CHALMERS

Games on the Surface

An exploration of the game design space for the Microsoft Surface

Master of Science Thesis in the Programme Interaction Design
Jonas Ekström Mattsson
Mikael Rosenqvist

Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, July 2011

Games on the Surface
An exploration of the game design space for the Microsoft Surface

Jonas Ekström Mattsson
Mikael Rosenqvist

Examiner: Staffan Björk

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

[Cover:
A Microsoft Surface, together with figures from different games to symbolise the merging of the two.]

Department of Computer Science and Engineering
Göteborg, Sweden July 2011

Acknowledgements

We would like to thank our supervisor Staffan Björk and our friends and families for giving great feedback on the prototypes and on the report. We would also like to thank all the game companies releasing games which both got us into this field and did their best in delaying this thesis.

Abstract

The Microsoft Surface is a multi-touch tabletop computer which often is used as a photo or map viewer, or as computer aid while playing board games. Multi-touch games, such as those on smartphones and surfpads do not exist in great amounts on any tabletop yet.

In this thesis we analyse and test which game mechanics suits a machine such as the Surface. Several games are designed, created and analysed with regards to how well they fit the Surface's way of user interaction. All games created can be played by several players at the same time and they also all uses the interactions provided by the Surface.

Table of content

Glossary

Touch
Touch is the ability of a screen to detect a finger, or other object, touching the screen.

Multi Touch
Multi touch is the ability to detect several fingers or objects at the same time.

Surface
A multi-touch tabletop computer from Microsoft.

Windows Vista
An operating system by Microsoft.

LED
Light-Emitting Diode, a type of light source.

Infrared
Light not visible to the human eye.

XNA
Xbox New Architecture is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development and management.

WPF
Windows Presentation Foundation is a computer-software graphical subsystem for rendering user interfaces in Windows-based applications.

MSNBC
An American cable news channel.

Smartphone
A mobile phone with a multi-touch screen as its main input utilising more features than a normal mobile phone, such as downloadable applications.

Android
Android is a software stack, developed by Google, for mobile devices that includes an operating system, middleware and key applications.

iPhone
A smartphone from Apple.

iPad
A surftablet from Apple, featuring a multi-touch screen as its main input

iTunes
Apple's online store for Apple devices, letting users buy and download music, applications and such from a mutual place.

Versus
Two or more players competing against each other.

Cooperative
Two or more players helping each other to reach a goal.

Computer aided game
Refers to games which are at least partially computerized, but which are actively regulated by a human referee.

API
An Application Programming Interface is a particular set of rules and specifications that software programs can follow to communicate with each other

Gamepad
A handheld device humans uses to control a game with. Often contains buttons and sticks.

Pixel
A single point element of a screen.

1 Introduction

Games and play is/are an important activity for many animals. For humans it was not enough to play with simple objects or each other, so we started playing more advanced games, introducing rules. Increasingly advanced tools were played with and in the 1940's, the oscilloscope was invented and in the 1950's engineers started to play a primitive version of pong on them [1]. Since then, video games and computer games has been a popular entertainment. The first home console dedicated to games, such as the Magnavox Odyssey, came in the 1970's [2] and the first generally affordable home computer first saw the light of day in the late 1970's and early 1980's [3]. After a lull period after the video game crash in 1983, the market revived after Nintendo released the NES in 1985 [1]. These machines laid the first stones in the foundation of the multimillion dollar industry that is game development and distribution today.

In recent years, touch-sensitive input devices such as smartphones, Internet tablets and the Nintendo DS have shown that touch-sensitivity is a viable solution and touch or multi touch are techniques which, according to Microsoft [4], might become a very large part of the future. Since games have evolved to fit every other technical innovation, it stands to reason that games will also have to transcend into this arena. Large screens with added multi touch functionality for inputs, and no attached external keyboard or mouse, will feature games. The reason for this is that devices which were not meant to be played on in the first place, have been introduced to gaming anyway [1].

Microsoft Surface, see Figure 1, is a combination of a computer and a touch-sensitive table whose main form of interaction is its screen. The system consists of a normal computer with a not so normal interaction. Currently the market for the Microsoft Surface is quite narrow and there are not many applications for it, and even fewer games. The price of the machine, 11.000€, and its size are probably the most discouraging features.

