

# Applying Tilt and Multi-touch to Increase Input Space for Overhead Strategy Games on Tablet Computers

by Nevin Flanagan

A Thesis Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE in INTERACTIVE MEDIA & GAME DEVELOPMENT

MAY 2014

APPROVALS:

Dr. Robert W. Lindeman, Professor of Computer Science, Principal Advisor

Brian J. Moriarty, Professor of Practice in Interactive Media & Game Development

Dr. George D. J. Phillies, Professor of Physics

## Abstract

Real-time strategy computer games are a popular form of participatory and spectator entertainment. Despite their wide popularity on desktop and laptop computers, few examples of this gameplay type exist that have been developed to run on handheld touchscreen devices (smartphones and tablet computers). Most of the examples discovered have extremely simplified gameplay compared to strategy games developed for traditional computers, presumably to simplify the controls that are required to take actions in the game.

Given the popularity of the genre on previously conventional computer environments, I posit that adapting these games to this new computing space has been challenging because the commonly recognized control schemes rely heavily on forms of input that are absent or awkward on touchscreen devices (hotkey input with keyboard, a mouse with multiple buttons). I suggest that the input space of handheld touchscreen devices is deeper than these attempts have allowed for (adding detection of the device's angle, as well as multiple simultaneous screen position inputs).

I propose a layout of control and informative elements common to RTS gameplay designed to make use of the differing depth profile of this input space, and adapted to the observed ergonomics of handheld computer use. I describe a software project created to implement these designs, and present the results of a user study which captures feedback about this project and examines the effectiveness of different interaction modes in selecting commands and adding specifying metadata to them (e.g. destination points for orders to move across the map). I find that tilt can become an effective form of camera control if extended with additional refinements, and that the methods of touch interaction appear roughly comparable in effectiveness.

## Acknowledgements

As with any author undertaking a work of this scope, I owe significant thanks to a number of other contributors who made the completion of this work possible.

Foremost is my advisor, Dr. Robert Lindeman, who imparted some measure of academic rigor and research ambitions into what would otherwise have been a very practical exercise in software development without a clear direction of inquiry.

Brian Moriarty and Dr. George Phillis served on my thesis committee and provided valuable advice about game design and the nature of war-game play. The members of WPI's research group for Human Interaction in Virtual Environments provided important feedback on early prototypes of the project, particularly Jia Wang, who contributed advice about the direction and reasonable scope of research.

The members of the #corona chatroom on [irc.freenode.net](http://irc.freenode.net), particularly Sergey Lalov, Rich Gushue & Raymond Delia, provided pilot testing for bugs and design, and members of the #unity3d room furnished me with vitally important technical help in completing the development of the test project.

I cannot sufficiently appreciate here the help and support of my family, both natural and discovered, in bringing me to this point, between instilling a love of scholarship, developing all the supporting skills, and encouraging my interest in teaching; and most especially, of my wife Jenna, who has contributed her enormous faith in me, her material support while I occupied myself with school, and her determination to see me finish. I have only the inadequate words: I love you all.

# Table of Contents

Abstract .....	ii
Acknowledgements .....	iii
List of Figures .....	v
Chapter 1: Introduction & Description of Inquiry .....	1
1.1: Overview of Exploration.....	2
1.2: Intended Audience .....	4
1.3: Organization of this Document .....	5
Chapter 2: Related Work.....	6
2.1: Desktop RTS Interfaces .....	6
2.3: Mobile Strategy Interfaces .....	11
2.3: Literature on RTS Interfaces .....	16
2.4: Literature on Touch Interactions & Menu Selection .....	17
2.5: Literature on Tilt Input to Handheld Devices .....	18
2.5: Implications of Findings.....	18
Chapter 3: Design Constraints & Proposed Interface .....	19
3.1: Task Categories .....	19
3.2: Constraints Crossing Platforms.....	24
3.3: Techniques Introduced by Mobile Devices .....	25
3.4: Proposed Interface.....	26
Chapter 4: Test Project & Experiments.....	35
4.1: Test Project.....	35
4.2: Experimental Protocol.....	37
4.3: Experiment Outcomes.....	40
Chapter 5: Conclusions & Future Work .....	44
4.1: Directions for Future Inquiry.....	44
References.....	47
Appendix: User Study Materials.....	I

## List of Figures

Figure 1: Player action flow in RTS games	3
Figure 2: Total Annihilation [11]	6
Figure 3: Command & Conquer 3 [6]	8
Figure 4: StarCraft 2 [7], Age of Empires [1], and Rise of Nations [8]	10
Figure 5: Anthill: Ants Ain't Saints	12
Figure 6: Command & Conquer: Red Alert for iPad	14
Figure 7: Amoebattle	15
Figure 8: Ergonomics of command selection	23
Figure 9: Selecting a group of units	27
Figure 10: Finger travel path with dynamically positioned menu	28
Figure 11: Radial menus in The Sims [22]	29
Figure 12: A selection proxy button (green) being loaded into a saved selection button (white)	31
Figure 13: The life-cycle of a rolling-release drag gesture	33
Figure 14: Proposed War Table interface, with training command in progress	34
Figure 15: War Table's test scenario maps	36
Figure 16: The interface as implemented for testing	38
Figure 17: Distributions of camera displacement distances for each user, executed through map (left) and tilt (right)	42

# Chapter 1: Introduction & Description of Inquiry

Effective play of real-time strategy games (*RTSs*) requires the rapid interleaving of several different tasks, including switching a close viewpoint rapidly between different zones of a large depicted area or *map*, switching between different groups of game agents, and giving different orders to those groups, including targets or destinations for those orders that may be at any range from the agents throughout the map. Over the development of these games on desktop (and later laptop) computers using mouse, keyboard and raster monitor, the interfaces for these games have become refined to make very effective use of this set of input and output modes; in fact it seems like a plausible argument that the set of gameplay features and challenges common to *RTSs* was in significant part shaped by the hardware available for players to address them (the term *desktop* will be used in this paper to refer to any computer used through this peripheral set, whether it is a desktop or laptop configuration). The complex interface demands of this game genre appear to be a likely reason why game developers have been slow in adapting the *RTS* genre to handheld touchscreen devices such as smartphones and tablet computers (collectively called *mobile devices* throughout this paper, although there are many other categories of mobile device which do not fall under these categories and are not under discussion here), despite the popularity of these devices as platforms for playing computer games.

I suggest that interfaces for managing the complex mechanics of these games efficiently on touchscreen are possible, but that directly translating the accepted keyboard-and-mouse interfaces onto a mobile device's touch-screen is a poor starting point for developing them. I propose an alternative layout intended to be ergonomic with common patterns of holding and interacting with mobile devices.

## 1.1: Overview of Exploration

The genre of games called real-time strategy is broadly characterized by the way in which there is no single agent in the simulated world that corresponds to the player; instead, the player has an outside view of the world, and can issue commands to any number of different agents in the world. These agents typically represent soldiers, base buildings and military vehicles, and the player orchestrates their actions to eliminate an enemy's forces. Popular examples include the Age of Empires and StarCraft series of games.

The principal form of player agency in RTS games follows a basic cyclical pattern: each action is composed of the actors or agents that will carry it out, the action for them to perform, and the target object or destination point in the game's depicted world. Typically in RTSs, the first part is done in a stateful way, making the player's choice of agent or group of agents persistent between several commands, or until deliberately changed; this state is usually referred to as the player's current *selection*. To command the currently selected agents in most desktop RTSs, the command to be performed is selected, followed by the destination point or object.

RTS scenarios typically take place in a virtual world which is larger than can be viewed on the screen at an acceptable level of detail. Therefore, the main view of these games typically shows a small portion of the virtual world at a useful level of detail, and the focus of this view can be moved around the larger map. This allows the player to coordinate the actions of groups of agents that are in different parts of the map, monitor the progress of his actions or the placement and movement of opposing agents, or target an action on a destination far from the agents performing that action. Because shifts in player view may be relevant to any part of the command cycle, includ-

ing choosing what actions should be taken, the cycle of player actions thus looks something represented in Figure 1.

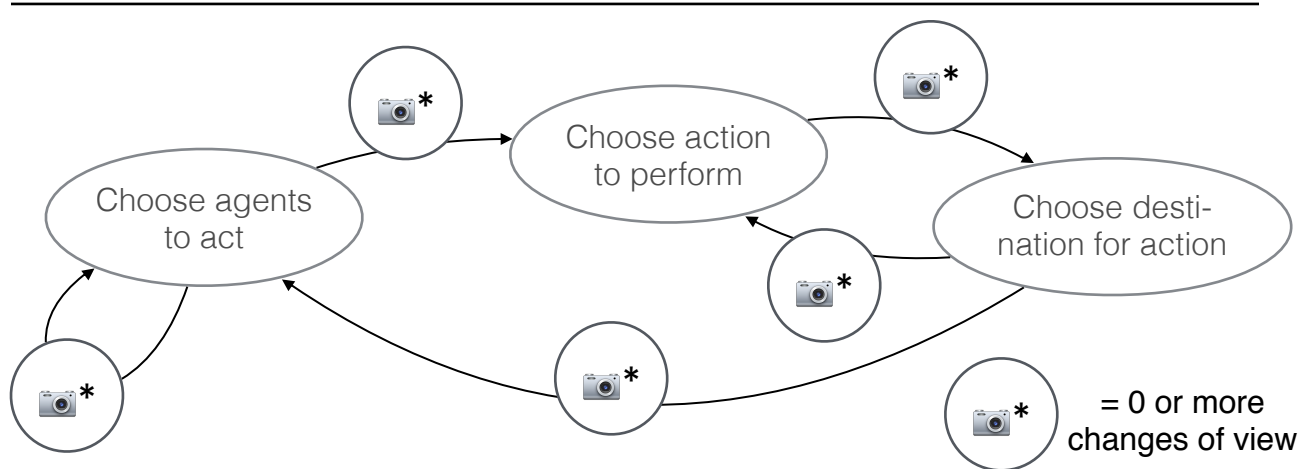


Figure 1: Player action flow in RTS games

---

Selection of a command is not usually stateful; once a destination is selected, a new command must be selected before any new destinations can be specified.

Sometimes, some of these steps become implicit. For instance, in desktop environments it is common for a right-click to issue the selected agents a command specified as their default action, with the clicked point as the destination. Similarly, some actions assume a given destination; for example, if an agent is instructed to remain at a position and defend it against enemies, it frequently assumes the agent's current position as the location to be defended. And depending on the interface, a player command to set the current selection to a previously memorized group of agents may implicitly change the player's view to the area around those agents.

Player agents are typically divided into two categories: *units*, which are mobile and act on other agents, and *buildings*, which are typically immobile and serve as points that receive resources from worker units and convert resources into units. *Worker* units collect resources from world objects,



convey them to buildings, and convert deposited resources into new buildings when ordered. RTS games frequently offer some variation on this baseline behavior.

