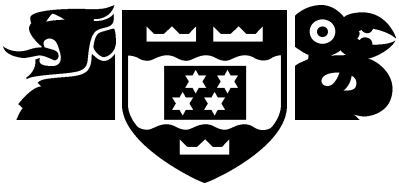
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Engineering and Computer Science
Te Kura Matai Pukaha, Purorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Sketch Interaction in Real Time Strategy Games

Elwyn Benson

Supervisor: Dr. Peter Andreae

Submitted in partial fulfilment of the requirements for
Bachelor of Information Technology.

Abstract

As real time strategy games are becoming increasingly complex and large scale, new interaction techniques need to be investigated to overcome the limitations found in conventional interaction methods. This project has explored freehand sketching as an interaction technique, and this report discusses some of the advantages and disadvantages of sketching for spatial tasks, which are common in real time strategy games. The report describes two novel sketch interaction techniques for giving precise orders to units— a technique for selecting groups of units, and a technique for specifying movement paths. A user experiment has been conducted that evaluated the novel interaction techniques, which is described. The results have shown that sketching does not provide a clear quantitative advantage in terms of efficiency for selecting units but seem to be comparable to the conventional technique. Further study is required to confirm this due to issues with the experimental design. The results for movement paths are positive, however not by a significant amount. Despite this, the qualitative feedback from participants of both experiments has been almost completely positive, which combined with sketching being comparable to the conventional techniques suggests there may be further work to be done to take this idea further.

Acknowledgements

Dr. Peter Andreae

For being a crucial part of both ends of my degree- from teaching me Java in 2007 to supervising this project in 2010. Your kindness, support and extensive knowledge make me feel privileged to have you as a supervisor.
Despite what Salient voters might say, you are *my* Academic Idol!

Dr. Stuart Marshall

For introducing me to the world of H.C.I. and piquing my interest in all things usability related, ultimately culminating in this project.

All of my lecturers and teachers over the last few years

For helping me get to where I am today and imparting your wisdom in the process.

Fonda

For proof reading everything and putting up with the espresso machine buzzing at all hours!

Contents

Chapter 1

Introduction

The real time strategy game genre requires the user to give lots of commands in a situation where time is a constraint. Users must constantly micromanage their units, buildings and resources. Many of their tasks involve selecting units, specifying commands and issuing instructions within a spatial, geographic context. The standard user interface in real time strategy games is less than satisfactory for many of these interactions.

This project has explored the use of freehand sketch based input for selecting units and issuing movement commands. These novel interaction techniques have been implemented in a custom built real time strategy game engine.

This engine was used in the core component of the project — a set of user experiments that measured and compared participants' performance using the sketch interaction techniques and the conventional mechanisms. There were a total of 28 users participating in three experiments.

The results for selecting units do not show any clear benefit for sketching, in fact, sketching was consistently marginally slower. Issues with the experimental design mean the results may not show what they are expected to.

Sketching movement paths however has shown a consistent, if slight, performance benefit in both time (specifically the time taken to execute the actions) and accuracy (the quality of the paths/waypoint set created).

Qualitative feedback from participants has been very positive, with the majority commenting that they preferred sketching and would like to see it used in actual real time strategy games.

I believe that with the novel features at least holding their own in terms of quantitative results, combined with the positive feedback from users, sketching merits further investigation.

Contributions

- Explored the use of freehand sketching for issuing commands within real time strategy games.
- Designed and implemented two novel sketch-based techniques
- Evaluated the effectiveness of these techniques with user experiments

A paper based on the work in this project has been accepted into the 7th Australasian Conference on Interactive Entertainment[1] and will be presented in November.

[1] `http://ieconference.org/ie2010/`

Chapter 2

Background

2.1 Real Time Strategy Games

Real time strategy games are a popular game genre where players generally harvest resources, build and defend bases, and usually create armies to conquer enemies. These games require the player to micromanage their resources, units and buildings effectively in order to complete their goals and objectives. Professional real time strategy gamers aim for 100-300+ actions per minute, so an efficient means of issuing commands is crucial to the success of games in this genre.

There are numerous types of actions in real time strategy games, such as producing units, placing buildings and harvesting resources. Many of these actions are spatial or geographic in nature, such as selecting a group of units or specifying destinations or paths a unit should move along. Many of the interaction techniques found in real time strategy games are inherited from conventional software and do not appear to be ideal for these spatial/geographic commands. This project has focused on two components of these spatial/geographic commands— selecting a set of units and specifying movement commands for units.

2.1.1 Selecting Entities

Selecting units is a vital part of the genre, and the task may be executed hundreds of times within a single game. Conventional real time strategy game interfaces allow unit selection in three ways:

1. Selection box
   A rectangular box is drawn corner-to-corner, and all entities within the boundaries of the box are selected. This is the conventional method of selecting groups of entities, and is common in fields other than real time strategy games (such as file managers in operating systems).

2. Individual selection
   The most basic type of selection— clicking on an entity causes it to be selected.

3. Selection by unit type
   Double clicking on an entity will select that entity and all other entities of the same type currently visible.

Modifier keys allow the chaining of multiple selections together, and modifications to the current selection to be made. There are two common modifier keys found in many real time strategy games:

1. Adding selections
   Usually activated by holding the Shift key while selecting, this modifier simply aggregates multiple selections together

2. Toggling selections
   Usually activated by holding the Control key while selecting, this modifier toggles the selection— that is, if an entity is already selected it will be deselected, and visa-versa.

Where the units to be selected are interleaved with other units that must not be selected, a large number of selection boxes must be used. This adds a non-trivial amount of time and complexity, as well as increasing the chances of making errors.

Another scenario where selection boxes can be problematic is when the target entity to be selected is near other entities that must be avoided. The placement of the initial corner of the selection box is critical for the success of the action— it is very easy to misjudge the placement and either miss the target entity or pick up the unwanted entities. This problem is compounded when the entities are spread over a larger area, as visualising the correct placement point can be a difficult task. Figure 2.1 shows an example of this problem.

Selection boxes offer no way to recover from mistakes like this. If the box is placed incorrectly, the only option is to start again.

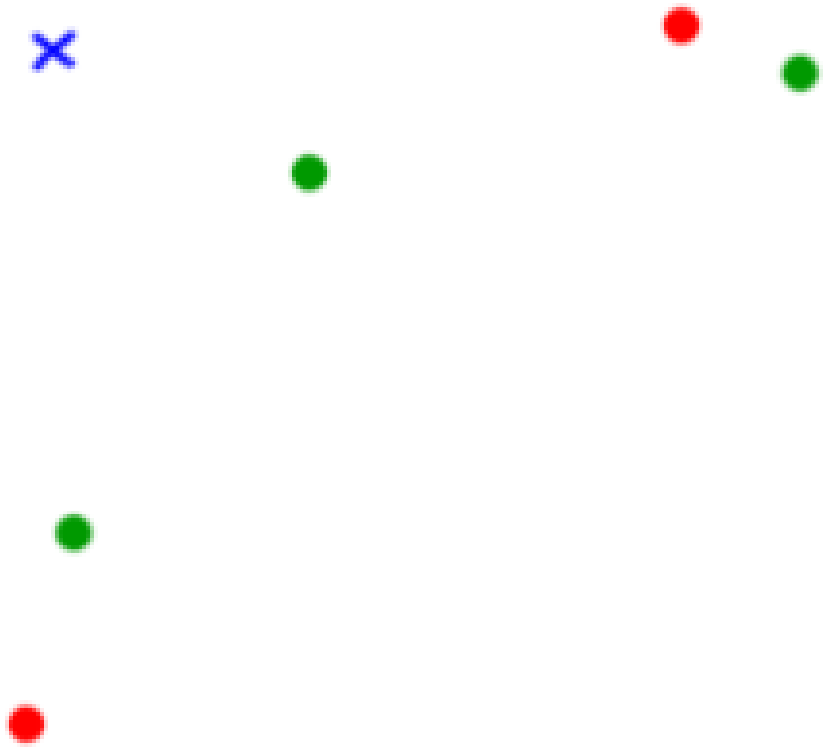


Figure 2.1: An example of a difficult case for a selection box. The green dots are desired, while the red dots are to be avoided. If drawing the box from the top left, the blue X is the only position the box can be successfully started from. Any errors and the green dots will be missed or the red dots selected.

2.1.2 Moving Units

Moving units is achieved by placing a sequence of one or more waypoints, that is, destinations the unit will move towards. A waypoint is conventionally placed by right clicking the destination, with multiple waypoints chained together using a modifier key (again, this is usually the Shift key).

Complex movement orders (for example, circling a forest while maintaining a certain distance to avoid detection by the enemy) are achieved by placing many waypoints to specify the path the unit should follow.

Where the path is not straight and must be specified precisely, users must place a large number of waypoints carefully to ensure the unit does not stray.

2.2 Sketching

Sketching with a mouse, tablet, or touch-screen as an input mechanism for creating diagrams or editing graphics has been around since Ivan Sutherland's seminal Sketchpad[13] system, and is used widely today. However sketching as an interaction technique is a novel concept, and little research has been undertaken on this topic.

In this report, sketching as an interaction technique (or simply sketch interaction) refers to freehand drawing with an input device, which is understood and interpreted as a command (or a parameter to a command). This type of interaction provides a natural way to issue spatial/geographic commands due to the inherently spatial nature of sketching.

The United States military has used sketching for a long time for developing battle strategies. The development of these strategies share many tasks found in real time strategy games. Recent efforts to digitise their battle strategy system have shown sketching to be an effective method of conveying spatial/geographic tasks[7]. nuWar[5] is a turn based strategy game that was developed from this work, and uses sketching to specify battle plans similar to those used in the United States military. The sketch interaction has shown promise in this context, and it is reasonable to expect real time strategy games may share the benefits of this interaction technique.

Aside from nuWar, sketching has not been used in many games, with the exception of games revolving around drawing, such as *Pictionary*. Some games use drawing to attempt to add realism/depth to the game (for example drawing a spell to simulate the player actually casting the spell). This type of drawing is distinct from the interactive sketching discussed in this report however, as it serves a different function (adding realism/depth to the game as opposed to providing an efficient interaction technique).