Figure 1. The Microsoft Surface

1.1 Problem description

A tabletop device has as many reasons to play games on as any smartphone or video game console, but with one major advantage, the tabletop device is more tangible. The same kind of objects can be placed on top of the screen and used as input for the system and be printed in screenspace and used as output to the user. The questions that immediately come to mind are: how could games take advantage of this feature, and what disadvantages it might bring along?

1.2 Stakeholders

Chalmers, other universities, Microsoft, game developers, game researchers and us.

1.3 Question formulation

Which types of game mechanisms are well suited for the Microsoft Surface?

The same game mechanisms would probably fit any other interactive tabletop multi touch device. The question will be answered by studying existing games for the Surface and by designing and implementing our own games.

1.4 Limitations

From the beginning of the project it was decided that 3D will be completely left out, no games or game ideas will have any 3D graphics. The reason for this was that, firstly, because it is not interesting for the project since the objective is to explore interactions

and, secondly, because it would take a lot of time while not delivering much of extra value. Gesture recognition was left out for the same time constraint reason.

The price of one Surface unit makes it unlikely to be bought by private customers and placed in homes, it is more common that companies buy the Surface to place it at a public place such as hotels, restaurants and airports. Players playing at home can spend much more time on a game than most players playing in public places and therefore games demanding excessive amounts of playtime or games with a too steep learning curve will also be left out. While one could consider a memory-card system similar to Japanese arcade machines to support games with higher learning curve, unlockable content and game progress. But since that system is virtually unused outside of Japan, it was decided against.

We also had no desire to port games from another platform to the Surface and try to emulate a mouse and keyboard, even though there exists several interesting ideas on how this could be done. We wanted to explore the Surfaces natural interaction, not emulated ones and thus ports were out while remakes were in.

Furthermore, since Surfaces are placed in public environment with a lot of people passing by, all games must be able to handle more than one player playing at any time, ether cooperative or versus each other, no games for one player only.

2 Background

Microsoft Surface uses multi touch technology to interact with users. Touch has been around for quite some time. The first research into the subject of touch-sensitive surfaces started in the 1960's and in 1965 E.A. Johnson published the first article regarding a touch screen [5]. In the early 1980's, the subject grew to include multi-touch, i.e. the ability to handle several objects touching the same surface at the same time, as well [6]. Lately, several devices that support touch and multi-touch have been released to the consumer market. Entertainment machines such as Nintendo DS, smartphones such as Apple iPhone and surfpads such as Apple iPad. Following this trend, Microsoft has constructed a touch-sensitive table called the Microsoft Surface.

The target audiences for the Microsoft Surface are hotels, retail establishments, restaurants and public entertainment venues [7]. Examples are described more in 2.2 Existing Work. The first iteration of Microsoft Surface was released in 2008. A Surface cost at release about 11.000 €, roughly SEK 100.000, but Microsoft planned to eventually press the price closer to a consumer level. The second iteration of the Surface, called the Samsung SUR40 or Microsoft Surface 2.0, was previewed at CES 2011 on January 6:th 2011 and is scheduled for release late in the year 2011 [8]. According to several previews, the Surface 2.0 will cost roughly 5400 €, or SEK 50 000 [9].

2.1 Surface Technology

Microsoft Surface is a combination of a computer and a touch-sensitive table whose main form of interaction is its 30 inch surface. It is capable of taking input from over 40 fingers and 12 objects simultaneously. The Surface can also recognise a certain type of square, two dimensional bar-codes. These bar codes, also called tags, come in two different variants; byte tags, see Figure 2, and identity tags, see Figure 2. The byte tags are 1.9 centimetres on each side and can have 256 unique values and the Surface reacts to these tags at the same speed as it reacts to fingers. The identity tags are 2.5 centimetres on each side and the Surface reacts to them a bit slower than byte tags, identity tags can effectively have infinite different unique values. To ease application handiness, the tags are often fastened to tokens such as game pieces or other small objects with a flat surface roughly with the same size as the tag.

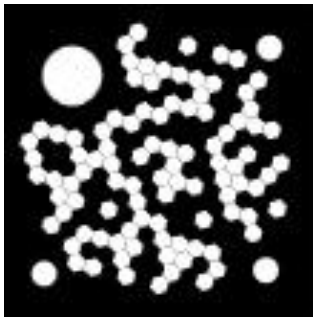

Figure 2. An identity tag

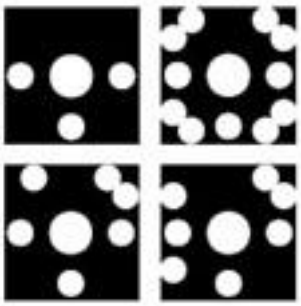

Figure 3. Four different byte tags

The Surface's hardware, which is hidden in a shell below the screen, consists primarily of a normal computer running Windows Vista. A projector is used to project the screen image to the table surface and an infrared LED-light source is directed onto the screen from below. Five IR-cameras are used to detect input and these work in such a way that if a user puts a finger or other object on the screen, the IR-light bounces on the object and the reflected light is detected by the cameras [7].

