manifested in scenarios like opponents shooting through walls and opponents shooting at the player while seemingly looking in another direction.

Pinelle et al. note that standardized input mappings for shooter games help players to easily move from a shooter game to another. Standard input mappings were discussed in Chapter 2 of this thesis. However, shooters, and action and sports games, often suffer from either under or over-sensitive controls, according to the authors. In this context, under sensitive controls means slow or inaccurate responses to user's actions. For example, an action meant to fire a projectile might happen seconds after the player has initiated the action. Over-sensitive controls behave in the
opposite way. For example, moving an over-sensitive mouse might move the view too fast, making effective target tracking and leading a frustrating task.

To combat these usability issues, Pinelle et al. suggest using guidelines and heuristics when evaluating a game. These methods can be accompanied with the previously mentioned game profiles in order to enhance the evaluation for a game in a specific genre. To conclude, the authors note that spending time and effort to thoroughly evaluate the usability of the game will provide significant improvements to the overall quality of the game.

6.7 Player experience

Cox et al. [2012] explore immersion in digital games. The authors start by introducing the concept of Flow, which is an experience where optimal challenge, skill and focus are combined. In the context of games, this means that the player is constantly experiencing challenges relative to his skill level. For example, if the game is too easy, expert players might get bored. On the other hand, if the game is too hard, novice players might quit in frustration.

Cox et al. conducted three studies in order to discern what affects player immersion the most. The results did suggest that the balance of challenge and player skill is important. However, the authors note that the player's perception of his own skills in relation to the challenge is also essential. Cox et al. conclude by stating that the feeling of immersion does not simply arise from quickly completing physical challenges, but from the cognitive challenge of solving problems.

Gerling et al. [2011] studied the effects of game controllers on player experience in FPS games. The authors tested the same game on the PC and console platforms. The results indicated that, even though players switching from their comfort platform to a new platform did experience usability issues, their overall enjoyment was positive. Gerling et al. conclude by noting that instead of adjusting the usability of FPS games on particular platforms, the efforts should go to exploring and designing the mechanics of FPS games.

Poels et al. [2007] present a categorisation of game experiences. The experiences have been divided into nine dimensions: enjoyment, flow, imaginative immersion, sensory immersion, suspense, competence, negative affect, control and social presence. The dimensions also have two aspects, in-game and post-game experiences. For example, suspense involves in-game feelings such as challenge, tension, and anxiety. The authors note, that even though this categorisation is tentative and not yet fully comprehensive, it can be still used to by game theorists and developers as a starting point for understanding game experiences.

6.8 Summary

The mouse and keyboard combination is clearly a dominant one. It was theorized that, when tracking and leading a target, a gamepad should offer similar performance to a mouse. However, results show that the mouse again dominated in these tasks. The ability to quickly position the view with the mouse is clearly a major advantage in first-person shooter games.
The difference between physical and touchscreen game controls is also interesting. Not only did the physical controls provide better game control and performance, the users in the study all preferred physical controls over touchscreen controls. This finding has a significant impact on how game developers should approach game and control design on a touchscreen device.

Summarizing research concerning first-person shooter games is rather easy: provide good frame rates even at the cost of resolution size, or else the enjoyment of the game itself will be degraded. FPS games have a lot of action happening concurrently on the screen, so it is not a simple task to provide excellent frame rates, especially if the graphics of the game are advanced. It is interesting to see how game developers have approached this issue on the iPod Touch platform.

Research on player experience was also briefly discussed. The main points from this research concern the effect of challenge and player skill on player experience. By having a good balance of challenge and skill, the player is able to immerse in the game and achieve a state of flow. A categorisation of game experiences was also presented. These nine dimensions can be used to roughly gauge the experiences or feelings the player can undergo during gameplay.

In the next Chapter selected first-person shooter games will be evaluated on the iPod Touch.
7. Game evaluation

This Chapter will present the evaluation of selected first-person shooter games on the iPod Touch. In order to fully understand how the controls in an FPS game work on a touchscreen device, it is necessary to look at the existing games on the platform and evaluate their controls. It is most likely that different developers have taken different approaches to solving the issue of controls. Also research regarding heuristics is presented and discussed. Completely new heuristics will be designed and then tested against the selected games.

7.1 Research summary

The previous Chapters presented a lot of new concepts, which is to be expected as this thesis touches on many different aspects, from touch input to game controllers to first-person shooter games. Here a summary of the related research is presented.

As a recap, Fitts' law is suited for measuring simple pointing tasks. For example, tasks where the subject points and selects rectangles in successive tries. But as has been discussed and shown in this thesis, first-person shooter games contain rather complex user interactions. The user has to move the avatar, acquire fast moving targets, perform mental leading on the acquired targets and shoot projectiles – all at the same time. Thus, designing FPS control evaluation heuristics solely based on Fitts' law is not ideal. However, other researchers, such as Looser et. al [2005], have shown that the FPS user interaction can be dissected into smaller pieces, where Fitts' law can then be successfully applied.

Another main topic for this thesis is the iPod Touch's touchscreen. Research based on finding high precision selection techniques for touchscreens was presented. It is important to observe what issues crop up when a traditional input mechanism, such as the mouse, is replaced by touch input techniques. For example, targeting and selecting smaller targets becomes a challenging task, requiring researchers to develop roundabout ways to make this task reasonably fast and pleasant.

It is important to note that the presented touchscreen techniques mostly considered targeting and pointing. As shown in Chapter 2, the player in a first-person shooter game will have to simultaneously move the avatar and acquire rapidly moving targets. As it stands, none of these techniques can be used as a baseline metric for evaluating FPS game controls, but rather they can be used as data points for developing a specialized game control heuristics.

Research on multi-touch was also explored. The research on unimanual and bimanual interaction presented differences in performance between the two interaction modes. Unimanual interaction was effective for visual rotation tasks, whereas bimanual interaction excelled in object manipulation tasks. It is to be expected that the FPS games on the iPod Touch will almost certainly have bimanual controls. It is nearly impossible to map all the actions in an FPS game to a single hand, unless the available actions in the FPS game are limited. For example, in games which
employ a fixed movement system, the player needs to only concentrate on shooting. However, this thesis will only examine FPS games which have a free movement system. Thus, it is good to keep in mind what the researchers suggested when designing a bimanual control scheme: performance degradation can be expected if the visual actions in the system do not correlate strongly with the actions the user is making.