Sketching does have limitations however. The amount of hand movement required to create the sketches can be large, which implies sketching may take a nontrivial length of time. The precise movements required may impose high cognitive demands on the user, which may create a large planning stage for each sketch being created. Finally, using a mouse as an input device has been shown to be slower than more familiar devices such as a pen or pencil[1].

Chapter 3

Design and Implementation

3.1 Novel Features

Sketching selections of entities was the first novel sketch interaction feature that the project addressed. As mentioned in the previous chapter, selecting entities in real time strategy games is one of, if not the most frequent action performed. Sketching provides a natural means of indicating which entities are required — simply freehand drawing around the units required.

Specifying movement paths for units is another common task that sketching lends itself to. Drawing paths on the map is a familiar, natural way of specifying exact destinations for units. This was the second novel sketch interaction feature that the project addressed.

This section discusses specific details of both these novel interaction techniques next, including functionality and design decisions, and implementation factors.

3.1.1 Sketch Selections

Selecting units by sketching is relatively simple — by holding down the left mouse button and dragging the mouse, a sketch is started. This is the same action that would activate a selection box in a conventional real time strategy game. The mechanism of activating the sketch was made the same as activating a box selection as this is likely the most obvious way of selecting a group of units for most player of real time strategy games due to their existing experience. This is also the standard way of activating a selection box in fields outside of real time strategy games, such as operating system file browsers like Windows Explorer or Konqueror.

Upon releasing the mouse, the sketch is closed automatically. That is, a line is drawn from the starting point to the point that the mouse was released. Everything inside the boundary is selected (or toggled if the toggle modifier key is being used).

The automatic closing of the sketch means users do not have to manually draw back to the starting point. This is an attempt to mitigate the fact that sketching selections requires more mouse movement than a box selection since it draws a boundary around the entire area to select, whereas a selection box draws a straight line from corner-to-corner. As well as reducing the mouse movement required, the automatic closing line may be used where a straight edge in the selection boundary is desirable. Figure 3.1 shows the two different uses of the automatic closing line. Once familiar with the automatic closing feature, complex selections are possible with relatively small amounts of mouse movement. Fully utilising the automatic closing feature requires additional cognitive load, however practice appears to reduce this requirement.

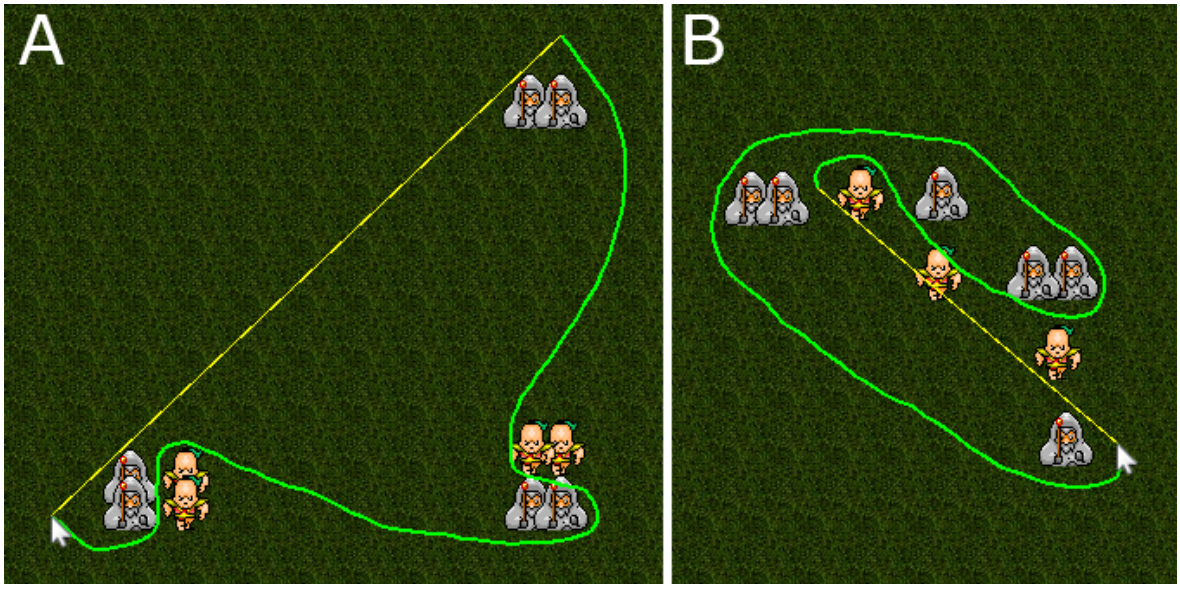


Figure 3.1: Two uses of the automatic closing line (in yellow). (A) shows the line simply being used to reduce the amount of mouse movement required while selecting. (B) shows the line being used where a straight edge is desirable for the selection.

The sketch is displayed on screen as it is drawn in neon lime green, which was chosen due to its high contrast against much of the terrain found in real time strategy games, as well as its use for selection boxes in other games. A hint line is displayed in yellow which shows where the automatic closing of the sketch would occur. This greatly increases users' ability to utilise the automatic closing functionality previously described.

It is worth noting at this stage that sketching selections does not aim to replace with the double click selection of all units by type, described in section 2.1.1. Figure 3.1 shows sketch selection being used to select a group of one type of unit. In an actual real time strategy game the same selection could be made by double clicking one of the target units, and all of the other units of this type would be selected. This only works when all of the units to be selected are of the same type, and there are no undesired units of the same type visible. Sketching is not restricted by this special case.

Implementation of Sketch Selection

Each sketch is stored as a series of points, which are generated as the mouse moves while sketching. This is useful for the experiment as it provides an exact record of every sketch made. Each point is timestamped, which allows the sketch to be replayed exactly as it was drawn by the user. If a sketch is drawn very rapidly, points are inserted further apart, while very slowly drawn sketches insert many points very close together. This is based on the assumption that users do not need such fine control when they are rapidly moving their mouse. When they are sketching slowly, it is assumed they are being careful with the accuracy of their sketch, so additional points are inserted to ensure the quality of the sketch.

The sketch can be rendered simply by drawing lines between each point, which gives the illusion of a curved line due to the proximity of the points.

A decision to make is whether units must be entirely within the bounds of the sketch to be selected, or if they will be selected just from one part of them, such as their center or origin (where they are standing on the ground). Given that the units may overlap, requiring

the entire unit to be inside the bounds of a sketch is very restrictive. Therefore a better choice would be choosing a single point of the unit to be the determining factor. The center of the unit is a good choice as it allows for a margin of error in the sketch. Users may overlap part of a unit when selecting another unit in close proximity but are unlikely to accidentally select across the center of the unit.

Collision detection, or determining what is 'inside' the sketch, is handled by the .NET `System.Drawing.Drawing2D.GraphicsPath` library, which allows complex sketches to be resolved and provides advanced functionality. For example, by circling a unit twice the unit will not be selected, (as the sketch has looped over itself, causing the loop to be counted as 'outside' the sketch). This is useful for allowing users to correct mistakes they have made in a sketch without having to start over, and provides an alternative to using the toggle modifier key, which may be preferable for some users.

3.1.2 Sketch Movements

Sketch movements are executed in much the same way as the sketch selections. Once one or more units have been selected, the user holds down the right mouse button and drags the mouse to define the path they wish the unit to follow.

For simple paths where a few waypoints are sufficient, sketching may require more time because the user must carefully place the entire path.

Therefor it is better to allow sketched paths to be combined with individual waypoints by using a modifier key. A combination of sketches and individual waypoints would be used if this technique were implemented in an actual real time strategy game. The two methods for specifying movements complement each other nicely, and the mechanisms of invoking each action do not interfere with each other. Sketching is a natural supplement to the existing techniques — where individual waypoints are good for specifying long straight lines, sketches are good for specifying small, complex curves.

As the sketch is drawn, waypoints are automatically added to the selected unit(s) queue of outstanding actions. This means the moment the user starts drawing the sketch, the first waypoint will be added and the unit(s) will begin moving. Earlier implementations waited until the entire sketch was finished, however it was clear this was not the correct, expected behaviour so it was changed to its present state.

Implementation of Sketch Movements

Sketch movements share much underlying code with sketch selections. The same set of timestamped points is present, and the variance of added points with regards to the speed of the sketch is present.

The sketch is drawn in the same way (straight lines between each point), however as waypoints have been added to each unit(s) queue of outstanding commands these are also visible as a series of dots underneath the sketch. Once the mouse has been released and the user is no longer sketching, the series of waypoint dots will remain after the sketch disappears. Figure 3.2 shows this in effect.

3.2 Designing the Experiment Framework

The two novel sketch interaction techniques were implemented in order to explore their effectiveness within the real time strategy game context. These novel features were implemented in a simple real time strategy game engine which was also developed as part of this project. This is discussed in section 3.2.2.

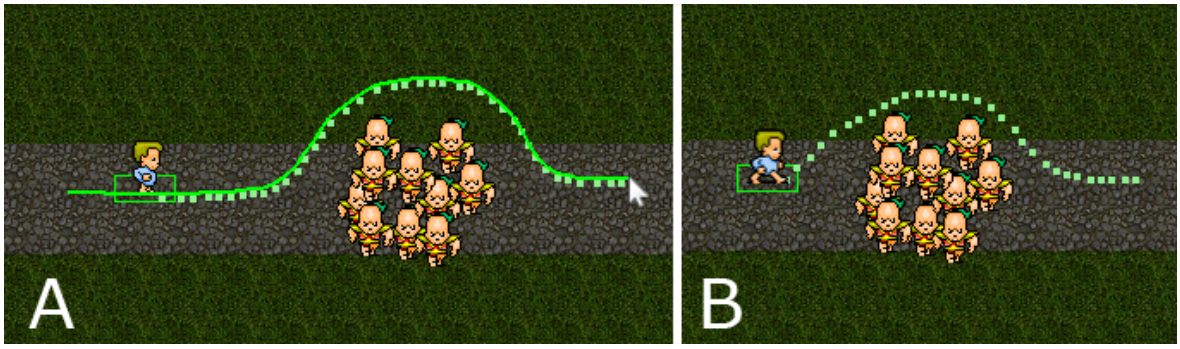


Figure 3.2: Sketch movement in use. (A) shows a sketch currently being drawn. Note the left edge of the sketch where the waypoints have already been consumed by the unit as it has reached them. (B) shows the waypoints generated by the sketch after the sketch has been completed.