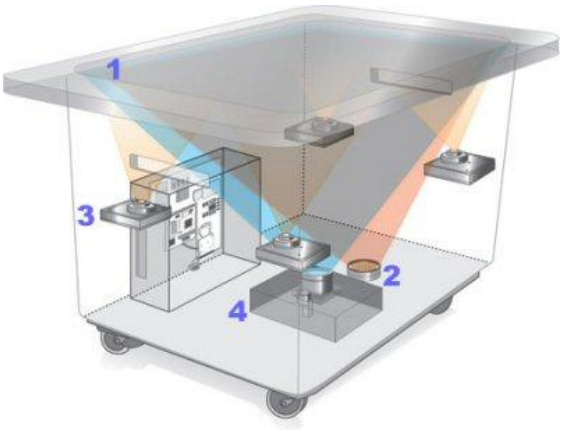

Figure 4. The inner parts of a Surface. 1: The screen. 2: Infrared lamp. 3: One of the infrared cameras. 4: Projector.

The development software on the Surface is utilising the XNA and/or the WPF frameworks. XNA is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development and management. WPF is a graphical subsystem for rendering user interfaces in Windows-based applications.

2.1.1 Surface Simulator

The Microsoft Surface simulator is a tool to simulate the Surface device with all its inputs on a normal computer without touch screen. It requires the computer running it to have a screen connected to it with a resolution higher than 1280 * 960, if this is not the case, the simulator will not run [10]. When launching a Surface application, the application is automatically put inside the Simulator if the simulator is already running. If the simulator is not running when an application is started, the application will still start but will have all of its touch inputs disabled.

The simulator can simulate a large number of fingers, all possible byte tag values and also all possible identity tag values. The user picks what type of input the mouse is to use and can then press the left mouse button on the surface to simulate a touch with the current input. If the right mouse button is pressed while the left mouse button also is pressed the input type will stay pressed on the spot. The scroll wheel works as rotation of the current object and works both when the object is pressed to the surface and when it is not pressed to the surface.

2.2 Existing Work

There exists several different kinds of large multi-touch machines and the Internet has many videos where people are using multi touch applications they have built and developed themselves. A problem is that almost none of them are playing games on the tabletops, most are moving, rotating and resizing images.

There exists plenty of research on the touch screen subject, but multi user interaction on the same screen is very sparse.

2.2.1 Surface Applications

Currently, Microsoft Surface is used in several pilot projects such as, and as diverse as, Sheraton's interactive guide to Seattle, Barclay's account information, MSNBC's application for covering the presidential election, Harrah's interactive bar-table and Churchend Primary School's program to help schoolchildren learn languages [11]. There are a couple of games available for the Surface, ranging from implementation of classic board games such as Settlers of Catan [12], Chess [13] and Go [14], via Surfacescapes [15] to The Jelly Reef [16] and Drift 'N Drive [17].

Figure 5. The Jelly Reef.

The guide to Seattle consists of several independent programs with different purpose. One displays postcards of famous landmarks, another shows a map of the city with cites of interest marked on the map and one lists restaurants, museums, cafes and other similar establishments of tourist interest. In all of the applications the user can change the size, rotate and move the different windows by moving theirs fingers in defined patterns, as well as opening new windows by tapping things [18]. Barclay's application displays the user's account information as well as information about the bank and its services. It allows the user to interact with this information in the same way that the Sheraton applications do [19]. MSNBC's applications follow the same pattern but adds tags to the mix, using them to highlight different information [11]. Harrah's bar-table follow the same pattern of showing and interacting with information and/or media but also allows the user to control an external camera and throws in some games such as bowling and pinball into the mix [20].