Game controller definitions and performance metrics were presented. Target leading and acquisition are obviously key concepts. In FPS games, most enemies will be moving, instead of being stationary, so naturally the player needs to acquire targets and then lead them in order to successfully fire and hit a projectile. Assuming, of course, that all the weapons in the game are not hit scan weapons, which eliminates the need for leading. The research also showed how moving the camera view is a position-based task, and how different input devices perform when performing this task. As manipulating the camera view is an important part of FPS game play, it will be taken into account when designing the evaluation heuristics.

The performance metrics were explored to give a sense of what is important in target tracking and leading. The mathematical nature of these metrics won't be analyzed in this thesis, but the essence of the performance metrics will be taken into account when designing heuristics for game controls evaluation.

It is important to note how dominant the mouse is in the controller studies. Coming in second is the gamepad, but with notably worse performance. The mouse's performance advantage leads to several questions. Is it wise to directly map FPS game controls, or even FPS game design, directly from the PC platform to the iOS platform, because the main components (mouse and keyboard) for an enjoyable FPS game play experience are missing? Will the absence of these devices completely ruin the game play experience, or is the experience tolerable? What about experienced FPS players and novice players; will the lack of high precision affect the other group more or less? Some of these questions were explored in the research of Zaman et al. [2010], where the researchers concluded that physical game controls clearly improved the overall game play experience when compared to touchscreen game controls.

Finally, some research on first-person shooter games was explored. It is quite straightforward to summarize this research: FPS games need to have good frame rates in order to provide the player an enjoyable game experience. This distillation can, and will, be made into a metric in the following evaluation heuristics.

Pulling all the presented research together, it is obvious that the following work will need to consider several options when designing FPS game controls for touchscreen devices.

First, touchscreen control design can be improved, for example, with back-of-device controls, as discussed in Zaman et al. [2010]. However, actual hardware design improvements are outside the scope of this thesis.
Second, first-person shooter game design in touchscreen devices should be examined. That is, should FPS games and their designs be directly mapped from their original platforms (PC, consoles) to smaller target platforms (mobile, tablet)? Do players expect the same kind of experience or difficulty levels on mobile platforms? Can the rules of the game be relaxed so that the original game design will be still intact, but the limitations of the target platform will be taken into account? These questions are interesting to look at, but they are not the main emphasis of this thesis.

Third, should first-person shooter game controls be directly mapped from their original platforms (PC, consoles) or can the touchscreen be used to innovate a complete new way to play FPS games? Can effective and meaningful heuristics be developed solely for touchscreen device games, or even for a sub-category of games on a touchscreen platform? These topics will receive great attention in this thesis.

7.2 Existing heuristics

Before fully delving into the design of heuristics, it is important to look at how other researchers have approached the task of game evaluation. Researchers have designed and invented many heuristics, one of the most famous being Nielsen's usability heuristics. Keeping with the theme of this thesis, only heuristics regarding game evaluation, and specifically game controls, are explored here.

Korhonen and Koivisto [2006] present three sets of heuristics in their study of mobile game evaluation. They are for game usability, mobility and gameplay. As this thesis focuses on controls and their usability, naturally the heuristics for game usability are most helpful. The game usability heuristics contain 12 rules or guidelines, ranging from UI advice to control design. The game control heuristics are as follows:

- 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.

These heuristics can used as a very high level evaluation guideline for game controls. However, as this thesis is focused on a specific genre on a certain type of a mobile device, the heuristics should also refer to the properties and characteristics discussed in previous Chapters.

Pinelle et al. [2008b] present usability heuristics for games based on the analysis of 108 PC game reviews. 12 usability problem categories were also identified in the process. Examining the problem categories is a good place to start. The categories related to controlling issues are as follows:

- Unpredictable or inconsistent response to user's actions.
- Mismatch between camera or view and action.
- Clumsy input scheme.
• Difficult to control actions in the game.
• Command sequences are too complex.
• Response to user's action not timely enough.

It is interesting to note how half of the problem categories are related to control issues. Considering the reviewed games were on the PC, which has excellent input devices, it is easy to postulate that these issues might be aggravated on a mobile platform. The authors devised 10 game usability heuristics following the specification of the problem categories. The heuristics related to game controls are as follows:

• 1. Provide consistent responses to the user's actions.
• 4. Provide unobstructed views that are appropriate for the user's current actions.
• 6. Provide intuitive and customizable input mappings.
• 7. Provide controls that are easy to manage, and that have an appropriate level of sensitivity and responsiveness.

These usability considerations are quite similar to the ones proposed by Korhonen and Koivisto [2006]. They are good for a high level review, but again, they do not contain detailed guidelines which are needed for this thesis. For example, the usability heuristic number 7 recommends controls that are easy to manage, but how do you define “controls that are easy to manage” in a mobile context for a first-person shooter game?

Unfortunately, it seems that there currently aren't heuristics designed specifically for the topics covered in this thesis. It is then necessary to design specialized heuristics for the purpose of evaluating controls in FPS games.

7.3 Designing heuristics

It must be first decided what the heuristics will take into account. As discussed earlier, hardware improvements and direct game design issues are not within the scope of this thesis. These topics can be then left out of the heuristics. One can argue successfully that game design and game controls are tightly linked, but in this context game design is understood as a higher level concept. For example, does the game allow free movement or fixed movement, ranged combat or melee combat and so on.

The following heuristics will concentrate on the aspect of controlling the avatar's actions, and the resulting feedback on user's actions.

On a high level, the heuristics will need to consider first-person shooter game controls on a nontactile touchscreen device using bimanual interaction. On a detailed level, the heuristics will need to address issues concerning target acquisition, target tracking, target leading and avatar movement. What follows is an attempt to capture these requirements as heuristics:

1. *Enable the user to effectively acquire static or moving targets.* The player should be able to easily acquire targets, whether they are moving or static. The game should provide a reticle for capturing targets under sight. In essence the reticle is usually a dot in the middle of the
screen. This simple dot can be then used to target opponents and line up shots. This heuristic will be referred to as **Target Acquisition** in the evaluation.