3.2.1 Alternative Approaches

Before deciding to develop a simple real time strategy engine of my own for this project, several other ideas were investigated for implementing the sketching techniques. They fell into three categories: use an existing game (open-source, or with source code available), use a middleware layer to interpret and pass sketch commands to an underlying game, or develop a minimal experiment framework. All options had advantages and disadvantages.

At the beginning of this project the scope included exploring the use of mouse gestures for specifying commands in addition to sketching. Support for this was an important criterion in choosing an appropriate implementation vehicle.

Using an Existing Game

Modifying the source of an existing game to include sketch interaction was an obvious choice. This would allow the sketch techniques to be implemented with minimal development time, and would allow the techniques to be tested in an actual existing game.

FreeCiv[1] was the first game to be investigated for this purpose: it is a 2D strategy game that is open-source. The source code is developed and maintained by volunteers. There were several disadvantages to using FreeCiv however which ultimately wrote it off as a viable option.

First, FreeCiv is a turn-based game, so while many of the actions being explored by this project were present (selecting entities, moving units) the game does not have the same time-critical atmosphere that real time strategy games do. Also there is less emphasis on micromanaging entities and resources, and less multitasking.

Second, its source code was very large and messy. It took a long time to figure out how the game was structured and where core functionality was implemented. It was very clear the game had been built by many individuals, without a strong cohesive plan to guide their development. This represented a significant obstacle to adding the sketch based interaction — the significant advantage of using an existing game was mitigated by the expected additional time required to use the existing codebase. As the game is written in an unfamiliar programming language development would have been difficult. There were no suitable libraries to utilise for aiding the implementation of mouse gestures.

[1] FreeCiv: `http://www.freeciv.net/`

This was one of the main reasons for deciding to develop an engine specifically for this project. Even other existing real time strategy games with much more maintained code, such as the Spring Project[2], contained a lot of features and code that was irrelevant for this project. The additional time required to understand the complexity of a fully completed, commercially used real time strategy engine (as well as working in unfamiliar programming languages) made the advantages of using an existing game less significant.

Using a Middleware Layer

A middleware layer was the next solution for implementing the sketch techniques. The middleware layer would sit on top of a real time strategy game, and intercept mouse events before they were received by the game. These events would be interpreted as either normal mouse clicks, or a sketch interaction. They would then be passed on to the game in a way that the game can understand and use.

The primary advantage of this option was that once implemented, the layer could be used on a number of different games, so user testing could be conducted on the sketch techniques in several commercial games which participants might be familiar with. This would help ensure that the validity of results was not influenced by one particular game being used.

The layer would have easily allowed the scope of the project to increase if sketching were successful, so it might be investigated in additional contexts to real time strategy games.

Implementing the layer however, represented a difficult task. There were several technical constraints that needed to be worked around for it to be possible. Games running in fullscreen constantly grab focus each time they redraw, so making the layer stay always on top of the game to receive mouse events appeared to be a non-trivial task. Writing interpretations for each underlying game was also a difficult task, as there is no uniform interaction between the games.

Another issue is the lack of control over the experimental environment. The tasks that participants of a user study would have to complete would differ across the various underlying games. Even if custom maps were made for each game to be as similar as possible, differences in the engine, graphics, types of units and other elements would limit the degree of control available when designing experiments.

Using a Minimal Experiment Framework

A third approach would involve developing a basic, minimal experiment framework that would support only very simple features. The framework would not have any functionality as an actual real time strategy game but would instead provide artificial mock-ups to attempt to simulate a real time strategy game.

The primary advantage is the small amount of development required compared to a full real time strategy game engine. As the framework would not need any core functionality of a actual engine (such as game world timing, collisions etc) only the bare minimum crucial features would be implemented, for example units and selections.

The first disadvantage with this idea is the lack of depth/realism in the experiment. This might have had negative affects on the behaviour of participants during experimentation.

Secondly, when deciding on an implementation vehicle, the project requirements included implementing mouse gestures for issuing a range of commands. This would have required a more rounded set of features, making this option less ideal compared to the other solutions.

[2]The Spring Project: `http://springrts.com/`

3.2.2 Custom Made RTS Engine

The last solution was to use a custom built real time strategy game engine. This has significant advantages over the alternatives previously discussed, and was ultimately the chosen solution.

Advantages

A custom solution allows the sketch interaction to be included as an inherent feature of the system instead of an afterthought. The codebase is well known, written in a language of my choice, and uses libraries and development tools of choice.

The significant advantage of the custom solution is the benefits for the user experiments. Timing and other measurements are built into the core of the engine, ensuring the recorded metrics are as valid as possible. For example the starting and stopping of timers happens at the correct places in the code, and is not affected by other running code (such as a timer being started and recording large blocks of code execution before the participant even starts to react).

As the system is built with user experiments in mind, there are some key differences to a traditional real time strategy game. In a normal game, a map is loaded once at the start of a game and is used continuously throughout the rest of the session. However, with the type of user experiments planned for this project, scenarios would be loaded, run for a short period of time (a few seconds while a task is completed) and then a new scenario would be loaded, in a quick fire manner.

Limitations

The primary limitation of developing a custom real time strategy game engine is the significant amount of time and work required. This was an ambitious goal that was easily the largest project I had developed. It was important to use good planning to stay on track and get the required functionality completed.

This limitation was mitigated by the choice of development environment, frameworks and libraries used (discussed next) which helped reduce the workload and speed up implementation time. Another significant reduction in this limitation was that at the beginning of the year I had already started building a real time strategy game engine, which had some of the basic functionality already in place. At the beginning of the project the engine was loading basic maps, as well as rendering the world, and had rudimentary support for units.

The second limitation was the type of functionality that was possible to implement in such a short amount of time. This was not a deal-breaker however as a fully developed real time strategy game was not the goal, just the crucial components of the engine required to test the sketch interaction. For example, it was important to have units, selections, and movements, but a fully balanced set of different unit types and a functional economy that would be found in most real time strategy games was not required.

One crucial decision was to keep the engine 2D, as this vastly simplified the task. Using 2D graphics was not seen to be a significant issue as most 3D engines present the world from an overhead view (sometimes known as "2.5D") and the sketching techniques would function very similar in a 3D real time strategy environment. In real time strategy games 3D graphics are primarily aesthetic in role, as opposed to a first person shooter, for example, where 3D graphics affect the game-play in a huge way.

In retrospect, given the restriction of the scope of this project to focus solely on sketching, many of the features available in the custom real time strategy engine were redundant.

It may have been possible to achieve the same results by using the minimal experiment framework previously described, with less development time.

Development Environment and Tools

The XNA Game Studio framework[3] was chosen to use for development. The XNA framework is a game development framework aimed at hobbyist and independent game developers. It takes care of much of the low level development (such as game timing, rendering and content management) which was ideal for this project. There is a lot of support available online, as well as many similar projects (in the real time strategy genre) to draw from. This, combined with some prior experience I had with the framework, made it a clear choice for speeding up the development time.

Choosing to use the XNA framework implied several other choices. XNA uses C# as a programming language, and is a part of the .NET family. The project was therefore developed in Visual Studio 2008 and utilised many .NET libraries available.

These tools made it much easier to implement the basic functionality of the engine, which meant there was more time available to focus on the novel sketch interaction.

3.2.3 Map Editor

A map editor was created to assist in the creation of all of the scenarios. With about 200 unique scenarios this was a critical tool for constructing the experiments. Figure 3.3 shows a screenshot of the editor in use.

The editor supported adding individual tiles of various sorts and complex pre-arranged sets of tiles (for adding parts of curves). There was significant support for placing units in particular arrangements also, as a large number of the scenarios were related to unit configurations. The editor works by running an instance of the game, so using the novel sketch techniques was possible. It was here that many of the informal observations and insights into sketching were discovered. The large amount of use the editor got provided me with extensive familiarisation of the sketch techniques, which I have noticed have improved in usefulness the more sketching is practised.

[3]XNA Game Studio framework: `http://msdn.microsoft.com/en-us/library/bb200104(v=XNAGameStudio.31).aspx`

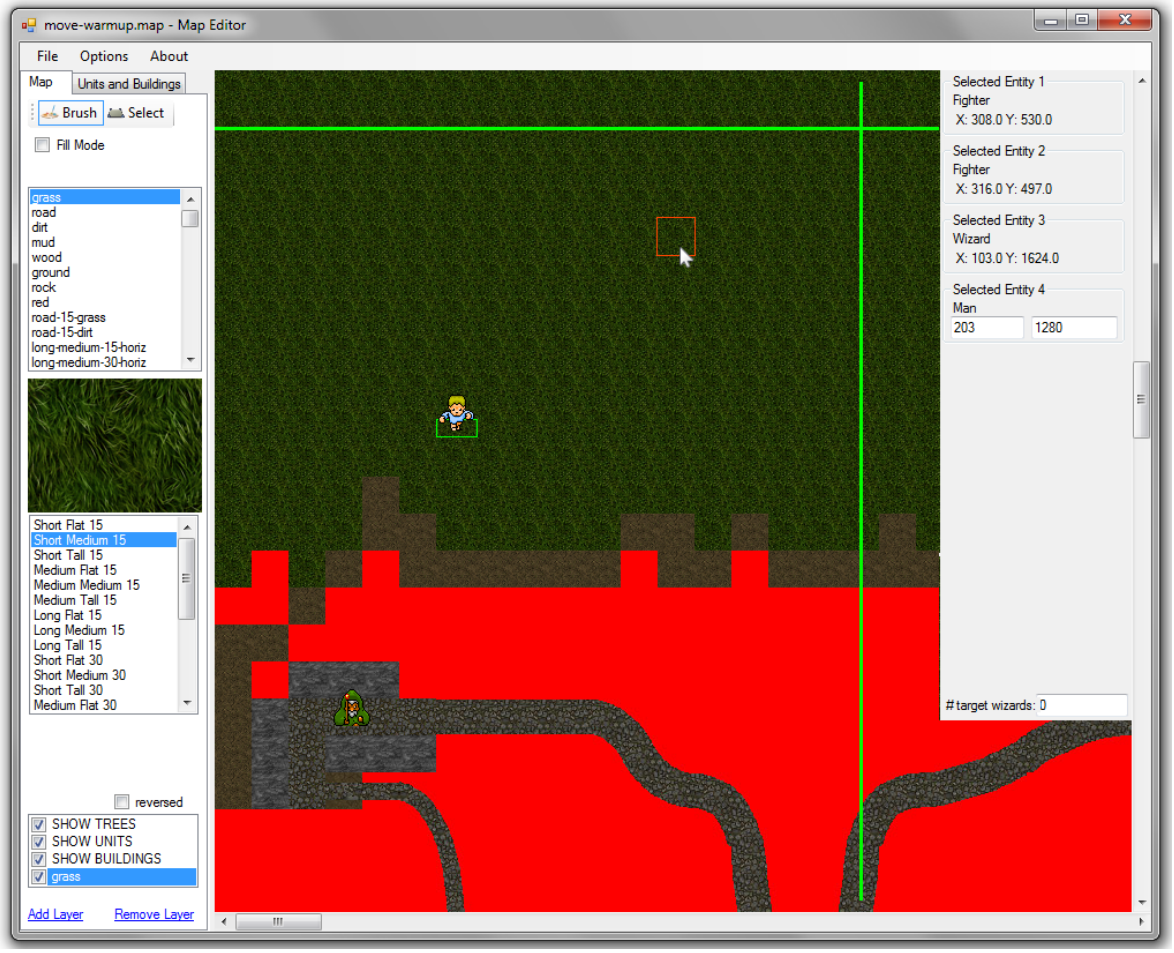


Figure 3.3: The map editor used to create all the experiment scenarios.