Several of the assumptions that guided the design of these techniques are subject to examination. During the extended interface discussion, I will discuss an alternative order where an order's destination is selected before the action is chosen. However, the question of whether the selected agents might be chosen in a stateless way, or might be chosen after actions or destinations, or whether statefulness might be applied to the selection of command or destination in a useful way, are left outside the scope of this investigation.

During the course of this investigation, I will examine how these actions have been made available in desktop environments and what existing attempts have been made to provide them in mobile environments. I will describe the disparities between desktop and mobile computing devices and the interfaces they present, and suggest how these can be accommodated to supply a functionally similar cycle of player agency using the available input and display systems. I will also attempt to address the question of whether some of these accommodations may be superior to others and how this may inform future design.

## **1.2: Intended Audience**

This document is intended primarily for developers of mobile game interfaces, particularly real-time strategy games, and researchers into interaction methods for mobile devices, but may be of interest to those investigating ways to navigate maps in other applications. The research includes inquiries into both different forms of touch interaction and the use of device tilt as a form of vector input.

### **1.3: Organization of this Document**

This chapter has presented the goal of this project and introduced the general parameters underlying its intent. Chapter 2 outlines existing research into relevant forms of interaction techniques, as well as existing interfaces developed for RTS games on both desktop and mobile platforms. Chapter 3 will discuss the parameters specific to interacting with either desktop or mobile platforms, and propose an interface layout intended to adapt techniques developed specifically for the desktop into the handheld interaction space. Chapter 4 outlines the test project created to implement these designs and the user study performed to assess their usefulness.

Chapter 5 discusses the findings of the user study and presents directions for future inquiry.

## Chapter 2: Related Work

There are two principal categories of related work relevant to this design study:

- Research investigations into interaction techniques such as menu formats and command structures for touch-screen and mouse-based platforms;
- Interfaces created for existing strategy games on both desktop and mobile platforms. Because comparatively few RTS offerings exist in the mobile space, some other strategy games will also be considered.

Because interaction techniques need to be considered in terms of the needs of the application interface, existing interfaces will be reviewed first.

### 2.1: Desktop RTS Interfaces

#### Total Annihilation (1997) [14]

The oldest of the desktop RTSs under review, Total Annihilation also has the simplest on-screen interface (like most RTSs, it also has a substantial set of hotkey shortcuts).



Figure 2: Total Annihilation [11]

Total Annihilation occupies most of the screen with a large view of the virtual world. The top edge of the screen is occupied with a display of relevant resources used to construct units and operate buildings. The left side presents a bar of control features, including a mini-map at the top which offers an overview of the virtual world in a fixed space. The center of this area contains a set of buttons to initiate actions common to various units, and the bottom is a dynamic area which can be toggled between unit-specific orders and build commands. The mouse is used to select units and targets for commands, and can click on command buttons (in addition to the hotkeys available for nearly all menu operations). The right mouse button is used to deselect units. Some functions are only available through hotkeys, including setting and selecting *squads*, memorized groups of units that can be used to quickly set the selection. Shift is used when issuing commands to queue commands to be completed in order.

### Command & Conquer 3: Tiberium Wars (2007) [4]

Command & Conquer 3 has the most obviously complex interface being described. The world display is extended to all edges of the screen by appearing to 'float' the UI elements over the 3D world, although parts of the world are too heavily obscured to be particularly usable.

This interface places the production of new agents permanently on the screen along with the selection of units currently receiving action commands. The production tabs (#9 in Figure 3) indicate the categories of production available, the sub-tabs (labelled #10 in Figure 3) select a single producer from the chosen category, and the build buttons (#11) display all the agents producible by the selected producer. This hierarchical choice is stateful and persistent. The interface displays information about the current selection in the lower right (#12 - 18), including overall health, what

types of units it is effective or ineffective at engaging efficiently, and buttons to quickly select special actions available to the selected agents. A small bar in the bottom left (#19) indicates commands available regardless of context, and a button under the mini-map gives access to a very detailed sub-screen intended to allow fine control of the current selection, including issuing commands to subsets of the group without changing the selection. Buttons on the left give access to commands that are not carried out by any single agent, and a display below them indicates the availability of similar powers controlled by the enemy, to aid in strategic decisions. Resources are overlaid on the mini-map in the upper right, in the form of a numerical value for a resource which is expended (credits), and an allowance bar for a resource which is allocated (power).



Figure 3: Command & Conquer 3 [6]

The left mouse button is used to alter the current selection and the right mouse button is used to target commands being issued. As with Total Annihilation, hotkey shortcuts are available for

most commands, and some features, such as squad assignment and selection, are handled through hotkeys exclusively.

### StarCraft 2 (2010) [10]

Probably the most widely recognized RTS, StarCraft 2 is the fifth strategy game from Blizzard Entertainment. It is the most familiar to the author and was the basis of several of the assumptions used in designing the prototype being presented. The interface is an extension of that used for Blizzard's earlier games, StarCraft and WarCraft 3 (the interfaces for WarCraft: Orcs and Humans and WarCraft 2 more closely resemble that used in Total Annihilation). The interface also resembles that of Age of Empires [15] and Rise of Nations [9] to a significant degree, as shown in Figure 4, and may owe some inspiration to Age of Empires, which pre-dates StarCraft slightly. All three franchises organize the major UI elements horizontally along the bottom of the screen and place a few informative elements such as resource displays at the top.

StarCraft 2 places the mini-map at the bottom of the screen, not the top, across the screen from a selection of possible actions called the *command card*. The contents of the command card vary in part depending on what units are selected and what special actions they have available to them. The actions in the command card also display the hotkeys that will activate them, as these hotkeys are reassigned to different commands depending on the contents of the command card.



Figure 4: StarCraft 2 [7], Age of Empires [1], and Rise of Nations [8]

The area between the mini-map and the command card displays the current selection. If a single agent is selected, the selection screen shows detailed information about that agent, including weapons possessed, special characteristics, and degree of experience, and the portrait for that unit type is shown between the selection area and the command card. If several units are selected (the

selection area can be scrolled between different pages to accommodate selections of up to 150 units), they are presented sorted according to their type, and one type is highlighted; it is this sub-group of units whose special abilities are presented on the command card (although generic commands given through the command card will apply to all units capable of carrying them out). The player can press Tab or Shift+Tab to switch which category of agents is presented in the command card. The portrait slot displays the portrait for whatever agent type is the highlighted sub-group.

StarCraft 2 is also the only desktop game surveyed to present a visual interface for memorized selections or control groups. While it uses a similar interface to the others for setting and selecting control groups using the keyboard, it also provides a thin strip of buttons above the selection display, one for each allowed control group (up to ten in all). Each control group that has any units assigned to it will have a corresponding button, and one empty button is available for assigning a current selection to a new control group. In addition to setting these groups with Ctrl+#, selecting them with #, or adding to them with Shift+# (where # stands in for a numeric key on the keyboard corresponding to the desired control group), these buttons can be left-clicked to select their contents or right-clicked to assign their contents.

### **2.3: Mobile Strategy Interfaces**

A small number of real-time strategy offerings have made their way into the mobile platform space. The first and simplest is not quite a true RTS, but has several common elements and illustrates how typically developers have adapted to the space by slimming down their game mechanics.



## Anthill: Ants Ain't Saints (2011) [3]

Anthill is one of the first serious attempts at a real-time strategy game specifically tailored to the touch-screen environment. It focuses on tracing trails for agents to follow, in order to determine where your agents will go in pursuit of their tasks (resource collection, mobile defense, base defense). You define trails by dragging a freeform path starting from your base location; when you lift your finger, a radial menu appears, as visible in Figure 5, around the point where you left the screen that allows you to choose what sorts of agents will patrol that path. Paths persist until you clear them by pressing your finger and holding it on a path. If you define multiple paths of the same type, the available agents of that type are divided between the paths of matching type.



Figure 5: Anthill: Ants Ain't Saints

Besides dispatching agents by laying trails for them, you can also produce targeted offense agents (bombers) who are dispatched by tapping points on the screen. A row of buttons along the bottom indicates how many agents of each type you have available and allows you to produce more.

Anthill occupies a border space between being an RTS and a zone defense game, but is considered significant here because its interface and mechanics are well-tailored to the touch-screen environment, and because it shares the RTS characteristic of requiring you to choose what portion of the resources you gather will go toward your mission goals and what portion toward increasing your resource gathering.

### Command & Conquer: Red Alert for iPad (2010) [5]

This game is an adaptation of an adaptation; the original Command & Conquer: Red Alert was ported to iPhone, then when the iPad came out, the iPhone port was adapted for it. This version is reminiscent of the desktop interface for Command & Conquer 3, but simplified for play on the iPad: a mini-map with a power supply/demand bar is in the upper-right, and the production menu on the right side is hierarchical based on what type of production is being managed. Buttons along the bottom allow the player to modify the current selection, and three tabs at the left side serve as memorized selection buttons.

Commands are kept to a tap or two where possible; tap a building to use it in the training menu, tap a unit to select it, tap a destination to move selected units there. The iPhone version required you to use a button on-screen in order to activate a group selection mode, but the iPad version allows you to select all agents within a box by using three fingers to define the sides. This mechanism is shown being used in Figure 6. Usefully, when you select a building, the buttons to

repair or sell that building appear right by the building rather than being situated over by a side, making it clear which building will be affected.



Figure 6: Command & Conquer: Red Alert for iPad

### Amoebattle (2012) [2]

Of the strategy offerings currently available on mobile devices, this is probably the one that looks most like a traditional RTS and has the most familiar UI elements. A mini-map sits in the upper-right corner, with resource values next to it along the top. Selection buttons are down the left side, including a button to select all units visible on the screen, one to deselect all selected units and up to four control group buttons to hold memorized selections. Along the right side are buttons to activate player-controlled abilities. On the bottom are buttons that increase and diversify your force of agents, and a strip that display the types and counts of units that you have currently selected. The arrangement of these elements is shown clearly in Figure 7.



Figure 7: Amoebattle

Touch controls on the world display are still simple, but deeper than in the other mobile games presented. Tapping the world view orders all selected units to move directly to the specified point, and tapping an enemy unit orders all selected units to attack it, even to the exclusion of other enemies. However, swiping a short distance across the screen with one finger will order the selected units to *attack-move* toward the point at the end of the swipe, an action in which they will engage any enemies they encounter, then resume moving toward the designated point when the enemies have fled or been defeated. Tapping a single friendly unit will set the current selection to that unit, but touching and holding on a unit for a significant fraction of a second will toggle that unit in or out of the selected group. Tracing a finger in an outline around a group of friendly units will select

all the enclosed units, but touching the screen with two fingers simultaneously and sliding them around will pan the player's view of the world around the larger map. The control group buttons are activated according to touch duration as well; a single tap will select the memorized group, a double tap will select them and move the view to include them, but a prolonged touch will reset the group's contents to the current selection.