2. **Enable the user to effectively lead moving targets.** The player should be able to easily lead targets when using projectile weapons. For example, if an opponent is moving to the right of the player, the player should be able to easily target at the mental proxy in order to successfully hit the actual target. This heuristic will be referred to as **Target Leading** in the evaluation.

3. **Enable the user to effectively track moving targets.** The player should be able to easily track moving targets without constantly needing to reacquire the target. For example, if an opponent is moving to right of the player, the player should be able to smoothly and consistently track the opponent without losing the track. This heuristic will be referred to as **Target Tracking** in the evaluation.

4. **Enable the user to effectively manipulate the view.** The player should be able to easily rotate the view or camera in any direction. View movement is an import element in game controls because it enables the player to observe the game world, and to complete the leading, tracking and acquiring tasks. The sensitivity of view controls should be customizable. If the view movement is too slow, the player cannot react to fast moving opponents or events at all. However, if the view movement is too rapid, the player cannot acquire or lead targets because the movement will not be smooth. This heuristic will be referred to as **View Manipulation** in the evaluation.

5. **Enable the user to effectively navigate in the game world.** The player should be able to navigate X-, Y- and Z-axes in the game world. For example, preventing the user to jump over obstacles will lead to frustrating situations, such as small rocks or objects completely stopping the advancement of the player. This heuristic will be referred to as **World Navigation** in the evaluation.

6. **Provide stable frame rates.** The game should provide stable frame rates, especially in action oriented sequences. Radically varying frame rates degrade player performance because the feedback provided by the game won't be stable and predictable. For example, if the frame rate of the game drops severely, the player will only get partial visual feedback for the duration of the frame rate drop, which might lead to unfavorable consequences for the player, such as death, loss of health et cetera. This heuristic will be referred to as **Stable Frame Rate** in the evaluation.

7. **Provide frame rates which are preferably over 30 frames per second.** The game should provide high frame rates. As discussed before, low frame rates lead to decreased player performance. Claypool and Claypool [2007] showed that the performance benefit of 60 FPS to 3 FPS is over seven fold. For example, if the game runs at 5 FPS, the player will
experience the game as choppy and jittery. This heuristic will be referred to as *High Frame Rate* in the evaluation.

8. *Make sure the user interface elements, which map to game controls, are clearly defined.* The game should provide user interface elements which are large enough for users to easily manipulate them. As discussed before, smaller user interface elements are harder to target and point. For example, if the size of the icons is smaller than the size of the fingertip, the player needs to then lift his fingers in order to discover the controls. This heuristic will be referred to as *Clearly Defined UI Elements* in the evaluation.

9. *Make sure the user interface elements, which map to game controls, do not obstruct the game view.* The game should provide user interface elements which are moderate enough in size, but not so large that they block significant parts of the game view. For example, if the user interface elements are too large, the player will block a good part of the game view with his own fingers. This heuristic will be referred to as *Non-blocking UI Elements* in the evaluation.

10. *Provide consistent and logical feedback to the user when control points are manipulated.* As is most likely, FPS games on the iPod Touch will have at least two control points. One for avatar movement and one for moving the view. The functionality of these control points should be clear to the user. For example, using the avatar movement controls should only move the avatar, they should not also move the view. Mixing the functionality between the control points will lead to decreased player performance. This heuristic will be referred to as *Consistent Feedback* in the evaluation.

11. *Provide customizable input mappings.* The player should be to change the default input mappings. For example, left handed players should be able to reverse the controls. The ability to change the sensitivity of the controls is also extremely important, as over or under sensitive controls will lead to degraded player performance. This heuristic will be referred to as *Customizable Controls* in the evaluation.

These heuristics provide a detailed method of evaluating first-person shooter game controls. They do not delve too deeply into implementation details, but are detailed enough to distinguish themselves from the heuristics presented in the previous section. The last heuristic was picked from the work of Pinelle et al. [2008b] because it is sound advice for every game, independent of genre, to have customizable input mappings.

### 7.4 Methodology

It would be tempting to compare the games on the iPod Touch directly with PC or console games, but as has been shown, the input devices for PC and console games are completely superior to the touch input screen in iPod Touch. Thus, a direct comparison study is not ideal.
In evaluating the controls, attention should be placed on how well the controls suit the platform, and how they assist the player in controlling his actions in the game. In order to have maximum effectiveness, the controls should adhere to the platform's strengths, while minimizing the weaknesses. As discussed before, controlling tasks such as acquiring targets, leading targets, tracking targets, hitting targets and moving the avatar are central in the context of FPS games.

The subjective evaluations were conducted in free form play sessions which lasted a couple of hours per each game. The evaluation was based on the heuristics presented in section 7.3. The observations made from the play sessions are shown in section 7.6 and analyzed in section 7.7.

All of the games were played in the easiest difficulty level. Difficulty level selection has implications for the gameplay, as some games might link the difficulty level to enemy movement. For example, the harder the difficulty level, the faster the enemies move. However, it is expected that acquiring and tracking targets will be in any case a hard task on the iPod Touch.

Any form of auto-aim or automatic target assistance was turned off, if it was possible. The aim of the evaluation was to study the effectiveness of the basic controls, so auto-aim needed to be toggled off. Otherwise the evaluation might be skewed in some direction because the game will subtly correct the mistakes evoked by the controls.

A scoring system for the evaluation based on heuristics was planned, but it was deemed too heavy for a subjective evaluation made by a single person. However, future work could contain a scoring framework based on the heuristics.

Three first-person shooter games were selected from the App Store. This amount provided just enough variety to see how different developers have approached the design of game controls. Future studies should include a larger sample size to fully capture most approaches to the design of game controls.

An analysis of the results of the evaluation is also presented. This analysis contains observations made from the application of heuristics and the state of game controls in first-person shooter games.

7.5 Selected games

This section contains the criteria for the selection of the games, the process of selection and the selected games.

7.5.1 Criteria

The game selection needs to be based on a solid criteria. These are the rules of the criteria:

1. First-person shooter games which allow free movement. Fixed movement shooters, such as Doom Resurrection by id Software [2009], will not be evaluated.
2. First-person shooter games where shooting and moving opponents are a part of the gameplay.
3. First-person shooter games where a single-player mode is available. The process of evaluation will be easier in a game with a single-player mode as there will be always be
opponents in the levels. An FPS game, which only has a multiplayer mode, would require
the evaluator to gather a group of people to play at the same time in order to have opponents
in the game.