Chapter 4

Experiments

In order to evaluate the effectiveness of the novel sketch techniques described in Section 3.1 a user study was undertaken. Approval was granted by the Victoria Human Ethics Committee for the user experimentation aspect of this project.

Following a pilot study, there were two experiments, one for unit selection, and one for specifying paths. Both experiments involved measuring and comparing the subjects' performance at a task using both conventional techniques and sketch based techniques. The common components of both experiments are described first, followed by the specifics of the individual experiments.

4.1 Experimental Design

The evaluation used a within-subjects experimental design, with participants of each experiment split into two sub-groups. Each experiment had two sets of tasks: one set using the novel feature being evaluated, the other set using the conventional interaction technique. Each sub-group of participants completed both sets of tasks, however the order in which the tasks were completed were reversed for one sub-group.

4.1.1 Participants

Participants were recruited through advertising in the Salient magazine, ECS forums, and word of mouth. All of the participants were students at Victoria University, with approximately half being from the Engineering and Computer Science faculty.

None of the participants identified as having colour blindness or any other issues that might impact their ability to participate in the test.

Incentives

Due to the widespread interest in computer games, recruiting participants was not particularly difficult. Therefore, incentives were offered primarily to influence behaviour during the experiment. Participants' results were ranked by their speed and accuracy and the top ten were given a movie voucher. This ranking caused participants to be emotionally invested in their results, and helped simulate the competitive nature of real time strategy games.

4.1.2 Experiment Environment

Participants used a standard QWERTY keyboard and optical laser mouse, on a 20“ monitor running at 1680x1050 resolution. The computer ran the custom testing system described in section 3.2.2.

4.1.3 Procedures

This section summarises the procedures of each experiment. The full script that was followed during the testing is available in Appendix A.1.

Participants began by answering a short questionnaire designed to evaluate their prior experience with real time strategy games and sketching. The questionnaire is in Appendix A.2.

Following this, participants were introduced to the novel feature and shown how it works. The remaining features were then explained, such as the use of modifier keys. Participants were then given time to familiarise themselves with the system (using both the novel and conventional features), and following this asked to complete a simple task to ensure they had grasped the basics.

Once they had completed the warmup/familiarisation task, and had confirmed they were ready to begin, the first task was loaded. The system loads tasks automatically once the previous task has been completed in a quick-fire manner. In between each task, the mouse cursor was reset to the center of the screen. This allowed the movement of the mouse to be measured accurately for each task. Dialogue boxes were shown at certain points during the tasks to give instructions or to offer the participants an opportunity for a rest. Instructions were shown before task 1, rest breaks were shown at 25% and 75% through the experiment, and instructions (and a rest break) were shown at 50% when the interaction technique being used was switched.

Measurement of the various factors, such as timing and accuracy, is built in to the system. As participants complete actions the system logs various data. Exactly what data was recorded is covered in the next sections.

4.2 Experiment 1: Selecting Units

The first experiment aimed to evaluate the effectiveness of sketching to select units, and to ascertain under which conditions sketching is better (if any). Participants were presented with a set of scenarios and asked to select the grey wizards, and avoid the decoy units. Figure 4.1 shows some example scenarios.

In order for the task to be considered complete (and thus load the next task) participants had to select all the wizards and not have any decoys selected. There was no penalty for selecting decoys along the way however, as it is considered a valid strategy to select everything and then deselect just the decoys.

There were 14 participants in this experiment, two female and 12 male. They were split into two sub-groups, which determined the order in which sketch/box selections would be used. The first group used sketch selections initially and box selections last, while the second did the opposite. Individual selections were available to participants at all times. The reason for this was to simulate the conditions sketch/box selections would be used in, in an actual real time strategy game. In retrospect, allowing individual selections was unfortunate, as discussed in Chapter 5.

The system recorded the following data:

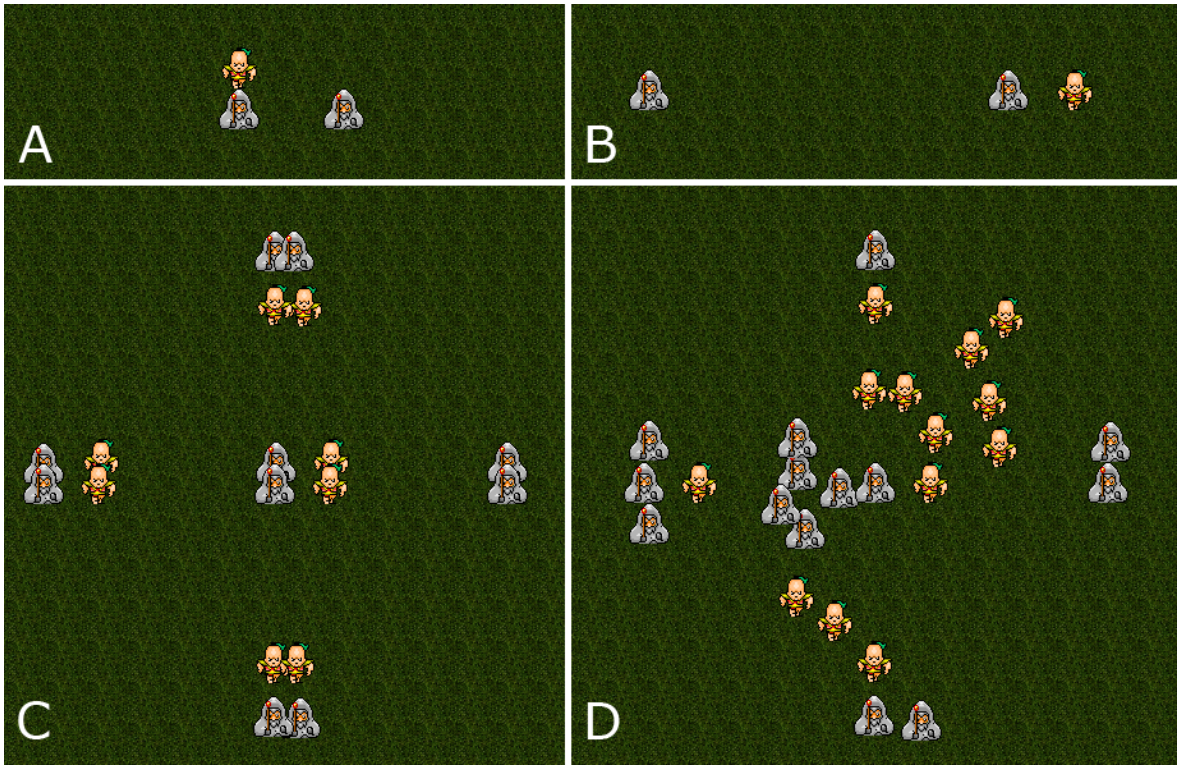


Figure 4.1: Some example scenarios from Experiment 1. Each scenario have a complexity, spread, and margin: (A) 1, 50px, 0px. (B) 1, 250px, 20px. (C) and (D) 5, 150px, 20px. While (C) and (D) have the same factors, (D) has 'filler' units added.

- Task time: total time taken to complete the task (timed from the start of the task up until the moment all wizards are selected)
- Action time: total time spent making selections (sum of the times from the mouse-down start of each selection event, up until the mouse up event that ends the selection)
- Total mouse movement: the distance the mouse moves during the entire task
- Action mouse movement: the distance the mouse moves during each selection

A full log of each action and the state of every entity in the scenario at all times was also recorded for further analysis if required.

4.2.1 Controlled Factors

There were several factors which I believed would have an impact on the efficiency of the selections— complexity, spread and margin.

The complexity of a scenario refers is the minimum number of selection boxes required to select all the target units. Figure 4.2 shows two scenarios, with complexity 1 and complexity 5. The critical determinant of this factor is the placement of decoy units that prevent single box selections.

The spread refers to the distance between each wizard/group of wizards. As sketching selections requires more mouse movement than selection boxes (as the mouse traverses the boundary of the units being selected as opposed to the corner-to-corner movement of a box) the spread will affect the performance of sketching.

Margin refers to the minimum distance found between a decoy unit and one of the target wizards. This factor will affect the accuracy/care required when making selections. Particularly, with box selections, it can be difficult to place the initial corner if it is not close to the point of the small margin.

Table 4.2.1 shows all of the factors and the values used for each.

	Lower Bounds				**U**pper Bounds
Complexity	1	2	3	4	5
Spread	50px	100px	150px	200px	250px
Margin	0px	5px	10px	15px	20px

Table 4.1: The factors varied across the scenarios and the values used.

There were 142 scenarios in total. 117 scenarios varied the factors previously discussed. There were only 117 (rather than 125) as some combinations of factors were impossible, for example a scenario with >1 complexity, 1 spread, and >10px margin. This is because the margin between units would be greater than the spread. The remaining 25 were duplicates of some of the 117, but instead of using the minimum number of units required to form the scenario, 'filler' units were added. These units did not affect the factors at all, but were included to see if participants would do better or worse when units were not arranged in strict grids. Figure 4.1 C and 4.1 D shows an example of this.