### **2.3: Literature on RTS Interfaces**

RTS games have featured repeatedly in literature on artificial intelligence research, and made some appearances in other learning and cognitive fields like cognitive decline resistance training. However, in October of 2012, William Hamilton et al. presented a paper at UIST [17] on their work developing a pen + multi-touch user interface intended for high-performance play of real-time strategy games. The comparison to the intended device field is not perfect; Hamilton et al. used large monitors against which the hands could be rested, rather than devices small enough to hold in the hand. Nonetheless there is significant overlap. Both environments allow for multiple simultaneous screen position inputs and thus also lend themselves to bi-manual interaction, and both environments function in the absence of symbolic input using a keyboard. The Hamilton research deals in switching between input modes by varying the number of fingers used, as well as alternating the dominant hand rapidly between holding the pen stowed in the hand to employ the fingers, and bringing the pen out to use the tip for very precise screen input. They also raise the importance of Fitts' law [16] in maximizing interface accuracy and of edge-constrained input in reducing targeting time and effort for actions with the non-dominant hand. Effort is made in the design of the proposed interface to incorporate some of these concerns.

## 2.4: Literature on Touch Interactions & Menu Selection

More research has been done on various methods for interpreting touch-screen gestures and efficiently selecting from graphical menus. Henze et al. produced a simple target-accuracy game and released it to the public as a way of gathering a large data sample about the natural accuracy of touch inputs [18]. They found that fast taps tend to land somewhat closer to the center of the screen than targeted, and that this effect is exacerbated by the closeness of the target to the edge of the screen.

Hamilton's research on real-time strategy interfaces above identifies the problem domain of selecting agents and targets as one where speed and accuracy crucially affect performance in play tasks, which suggests that Fitts' law affects the choice of likely interaction techniques. Based on this, the work of Callahan, Hopkins et al. suggests radial or pie menus are a good starting point to maximize target size and minimize travel distance [13].

Banovic et al. propose a novel system of menu invocation and selection intended for multi-touch surfaces [12], to be used when menus are desired to be visible only when needed (because tablet screens are small, screen space is at a premium and minimizing clutter is desirable). The first touch is assumed to be the thumb, and command regions are therefore arranged radially around the inferred center of the hand, placing them under the fingers of the activating hand. These regions may be further subdivided concentrically from the center of the hand, allowing each finger multiple potential actions to choose from, and each target region may modify its behavior if touched by multiple fingers simultaneously. While this system has its appeal, it was not adopted for this project because the proportion between hand size and screen size is often small when using handheld tablets. This limits the freedom to make placement of the hand on the screen relevant to

the input. Also, the dynamics of the wrist make it difficult to engage multiple fingers with the screen at the same time as the thumb unless the target surface is held quite level, or awkwardly far from the body.

Samp & Decker propose a *compact radial layout* as a way to further limit travel distance when choices have nested sub-choices called the compact radial layout [26]; the existing interfaces reviewed so far frequently involve hierarchical choices, particularly for building and training new agents, so this system will be revisited.

## **2.5: Literature on Tilt Input to Handheld Devices**

Because most mobile devices contain accelerometers, the possibility of providing two-dimensional input to the device by positioning it arises. Hinckley et al. proposed the use of device tilt to scroll across a larger viewable area in 2000 [19] (Oakley and O’Modhrain proposed in 2005 to use tilt as a way of selecting from linear menus [24], but this is somewhat at odds with the proposal to focus on radial menus). Van Tonder and Wesson advance the notion by suggesting tilt specifically as a one-handed mechanism for controlling map applications on mobile devices [28], and refine their suggestion with a scheme, IntelliTilt [27], to make it easier to focus on specified points of interest.

## **2.5: Implications of Findings**

Several of the features encountered have the potential to be useful in the forthcoming design, particularly the two-finger pan to select, the compact radial menus, and tilt control of the map view. The use of additional screen touches to select the mode of interaction also has promise.

## Chapter 3: Design Constraints & Proposed Interface

Here I compile several of the tasks that a complete interface must be able to accommodate, and which of them require new approaches to effect, and suggest a format to facilitate these interactions.

### 3.1: Task Categories

Player actions in most RTS games fall into three broad categories: moving the view of the map, selecting objects, and issuing commands.

#### Moving the Player View

Most RTS scenarios take place in a world which is intentionally larger than can be viewed at one time in the main window; the mini-map displays the entire world, but at a very low level of detail, adequate for making only very broad decisions. This means that to carry out precise tasks like placing buildings to leave adequate travel lanes between them, attack specific enemies, or maneuver for advantageous position in skirmishes, the main detail view must be shifted to show the location of interest. There are three basic operations to move the view:

- Jumping to a location. For any of several reasons, such as intervening in a fight or checking your base to research upgrades or create buildings, the player may be required to move his or her view as quickly as possible to the location of interest. In most desktop RTS games, this is accomplished by clicking or dragging the mouse over the desired location on the mini-map.
- Scouting an area. Bases or forces of units are frequently larger than can fit comfortably in the detail view, but long-distance jumping of the view is often not precise enough to easily and efficiently move short distances. In these short-distance hops, movement relative to the existing



camera position is preferred. Desktop RTS games typically enable this relative movement with the arrow keys or by hovering the mouse near the edges of the view, or by dragging the terrain with a specific mouse button held down.

- Jumping to an event. This is a specialized application of jumping to a location. RTS game systems frequently provide some way of notifying the player of news such as the completion of a building, or coming under attack by newly-discovered enemy units, and there can be methods of transferring the player view directly to the location of these notifications. For example, in Star-Craft 2, when a unit comes under fire in a location well outside the player's view, there is typically a voice line such as "We could use a little help over here!" and the portrait of the attacked unit will briefly replace the the portrait of the selected unit or group; the player can click this temporary portrait or press the Space bar to navigate the view instantly to center on the relevant unit or building.

### Adjusting the Selection State

Selecting what agents will perform subsequently commanded actions is one of the dominant procedures of an RTS game, frequently executed several times in succession, and there are several variations on how it can be executed; exact mechanisms vary from one game to another.

- Selecting a specific agent visible on the screen. The player can set the selection so that only the specified agent is included. This is typically done on a desktop by left-clicking the agent when it is visible in the detail view.
- Selecting a group of agents that are geographically contiguous. The selection can be set to a group of agents that are close together in the virtual world and all visible on the detail view at

once, usually by holding the left mouse button down while near one corner of the group, dragging to create a selection box, and releasing the button while the desired units are enclosed.

- Selecting agents of a specific type. By double-clicking or clicking with the Control key held down on a single agent, the selection can be set to the group of all agents of that kind that are visible in the detail value.
- Adding one or more agents to the current selection. By holding Shift while performing one of the above techniques on an agent not currently selected, all affected units will be included in the selection, in addition to those units previously in the selection.
- Removing agents from the current selection. Holding Shift in conjunction with any of the first three techniques on an agent in the current selection will remove the affected agents from the selection. Using this with agent-type selection will not usually be limited to the visible world area.
- Recalling a memorized selection. Because RTS play frequently requires acting in multiple positions throughout the virtual world, it is convenient to have some way to quickly switch between groups of agents in those disparate areas. For this reason, most RTS games include some sort of mechanic for memorizing the current selection and recalling it later, variously referred to as squads, control groups, forces or similar terms. Memorized selections are usually associated with the number keys; pressing one recalls the selection committed to it, and double-pressing the hotkey also changes the view to center on the selected group of agents.
- Setting or extending a memorized selection. In order to populate memorized selection groups in the first place, commands exist to commit the current selection into a particular

stored group. On most game clients, this is accomplished by holding the Control key and pressing the desired number key. Some games also allow you to hold Shift while pressing the desired number key in order to add the current selection to any agents in the specified group.

## Issuing Commands to Agents

Assigning actions to agents is the part of the player action cycle that actually modifies the game world (some games do also allow the player a small set of actions that do not require an agent to carry them out, usually sweeping actions like calling down a satellite laser strike on a portion of a battlefield; these actions are usually carried out in similar fashion, except that they ignore the selection state). In desktop RTS games, there are usually three ways to issue a command:

- Click a GUI button on the screen corresponding to the action or press the corresponding hotkey, if the command requires no further information, typically because it takes place at the agent's current location, such as "defend" or "burrow." The command takes effect as soon as the button or hotkey is released.
- Click the command's corresponding GUI button or hotkey, then click the object or point on the detail view which should act as the target or destination for the action. A building might be ordered to rally the new units it creates to a specific location, a worker unit could be ordered to harvest a specific resource, or a unit could be ordered to attack a specific enemy building or follow an allied unit. Typically, the mouse cursor changes to reflect the fact that the next click will be interpreted as the destination of an order.
- Several clients interpret a click or tap on the detail view as a default order to the currently selected agents; typically attack-move for combat units, harvest or move for worker units, and

rally for buildings. In some cases, only specific clicks such as right-clicks are interpreted in this way.

In many desktop clients, holding Shift while assigning an action to units queues the action on those units, so that they do not begin that action until their current action is complete. For instance, a worker unit might be ordered to construct a new building, but not until the resources it was carrying were deposited at a depot building. Or, a group could be ordered to destroy the buildings of an enemy base in a specific sequence. If Shift is not held when an action is assigned, then that action supplants whatever actions those units are currently executing or have queued.

Hotkeys are regarded by experienced players as significantly more efficient for issuing orders quickly than using the mouse to select actions in the GUI. It is not hard to understand why when the required motions are considered.

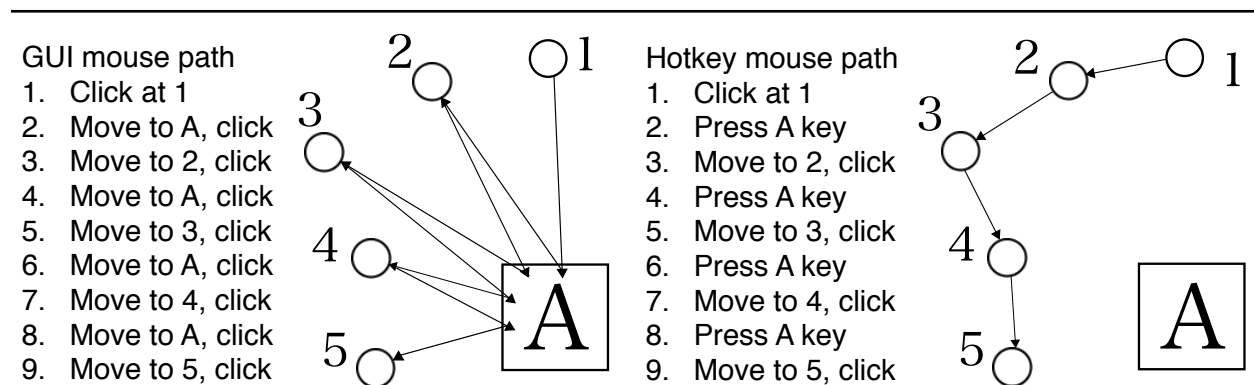


Figure 8: Ergonomics of command selection

---

Figure 8 makes it apparent that not only must the mouse travel a substantially greater total distance before the first path, but that the first path is much more error prone as a Fitts' Law task.