4. First-person shooter games which are relatively bug free. The games should run without
constantly crashing or preventing the player from successfully completing gameplay related
tasks.

5. Emphasis on most recently released games. Game developers are bound to get better at
designing controls as they get more familiar with the development platform. Hopefully, user
feedback over the years will have also shaped the design process of game controls for
developers.

6. The games should be selected from different developers. A game studio might use the same
controls for every FPS game they develop. To avoid this issue, the game developer should
be different for each evaluated game.

These criteria already contain some evaluation of the games. For example, points 1, 2 and 3 can
only fulfilled if the game is played for a while. Selection can also be only based on game reviews,
but to guarantee that the games contain the elements the evaluation needs, the games need to be
examined beforehand.

7.5.2 Selection
The process of finding and selecting the games started from the Apple's App Store, but
unfortunately, the web interface in the store only allowed sorting by very generic genres, such as
“action” or “adventure”. No sub-categories for genres were present. Luckily, many game review
and aggregation sites exist today, such as Slide To Play. Selecting the category “first-person
shooters” and ordering them by release date yielded a good crop of potential candidates
[SlideToPlay, 2012].

Three FPS games were selected from this list. As discussed before, rail shooters and multiplayer
only games did not fulfill the criteria of selection. Some games were left out due to technical
difficulties. For example, N.O.V.A. 3 and Call of Duty: Black Ops Zombies required OpenGL ES
2.0 functionality which is not supported by the first generation iPod Touch. Painkiller Purgatory by
MachineWorks Northwest [Chillingo, 2011] was also considered, but the game suffered from bad
frame rates and consistency issues, such as enemies going through walls.

In the end, N.O.V.A. 2 by Gameloft [2010], Call of Duty: World At War: Zombies by Ideaworks
Game Studio [2009] and Battlefield: Bad Company 2 by Digital Legends [2010] were selected as
the target games. It is to be noted that Gameloft is also a publisher, but it seems that the N.O.V.A.
series is developed by their own internal development team. Battlefield: Bad Company 2 was
originally developed for PC, and the iOS version was ported by Digital Legends. The very first Call
of Duty game was also originally developed for PC, and then ported to the iOS platform by
Ideaworks Game Studio. Naturally, these ports are not trying to completely emulate the original experience, but rather to provide a familiar FPS experience on a touch device.

7.6 Evaluation

This section contains the detailed evaluation of the selected iPod Touch games.

7.6.1 N.O.V.A. 2 evaluation

The evaluation started with N.O.V.A. 2. The game is set in a sci-fi world with futuristic weapons and gameplay. It features a campaign mode where the story advances after the player completes levels with different objectives. The game is played with the device in landscape mode. In the test session, the game was played for a couple of hours with the easiest difficulty level. Figure 6 illustrates a typical gameplay situation in N.O.V.A. 2.

![Figure 6. N.O.V.A. 2 gameplay.](image)

N.O.V.A. 2 has three control modes the player can select. The first one has a virtual stick for movement, a virtual button for firing projectiles and the screen is used for free camera movement. Figure 6 illustrates this control mode. In this mode, the player can freely swipe at the screen in any location on the screen to move the camera. The virtual stick is placed in the bottom left corner and the button for shooting in the bottom right corner.

The second control mode is a variation of the first one. In this mode the screen is divided in the middle: the left side is for player movement and right side for camera movement. The virtual stick will appear in the first finger contact location. The virtual stick will disappear after the finger is released. The way the camera is moved is identical to the first mode. The button for shooting is also in the same location.

The third control mode introduces another virtual stick which handles the camera movement. Firing projectiles is achieved by pressing any location on the screen.
There is also a plenty of options for tweaking the controls. Auto-aim can be turned on or off. If the auto-aim is turned on, it will subtly assist target acquisition by automatically acquiring a target if the reticle is close enough. All controls can be also be inverted for left-handed playing. All of the control icons on the screen can be moved. Y-axis control can also be inverted. The sensitivity of the camera movement can be adjusted. And finally, the screen orientation can be set to 270 or 90 degrees.

Now it is time to look how the game performed while evaluated against the heuristics developed in Section 7.3.

**Target Acquisition, Target Leading and Target Tracking.** The game was played with all of the aforementioned control modes. As expected, most of the issues appeared when the combat was first initiated. Tracking or leading opponents was extremely challenging with all of the control modes. The third control mode was the worst for target acquisition because the movement of the virtual stick, which controls the camera movement, was implemented as a rate-control based input. As observed before, rate-control offers lesser performance when compared to a position-based control. Regardless of the control mode, it was challenging to acquire moving targets. Trying to acquire a moving target always led to inaccurate movements which moved the reticle over the target. This led to a cycle of corrections where it took a relatively long time to acquire the target. The best strategy for targeting opponents was to keep the reticle in the center of the view and wait for the opponents to stop moving, then strafing to line the opponent under the reticle. Also, the lack of high precision in the controls clearly hindered target acquisition when the opponents stayed at a distant range.

**View Manipulation.** View manipulation was implemented quite well. The configuration options allowed the player to have a custom sensitivity for the camera controls. However, even the with the custom sensitivity, tasks related to leading, tracking and acquiring targets were hard to complete. As discussed previously, it would seem that the resolution of the touch input is not enough for completing these tasks successfully.

**World Navigation.** Player movement was the same for all the of the control modes. The developers had implemented a standard movement system where the up and down arrows moved the character forward and backward and the left and right arrows allowed the character to strafe (instead of turning left or right). Jumping could be performed by pressing the virtual button designated for the jump action. The implementation was solid and no problems occurred with player movement.

**Stable Frame Rate and High Frame Rate.** The game provided stable frame rates even during heavy action. There was no way to check the frame rates, so it was unclear whether the actual frame rate was over 30 FPS. However, smooth player movement and animation in the game indicate a high frame rate.
Clearly Defined UI Elements, Non-blocking UI Elements and Consistent Feedback. The user interface elements which mapped to game controls were implemented nicely. The virtual stick is big enough for thumb manipulation, but not too big to cover a lot of space. The other control icons are moderate in size which makes them easy to locate and press. Control points were also clearly defined and performed logically.