Surface Games

The Surface's game department is less well populated than its library of ordinary applications, but there are some dedicated games for the machine. For example, Microsoft has released a bundle of four games, consisting of implementations two classic board games, chess and checkers, a "match-three" game and a "draw-moving-snakes" sandbox [20]. Apart from Microsoft, some of Microsoft's partners, such as Vectorform, have made games for the Surface. Among other things, Vectorform has made a copy of the board game Settlers of Catan [21] and the "tower defence" game Galactic Alliance [14]. There has also been some independent development for the Surface. A group of Carnegie Mellon students are working on implementing a computer version of Dungeons & Dragons 4e, called Surfacescapes [15], and another student group at Utrecht School of the Arts has made The Jelly Reef, a game about leading a group of jellyfish through a maze filled with deadly obstacles.

2.3 Similar systems

Devices with touch as a possible way of interaction have been on the market for several years and there exists a multitude of commercial application, see 2.3.1 Smartphones to 2.3.4 Reactable for examples, that use touch and multi-touch as their primary mode of interaction. These with intended use ranging from solely office work to pure gaming.

Modern smart-phones often make use of touch technology as well and, thus there are many games to these systems utilising touch mechanics. For example, the iPhone has Angry Birds where the player stretch and aim a slingshot to shoot down structures containing pigs [22][23].

2.3.1 Smartphones

The first smartphone, the IBM Simon, was released to the public in 1993. It had no keyboard and the user interacted using a touch-sensitive screen [24]. Development in the field continued in the years that followed and the big systems today are Apple's iPhone and various systems based on Google's Android. The iPhone utilises a multi-touch screen as it main form of interaction [25].

2.3.2 Nintendo DS

The Nintendo DS was released in Europe on 11 of Mars 2005. It has two screens with the bottom one sensitive to touch but not multi-touch. The consoles games utilizes this by finger, the needle-like stylus, a curved plastic tab attached to the optional wrist strap or not at all [26].

For an example of a DS game that uses touch, the DS have Elite Beat Agent were the player presses a stylus on specific spots on the screen to a music beat [27].

2.3.3 Apple iPad

The iPad is basically an enlarged iPhone, utilising multi-touch and featuring a screen that is 9.7 inches diagonally in comparison to the iPhone's 3.5 inches [28]. The iPad can play all games supported by the iPhone but also have some exclusive titles. Many iPad games are remakes of iPhone games with higher resolution, allowing more players easier access to the device.

Figure 6. An iPad showing its home screen

2.3.4 Reactable

The Reactable is, just as the Surface, a multi-touch tabletop, but the Reactable is built as a music instrument instead. Its primary mode of interaction is tags placed on the screen. The positioning and rotation of the tags controls the speed and many other properties of the music [29].

2.3.5 Various Home-made Touch-tables

Many universities have their own home-made touch-tables they use for research into various fields, such as user interaction and multi-touch technology. Examples include the Swiss Federal Institute of Technology [30] and Chalmers University of Technology.

3 Theory

This chapter will handle common ways of giving input to a computer and common movements used in various touch and multi-touch devices. It will discuss games and game design patterns, as well as benefits and drawbacks of touch input.

3.1 Common Pointing Input Devices

There exist a number of ways of controlling a computer with different pointing devices [31]. The most common are with a mouse, a finger or a stylus.

3.1.1 Mouse

The mouse is a small device connected to the computer, controlled by one hand and it usually has two or more buttons. If a user moves the mouse, a pointer on the screen moves in the same direction. The appearance of the pointer depends on its position on the screen or what is currently beneath the pointer. If the user wants to interact with something, he or she positions the pointer over the object and presses a button on the mouse [32].

3.1.2 Finger

Touch input is, for example, when a user uses fingers to control the computer. One way to use fingers is to utilise a touch pointer, which works in the same way as a mouse pointer but the pointer is positioned where the user points with the finger instead. This only corresponds to one button on the mouse though. A more common way of interaction is to use the special control types mentioned later in this chapter [33].

3.1.3 Mouse versus Finger

There have been several studies comparing the interaction of mouse with that of a finger. For example, in Direct-Touch vs. Mouse Input for Tabletop Displays Forlines et al. [34] which concludes that users are faster and more accurate using a mouse for unimanual, or one-handed, interaction while fingers are better for interactions requiring more hands. However, they also note that other things such as fatigue, spatial memory, and awareness of other user's actions in a multi-user setting might affect an interaction designer's decisions.