4.2.2 Pilot Experiment

A pilot of this experiment was initially run to discover any issues with the experimental design. The pilot was run on seven participants and did find issues, most notably:

- Scenarios needed groups of units to discourage use of individual selections.
  Initially scenarios had single units rather than groups, and (understandably) participants rarely used the box/sketch selections and instead opted to individually select the wizards. All of the scenarios had to be modified to be more suited for using box/sketch selections.

- Participants needed rests.
  The full set of 284 scenarios took about 20 minutes to complete. Initially the only rest participants could take was at the half way point when the selection method switched over. Participants commented that it was too strenuous physically to complete that many tasks in a row. Rests were added at the 25% and 75% points.

4.3 Experiment 2: Movement Paths

The second experiment aimed to evaluate the effectiveness of using sketching to specify movement paths for units, and to ascertain under which conditions sketching is better (if any). Participants were presented with a set of scenarios and asked to guide a unit along a path to a tower on an island at the end of the path. The path spanned an expanse of lava which meant accuracy was required to keep the unit on the path. Once the unit was within 50 pixels of the tower (just stepping onto the island) the task was considered complete. Figure 4.2 shows an example of a scenario from this experiment.

Units did not have pathfinding, so if a waypoint was placed off the path in the lava, the unit would walk to the waypoint in a straight line. The unit could not die, however, but participants were told their accuracy was being measured as well as time so they needed to

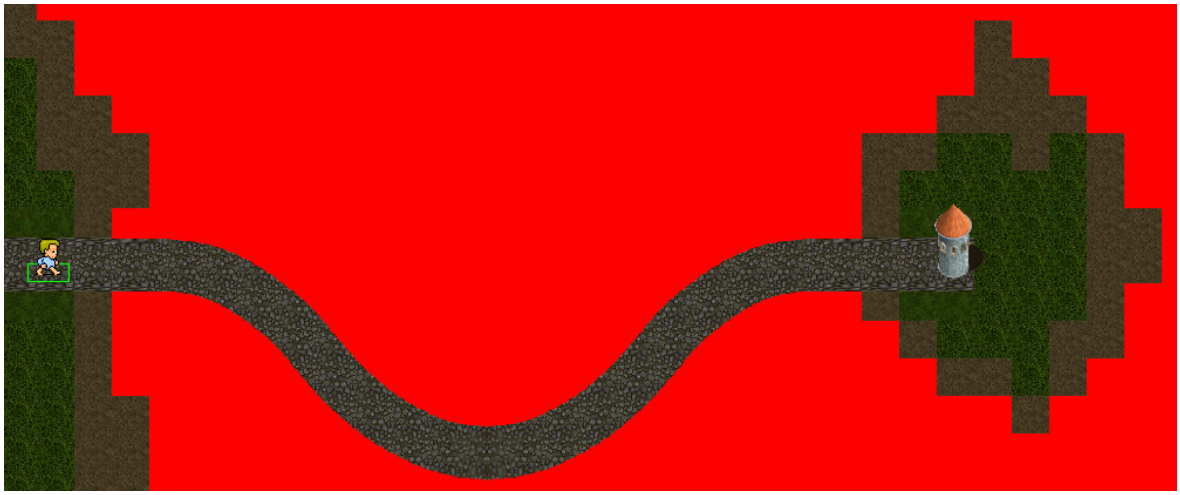


Figure 4.2: An example scenario from Experiment 2.

keep the unit off the lava. Participants gained feedback when the unit was in the lava, as the sprite was shaded red. This provided participants with an opportunity to see and correct mistakes.

There were 8 participants for this experiment (1 female, 7 male), and again they were divided into two sub-groups. The first group used sketching for movement paths in the first half, and individual waypoints in the second half, while the second sub-group again used the reverse order. This time, while using sketching to specify movement paths, individual waypoints were disabled.

There was no pilot study for this experiment, however results from the initial pilot and experience gained while running the selections experiment influenced the design of the movement experiment.

4.3.1 Controlled Factors

The important factors in this experiment are all related to the shape and size of the path being traversed, specifically the type of curve the path has. The factors that affected the difficulty of the task included the thickness and 'curviness' of the path.

The thickness of the path dictates the accuracy and care required when placing movement orders. On very thick paths, fewer waypoints are required and may be placed rapidly. On thinner paths more care (and therefore time) is required to ensure the waypoints are on the path. Also, thinner paths require more waypoints to ensure the unit stays within the bounds of the path.

The 'curviness' of the path is determined by the horizontal and vertical spread of the curve. For example, a curve may be 192 pixels wide from end to end, and 384 pixels high. Figure 4.3 shows some example curves. The curviness affects the number of waypoints required to successfully navigate the unit along the path while staying within the bounds of the path. A highly curved path will require many points to keep the unit on the path, while a flatter curve will require less. Highly curved paths are susceptible to error as it can be difficult to visualise when a path between two waypoints strays off the path, 'clipping' corners.

Table 4.3.1 shows the three factors that were varied across the scenarios and the values used. There were 24 scenarios in total.

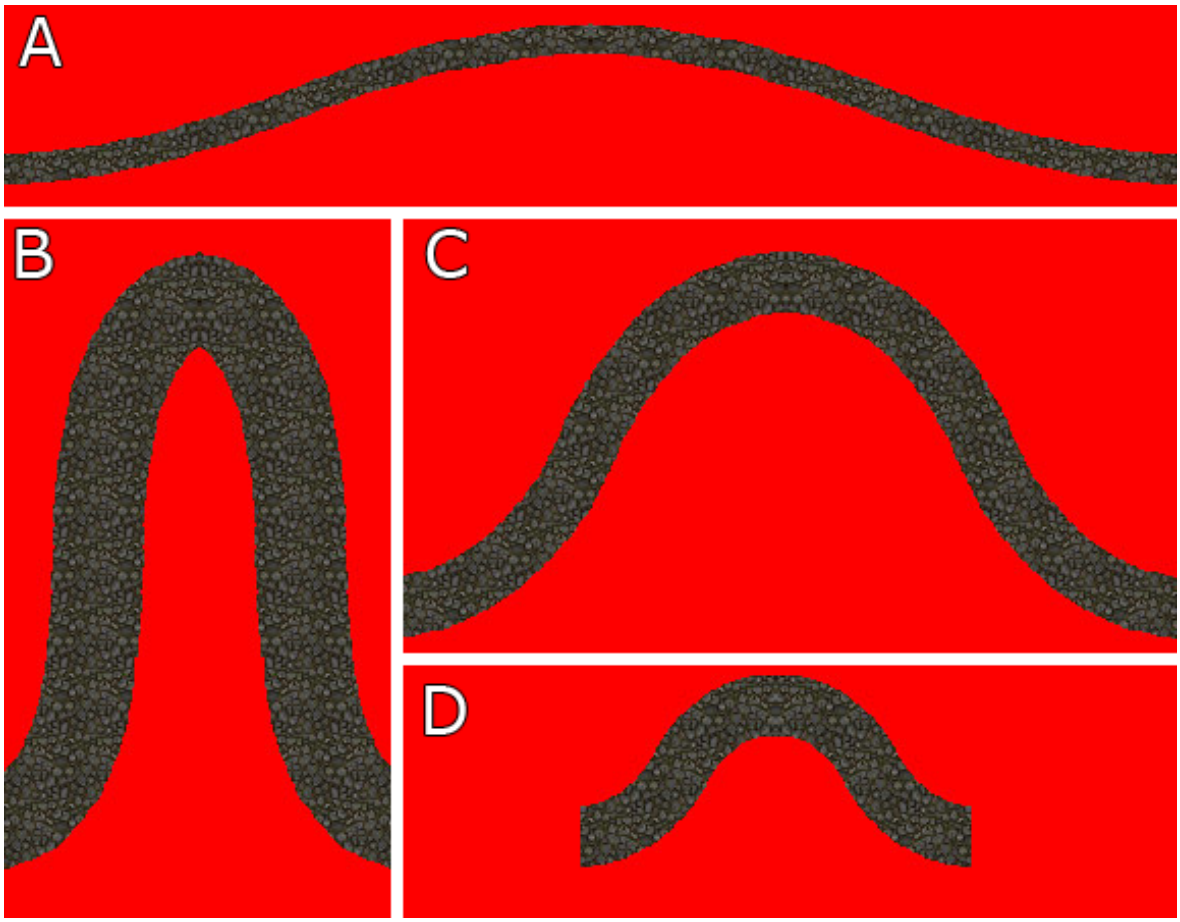


Figure 4.3: Some of the different curves used in Experiment 2. The scenarios have varying thickness, horizontal span and vertical span: (A) 15px, 576px, 96px. (B) 45px, 192px, 288px. (C) 30px, 384px, 192px. (D) 30px, 192px, 96px.

	Lower Bounds		**U**pper Bounds
Thickness of Path	15px	30px	45px
Horizontal Spread	192px	384px	576px
Vertical Spread	96px	192px	288px

Table 4.2: The factors varied across the scenarios and the values used.

Chapter 5

Results and Discussion

The experiment framework collected a very large set of data about the participants' actions. this section presents the results of the most significant parts of this data. This data yielded mixed results, and did not follow all the expected patterns.

All of the results presented in this section, unless otherwise specified, are determined in the following manner. For each participant, the difference between the costs of conventional and novel interaction technique for each task is calculated (conventional cost - novel cost). The costs included time taken and mouse movement distances for either the whole task or the action component of the task. The difference in the quality of the paths (for the movement experiment) were also calculated, using the times the unit spent in the lava. The statistics presented like average and standard deviation are based on these differences.

5.1 Experiment 1: Selecting Units

The key result in this experiment is the total time taken to complete a task, however the action time and mouse movements provide interesting data also. Table 5.1 shows the overall results for this experiment, where a negative number means the novel interaction technique had a higher value (performed worse).