Customizable Controls. As shown before, the game has extensive configuration options. It was actually surprising to see so many options. The option for choosing a left-handed control system is a notable addition.

Swiping at the screen to move the camera also disrupted the gameplay. On the other hand, it was easy to move the view because the screen could be swiped at any location, thus eliminating the need to remember where to swipe. However, swiping at the screen often blocked important game information, such as approaching enemies or incoming projectiles.

The main weapons, which were unlocked during the test session, were all hit scan weapons. It is easy to understand why the developers had chosen to include mostly hit scan weapons, as it was tough kill opponents even with them! Secondary weapons, such as grenades, were projectiles weapons, but with a large area of effect. This made hitting enemies with secondary weapons an easy task.

Overall, the game's controls performed well in the evaluation. Good configurations options allowed the player to customize controls properly. However, the acquisition of moving targets proved to be the hardest task to complete. The resolution of finger-based input is clearly not enough as it greatly decreased the ability to target opponents.

7.6.2 Battlefield: Bad Company 2 evaluation
The second game evaluated was Battlefield: Bad Company 2. The game will be referred to as BBC2 from now on. The setting for the game is based on modern warfare. As with N.O.V.A. 2, the player advances through the levels by defeating opponents and completing objectives. The game is played with the device in landscape mode. In the test session, the game was played with the easiest difficulty level for a couple of hours. A typical gameplay situation in BBC2 can be seen in Figure 7.
Similar to N.O.V.A. 2, BBC2 has three different control modes the player can select. In the first control mode, the screen is split into two sides. The left side is for controlling the character movement and the right side for moving the camera. Again, very similar to N.O.V.A. 2. When the left side of the screen is pressed, a virtual stick will appear. This stick can be then used to control the character's movement. If the player slides his finger off the area of the stick and then continues to slide the finger, the stick will follow the movement of the finger. This is a nice addition to the movement controls, as sliding off the virtual stick won't abruptly cease the movement of the player. Sliding or swiping a finger on the right side will move the camera. Shooting is accomplished by tapping the right side of the screen. This implementation is a little confusing as it mixes both movement and shooting functionality under similar finger movements.

The second control mode is nearly identical to the first one. The only difference is that the action for shooting is assigned to a virtual button which is in the bottom right corner of the screen. This is a clear improvement as it decouples the movement and shooting functionality present in the first control mode.

The third control mode replaces the free position virtual stick by assigning it to the left bottom corner. The user actions for camera movement and shooting are similar to the ones present in the second control mode. However, this mode is not split into two sides. This means that the player can move the camera by swiping at the screen in any location.

Other user interface functionality is the ability to target opponents through the scope of the weapon. The button for toggling scope on and off is placed in the bottom left corner. The player can also toggle the character to stand or kneel. The standing position button is placed in the bottom right corner. Reload and throw actions can be accomplished by touching the icons found in the top right corner. Placing these buttons in top part of the screen is awkward as it requires the player to shift his fingers from the bottom part of the screen where most of the user actions are initiated.
BBC2 offers some additional configuration options. Target assistance (auto-aim) can be toggled on or off. Left-handed gameplay is also supported. Toggling the left-handed mode on will invert the location of the icons and the screen areas of the camera and character movement. The sensitivity of the camera movement could also be changed on a scale of 0% to 100%.

As with N.O.V.A 2, the game's controls were evaluated against the heuristics developed in Section 7.3.

**Target Acquisition, Target Leading and Target Tracking.** BBC2 has similar problems in target acquisition as N.O.V.A. 2. Regardless of the control mode, movements needed for capturing targets were too coarse. It was extremely challenging to lead or track targets. In addition, the weapons in the game had recoil implemented which in turn made re-acquiring targets a necessity. Again, a solid strategy for defeating opponents was to keep the reticle and camera movement stable and line up the opponents by strafing. A particularly useful tactic was to rush up to an opponent and use the shotgun to finish the opponent. This illustrates the problem of target acquisition perfectly as the shotgun doesn't need to be targeted accurately in order to hit a target.

**View Manipulation.** There were no issues with view manipulation. However, as shown before, the low resolution of touch input doesn't really lend itself well to leading, tracking and targeting tasks. The view sensitivity was easily tuned by shifting the corresponding sensitivity option.

**World Navigation.** The navigation of the game world was lacking for the Y-axis, as the player could not jump. The stance of the player could be adjusted, but for some reason the developers had disabled jumping. This of course led to comical situations where the player's character couldn't jump over knee-high obstacles.

**Stable Frame Rate and High Frame Rate.** The frame rates of the game were passable. In most situations the frame rate was good enough, but during heavy combat the frame rate dipped noticeably. This led to situations where targeting was very difficult due to the already established inaccuracy in controls and the addition of weak frame rates. The bigger the dip in frame rates, the larger the miscalculation in targeting.

**Clearly Defined UI Elements and Non-blocking UI Elements.** All of the user interface icons were reasonably sized. The icons were also placed properly so that did not block or obstruct any key gameplay information.

**Consistent Feedback.** The control points provided logical and consistent feedback, for the most part. As was noted earlier, the shooting and camera movement actions were coupled in the second control mode. This led to confusion during the gameplay as sometimes swipes or slides meant to move the camera were read as actions to initiate gunfire.

**Customizable Controls.** The game had good customizable input mappings. As shown earlier, the player could choose from three different control modes. Left-handed gameplay was also supported.
The controls in the game performed nicely when evaluated with the heuristics. Unfortunately, the issues familiar from N.O.V.A. 2 also plagued BBC2. Finger occlusion was also present. All of the unlocked weapons during the test session were hit scan based. Overall the performance was good but slightly lacking when compared to N.O.V.A 2.

7.6.3 Call of Duty: World at War: Zombies evaluation

The third and last game in the evaluation was Call of Duty: World at War: Zombies. The game will be referred to as CODWAWZ from now on. The setting for this game is half modern and half fantasy. The game is different from the other evaluated games in that it doesn't have a campaign based mode. In CODWAWZ, rather than completing levels, the objective of the player is to survive as long he can in the level. The game will award score based on how long the player can survive and how many zombies the player can kill. As with the other evaluated games, CODWAWZ is played with the device in landscape mode. In the test session, the game was played for a couple of hours with the easiest difficulty level. Figure 8 illustrates a typical gameplay situation in CODWAWZ.