This relates to the earlier description of RTS play as a cycle of agent selection, action selection and target selection, because it solidifies the cycle of interleaved action and destination selections.

### **3.2: Constraints Crossing Platforms**

Several of the interface implementations discussed above are clearly not applicable to a tablet environment, so before proceeding, I will outline some of the major discrepancies.

#### Hotkeys

An overriding concern is the inability to use hotkeys to trigger selection of commands in an environment with no keyboard. This means that menu design must minimize the distance between targets and commands in order to avoid the trap of long travel times outlined above.

#### Relative Camera Navigation

The two dominant techniques used to slide the camera to nearby areas—arrow keys and mouse hovering—don't apply to tablets, so an alternative method of relative view control needs to be provided, or jump navigation made easier and more precise.

#### Additive Actions

There are several examples where a modifier key, usually Shift, is used when making assignments to make the additions rather than replacements: additions to the current selection, additions to a memorized selection, queued actions being added to a sequence rather than replacing it. In the absence of modifier keys, other ways of distinguishing interaction modes need to be found.

## Alternate Mouse Buttons

Some techniques, such as issuing default commands by selecting a destination, or dragging the terrain to control relative camera movement, make use of multiple mouse buttons to discriminate the different actions that might be targeted by the click or drag interaction. Mobile devices cannot distinguish which finger or object is touching a given point, only which points are being touched at a given moment and whether they represent the same touch that contacted some point previously.

## Ergonomics of Motion

Several visual elements on desktop interfaces are arranged in ways which are either more comfortable when using a mouse, or whose GUI ergonomics are made irrelevant because of the widespread use of hotkey alternatives. Which elements come most readily or comfortably within reach should be considered carefully in their placement.

### **3.3: Techniques Introduced by Mobile Devices**

Tablets and smartphones give up their keyboard and multi-button mouse, but they do offer some features that are not present in desktop environments.

## Accelerometers

Most mobile devices come equipped with accelerometers that allow them to determine the gravity vector relative to the device, as well as ascertaining the instantaneous motion of the device to some degree. While extreme motions may interfere with the ability to use the touch-screen, device tilt can be used as an input vector within a reasonable range, and other motion gestures such as shaking the device may also be useful.

## Multiple Position Inputs

Modern touch-screens are capable of sensing very large numbers of simultaneous inputs; the device used by Hamilton et al. in their pen-in-hand RTS scheme supports up to 28 [17]. Most smartphone and tablet devices appear to support four to ten simultaneous touches, which makes sense since these devices are generally intended for a single user and most users employ only their fingers when operating one. However, compared to the single positional input of a desktop computer, this offers new options, notably the ability to interact with multiple GUI elements simultaneously, indicating more than one point in the world display, or modifying gestures by using additional fingers.

## Bimanual Operation

Strictly speaking, traditional desktop computer operation is bimanual; when playing RTS games, one hand operates the keyboard to choose actions and agent groups, the other operates the mouse to choose targets and move the view. However, as noted previously, a touch-screen environment does not distinguish between the sources of inputs performed with various fingers. This means that tasks can be performed with either hand freely, and interface design can place certain controls near the non-dominant hand, which is typically employed in holding the device but can generally maneuver a thumb across the screen.

### **3.4: Proposed Interface**

I will outline here ways in which a combination of revised GUI design and the new input features described above can mitigate the challenges created by losing inputs considered fundamental to functional play.

## Selection Mechanics

Selection remains fairly simple, being performed through the world detail view (the *main map*) with single-finger touches. Tapping a single unit selects it; dragging a finger across the screen defines an area and selects each unit in it. There are several ways to define a selection area based on a drag path, including Hamilton's isosceles lasso technique [17], but for simplicity's sake a simple rectangle with its corners defined by the start and end of the drag was adopted, shown in Figure 9. Other options will be mentioned under future work.

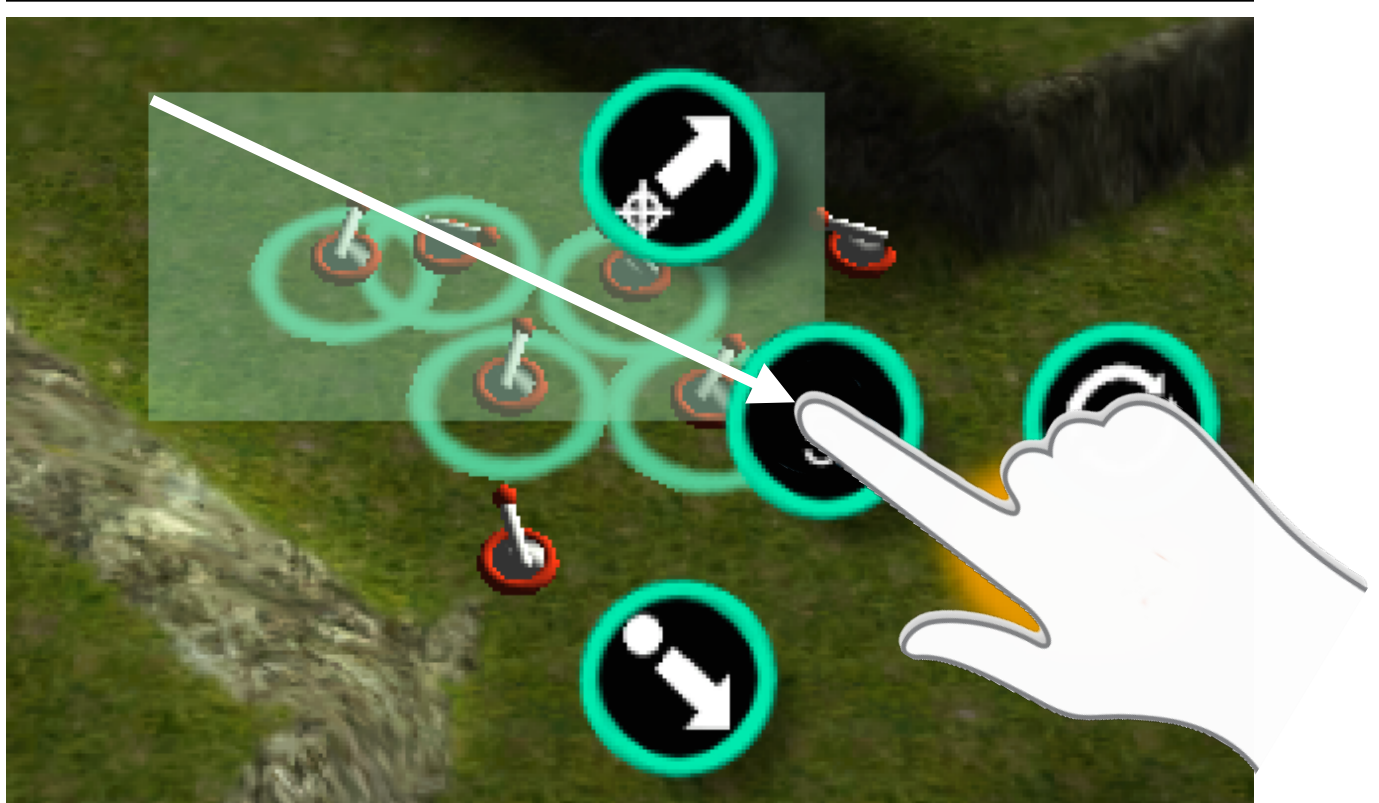


Figure 9: Selecting a group of units



## Dynamic Menu

The principle challenge created by losing the use of hotkeys is the increased travel time and error rate associated with constantly having to revisit the buttons that select specific actions. This can be vastly reduced by moving the menu close to the touch inputs that activate or interact with it. To further reduce travel times, the menu can be implemented as a radial or pie menu, insuring that the options are equally distant from the finger that activates the menu.

For example, if we assume a radial menu with three buttons distributed around the center of the menu, we can see in Figure 10 that the path needed to target five points in succession is still much shorter than if a fixed button has to be revisited.

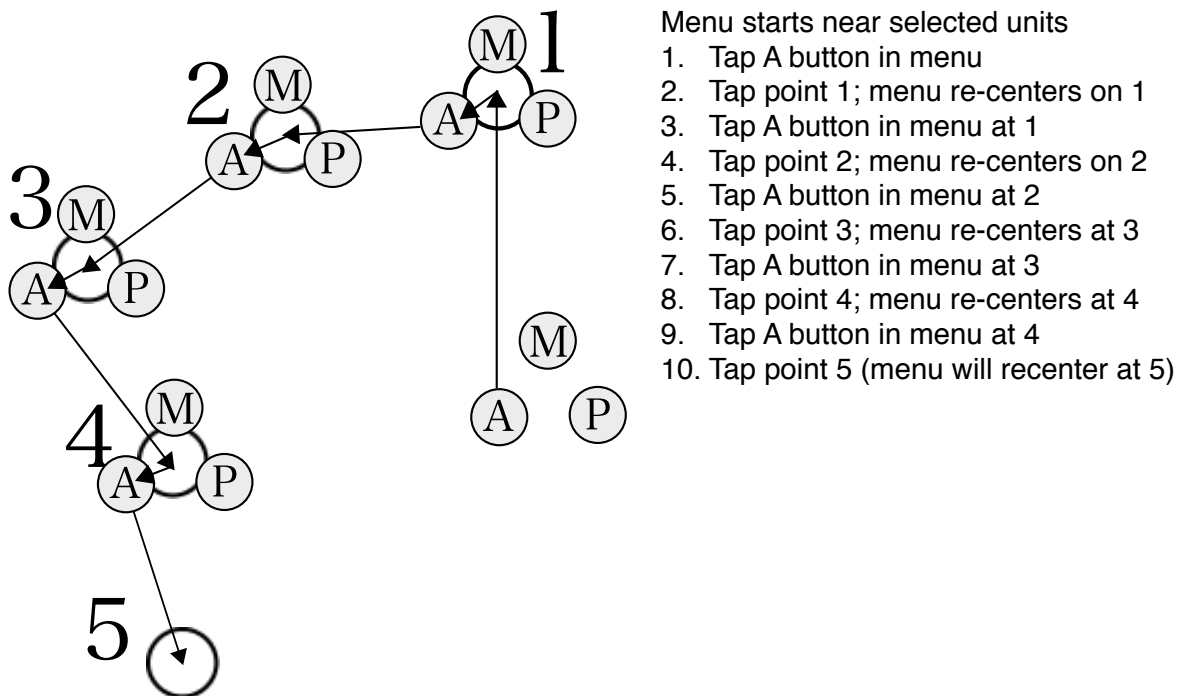


Figure 10: Finger travel path with dynamically positioned menu

---

While the original pie menus described by Hopkins [20] were solid circles divided into wedges, these menus are positioned on top of an environment that will continue to change while the player makes choices, so there is a concern that solid menus would obscure too much. Therefore, the menus are composed of clusters of round buttons, in order to leave the space between them visible and accessible, in a fashion similar to menus like that seen in Figure 11, from The Sims [21], or Anthill.