3.1.4 Stylus

A stylus, also called a pen, can be used to bridge the gap between mouse and finger. A stylus can be used both as mouse pointer and to draw or write. Styluses sometimes have a button on it for right clicking. Mack and Lang concludes that it is no worse than a mouse regarding speed and accuracy [35] MacKenzie et al. concludes that a stylus is at

least as good as a mouse in pointing and dragging tasks, but outperforms the mouse in drawings and gesture tasks [36].

3.2 Control Types

There exists a number of ways to control an application with previously mentioned pointing devices; these will be briefly mentioned in this chapter.

3.2.1 Touch Control Types

Most touch devices and operating systems using touch input has a number of control types which are more or less standardised [33][38]. They can all be used in applications utilising any form of multi touch pointing. Some touch pads for laptop computers also use a number of different touch control types [37] [39].

Tap
Tap is the most common input method on any touch device; it is the input method most similar to clicking with a mouse and often is the one action where the program reacts in the same way as a click would. Tap is when a user touches the screen and then quickly pulls it back. This motion is often used to open programs and selecting characters on a virtual keyboard.

Double tap
This input method works in the same way as a normal tap, but with a new tap motion quickly after the first. This motion is just like tapping similar to a normal mouse motion, double clicking.

Hard press
This input method is when the user presses on the screen and then keeps the finger pressed at the same spot for a certain amount time. This motion sometimes brings up a menu close to the finger and therefore sometimes it is the motion which represents right clicking with a mouse.

Press and Tap
Press and tap is one way to simulate a right clicking with a mouse. While holding one finger on the screen where the user wants to right click, the user performs a tap action with another finger close to the first one.

Swipe
Swipe is the input method of sliding one finger across the screen. Swiping is similar to real life interaction with objects, like throwing or moving them.

Flick
Flicking is when the user slides a finger a short distance in one direction on the screen, and then releases it. While there is no technical limit in the amount of directions, eight is a common number. According to Microsoft, flicking roughly represents keyboard shortcuts such as copy and paste [33].

Pinch
The pinch motion is similar to two swipes either directed towards each other or away from each other. The user touches the screen with two fingers and then while maintaining contact with the screen drags them together or apart. The motion of dragging them apart is often called reverse-pinch. Pinching is often used for zooming.

Rotate
Rotate is done in a similar way as pinching but instead of changing the distance between the fingers, the user modifies the angle between them relative to the edge of the screen.

3.2.2 Game Control Types

A game on a touch device can be controlled in a number of ways; gesture, tilt, analogue sticks and buttons [40].

Gesture
A gesture is a combination of consecutive swipes. A gesture often correspond directly with the physical properties of an object, like cutting or throwing.

Tilt
Tilting uses the built in accelerometer in the device and lets the user modify the whole devise physically and use the motion as input to applications. Tilt is a input method which is uninteresting from the Microsoft Surface point of view. Tilting the table is not possible and this control type will not be researched in this project.

Virtual Directional Pad and Analogue Stick
The Directional pad, also called the analogue stick is a common way to control a game. When using a virtual version of the traditional gamepad's directional pad, a virtual version of it is added to the screen. With this, the player can control movement, directions, rotation or such in the game by moving and holding a finger over it. The directional pad can both be visible or invisible, it can sometimes be movable by the player and sometimes it is set to always have its origo where the finger is positioned at first.

Buttons
Buttons are objects on the screen a user in some way can interact with, by using the different input methods mentioned earlier. A button can have any form or size and be

positioned anywhere on the screen. A character on the screen might function as a button as well as representing the player.

3.3 Touch as Input

Today, Touch is a popular feature in many devices, such as phones and computers, in this chapter, benefits and limitations of touch will be mentioned. The focus will mostly be on tabletop touching, more Surface specific properties will be mentioned more in chapter 7 Discussion. Several people have explored the area of touch interaction, for example H. Benko et al. discusses selection techniques for multi-touch [41], W. Buxton et al. discusses issues and techniques in touch-sensitive table input [43], H. Benko and D. Wigdor collects and discusses common problems with touch interfaces [44] and O. Hilliges et al. discusses different tabletop interactions [45].

3.3.1 Touch Benefits

The ability to touch objects directly on the screen has a very strong appeal to users [41]. Touch removes one layer of abstraction; input and output is placed on the same place and it becomes easier for the user to understand what is happening. A user can drag an object or press buttons just like in reality.

An advantage a big shared touch screen has over smaller individual screens is the fact that the controls are mimetic, a newly arrived player can look at what other players are doing and easily understand how the controls work. J. Juul states that it is important to easily and quickly learn the controls of a game [42].