As with the other evaluated games, CODWAWZ has three control modes the player can select from. These modes are called Dual Stick, Tilt, and Touch screen.

Dual Stick has two virtual sticks assigned on the screen. The virtual stick placed in the bottom left corner handles character movement, and the virtual stick in the bottom right corner handles camera movement. The player can quickly turn 90 degrees right or left by swiping in the corresponding direction. Shooting is accomplished by pressing either the center of the bottom right virtual stick or pressing a virtual button. The virtual button for shooting can be activated by turning off the “Tap Aim To Fire” option. The sight of the weapon can be toggled by pressing the respective
button found in the right side of the screen. Using the sight also toggles the auto-aim which helps
the player in targeting opponents. Sight-assisted shooting is more accurate but slows down other
player actions, such as turning the view.

The Tilt control mode is based on the movement of the accelerometer. Specifics of the
accelerometer were discussed in Section 3.3. In this mode the player will control the camera by
turning and adjusting the position of the device. The accelerometer needs to be calibrated in the
game options before it can be used properly. The player can move the character by manipulating the
virtual stick found in the bottom left corner, just as in the Dual Stick control mode. Firing weapons
is accomplished by pressing the on-screen virtual button.

The Touch screen control mode is similar to control modes found in the previously evaluated
games. The screen is divided into two equally sized areas. The left side has the virtual stick for
movement, just as the two previous control modes. The right side can be swiped at freely to move
the view of the player. Shooting is implemented in the same way as in the two other control modes.
A specific configuration option in this mode is the ability to tweak the sensitivity of the camera's Y-axis.

General configuration options for all control modes include options for inverting the Y-axis,
adjusting the sensitivity of the camera movement's X-axis and changing the level of the auto-aim.
For some reason auto-aim couldn't be turned off completely.

Out of the three control modes, Tilt had clearly the worst performance. Having to constantly
adjust and twist the device in order to acquire targets was extremely frustrating. Tilting the device
does not offer the needed accuracy for targeting, let alone for leading or tracking targets.

Dual Stick and Touch screen control modes were neck and neck in performance. All of the
control modes had implemented the view manipulation as based on rate-control, so in this regard
the Touch screen mode's larger area allowed for better accuracy. On the other hand, in the Touch
screen mode, the buttons for shooting and toggling the sight were placed in the middle of the area of
view movement which caused unintended shooting and accidental sight activations. However, the
improved targeting made the Touch screen control mode the best choice.

As with the other evaluated games, CODWAWZ was also evaluated against the heuristics
developed in Section 7.3.

**Target Acquisition, Target Leading and Target Tracking.** It was fairly easy to acquire, static
or moving, targets in CODWAWZ because the opponents were zombies which moved at a really
slow pace. There were no leading or tracking tasks because the zombies just walked at the player,
inviting the player to shoot them immediately. The zombies did not also shoot any projectiles at the
player which made target acquisition even easier. Forgetting the opponents for now, the controls
suffered from the same issues as discussed in previous evaluations. Rate-control made the problems
even worse, especially in situations where the player needed to acquire the target quickly.
View Manipulation. There were no problems in manipulating the view with the Dual Stick and Touch screen control modes. However, the Tilt control mode made camera movement extremely challenging. Moving the player's avatar in the Tilt control mode was like being in a drunken stupor.

World Navigation. The player was not able to jump in CODWAWZ. Again, this led to situations where the player needed to go around knee-high objects instead of climbing or jumping over them. Otherwise navigation of the game world was free and unencumbered.

Stable Frame Rate and High Frame Rate. The game had stable and high frame rates. There were no hiccups or slowdowns observed during the test session.

Clearly Defined UI Elements and Non-blocking UI Elements. The user interface icons were placed properly and were of reasonable size. As noted before, in the Touch screen control mode, the buttons for shooting and toggling the sight were placed in the middle of the screen. This led to unintended user actions as the buttons were in the way of view manipulation actions.

Consistent Feedback. The control point manipulation in CODWAWZ provided logical and consistent feedback. As a side note, the view manipulation and shooting actions were combined under the same control point in the Dual Stick control mode. This was not a huge issue as the area which toggled the action for shooting was in the middle of the virtual stick, and the user action for shooting could also be assigned to a different virtual button.

Customizable Controls. CODWAWZ had reasonably customizable input mappings. The mappings were similar to the ones discovered in the other two evaluated games. However, the game did not have the option for left-handed gameplay which present in the other two games.

The game's controls performed adequately when reviewed against the heuristics. However, it is questionable whether the game can be directly compared to the other games in the following analysis as it had significantly slower opponents in gameplay. Also, the fact that the auto-aim couldn't be turned off made targeting much more easier. By using the auto-aim the player could just approximate targeting and then toggle the sight of the weapon which would then activate the auto-aim.

7.7 Results

The evaluation consisted of three games and nine control modes. The implementation of the control modes was similar from game to game, except for the accelerometer based one in CODWAWZ.

All games had implemented character movement in the same way as discussed in section 2.8.2, i.e. pushing the virtual stick left or right initiated strafe movement instead of turning the player. This convention seems to be the industry standard for movement in FPS games as it appears in almost every FPS title regardless of the platform the game is developed for.

View manipulation was implemented as both position-control based and rate-control based. The feedback from the evaluations follow the results discussed in Klochek and MazKenzie [2006]
where it was suggested that position-control fits a fast moving FPS environment better than rate-control.

Target acquisition for moving targets was challenging, as was expected. There seemed to be no easy way to hit a moving target even with a hit scan weapon. This forced the player to wait for the opponent to stop and then acquire the target. The downside of this tactic is that usually when the opponents stopped, they started firing projectiles at the player, which then forced the player to start moving.

The difficulty of acquiring static targets was dependent on the distance of the targets. For example, in N.O.V.A. 2 some of the opponents stayed constantly at a distant range which made the targeting area for the opponents miniscule. As was discussed in Chapter 5, high precision targeting is quite challenging with a touchscreen device without a special strategy implemented for the targeting.