Figure 11: Radial menus in The Sims [22]

The proposed design identifies any touch action that changes the selection, such as tapping an agent, dragging a finger across a group of agents to select them, or touching a button that stores a memorized selection. The menu appears centered on the point of contact and follows it as the touch continues, updating if needed as the selection's contents change; this can be seen in the selection example shown in Figure 9. When the selecting finger leaves the screen, the menu remains centered on the last point where the finger touched the screen, so that the action buttons are all near the finger's presumed position, keeping travel distances short. When a command is selected from the menu and a destination indicated for it, the menu re-centers on the point touched to

make that indication, so that it still remains near the finger being used. This system allows the user to carry out the cycle of interleaved action selection and target selection with minimal unneeded displacement, either by consecutive taps on the action and the destination, or by touching the action button and dragging it to the destination point before releasing it (both of these methods will be explored throughout the project).

### Relative Camera Movement

As discussed before, all of the common ways to implement this on desktops (right-drag, arrow keys, hovering the mouse cursor by the edges of the view) are inapplicable to touch-screen mobile devices. New techniques must therefore be devised. One way is that adopted by the tablet RTS Amoebattle [2], which allows the user to put down two fingers next to each other and move them together (a mechanism called a *two-finger swipe* or *chordic drag* [29]). This is the same mechanism that Macintosh computers with touch pads use to enable vertical and horizontal scrolling and is therefore familiar to some users.

Another, less tested technique, is to use the tilt angle of the device as a rate-control (or possibly acceleration-control) vector input to move the camera in two dimensions. Side-to-side tilts of the device will slide the camera view laterally, and tilting the device toward vertical or horizontal will cause the camera to dolly sagittally over the terrain. The degree of tilt considered neutral can be adjusted to a user's preferences, as can the gain applied to the tilt angle to determine camera motion rate. It is also trivial to insert an option for inverting the vertical tilt input, similar to the way many shooters allow you to toggle between flight-stick and view-shift control for mouse camera control.

## Memorized Selections

Without number keys, it is necessary to add GUI controls for saving and recalling control groups or memorized selections. Some games have already experimented with this (StarCraft 2 [10], Command & Conquer: Red Alert for iPad [5], Amoebattle [2]) and have suggested multiple mechanisms for interacting with these buttons. This design proposes two sets of elements: a set of saved selection buttons, placed along the border between the detail view and the other static UI elements, and a *selection proxy button* placed at the center of the action radial menu. This button resembles the other action buttons in the menu, but instead of an action icon, it shows a number indicating how many agents are currently selected. If dragged, it issues all selected agents whatever is designated as their default order targeting the dragged location; if tapped, it resets the camera view to show the selected agents. It also serves as a place to drop dragged action buttons in order to cancel them. But it can be dragged onto a saved selection button in order to set that button's saved agent roster to the current selection of agents, as shown in Figure 12.

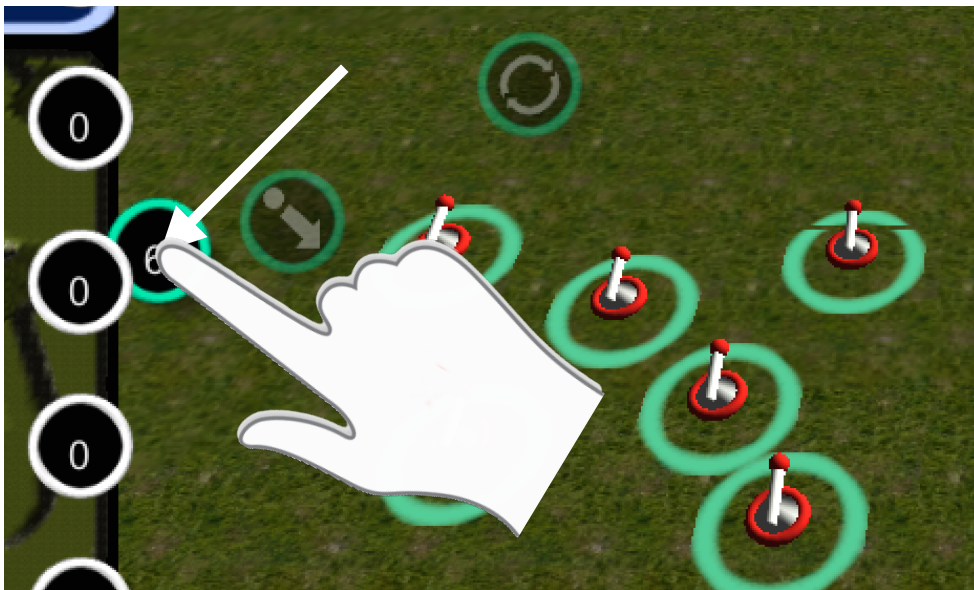


Figure 12: A selection proxy button (green) being loaded into a saved selection button (white)

The saved selection buttons, like the selection proxy button in the menu, show a number indicating the number of agents in the set saved to them. Tapping any of these buttons will select its saved group, jump the camera to view those agents, and activate the floating pie menu centered on that button. Touching a saved selection button and dragging off of it will select the saved agents without moving the camera, and allow you to position the menu wherever you wish to leave it.

### Additive Actions

In most desktop RTS clients, there is a common convention to use Shift for actions that are cumulative in some way, such as increasing the current selection, adding currently selected units to an existing saved agent set, or queueing orders to be executed after an agent completes its current commands. Because modifier keys are not typically present on mobile devices, this design proposes a new convention called a *rolling release*.

A rolling release is a variant of a single-finger gesture such as a drag or tap (it could potentially be used with some multi-finger gestures as well). It is performed by placing another finger, generally from the same hand, near the existing screen gesture and holding it there as the original gesture is released. So, a rolling tap would be performed by putting down one finger, setting another finger next to it, and then releasing the first finger before lifting the second. A rolling drag consists of putting down one finger and moving it across the screen, setting another finger by it at some point during the motion (possibly moving both together), and then releasing the original finger first, as demonstrated step by step in figure 13.

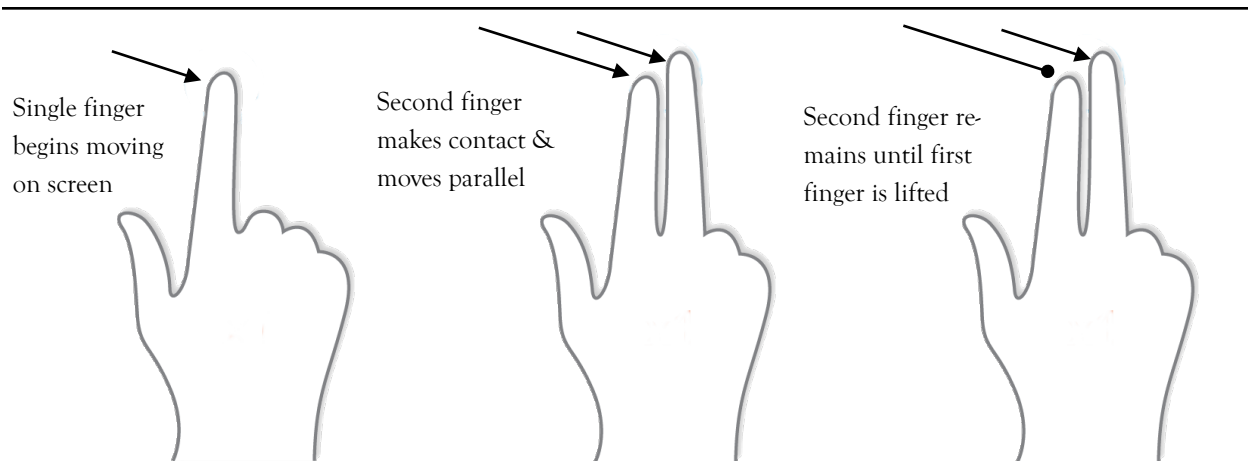


Figure 13: The life-cycle of a rolling-release drag gesture

---

## Ergonomics

Using hands and fingers for input poses some challenges that a mouse cursor does not; they are bulky and opaque and may obscure parts of the screen inconveniently. This suggests that the design should avoid placing any critical information (such as the contents of the current selection or the player's current resource availability) along the bottom of the screen, especially towards the corners, as these will frequently be obscured. It also suggests that care should be taken when placing interactive elements near the lower-right corner (or the lower left corner if the interface is configured for left-handed use), where they may accidentally be triggered by parts of the dominant hand passing across them as it moves.

While existing examples of mobile RTS games place the mini-map at the upper-right corner, presumably to make it convenient to see and to touch specific locations with the dominant index finger, observation suggests that most mobile users hold the device with their non-dominant hand low on the appropriate side (the left side of the device in most cases), with the thumb folded over the edge. The proposed design proposes to use this to advantage by allowing operation of the mini-map with the non-dominant thumb, by placing the mini-map in the lower-left corner (or the lower-

right corner). This leaves the area above the mini-map for other UI elements, such as a display of selected agents, as well as a button to switch the selection back to whatever the selection's previous contents were. This arrangement allows the non-dominant hand to tilt the device and use the thumb to move the view using the mini-map, easily pull back from the mini-map to make it visible, and leave the other side of the display available to the dominant hand to maneuver, select agents, and issue commands. Figure 14 presents the overall layout of the described elements.



Figure 14: Proposed War Table interface, with training command in progress

## Chapter 4: Test Project & Experiments

In order to test the usability of the interface and compare some of the interaction techniques being incorporated, a prototype was developed and a user study conducted.

### 4.1: Test Project

The prototype was produced in Unity 3D version 4.2. An existing project was discovered which provided a minimal example of an RTS produced in Unity and used as a foundation for the original structure, although no code assets from this project were used in the final test prototype. The prototype was generated as an Android application because the available test device was a Nexus 7. The project, dubbed *War Table*, was ultimately divided into four principal sections:

1. TouchUI wraps an objects-and-events layer around either multi-touch input or mouse input through a common set of APIs (mouse input being used primarily for debugging). It provides three classes, founded around a Contact class that represents a single touch on the screen from start to finish, or a mouse action from when the button is pressed until it is released. A Gesture class encapsulates one or more Contacts and provides abstract interpretations of these contacts, such as whether one represents a tap, drag or other gesture. The UIElement abstract class provides mechanisms for recognizing gestures on various areas of the screen such as GUI elements and responding to them. Use of Unity's built-in GUI package was kept light because the standard elements are not designed for interactions like dragging.
2. The Button package uses the TouchUI package to provide standard elements such as round and square buttons to tap on or drag, and an abstract Menu class that provides the foundations for various formats such as the radial menus used to control the active selection.