Multiple position sensing is a feature a touch screen is better at than a mouse [43]; a mouse can only report one position at a time and one user can physically only use two mice at the same time. If a touch screen is constructed for multiple inputs, one user can physically control up to ten points at the same time; one point for each finger.

3.3.2 Touch Limitations

H. Benko and D. Wigdor [44] discusses a number of limitations with touch and tabletop usage compared to mouse and keyboard interfaces.

One of the limitations, with all touch screens, that they mention is the lack of haptic feedback in the interfaces. A common button returns to its unpressed state when it has been pressed whereas a button on a touch screen does not. Hilliges et al. states that a user might be used to the mouse and keyboard feedback. This feedback is important for motor learning and automation of repetitive tasks [45].

Another limitation, noted by Benko et al. is a lack of a hover state, since a mouse has at least three states. Out of range when the mouse is not on any surface, hovering when the

mouse is on a surface and pressed when the mouse is on the surface with a button pressed. The more buttons, the more states can be made. A touch interface on the other hand does only have two states, out of range and pressed.

The fat finger problem is also discussed by Benko et al. This problem is defined as a finger being much bigger than the pixel the user intends to touch, and thus it is impossible for the user to see the pixels he or she intends to press because the finger blocks the view. This creates a problem with accuracy of aim.

Accidental activation and tabletop debris are other problems arising on tabletop touch systems. These problems are mentioned by Benko et al. as well. When using the touch device, a user might accidentally activate unwanted input with elbows or physical objects lying on the screen. In these cases the user might become frustrated when the device is not acting like expected and it might be hard to find what is causing it. If a stylus is used, the hand might be resting on the screen and might activate unwanted actions.

3.4 Games

One can divide games into two major groups; casual and hardcore. According to Jesper Juuls casual games are positive stories, which are easy to play and require little time. Hardcore games on the other hand often have a negative story, are harder and require much more time. Juuls states that casual games have five components; fiction (emotionally positive fiction), usability (easy to understand directly), interruptibility (the game can be played in brief bursts), difficulty and punishment (not very devastating if the player fails) and juiciness (Excessive positive feedback on successful actions). The best and most loved casual games are those where the player fails some but succeeds more [42].

3.5 Game Design Patterns

Use of game design patterns is a way to define and describe recurring themes and mechanics in game design. According to J. Holopainen and S. Björk, game design patterns have several uses. They serve as a way to discuss problems during game interaction design conundrums, inspiration, creative design tools and ways to communicate with peers and other professionals [46].

4 Method

The goal of the project is, as previously stated in chapter 1 Introduction, to explore the design space of games for the Surface and to find out what works considering the Surface's method of interaction. To achieve this goal a number of prototypes will be designed and tested. The group consists of two developers and all game play ideas will be discussed in the group before they are implemented to any prototype.

The Surface Simulator will be used but as it requires a screen resolution higher than that of most laptop screens and therefore most programming will be done at separate places where a screen with sufficient resolution to run the simulator can be accessed.

Because of the fact that each prototype is very small, the programming on them will be split between the two developers in the group in a way such that as few conflicts as possible will be made. The developers will start on one prototype each and can then switch to the other developer's prototype if this is desired. Both developers will never work on the same prototype at the same time. This will minimise documents and documentation needed and will instead let developers get working code faster.

Internal discussion between developers while working apart will be made using a chat client, allowing instant messages and code chunks can be sent and discussed in real time. Some development and bug fixing will be made directly on a Surface to be able to directly test newly added parts.

It should be noted that there exists many more methods than those mentioned in this chapter. However, during the course of our Master program, we have received training in these methods and we know that they are useful for getting us to where we want to be. Thus, to minimize time spent searching for and learning new methods, we decided to stick with only them.

4.1 Initial time plan

This plan follows Chalmers standard study plan, with the exception of Easter break.

Week 46 (November 15:th) to ***week 49*** (December 10:th)
Initial research into the Surface and touch, get development environment and simulator to work, check out games for other touch-systems

Week 50 (December 13:th) to ***week 4*** (January 28:th)
First prototype.

Week 50 (December 17:th) to ***week 2*** (January 14:th)
Christmas break

Week 5 (January 31:th) to ***week 7*** (February 18:th)
Second prototype.

Week 8 (February 21:th) to ***week 10*** (March 12:th)
Third prototype.

Week 11 (March 14:th) to ***week 14*** (April 1:st)
Fourth prototype.