Time spent on leading or tracking tasks was extremely low in the evaluations. The imprecise controls did not lend themselves well to either of these tasks. Usually, by the time the player acquired the target, the target had already stopped moving. A key observation can be made from this: using touchscreen controls can be used to eventually acquire targets, but the time it takes to acquire a target is not sufficient for leading or tracking tasks where the target is moving at a rapid pace.

All games allowed the player to navigate X- and Z-axes properly, but only N.O.V.A. 2 had implemented jumping. As noted before, this led to humorous situations where the player had to go around the objects, even if they were relatively small. It can be postulated that the other developers did not want to implement jumping as it was not a core part of the gameplay, and implementing the action would have required changes or additions in the user interface. The lack of jumping was not a significant obstacle but it still caused minor irritation during the gameplay.

N.O.V.A. 2 and CODWAWZ had adequate and stable frame rates during the gameplay. BBC2 suffered from dropping frame rates during heavy combat and it significantly impacted the gameplay. Claypool and Claypool [2007] note that bad frame rates harm the performance of the player a great deal, and it can be said that the frame rate drop seems to be even worse for a touchscreen controlled game. Having the frame rate dip dramatically made target acquisition extremely difficult. This led to situations where the player just waited for the frame rate to stabilize before engaging in the gameplay again.

All of the games had implemented the related user interface icons and buttons properly. There were no issues in accessing or using them. One of the control modes in CODWAWZ had some of the icons in the middle of the screen which caused unintended user actions.

Control point feedback was for the most part logical and consistent. One control mode implementation in BBC2 had mixed shooting and camera movement under the same control point. This caused confusion as the sometimes a little swipe would not turn the camera but instead fire the
weapon. The rest of control modes had proper separation of control, for example the action for shooting was assigned to a separate virtual button.

All of the games had a good amount of customization built into them. Each game had three control modes to choose from and an assortment of other configuration options, such as sensitivity of the camera, inverted Y-axis and so on.

The heuristics were a good tool in evaluating the games. For example, the heuristics related to frame rates should be strictly followed as frame rate issues really impact the performance of the player. Decreased performance will of course lead to decreased enjoyment.

However, all of the games failed heuristics 2 and 3 for the most part. One can question whether these heuristics should be replaced with something else because the absence of accurate controls on touchscreen devices really hinders the ability to complete these tasks properly.

The heuristics should be appended with at least one more heuristic. Some of the evaluated games had rate-control based control modes which significantly decreased the targeting performance. The new heuristic should take into account the fact that position-control works better for FPS games, and if it is possible, the developer should always default to position-control when designing the controls.

7.8 Summary
In this chapter existing heuristics for gameplay evaluation were explored. The existing heuristics were deemed to be too general, so new heuristics were designed. These heuristics were then used to assist the evaluation of the selected iPod Touch games.

In total three games and nine controls modes were evaluated. A persistent problem appearing in all of the games was the inaccuracy of the controls. Especially target leading and acquisition tasks were difficult to complete. Judging by the results, mapping PC or gamepad controls to a virtual touchscreen does not seem to be a good fit.

In the next Chapter results, player performance and future work relating to first-person shooter games and touchscreen devices will be discussed.
8. Discussion
In this final Chapter, topics such as player performance and experience, game design, and future work will be discussed. A summary of this thesis will also be presented in section 8.5.

8.1 Player performance
The game evaluation consisted of three games and nine control modes. The results of the evaluation point to a significant flaw with game controls on a touchscreen device. As it was noted many times during the evaluation, acquiring or tracking targets was extremely challenging. Especially moving targets were hard to target and kill.

The results also showed that the control schemes of the games shared many properties, even though the evaluated games were all developed by different developers. The control schemes followed the model presented in section 2.8.2. One of the control schemes deviated significantly from the standard way of controlling by utilizing the tilt control found in the iPod Touch. This proved to be the worst control scheme found in the evaluation. It seems that the standard way of controlling FPS games, which was developed in the late nineties, works even for mobile devices.

The custom heuristics proved to be a good tool in the evaluation. Even though some of the heuristics are quite obvious, such as the game having good frame rates, it is good to have these heuristics in order to capture the full range of properties of an FPS game.

Overall, the results follow existing research quite predictably. The research done by Zaman et al. [2010] clearly shows the performance difference of players when the same game is tested on physical and touchscreen platforms. The same issues cropped up in this thesis also. The performance of the player was severely degraded because incoming enemies couldn't be eliminated at a reasonable pace which led to the player losing lives or health.

It seems that the process of designing game controls for a touchscreen platform suffers from the type of issues as explored in the research of Albinsson and Zhai [2003] which concentrated on high precision selection strategies on touchscreen devices. On the one hand, the developer can design controls which enable the player to track and acquire enemies rapidly, but with a large error rate. On the other hand, the developer can devise controls which are extremely precise, but are really slow, rendering the controls useless.

Bad frame rates were not a large issue during the evaluation, but when the frame rate dipped, it caused the evaluated game to become nearly unplayable. That is, acquiring or tracking targets was impossible. This finding links directly back to the research of Claypool and Claypool [2007] where it was concluded that low frame rates clearly impacted the player's performance.

The evaluation only consisted of three games so it is possible that an existing or upcoming game might have clearly superior controls. However, the possibility is quite small as the properties of the touchscreen platform seem to inherently prevent precise and quick controls. Nevertheless,
trying to find improvements for game controls by iterating existing control schemes, like in the work of Albinsson and Zhai [2003], could yield surprising results.

### 8.2 Game design and controls

It is easy to see why some game developers have decided to forgo the typical FPS controls and instead develop a fixed-movement FPS shooter for touchscreen devices. By eliminating the movement part, the player can concentrate on shooting at the opponents. This kind of simplification is necessary, especially if the game has really aggressive and fast moving enemies. As has been observed, current touchscreen control mechanics do not enable the player to rapidly target enemies.

Even though the main topic of this thesis is not game design, the observations made during the evaluation point to game design issues. It is clear that emulating PC or gamepad controls on a touchscreen device does not lead to satisfactory results. Thus, improving the gameplay experience could be done through game design changes. This was also noted by Gerling et al. [2011]. For example, it was observed that the slow moving enemies in CODWAWZ were a good challenge for the player, and the long distance targets in N.O.V.A. 2 were a nuisance. These issues could be rectified by making sure the enemies move at a slow pace and that they remain at a relatively close distance. This will naturally require the game designers to develop a some kind of back story for the leisurely pace of the opponents (e.g., opponents are zombies).