	Total Time	**Action Time**	**Total Mouse Move**	**Action Mouse Move**
Average	-0.46	-0.57	-391.22	-564.06
Standard Deviation	2.30	1.23	975.41	739.58
Median	-0.31	-0.31	-296.33	-366.55
25% Quartile	-1.14	-0.89	-722.72	-743.42
75% Quartile	0.30	0.03	43.35	-170.81

Table 5.1: The overall results, based on each participants (conventional result - sketch result) for each task. Time values are in seconds while mouse movement is in pixels. Negative numbers indicate sketching performed worse than box selections.

The total time taken to complete the task was overall 0.46 seconds slower on average. While this is only a very small value, sketch selection was consistently slower across all types of scenarios. The same analysis was performed on subsets of the data for different categories of scenarios. The results were essentially the same in all cases, and therefore are not reported here. This was an unexpected result, as I believed sketching would have been faster for particular sets of factors.

One possible explanation for this is the types of sketches being performed. Rather than completing a few large complex sketches, participants treated sketching as they would

boxes, that is, circling small individual groups of units, tokenising the group into small sections that could have been selected by a box selection. This may be due to the lack of experience with sketching and the inability to break the habit of chaining multiple small selections together. By treating the sketch as if it were a box selection, the primary advantage of sketching (the ability to select units in complex arrangements with a single selection) is unused. Any future work should explore this idea, perhaps having a control group that is penalised for making many selections/encouraged to make few selections.

Some users attempted to execute complex selections but got confused about which parts of the sketch were 'inside' and which parts were 'outside'. This is an easy mistake to make, especially when using the advanced features of sketch selections, such as double looping units to not select them. A possible solution for this to be implemented in future versions of the sketch technique is to lightly shade the inside of the sketch so it is very clear about which units are and are not going to be selected. Figure 5.1 shows a mockup of this solution in use.

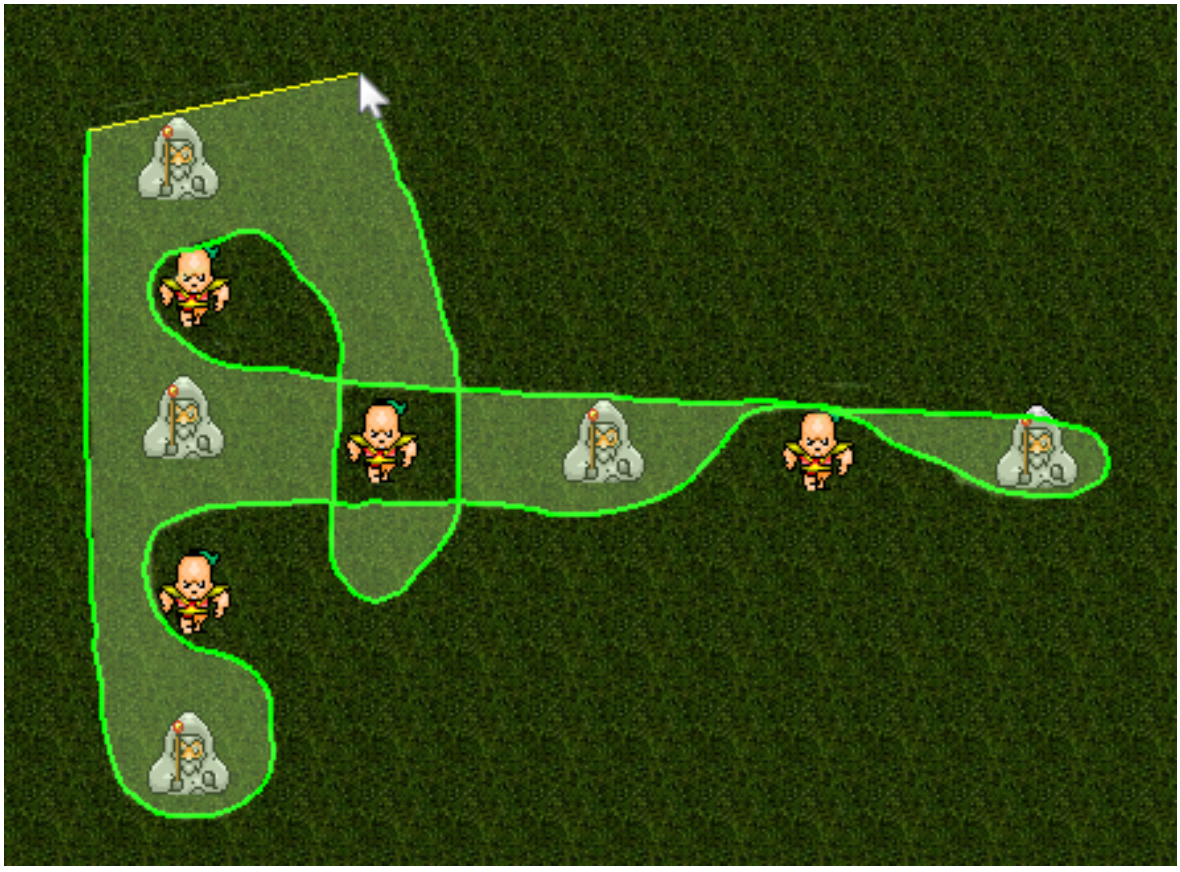


Figure 5.1: Mockup of an improved version of the sketch selection. The 'inside' of the sketch is shaded to clearly show which regions are currently selecting. Note the decoy unit in the middle which has been circled twice— this removes that section of the region from the 'inside'.

The action time is the time spent issuing selections (measured from the start of the first selection command through until the end of the last selection command). On average the action time was slower overall for sketching, by 0.57 seconds. This is expected due to the additional mouse movement required when executing the sketches.

The total and action mouse movements refer to the distance moved by the mouse in

pixels during the entire task, and while specifying actions respectively. They were both greater for sketching for the same reason as the action time, as was expected.

It is clear from this data that there is no significant time advantage to using the novel sketch selection technique for "novice" users who are experienced at box selections and have had little practice at sketching. From this, we would not expect an immediate take up of sketching by most users if it were provided in real time strategy games.

None the less, and surprisingly given the timing results, qualitative feedback from the participants was positive, with the majority indicating they preferred to use sketch selections over boxes. Several participants commented that sketching was "more fun", while others made mention of the ability to recover from errors when sketching.

With this positive qualitative feedback in mind I believe further quantitative study is required. Even though the timing results were consistently worse the differences were very low (fractions of a second). Furthermore, there are some considerations that suggest that the results do not represent an entirely valid evaluation.

5.1.1 Experiment Limitations

Before starting the measured tasks, participants were given an opportunity to practice using the system and familiarise themselves with the sketching (and box selections if they wished). This was an important aspect of the experiment as sketching has a nontrivial learning curve. Ideally participants would have spent a large amount of time using the novel feature to ensure they were familiar with it. However most participants only spent about 5 minutes in total on the warmup map with some participants spending as little as 3 minutes (the warmup map also includes some time spent being shown how the novel features function, so the actual amount of time spent practising is less than this).

Also during the warmup participants did not have clear objectives to complete, so many were not practising sketching in the time-critical context the measured scenarios were in. This caused many of them to be caught off guard once the actual scenarios began.

This lack of quality practice will have skewed the results in favour of box selections, as all the participants were familiar with this conventional selection method. Over half the participants commented in the post-experiment survey that it took a while to get used to sketching, or that they would liked to have spent more time becoming familiar with the novel feature. Providing this practice might have resulted in better performance for the sketching technique.

One decision when designing the experiment was to allow the use of individual selections in addition to the box/sketch selections. The justification for this was that the two types of selections, individual and box or sketch, would normally be used together in an actual game, and allowing participants access to both it would provide additional realism and depth to the testing experience. In retrospect that was a mistake— it introduced an uncontrolled factor into the experiment and by allowing this additional selection type the results become unclear. Some participants used a lot of individual selections, and barely used box/sketch, while others only used individual selection occasionally, usually to fix a mistake they had made with box or sketch selections. Experiment 2 was designed after running this first experiment, and provides a concise comparison between the two interaction techniques.

The existence of this uncontrolled factors casts doubt on the validity of the results— while it provides some feedback on how the interaction techniques might be used in a actual real time strategy game, it does not provide the direct comparison between box and sketch selections I was seeking. It is unclear how the results are affected by the factors in the scenario and the type of selection used, and how they are affected instead by the participants

use of individual or box/sketch selections.

Future studies should require the participants to practice for much longer, completing tasks similar to the ones that will be found in the measured scenarios. Also, the use of individual selections should be disabled to provide quantitative comparisons of the novel and conventional selection techniques.

5.2 Experiment 2: Movement Paths

The key results for the movement experiment are given in table 5.2.

	Total Time	Action Time	Planning Time	Time Off Path
Average	0.06	1.10	-0.12	0.11
Standard Deviation	1.24	2.17	0.59	0.63
Median	-0.05	1.08	-0.05	0.00
25% Quartile	-0.45	-0.21	-0.23	0.00
75% Quartile	0.35	2.13	0.17	0.19

Table 5.2: The overall results, based on each participants (individual waypoint result – sketch result) for each task. Values are in seconds. Negative numbers indicate sketching performed worse than individual waypoints.

For the movements total time is less relevant than for selections, as this is measuring the time taken for the unit to reach the goal. Understandably, this did not vary much, as the speed of the unit is fixed, which limits the possible variances in time. Overall, the unit got there 0.06 seconds faster on average on a sketched path, which is an insignificant increase, especially considering the standard deviation was 1.24 seconds.

However the action time is a key result. Action time is the time spent issuing movement commands, which is usually finished well before the unit reaches the goal. Overall, the action time was 1.10 seconds faster with sketching on average, with a standard deviation of 2.17 seconds. All but one of the eight participants showed an average improvement when using sketch, as shown in figure 5.2. Participants in group 2 (who used individual waypoints for the first set of tasks and sketching for the second set) had over twice the improvement, with an average difference of 1.52 seconds instead of 0.68 seconds. This suggests additional time should have been spent during the warmup round familiarising the participants with the system, and the types of tasks participants should expect. By the time these participants were up to the scenarios using sketching, they may have formed strategies for optimal selections of units for the types of arrangements found in the experiment.

Since the action time varied for paths of different horizontal and vertical spread, I also calculated the relative improvement for sketching: (individual waypoint movement time – sketch interaction time)/ individual waypoint movement time. The average relative improvement was 10

The time for each task consists of three parts- a planning time, (before any movement commands are issued), the action time(actually specifying the paths), and the remainder time (time taken for the unit to follow the path after the path has been specified). I wondered whether there would be a difference in the amount of planning time required for the sketching vs individual waypoints. in fact there was only 0.12 seconds difference on average which doesn't seem significant.