3. The Battlefield package provides the abstract mechanisms that underlie the RTS mechanics of the game, such as agents, orders, weapons, damage, resources, pathfinding and a battlefield.
4. The WarTable package provides things specific to this project, such as specific agents and weapons, dynamic radial menus, unit group buttons, and other UI behaviors and elements.

The TouchUI package was written in C#; the rest of the program was originally written in Boo, a Python-inspired language compatible with the .NET/Mono common language runtime. A simple set of automation behaviors was also created to provide a computer-controlled opponent.

Two scenarios were created. The first one was intended as a training scenario, where the human and computer players controlled bases at opposite ends of a narrow lane with a crook in the center. The second scenario was more involved and asymmetrical, where the player controlled bases in each of the four corners of a square area, with lanes between the bases along the sides of the square, and additional lanes from the centers of those sides leading to an open area in the center, where a single large base belonging to the computer player was situated. Figure 15 shows how these two levels were arranged.



Figure 15: War Table's test scenario maps

---

Pilot testing was done with a project that included a full set of common RTS elements, including resource gathering, worker units engaged in harvesting and building construction, buildings training new units, additive actions and saved selection management. Observation and responses of pilot testers indicated that while the proposed interface might prove robust, the tests were too short and the participants' exposure too shallow to provide meaningful feedback on specific elements of the interface.

## 4.2: Experimental Protocol

In light of the pilot test observations, simpler research questions were instated and the prototype's complexity was reduced significantly to promote focusing on those. The final research questions incorporated into the study were:

1. Is tilt input a functional technique to drive relative camera motion, particularly for short-range local view adjustments?
2. Between the command composition techniques of a) *tap-tap* (tapping the desired action, then tapping the destination point), or b) *drag-release*, (dragging the action button to the desired destination and releasing it there), is there a significant difference in terms of accuracy, speed or user preference?

In order to focus on these questions, several features were removed from the prototype:

- Resources and worker units were removed from the test scenarios. Players were provided with a selection of pre-determined buildings at the start of the scenario and were not able to create new buildings.

- Players were not able to control the training of new units at their buildings; buildings automatically produced a single type of unit at a regular interval and dispatched them to staging areas.
- Additive action inputs and memorized selection buttons were identified as advanced techniques that received little interest from pilot testers, and were backed out for the duration of the study.
- While the original prototype allowed tap–tap and drag–release to be used inter-operably, for testing purposes each participant in the study was restricted to using either one or the other, in order to do between-subjects comparison.



Figure 16: The interface as implemented for testing

In addition to removing these features, a feature was added where the staging area for each building’s new units received an associated button in the GUI, which displayed the number of idle units located near that area, and functioned similarly to saved selection buttons, allowing the play-

er to quickly select and command all idle units in that area. The prototype was also programmed to record details of the players' actions during the test sessions, including the device angle, contacts on the screen, selections from menus, and damage inflicted on agents. Figure 16 shows the interface as it was presented to participants in the user study.

Participants signed up through a campus study system, and were primarily students in a psychology course who received course credit for participating in various studies. Each participant was first required to sign a consent form, then filled out a demographic questionnaire regarding their experience with mobile computing devices, computer games in general, and real-time strategy games in particular. The participants were then presented with the test device, and played through a training scenario to become familiar with the controls and mechanics of the game. As they used the game software, the various features were explained to them by the investigator. The training scenario ran up to ten minutes, or until all the agents on one side were eliminated. In most of the cases, the investigator observed quietly for the second half of the training scenario.

After the test scenario, participants were allowed up to a five-minute break and invited to ask any questions they might have about the prototype game or its controls. After break, they were presented with the test scenario, and told that they had up to thirty minutes to complete it and that the investigator would not be allowed to answer any questions once the test scenario began.

After they completed the test scenario, participants completed a written questionnaire about their subjective experience of the game, and were invited to make verbal comments about their experience as well. Participants were then offered a debriefing form explaining the specific goals of

the study, as well as a chocolate, and dismissed. The study proposal was submitted to the WPI IRB and approved under expedited review.

### **4.3: Experiment Outcomes**

There were 18 participants in the user study, ten males and eight females. All but one were university undergraduates. Fourteen of these said they used mobile computing devices on a daily basis, with the other four ranging between “never” and “occasionally.” Participants on average reported their engagement in RTS games as between “familiar, but don’t play” and “play occasionally” with one “play regularly”. They reported their experience with computer games overall as typically between “play occasionally” and “play regularly” with one “play extensively.” (There were also multiple “don’t play” responses for both categories.)

The participants were all able to complete the presented scenarios without technical difficulties; eight participants were able to defeat the computer in the test scenario and ten people were defeated by the computer.

#### Camera Control

Interestingly, desktop schemes for relative camera control are difficult to incorporate into the standard flow of RTS play; hovering the mouse cursor over the edges interferes with using the mouse to select targets, and the arrow keys are not close to the standard left-handed keys used as hotkeys in most setups. As a result, skilled players tend to focus on using hotkeys or the mini-map to control the camera. It was therefore going to be of interest how tilt camera control would be received, as this method of relative camera control interoperates without directly obstructing other input vectors. Based on observations from Hamilton et al. [17], as well as van Tonder & Wesson’s

difficulties in developing an acceptable implementation of speed-dependent automatic zooming (SDAZ) [28], it was hypothesized that tilt operation of the game camera would be usable for short-distance or exploratory adjustments, but that camera targeting via the mini-map would dominate in long-range applications.

Participants used a mean of 296.5 tilt-driven adjustments to view position during their sessions, with a range from 23 to 1322 and a standard deviation of 345.75. In contrast, participants used a mean of 151.9 map-driven changes in view, with a range from 9 to 592 and a standard deviation of 146.2. An analysis of variance on tilt vs. map view event counts yielded a  $p$  of 0.1114, which suggests that the difference in frequency of use for the two methods may be significant, but further study would be required to make definitive statements.

An analysis of variance on the distances traveled across the world by means of tilt yielded a  $p$ -value less than 0.00001, indicating a decisive difference in the distributions of typical travel distances. The mean displacement to camera by means of map interaction was 200 game units, while the mean displacement applied through device tilt was 55 game units. The disparity was pronounced enough that despite the more frequent use of tilt transaction to move the camera, the mean total displacement applied per session by using the map was nearly twice what was applied via tilt (30,412 vs 16,314 game units). Figure 15 shows clearly the difference in the distribution of tilt events versus map events for each user, according to distance traveled.

Several participants commented on the tilt mechanic and its effectiveness. Five participants observed that they found it very easy to overshoot their target area when using tilt control or that it was too sensitive. Three users commented that the mini-map was easier to use than the tilt con-

trols. However, three users reported that tilt was usable or convenient, and one user particularly commented that the combination of tilt-driven relative control with map-driven absolute positioning was extremely flexible.

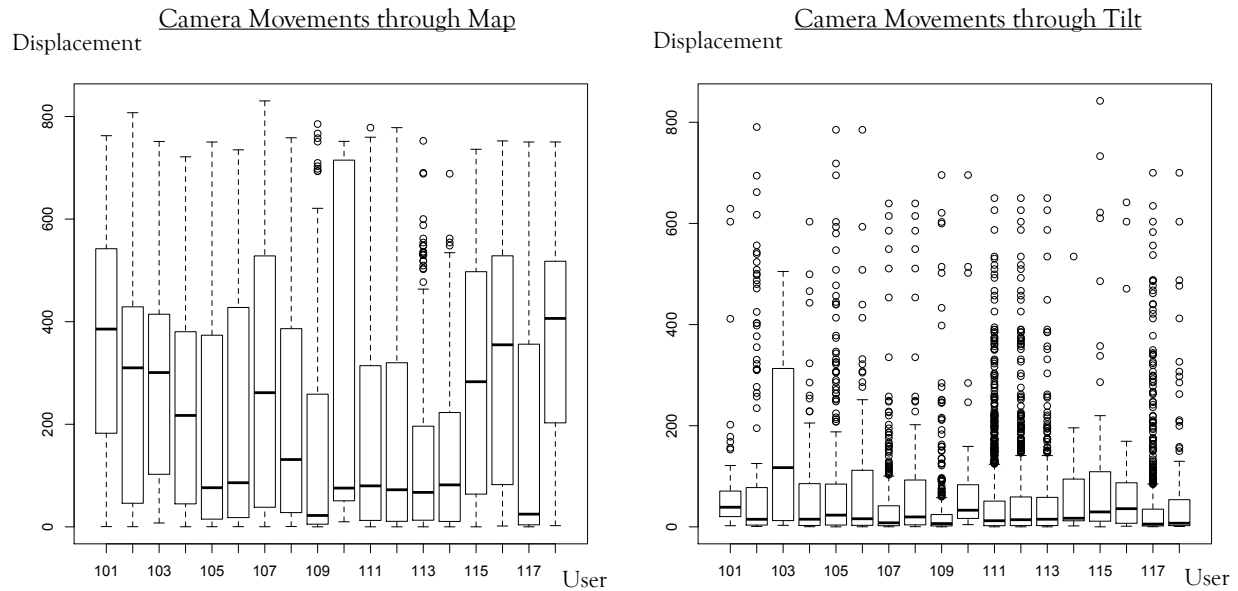


Figure 17: Distributions of camera displacement distances for each user, executed through map (left) and tilt (right)

### Menu Interaction Mechanisms

The other portion of the study conducted was a between-subjects examination of whether there was a significant difference in order execution between participants who first tapped their chosen command, then the target, and those who dragged the command to the target and released it there. The goals were to evaluate whether the interaction mechanism made a difference to the accuracy or speed of individual orders, the effectiveness of over-all play outcomes and user opinions of interface naturalness. As a proxy for order accuracy, the fraction of orders that were cancelled or overridden within five seconds of being issued was estimated. Speed was determined from the interval between the menu becoming reserved for a specific action and losing that reservation. Play

outcome was gauged between two metrics; the ratio of all damage dealt by the participant's agents to all other agents over the total damage dealt to the participant's agents, and rank in terms of whether they won and how quickly they resolved the scenario. This was determined by taking the time until they completed the scenario, negating it if they lost, and taking the reciprocal. This scheme ranks a fast win as the best and a fast loss as the worst outcome, with a slow win being counted better than a slow loss. User perception was evaluated through Likert-scale assessments of the user interface's overall naturalness as well as of satisfaction in using the touch UI elements. While no significant sources were found that discussed directly the effect of moving in contact with a surface on Fitts' Law-applicable tasks, the work of Potter et al. [25] as well as that of Lindeman, Sibert & Templeman [23] suggests that contact with a surface could increase the accuracy of pointing tasks, so it was hypothesized that participants using drag-release interaction could execute agent commands more accurately.