The observations made from the evaluation link to the research done by Zaman et al. [2010]. The researchers noted that virtual touchscreen controls offer severely worse performance than their physical counterparts. It is then poignant to ask whether it is wise to directly emulate gameplay in PC games on touchscreen devices. Based on the observations and the research presented in this study, it would not seem like a good route to take. Rather than emulating the gameplay in PC games, developers should try to design for the device and make use of the device's strongest points. A direct emulation will lead to weaker user performance and thus to decreased enjoyment of the game.

### 8.3 Player experience

The results of the evaluation should also be discussed in relation to player experience. In section 6.7 the concept of flow, or total immersion, was presented. Badly performing controls break the flow of the player because the player is fighting with external challenges (controls), rather than internal challenges (game objectives, enemies). The results of this study seem to indicate that achieving a state of flow in FPS games on touchscreen devices is hard because the controls themselves inhibit the player from total immersion. The controls obviously do not prevent some immersion, but achieving a state of flow seems to be a difficult task.

On the other hand, Gerling et al. [2011] noted that usability issues did not prevent players from having positive experiences. A novice, both to FPS games and touchscreen devices, might have a fully enjoyable game experience because the novice does not have a frame of reference for FPS
controls or gameplay. The game developers also aid novice players achieve a better game experience by enabling the auto-aim on by default.

The categorisation of game experience was presented in section 6.7. Judging by the results and the presented discussion, the dimensions affected by the effectiveness of the controls seem to be enjoyment (fun, amusement), flow (concentration), suspense (challenge, tension), competence (accomplishment), negative affect (frustration, disappointment) and control (autonomy, power, freedom). The controls affect six out of nine dimensions which indicates that they are a large part of the player experience. Experiments which comprise of the categorisation and FPS controls should be conducted in order to find out deeper links between the effect of controls and player experience.

8.4 Future work

Future work in this area spans several areas. First, more FPS games on touchscreen devices should be evaluated in order to find out the effectiveness of different implementations. The heuristics developed in the thesis can be used to assist the evaluation of the games. Naturally, the heuristics used in the present thesis should be reviewed by other researchers. It is certain that they can be improved to produce better feedback for both researchers and game developers.

Second, empirical testing between PC and touchscreen controls should be used to find out detailed metrics of performance. Zaman et al. [2010] presented an effective way to test a cross-platform game, thus a similar kind of testing could be arranged for a cross-platform FPS game. For example, the test setup could follow these kinds of rules:

1. Select an FPS game which plays nearly the same on PC and on a touchscreen device.
2. Select a level which is long enough to expose the player to typical FPS gameplay.
3. Measure the time it takes the players to complete the level.
4. Measure how many times the player dies during the level.
5. Measure the average time it takes the player to acquire a target.
6. Calculate the ratio between hits and misses.

This set of rules is by no means complete; improving it should lead to more defined results. However, these rules provide a starting point for developing a cohesive test setup.

Third, improvements in physical platforms should also be explored. This was also noted by Zaman et al. [2010]. For example, implementing actuators in a touchscreen device should greatly improve user performance [Hoggan et al., 2008]. An experiment between a touchscreen device with no tactile feedback and a touchscreen with tactile feedback should provide interesting insights into game development on touchscreen devices.

Fourth, evaluating different FPS game designs on touchscreen devices could yield potentially better gaming experiences for players. It is relatively easy for game developers to port a game to a new platform by emulating the controls on the original platform. For example, virtual stick controls emulate the functionality of the gamepad's analog sticks. Of course, game enjoyment is a subjective
feeling and what might work for inexperienced users, might not work for hardcore players. Fixed-movement shooters might not be at all enjoyable for experienced users but highly enjoyable for novice players.

Finally, inspecting the effectiveness of different control modes explored in Section 7.3 could yield improvements for game design. This kind of research could be split into a few phases. The first phase should include systematic gathering of different control mode implementations from a large group of touchscreen games. The middle phase should incorporate research and ideas into a framework for evaluating the control modes. The final phase should be a controlled test where the performance of the users is measured by the evaluation framework.

The present study can only give directions and guidelines for further development of FPS games on touchscreen devices because the evaluated games only represent a small portion of the available games. Broad generalizations cannot be made without further study. However, the results clearly point to issues noted by other researchers. Using the findings and the developed heuristics could be used further in future research and in game development.

8.5 Summary

This thesis concentrated on the effectiveness of FPS controls on a touchscreen device. First-person shooter games are complex games where the player needs to handle multiple interactions at once: maneuvering, targeting opponents, collecting power-ups, dodging incoming projectiles and so on. The target device of this thesis was the iPod Touch. Thus, the research question was what are the effects of low resolution input on complex gameplay.

Presented touchscreen device research pointed to difficulties in high precision targeting and selection tasks. Research on multi-touch input revealed possible issues with control point manipulation design. Game controller research showed that when comparing physical input devices, the mouse offers superior performance to other input devices. Research on first-person shooter games revealed that frame rates play a huge part in the performance of the player. The previously mentioned research was explored because it was necessary to understand related problems and to find potential heuristics for evaluation.

Eleven heuristics were developed based on the explored research and existing heuristics. The design motivation of the heuristics was to enable the evaluator to gather detailed feedback and pinpoint specific issues regarding first-person shooter gameplay on touchscreen devices. The heuristics proved to be useful in the game evaluation. Future work on the heuristics included deeper analysis and refinement.

Three iPod Touch games were evaluated by a subjective evaluation. In the process of evaluation, nine control modes were discovered. Majority of the problems occurred when the player tried to acquire or track moving targets. Even though game developers had implemented several different control modes, the lack of high resolution input prevented the player from effectively
tracking or acquiring targets. This led to decreased enjoyment of the game, as first-person shooter
games rely heavily on moving targets and fast gameplay.

Overall the results of this thesis comply with the general observations made by other
researchers. The performance of the player is drastically reduced when using virtual controls on a
touchscreen device. Future work should explore how different approaches to game design can
improve the FPS gaming experience on touchscreen devices. Potential research questions are the
performance differences between PC and touchscreen device controls and hardware improvements.
References