The time spent off the path is the other crucial result from this experiment. This factor is a measure of how accurate the paths/set of waypoints draw by the participant were. Overall there was a minor increase in accuracy when using sketching, 0.11 seconds with a standard

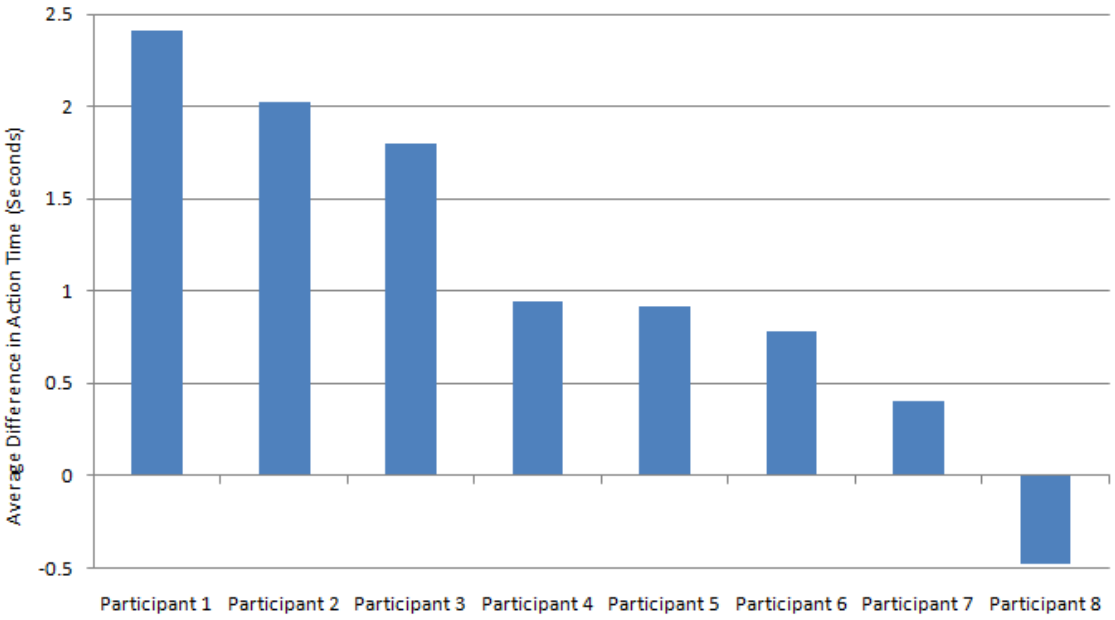


Figure 5.2: The average improvement in action time (seconds) when using sketching rather than individual waypoints.

deviation of 0.63 seconds. While the improvements are not significant, the fact that there was a small improvement on each subset of scenarios, which suggests sketching may be more accurate for specifying movement paths. Also, the qualitative feedback from participants was overall positive, with half the participants commenting on the post-experiment survey that sketching felt easier. Several mentioned sketching felt natural for specifying paths.

With the minor improvements shown in the results and the positive qualitative feedback from participants, using freehand sketching as an interaction technique for specifying movement paths certainly shows promise.

As the number of participants for this experiment is low, these results are indicative only, however they certainly suggest there is further research to be done. A larger experiment needs to be run to supply quantitative results.

Chapter 6

Conclusions

For the many spatial and geographic commands found in real time strategy games, using freehand sketching as an interaction technique provides a natural means of interaction.

This project has investigated the use of sketching for two actions: selecting groups of units, and specifying movement paths. A basic real time strategy game engine was developed to provide a vehicle for implementing the two novel interaction techniques. This was developed with the goal of providing an automated experiment system to run user experiments to evaluate the novel techniques.

Two user experiments were run following a pilot study on a total of 28 participants. The results did not indicate a definite advantage, specifically for the unit selections where issues with the experimental design have likely affected the results that were collected. However the second experiment for evaluating sketching movement paths provided some indication it may have benefits. Further studies are required on larger populations to confirm this.

The qualitative feedback received from participants using the system was nearly all positive, and this, combined with the comparable results of sketching to the conventional interaction technique suggest there may be an advantage in the sketch interaction after all, if not in efficiency, at least in the usability of the technique.

I believe with increased use and practice the sketching will be both more usable and more efficient than the conventional interaction techniques, particularly if sketching was used in multiple facets of the game rather than only two actions.

Further, as all the major gaming consoles have recently released their own forms of gestural, motion sensing devices for advanced interaction in games, and gestural, multimodal interfaces are becoming more prevalent in computers, especially with the rise of touch-screen devices, modern computer and video games are pushing the bounds of conventional interaction.

This project has represented only an exploratory investigation of the viability and effectiveness of sketching as an interaction technique.

Bibliography

[1] APTE, A., AND KIMURA, T. D. A comparison study of the pen and the mouse in editing graphic diagrams. In *Proceedings of 1993 IEEE Symposium on Visual Languages* (1993), pp. 352 – 357.

[2] ASSOCIATION, E. S. Esa report on the sales, demographics and usage data of the industry. Accessed May 2010 from `http://www.theesa.com/facts/pdfs/ESA_EF_2009.pdf`, 2009.

[3] CALLAHAN, J., HOPKINS, D., WEISER, M., AND SHNEIDERMAN, B. An empirical comparison of pie vs. linear menus. In *CHI '88* (1988), pp. 95 – 100.

[4] DULBERG, M. S., ST. AMANT, R., AND ZETTLEMOYER, L. S. An imprecise mouse gesture for the fast activation of controls. In *INTERACT '99* (1999), pp. 375 – 382.

[5] DUNHAM, G., AND FORBUS, K. nuwar: A prototype sketch-based strategy game. In *First Artificial Intelligence and Interactive Digital Entertainment Conference* (2005).

[6] FITTS, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47*, 6 (1954), 381 – 391.

[7] FORBUS, K., USHER, J., AND CHAPMAN, V. Sketching for military course of action diagrams. In *IUI '03* (2003).

[8] HOPKINS, D. The design and implementation of pie menus. *Dr. Dobb's Journal 16*, 12 (December 1991).

[9] KURTENBACH, G., AND BUXTON, W. User learning and performance with marking menus. In *CHI '94* (1994), pp. 258 – 264.

[10] OVIATT, S., ET AL. Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions. *Human-Computer Interaction 15*, 4 (December 2000), 263 – 322.

[11] RUBINE, D. Specifying gestures by example. In *18th annual conference on computer graphics and interactive techniques* (1991), ACM, pp. 329 – 337.

[12] SAFFER, D. *Designing Gestural Interfaces*, 1st ed. O'Reilly, 2008.

[13] SUTHERLAND, I. Sketchpad: A man-machine graphical communications systems. In *Spring Joint Computer Conference* (1963), Spartan Books, pp. 329 – 346.

Appendix A

Appendix

A.1 Procedures Script Followed During Testing

A.1.1 Experiment 1: Selecting Units

- Read and sign HEC forms
- Fill out pre-experiment survey
- While they are doing that, launch system, enter name, assign a group
- When they are ready, launch warmup map.
- Explain purpose of project
  Exploring the use of freehand sketching to select units
- Show them how it works
  Select some stuff, explaining how it is done (click and drag)
  Explain about auto closing line
  Modifier keys (Control for toggle, Shift for add)
  Unit collision (units considered "in" the sketch/box based on their center point)
  Switch to box selection and briefly demonstrate that
- Allow participant to have a go at sketch selections (No time limit, wait until they are ready)
- Switch to box selection and allow them to practice that
- When they are ready to continue, scroll map across to reveal set of units
  Ask them to select all of the grey wizards in one single sketch movement without selecting any of the other units
- Explain the format of the test
  Tasks will be similar to the selection they just made (although they do not have to do just one single sketch movement)
  Once they have selected all the wizards, (and no decoys), next task will auto load
  Mouse cursor resets to center of screen in between each task
  It is an efficiency test, so speed matters
  How to win voucher (fast speed)
  Dialogue boxes will inform when rest breaks are available, and when switching over to alternate selection technique

- Prompt if ready, and start the test
- After they have completed, fill out post-experiment survey

A.1.2 Experiment 2: Movement Paths

- Read and sign HEC forms
- Fill out pre-experiment survey
- While they are doing that, launch system, enter name, assign a group
- When they are ready, launch warmup map.
- Explain purpose of project
  Exploring the use of freehand sketching to move units
- Show them how it works
  Move unit around, explaining how it is done (right click and drag)
  Explain about individual selections (available during warmup but once task starts it will be either individual or sketch, not both)
  Modifier key (Shift to chain multiple commands)
  Accuracy- staying on the path (unit turns red when off the path)
- Allow participant to have a go at moving units (No time limit, wait until they are ready)
- When they are ready to continue, prompt them to explore lower map to reveal set of paths and lava
  Ask them to practice traversing the paths
- Explain the format of the test
  Tasks will be similar to the paths they are practising on
  Once unit is within 50px (approx 1cm) of tower, next task will auto load
  Mouse cursor resets to center of screen in between each task
  It is an efficiency test, so speed matters, however accuracy is also very important- staying on the path
  How to win voucher (fast speeds, high accuracy)
  Dialogue boxes will inform when rest breaks are available, and when switching over to alternate movement technique
- Prompt if ready, and start the test
- After they have completed, fill out post-experiment survey

A.2 Questionnaires Used in User Studies

These are the questionnaires participants completed before and after completing the user study.