The mean time that the menu had an outstanding command selection was 1.456 seconds for drag-release users and 1.557 seconds for tap-tap users, with a p-value of 0.68. The mean error rate was 9.7% for participants using drag-release and 14.1% for those using tap-tap, with a p-value of 0.34. For damage ratios, the mean was 0.927 for drag-release participants and 1.055 for tap-tap users, with a p-value of 0.528; the mean rank for drag-release participants was 8.8 and that for those using tap-tap was 10.1, with a p-value of 0.641. Wilcoxon signed-rank tests on user opinions of the overall game experience and satisfaction at using the touch UI provided a slightly higher pleasure at the touch interface for tap-tap users, with a p-value of 0.814; but interestingly, the typical overall game experience was ranked as neutral on average by tap-tap users but consistently ranked as "enjoyable" by drag-release users, with a p-value of 0.119.



## Chapter 5: Conclusions & Future Work

Analyzing the play data and subjective feedback yielded some insights into the proposed interface elements. Despite a slight preponderance of negative comments about the tilt interface, the extent to which it was used by all participants indicates that it was readily accessible as a mechanism to move the camera relative to its current position, and that the participants had an interest in local navigation which was not always served by using the mini-map. It also becomes fairly clear that the tilt controls for the camera are not as suitable for regular long-distance navigation of the map.

Results regarding the different types of menu selection mechanisms rarely even approached statistical significance, although it is possible that future, more specific work might bear out higher accuracy with drag interaction over sequential taps. Moreover, only one user even mentioned the tap or drag characteristics of their session in comments. This suggests that since the two modes are interoperable, both could be included in a game client in order to appeal to a broader selection of players; alternatively, one scheme or the other could be replaced with other forms of input interpretation, if they were devised.

### 4.1: Directions for Future Inquiry

While the data indicate that tilt input for relative view control is easy to access and fulfills a significant function, there was consistent feedback regarding its sensitivity and difficulty to control finely. Van Tonder and Wesson propose a system called IntelliTilt [27] wherein tilt control of a map can snap towards points of interest when they pass close enough to the application focus. One future test would be to modify the tilt control so that large concentrations of agents could attract

the camera to center on them when it was in motion. Groups that consisted of agents from diverse factions in battle would probably have a higher coefficient of attraction.

It might be useful to have a more focused test to more clearly settle the question of whether tap or drag offers any difference in speed or accuracy, particularly in an environment where users can use the program repeatedly and become familiar with its nuances. A publicly available target-touching game similar to Hit It! by Henze et al. [18] would be a way to generate a large body of data and settle the question more definitively. While no serious sign of statistical relevance appeared in the data of this experiment, the small participant pool and limited experience of each participant make it hard to consider the experiment fully definitive. It would also be useful to test the different forms for particular tasks, like targeting an attack order on an enemy agent that was in motion.

There are a number of possible mechanisms available for converting the start and end of a drag motion into a selection area. For simplicity of implementation, this project adopted a square area of the virtual battlefield with its corners defined by the specified points, but other possibilities include the use of more elaborate shapes such as Hamilton's isosceles lasso [17]. Options also include tracing a freehand shape around the units one wishes to include, or using the start and end points to define either the diameter or the radius of a circle. The diameter approach in particular seems as though it might lend itself to considerable nuance in adjusting a selection area as it was being defined. A categorical comparison of these techniques could be very useful.

The use of additive actions through rolling-release proved to address a greater level of complexity than the single-session participants in this study were ready to address. It could be valuable to pursue a more protracted study with repeat participants, in order to assess the usability of this

technique and compare it with other mechanisms to distinguish possible intentions of a touch input, such as long presses or the use of a defined area to select modes through chordic finger presence, similar to Hamilton's PiHC design.

Two other mechanisms were developed for the first prototype that did not appear in the test scenarios. One was an alternate command composition where the desired actor is selected, then the destination, then the action; by dragging from the agent to the destination, the command can be composed as soon as an action is selected from the menu that appears on release. This method is interesting for three reasons: it is interoperable with the other two, it allows the menu to consist of only those commands relevant to the selected actor and target, and it adds the possibility for the agent selection to not be stateful. For example, a worker unit could be dragged to an empty space on the map and directed to construct a building there, after which the current selection group would be reactivated.

The other mechanism that was developed for the pilot was a menu structure intended to let the player train new units without having to return the view to his base. A button was placed below the saved selection buttons which toggled a list of building types in a wedge around it. Each building button opened a sub-menu displaying the units that building type could produce, in a wedge concentric with the first menu, in an angle centered on the button that opened it. Tapping one of these buttons would issue a training order to one building of that type; however, the buttons could also be dragged to a point on the map, causing the newly created unit to travel there as its first order when it finished training, or to a saved selection button, in which case the new unit would be added to the specified saved group and move to join it when it appeared. This functionality is usu-

ally difficult to incorporate into desktop RTS control schemes and received favorable feedback from pilot testers.

The variety of control techniques available to bring deep and engaging RTS games to the touch-screen platform spaces appears to be rich with avenues worth investigating.

## References

- [1] "Age of Empires 1.0." *1Download1.com*. Web.
- [2] *Amoebattle*. Grab Games, 2012.
- [3] *Anthill: Ants Ain't Saints*. Image & Form, 2011.
- [4] *Command & Conquer 3: Tiberium Wars*. Electronic Arts, 2007.
- [5] *Command & Conquer: Red Alert for Ipad*. Vers. 1.8.20. Computer software. Electronic Arts, 2011.
- [6] "Command and Conquer 3: Control Interface & Side Bar Definitions." Video Games Blogger 2006. Web.
- [7] "Disco's Starcraft Ii: Wings of Liberty Review." *High Def Junkie*. Web.
- [8] "Rise of Nations: Thrones & Patriots Screenshot #1 for Pc." *GameFaqs*. Web.
- [9] Big Huge Games, *Rise of Nations*. Microsoft Game Studios, 2003.
- [10] *Starcraft 2*. Blizzard Entertainment, 2010.
- [11] "Total Annihilation/Controls." StrategyWiki. Web.
- [12] Banovic, Nikola, et al. "Design of Unimanual Multi-Finger Pie Menu Interaction." *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. 2076378: ACM, 2011. Print.
- [13] Callahan, J., et al. "An Empirical Comparison of Pie Vs. Linear Menus." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 57182: ACM, 1988. Print.
- [14] Cavedog Entertainment. *Total Annihilation*. GT Interactive, 1997.
- [15] Ensemble Studios,. *Age of Empires*. Microsoft Studios, 1997.
- [16] Fitts, Paul M. "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement." *Journal of Experimental Psychology: General* 121.3 (1992): 262-69. Print.
- [17] Hamilton, William, Andruid Kerne, and Tom Robbins. "High-Performance Pen + Touch Modality Interactions: A Real-Time Strategy Game Esports Context." *Proceedings of the*

- 25th annual ACM symposium on User interface software and technology. 2380156: ACM, 2012. Print.
- [18] Henze, Niels, Enrico Rukzio, and Susanne Boll. "100,000,000 Taps: Analysis and Improvement of Touch Performance in the Large." *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. 2037395: ACM, 2011. Print.
- [19] Hinckley, Ken, et al. "Sensing Techniques for Mobile Interaction." *Proceedings of the 13th annual ACM symposium on User interface software and technology*. 354417: ACM, 2000. Print.
- [20] Hopkins, D. "The Design and Implementation of Pie Menus." *Dr. Dobb's Journal*. December (1991) Print.
- [21] Maxis, *The Sims*. Electronic Arts, 2000.
- [22] Legault, Nicole. "Sims-Like Pie Menu Using Storyline." *E-Learning Heroes 2013*. Web.
- [23] *The Effect of 3d Widget Representation and Simulated Surface Constraints on Interaction in Virtual Environments*. Virtual Reality, 2001. Proceedings. IEEE. 17-17 March 2001 2001. Print.
- [24] *Tilt to Scroll: Evaluating a Motion Based Vibrotactile Mobile Interface*. Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint. 18-20 March 2005 2005. Print.
- [25] Potter, R. L., L. J. Weldon, and B. Shneiderman. "Improving the Accuracy of Touch Screens: An Experimental Evaluation of Three Strategies." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 57171: ACM, 1988. Print.
- [26] Samp, Krystian, and Stefan Decker. "Supporting Menu Design with Radial Layouts." *Proceedings of the International Conference on Advanced Visual Interfaces*. 1843021: ACM, 2010. Print.
- [27] Tonder, Bradley van, and Janet Wesson. "Intellitilt: An Enhanced Tilt Interaction Technique for Mobile Map-Based Applications." *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part II*. 2042167: Springer-Verlag, 2011. Print.
- [28] ---. "Is Tilt Interaction Better Than Keypad Interaction for Mobile Map-Based Applications?" *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*. 1899539: ACM, 2010. Print.
- [29] Westerman, Wayne. "Hand Tracking, Finger Identification, and Chordic Manipulation on a Multi-Touch Surface." University of Delaware, 1999. Print.

# Appendix: User Study Materials

## Script used by Investigator

You are being asked participate in a study to evaluate a control scheme for playing overhead strategy games on touch-screen devices. Participation will take about an hour and consists of completing a consent form, playing through two scenarios in a simple strategy game using a tablet computer and providing feedback on your experience, in the form of two short questionnaires and verbal comments. In addition, the game will collect anonymous information about your play actions for analysis.

Please read and sign this form indicating your consent to participate in the study.

Please take a minute to complete this questionnaire about your level of experience with computer games and mobile devices.

First, you will complete a simple training scenario to become familiar with the game and controls, using this tablet computer. You should see a green area with rocky borders containing gray buildings with red trim, as well as two small vehicles with red accents. The buildings will automatically generate new vehicles as time passes. Your computer opponent controls similar buildings and vehicles which are blue. You will win the scenario if your vehicles destroy all the blue buildings and vehicles, or lose if all of your vehicles and buildings are destroyed.

Tap on the screen when you're ready to begin.

You will need to move your attention between different parts of a large area in order to play effectively. Try tilting the upper left corner of the tablet down and away from you so that your view begins to slide toward the other corner of the area. You can control the speed at which the camera moves by tilting the device more or less steeply. You should reach the other corner and see the blue buildings there.

You can also look around an area by touching the 3D view on the screen with two fingers and sliding them around. Try this to look around the area near the enemy base.

In the lower left (right) corner of the screen, there is a top-down view of the entire area, with a white frame indicating the portion you see in the 3D view, and colored dots indicating your buildings and vehicles in red, as well as enemy buildings and vehicles near your forces, in blue. Touch the lower-right of this view, where the red dots are, to move your view back to your starting area.