A.2.1 Experiment 1: Selecting Units

Before

- Do you have any physical ailments that may affect your abilities during this experiment? If so, please list them here. *For example, colour blindness, arthritis, etc.*
- How much experience have you had playing RTS games? *RTS: Real time strategy. For example, Starcraft, Warcraft III, Age of Empires, Supreme Commander.* (1-5 scale)
- Please list which RTS games you have played, and how much experience you have had with each. Please don't include games you have only played a few times *For example: "Supreme Commander— completed campaign and play online about once per week"*
- How much experience have you had sketching with a mouse? *For example, graphics editing or creation* (1-5 scale)
- Please list which applications you have used that require sketching, and how much experience you have had with each. Please don't include applications you have only used a few times. *For example: "Photoshop— used daily in job, very experienced"*

After

- Which selection technique did you prefer? (Choice of Sketching and Box selections)
- How did you find using sketching to select units? *General comments, suggestions, problems, etc*

A.2.2 Experiment 2: Movement Paths

Before

- Do you have any physical ailments that may affect your abilities during this experiment? If so, please list them here. *For example, colour blindness, arthritis, etc.*
- How much experience have you had playing RTS games? *RTS: Real time strategy. For example, Starcraft, Warcraft III, Age of Empires, Supreme Commander.* (1-5 scale)
- Please list which RTS games you have played, and how much experience you have had with each. Please don't include games you have only played a few times *For example: "Supreme Commander— completed campaign and play online about once per week"*
- How much experience have you had sketching with a mouse? *For example, graphics editing or creation* (1-5 scale)
- Please list which applications you have used that require sketching, and how much experience you have had with each. Please don't include applications you have only used a few times. *For example: "Photoshop— used daily in job, very experienced"*

After

- What did you like about sketching movement paths?
- What did you not like about the sketching?
- Any other comments/feedback

A.3 Additional Results

This section (the remainder of this appendix and report) contains additional results from the data collected during the experiments. The results are in the same form as described in Chapter 5, that is, the difference between each participants' performance for the the novel technique and conventional technique for each task.

A.3.1 Experiment 1: Selecting Units

Table A.1: Statistics from all scenarios with 5 complexity

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.82	-1.08	-542.65	-889.19
Standard Deviation	3.01	1.76	1243.67	988.34
Median	-0.68	-0.73	-459.03	-633.34
25% quartile	-2.00	-1.85	-1043.00	-1216.36
75% quartile	0.61	0.00	107.79	-291.50

Table A.2: Stats from all scenarios with 4 complexity

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.66	-0.81	-523.82	-784.97
Standard Deviation	3.09	1.56	1337.32	952.88
Median	-0.43	-0.55	-345.52	-507.05
25% quartile	-1.64	-1.45	-953.09	-1060.98
75% quartile	0.64	0.02	102.58	-248.40

Table A.3: Stats from all scenarios with 3 complexity

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.45	-0.58	-433.46	-552.69
Standard Deviation	2.14	0.99	807.71	590.13
Median	-0.45	-0.40	-397.86	-384.68
25% quartile	-1.21	-1.09	-807.30	-839.62
75% quartile	0.27	0.00	-51.69	-196.32

Table A.4: Stats from all scenarios with 2 complexity

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.70	-0.77	-534.87	-736.01
Standard Deviation	2.69	1.49	1252.34	942.44
Median	-0.35	-0.38	-362.18	-445.62
25% quartile	-1.58	-1.38	-981.76	-991.73
75% quartile	0.52	0.02	57.24	-217.23

Table A.5: Stats from all scenarios with 1 complexity

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.55	-0.70	-430.97	-626.34
Standard Deviation	2.37	1.28	977.19	772.02
Median	-0.49	-0.47	-351.22	-388.43
25% quartile	-1.36	-1.21	-876.03	-892.85
75% quartile	0.43	0.00	94.44	-176.20

Table A.6: Stats from all scenarios with 20px margins

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.48	-0.55	-426.88	-588.77
Standard Deviation	1.45	0.92	855.22	734.07
Median	-0.38	-0.36	-374.08	-400.98
25% quartile	-1.08	-0.85	-713.56	-735.02
75% quartile	0.10	-0.04	-27.29	-191.79

Table A.7: Stats from all scenarios with 15px margins

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.73	-0.66	-475.64	-586.28
Standard Deviation	1.97	1.25	1011.37	790.53
Median	-0.39	-0.36	-362.03	-379.34
25% quartile	-1.19	-0.96	-778.28	-769.84
75% quartile	0.09	0.00	10.81	-172.48

Table A.8: Stats from all scenarios with 10px margins

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.66	-0.66	-406.84	-574.53
Standard Deviation	2.83	1.51	1135.21	802.35
Median	-0.29	-0.28	-273.76	-374.90
25% quartile	-1.25	-0.86	-729.02	-729.19
75% quartile	0.33	0.02	58.19	-159.37

Table A.9: Stats from all scenarios with 5px margins

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.35	-0.50	-366.55	-539.53
Standard Deviation	2.29	1.26	924.23	708.82
Median	-0.20	-0.23	-243.62	-332.32
25% quartile	-1.05	-0.86	-652.06	-683.08
75% quartile	0.50	0.12	73.48	-170.85

Table A.10: Stats from all scenarios with 0px margins

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.12	-0.50	-299.01	-538.22
Standard Deviation	2.54	1.12	913.87	663.12
Median	-0.24	-0.32	-275.37	-358.76
25% quartile	-1.15	-0.98	-716.98	-759.44
75% quartile	0.48	0.04	104.20	-159.99

Table A.11: Stats from all scenarios with 250px spread

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.64	-0.56	-517.87	-666.07
Standard Deviation	2.10	1.14	1115.84	882.66
Median	-0.38	-0.32	-404.12	-396.10
25% quartile	-1.30	-0.95	-974.12	-857.62
75% quartile	0.27	0.07	22.07	-163.27

Table A.12: Stats from all scenarios with 200px spread

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.59	-0.58	-440.26	-621.54
Standard Deviation	2.30	1.39	1235.25	947.80
Median	-0.31	-0.25	-326.47	-372.57
25% quartile	-1.13	-0.84	-749.87	-729.74
75% quartile	0.41	0.06	113.61	-152.31

Table A.13: Stats from all scenarios with 150px spread

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.39	-0.65	-380.14	-602.26
Standard Deviation	2.52	1.29	924.90	677.93
Median	-0.37	-0.39	-305.39	-415.99
25% quartile	-1.16	-0.98	-748.28	-836.17
75% quartile	0.23	0.00	-2.64	-209.09

Table A.14: Stats from all scenarios with 100px spread

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.39	-0.53	-322.82	-484.04
Standard Deviation	1.89	1.10	770.45	594.62
Median	-0.24	-0.29	-233.94	-338.62
25% quartile	-1.03	-0.86	-616.78	-644.41
75% quartile	0.35	0.02	63.64	-161.55

Table A.15: Stats from all scenarios with 50px spread

	Task Time	Selection Time	Total Mouse Move	Selection Mouse Move
Average	-0.29	-0.42	-265.37	-337.13
Standard Deviation	2.47	1.13	665.68	385.69
Median	-0.17	-0.19	-237.17	-260.18
25% quartile	-0.93	-0.68	-502.64	-450.59
75% quartile	0.31	0.05	56.32	-111.72

A.3.2 Experiment 2: Movement Paths

Table A.16: Stats from all scenarios with 15px width

	Total Time	Action Time	Planning Time	Time Off Path
Average	0.32	0.84	-0.18	0.18
Standard Deviation	1.47	1.66	0.64	0.74
Median	0.07	0.72	-0.10	0.05
25% quartile	-0.39	-0.22	-0.26	-0.12
75% quartile	0.76	1.78	0.18	0.42

Table A.17: Stats from all scenarios with 30px width

	Total Time	Action Time	Planning Time	Time Off Path
Average	-0.14	1.19	-0.18	-0.01
Standard Deviation	0.87	2.00	0.50	0.55
Median	-0.12	1.39	-0.05	0.00
25% quartile	-0.52	-0.14	-0.34	0.00
75% quartile	0.16	2.06	0.15	0.00

Table A.18: Stats from all scenarios with 45px width

	Total Time	Action Time	Planning Time	Time Off Path
Average	-0.05	1.34	0.04	0.19
Standard Deviation	1.30	2.94	0.62	0.52
Median	-0.02	1.17	0.00	0.00
25% quartile	-0.32	-0.22	-0.15	0.00
75% quartile	0.43	3.50	0.18	0.03

Table A.19: Statistics from all scenarios with 576px horizontal spread

	Total Time	Action Time	Planning Time	Time Off Path
Average	0.33	1.70	-0.05	0.26
Standard Deviation	1.26	2.01	0.64	0.77
Median	0.08	1.56	0.00	0.00
25% quartile	-0.30	0.28	-0.23	0.00
75% quartile	0.69	2.85	0.24	0.55

Table A.20: Statistics from all scenarios with 384px horizontal spread

	Total Time	Action Time	Planning Time	Time Off Path
Average	-0.11	0.87	-0.14	0.04
Standard Deviation	1.23	2.14	0.53	0.55
Median	-0.13	0.96	-0.05	0.00
25% quartile	-0.50	-0.61	-0.17	0.00
75% quartile	0.32	2.01	0.11	0.10

Table A.21: Statistics from all scenarios with 192px horizontal spread

	Total Time	Action Time	Planning Time	Time Off Path
Average	-0.05	0.75	-0.17	0.03
Standard Deviation	1.21	2.28	0.60	0.52
Median	-0.11	0.77	-0.12	0.00
25% quartile	-0.46	-0.61	-0.27	-0.01
75% quartile	0.26	1.96	0.15	0.12

Table A.22: Statistics from all scenarios with 288px vertical spread

	Total Time	Action Time	Planning Time	Time Off Path
Average	0.19	1.37	-0.15	0.11
Standard Deviation	0.98	1.99	0.48	0.50
Median	0.00	1.16	-0.08	0.00
25% quartile	-0.30	0.05	-0.23	0.00
75% quartile	0.33	2.61	0.15	0.12

Table A.23: Statistics from all scenarios with 192px vertical spread

	Total Time	Action Time	Planning Time	Time Off Path
Average	-0.02	0.97	0.01	0.13
Standard Deviation	1.57	2.61	0.60	0.60
Median	-0.11	0.92	0.00	0.00
25% quartile	-0.49	-0.45	-0.16	-0.01
75% quartile	0.52	2.06	0.19	0.20

Table A.24: Statistics from all scenarios with 96px vertical spread

	Total Time	Action Time	Planning Time	Time Off Path
Average	-0.03	0.88	-0.30	0.08
Standard Deviation	1.01	1.61	0.69	0.83
Median	-0.15	1.08	-0.15	0.00
25% quartile	-0.50	-0.26	-0.72	0.00
75% quartile	0.29	1.68	0.18	0.22