In order to destroy enemy buildings and vehicles, you have to move your own vehicles close enough to engage them. You do this by first selecting the vehicles you want to give an order to; try tapping one of the two round vehicles in front of your base. A green ring should appear around it, along with a wheel of round command icons. Tap the other vehicle and the ring and icons will jump to that one. Tap the arrow button in the upper left (right) just next to the 3D view to re-select whatever you mostly recently had selected.

Tap the red circle with a 0 in it, in the upper left (right) of the screen, to clear your selection; the menu will also disappear.

There are three colored buttons on the left (right) side above your map. Each one corresponds to a matching colored zone in front of your base, and shows the number of vehicles near that zone that are not currently doing anything. You can tap any of these circles to jump to the matching zone and select those vehicles, or drag off of that button to select them without jumping your view there and position the menu wherever you like.

To select multiple units, you can drag your finger across the group. Try selecting both vehicles in front of your base.

To send your selected vehicles to another point on the map, use the command circles that appear when you select them. ([1] First tap the desired command so that the others become dim, then tap the point in the 3D view where you want the vehicles to go in order to complete the command.) ([2] Touch the desired command, then drag it to the point in the 3D view where you want the vehicles to go in order to complete the command.) You can also tap/drag to a point on the minimal in the lower left (right) corner.

The arrow pointing up is the advance command; it will send the selected units to the targeted point, so that they stop and attack any enemies encountered along the way. The arrow pointing

down is the withdraw command; it sends the selected vehicles straight to the intended point, even if they are attacked or see enemies. The double circular arrow is the patrol command; it causes the selected units to move between their current point and the targeted position, fighting any enemies they encounter on the way.

In the top left (right) corner is a pause button; this pauses the game and lets you tweak game settings, such as changing the game to a left-handed layout, or adjusting the sensitivity of the tilt controls to change your view.

You will have up to ten minutes to destroy all the blue buildings and vehicles. The scenario will also end if all your vehicles and buildings are eliminated. During that time, you are free to ask me questions about the controls or other aspects of the game. Once you are done with this scenario, there will be a short break followed by a longer test.

(observe session)

You can now take up to five minutes before starting the final session. You are free to ask me questions during this time, but once the scenario starts I will not answer any more questions unless the software appears crashed or broken.

(break)

You are now ready to begin the test scenario. This takes place in a larger, more complex map, where you have multiple bases placed at all corners. Your goal is to destroy the large blue base and vehicles in the center of the map within 30 minutes. During this time, I will not answer questions, but you may give feedback about your play experience. Because the scenario is more complex, the software may start to become sluggish; if this happens, you may need to leave your fingers on the screen for longer in order for the software to pick up your input.

Once the test is done, you will be asked to complete a short questionnaire and make any comments.

You may tap the screen to begin.

(observe session)



Please take a moment to answer these questions about your play experience.

Before you go, what other comments would you like to make about the game prototype and its control scheme?

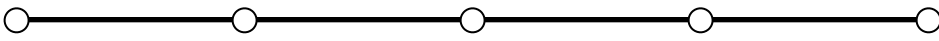
Please feel free to take a chocolate if you'd like one. Thanks for participating in our user study, and have a great day!

# Usability Study in Dynamic Touch-Based Interfaces for Real-Time Strategy Games

## Demographic Questionnaire

Participant ID: \_\_\_\_\_

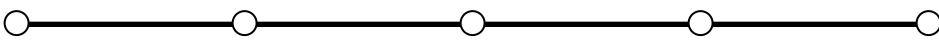
1. Please describe the degree to which you use tablet computers or other mobile devices:



○ ————— ○ ————— ○ ————— ○ ————— ○

Never use                  Once or twice                  Monthly                  Weekly                  Daily

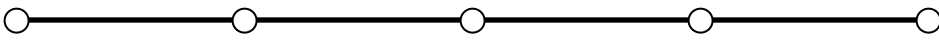
2. Please indicate how familiar you are with real-time strategy games:



○ ————— ○ ————— ○ ————— ○ ————— ○

Unfamiliar          Familiar, but don't play          Play occasionally          Play regularly          Play extensively

3. Please indicate how familiar you are with digital games in general:



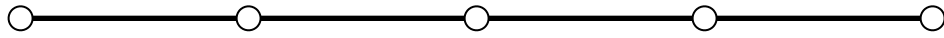
○ ————— ○ ————— ○ ————— ○ ————— ○

Unfamiliar                  Familiar                  Play occasionally                  Play regularly                  Play extensively

Usability Study in Dynamic Touch-Based Interfaces for Real-Time Strategy Games  
Experience Questionnaire—Tablet

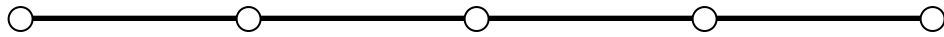
Participant ID: \_\_\_\_\_ TT DR

1. Please indicate your feeling of the overall experience of playing the game:



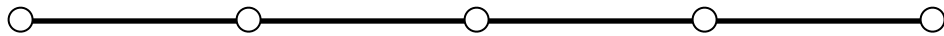
Very unpleasant      Unpleasant      Neutral/indifferent      Enjoyable      Very enjoyable

2. Please indicate how natural you found the game's controls:



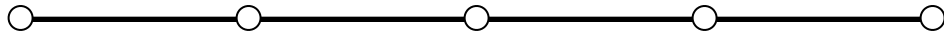
Very awkward      Awkward      Uncertain      Natural      Very natural

3. Please indicate how you felt about tilting the device to control your view:



Very frustrated      Frustrated      Indifferent      Pleased      Very pleased

4. Please indicate how you felt about using your fingers to control the menus:



Very frustrated      Frustrated      Indifferent      Pleased      Very pleased

5. Thinking about the play session, please describe a moment, if any, when you were conscious of being frustrated by the game controls:

6. Thinking about the play session, please comment on anything that struck you as easy or convenient:

## Debriefing Form

**Project Title:** Tilt, Touch and Strategy Game Play

Real-time strategy games are still scarce in the mobile device space. While this is partly due to the high computing demands of these games, they also place considerable demands on the interfaces used to stay informed of what is happening in the game and issue commands to the player's agents in the game.

Because successful play in real-time strategy games requires coordinating the actions of separate squads and reinforcements across a wide area, the interfaces for playing these games have to make it easy for the player to do certain common jobs such as rapidly moving their view from one part of the game map to another and giving groups of minions precisely targeted orders, while still letting the player see the details of action as clearly as possible. This project proposes an interface which should make it easier to handle these tasks.

This study seeks to evaluate the overall usability of the new interface and how well it facilitates enjoyable play. It also specifically tries to assess how players use tilt controls to manage their point of view in the absence of arrow keys or mouse position, and to compare the accuracy of interactions based on the common mobile paradigm of tapping a command, then tapping a target, with those based on dragging the command to the target.

We predict that tilt control is more useful and precise for short-range changes in view, and that drag interactions will be more accurate in the placement of command targets at higher net speeds than sequential tap interactions.

If you have any questions or comments, feel free to ask me now. If you have any further questions or comments, please contact Robert Lindeman at 508-831-6712. Because other students may be participating in this study in the future, we ask that you not discuss the details of this study with your friends or classmates through the end of the Spring 2014 semester.

If you are interested in reading more about this topic, try the following:

William Hamilton, Andruid Kerne, and Tom Robbins. 2012. High-performance pen + touch modality interactions: a real-time strategy game eSports context. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (UIST '12). ACM, New York, NY, USA, 309-318. DOI=10.1145/2380116.2380156 <http://doi.acm.org/10.1145/2380116.2380156>

Bradley Paul van Tonder, Janet Louise Wesson, Improving the controllability of tilt interaction for mobile map-based applications, *International Journal of Human-Computer Studies*, Volume 70, Issue 12, December 2012, Pages 920-935, ISSN 1071-5819, <http://dx.doi.org/10.1016/j.ijhcs.2012.08.001>.

**Thank you for your participation!**

### Selected Comments: Criticism

"When the game would slow down, the tilt controls would only respond to large tilts. In general however, the tilt was too sensitive. There is no need for high sensitivity since it is more intuitive to use the minimap for large movement."

"When I would tilt the screen so that I could send my cars to the middle and it would go much farther away than I anticipated."

"Sometimes touching didn't feel responsive. This was especially the case when there were FPS drops."

"Tilting the tablet sometimes made me move further then I meant to, and it was sometimes difficult to select new groups of units."

"When the game did not differentiate between selecting units or moving the camera, I was aware of the limitations of the game controls. The tilting of the tablet limited the user's ability to select and utilize newly-formed units."

"Occasionally when trying to use my finger to control the actions of the vehicals the device would tilt slightly and alter what I was trying to do. Sometimes, the vehicals would get stuck on the ridges and not move again until I told them to."

### Selected Comments: Appreciation

"I enjoyed being able to click anywhere on the expanded map that allowed me to view a specific area."

"This ability to control groups of unused vehicles was convenient and simple. The drag to select was also easy to use."

"I liked that the map in the bottom corner could easily get you where you wanted to go when you pressed on it. That was convenient instead of tilting the screen around."

"being able to select multiple vehicles at a time was very convenient, (directing them all at once). The colored buttons to control all vehicles in that area at once made the game a lot easier."

"Scrolling and drag box selection was convenient and I think the wheel UI is a good idea, it was just a tad not sensitive enough and didn't always register input."

"The circular buttons that were dragged onto the map were easy and intuitive to use."

"Sliding control + minimap offers great flexibility for movement. Accelerometer controls could cut down on UI clutter. Movement menu being context-sensitive was a nice touch."

"I did like the tilt to move the view around, it was easy and convenient once I got the hang of it."

## Participant Questionnaire Data

---

Participant	Mobile Experience (0-4)	RTS Experience (0-4)	Game Experience (0-4)	UI Type	Game Enjoyment (-2--+2)	UI Transparency (-2--+2)	Tilt Naturalness (-2--+2)	Touch Naturalness (-2--+2)	Rank
101	4	0	0	TT	0	-1	-1	-1	12
102	2	1	4	DR	2	1	2	1	1
103	4	2	3	TT	1	0	-1	1	3
104	4	0	1	DR	1	1	0	1	8
105	4	1	4	TT	-1	-1	-1	1	10
106	1	1	2	DR	0	-1	-1	0	14
107	4	2	2	TT	1	1	0	2	16
108	4	2	2	DR	1	-1	0	1	15
109	4	1	0	TT	1	0	-1	0	13
110	4	0	0	DR	1	-1	1	-1	18
111	0	2	3	TT	0	-1	1	-1	4
112	4	1	1	DR	1	-1	0	1	6
113	4	2	3	TT	-1	1	2	1	9
114	4	2	1	DR	-1	0	1	-1	11
115	4	2	2	TT	1	-1	-1	1	17
116	4	3	3	DR	1	-1	-1	0	5
117	1	0	3	TT	-2	-1	0	-1	7
118	4	2	3	DR	1	0	1	0	2