Week 14 (April 4:th) to ***week 17*** (April 29:th)
Dedicated to writing the report and preparing for the presentation.

The idea behind this plan was to define the search-space early and then sequentially design and develop prototypes which explored this space. During a three week iteration, the focus would be on developing and testing a particular prototype with other prototypes receiving less attention. This plan begun to fall apart when it was discovered that the prototypes could be handled a lot faster and in parallel.

4.2 Design Methods

Brainstorming is described in the book Design Methods by J.C. Jones [47] as a way to stimulate people to generate a lot of ideas quickly, working from the principle that the more ideas that are out in the open, more ideas are good and both good and bad ideas can be built further upon. Consists of the participants speaking/writing everything that comes to mind with no criticism allowed for an hour or so, constantly generating new ideas and/or building upon old ones. After this is done all ideas are gathered and reviewed.

The hat method consists of throwing a lot of keywords into a hat and then drawing and combining keywords to get inspiration from the new combinations that arise. This is done to force the participants' brains to find new connections. Jones define such methods as "Removing Mental Blocks" and said that even those put off by mental trickery find such techniques very acceptable when facing real difficulties.

Literature searching is the process of searching for published information regarding a particular subject. According to Jones, its aim is to find relevant information while spending a relatively small amount of time in obtaining it. However, Jones also warns that if one is using an inefficient search method, the methods output approaches zero.

Paper prototyping consists of testing a design on paper before it is implemented. This to be able to find, and easily and cheaply, fix design problems. In her paper on paper prototyping, C. Snyders states that paper prototyping is useful for testing your design

with users before implementation, making fast changes and eliminating technology constraints from the process [48].

Quick and dirty prototyping consists of hacking together something that works well enough to test a concept in as small amount of time as possible and to get a working proof of concept. It allows for quick development of a concept to see if a particular approach seems feasible or not.

4.3 Prototyping

The prototypes will follow the Quick and dirty prototyping pattern and thus graphically be very simple and will mostly test game mechanics present when more than one player is playing the game. All of the prototypes to be implemented will be casual, which means they will not be hardcore. One play session will not take excessive amounts of time to learn and play through. All of the prototypes will be made with the Quick and Dirty design pattern in mind. This to be able to develop and test a lot of ideas in a short amount of time.

4.4 Analysis using Game Design Patterns

According to J. Holopainen & S. Björk, game design patterns have several uses. They serve as problem solvers for game interaction design conundrums, inspiration, creative design tools and to communicate with peers and other professionals [46].

Design Patterns will be used to help analyse both existing games and the prototypes created and as an inspiration for further prototypes. The patterns are to be defined internally, with inspiration from J. Holopainen and S. Björk's book Patterns in Game Design [49] and the Gameplay Design Patterns [50].

4.5 Testing

The prototypes will be tested by observation with added interaction from the developers with the test subject. Testing will be done continuously throughout the project with primary test subjects being the developers themselves, people at the interactive institute at Chalmers and friends and family of the developers.

5 Process

This chapter describes the path taken in this thesis; from pre-study and initial idea, to development and testing.

5.1 Pre-Study

While developing some of the early prototypes, research was done in parallel to the development to find ideas for new games and gameplay patterns in already existing games. The Apple iPad was the device most testing was carried out on. The iPad was chosen because it is very similar to the Surface in many ways, both use a multi-touch screen as the only user interaction for example, and there is a lot more games available for the iPad than for the Surface.

Also during this time, we did a lot of reading on the subject of touch, game design and the combination of the two, as well as scouring the net after all things related to the Surface.

A decision regarding methods was also reached early in the development. To allow for more time spent researching touch, games and the Surface as well as doing development, implementation and testing, we would stick to the design methods we knew from before and not spend time researching new ones. We felt that the design courses that we both had read provided sufficient knowledge and training to get us were we wanted to go.

5.1.1 iPad Game Testing and Analysis

Some time was spent on testing and analysing games for the iPad, resulting in a rather large matrix of games and some of the game design patterns used in them, see Figure 7.

The iPad games chosen to be analysed were mainly selected to be games from the "Free Games Top Charts" list in the Apple App Store, from late 2010 to early 2011. 37 games were played and analysed to find out which game design patterns made good games, i.e. games that fit in a top chart. While analysing the game design patterns in the games, those applied to either all or none of the games were ignored and not listed. One could argue that such patterns were must-haves, but figuring out the norm and then look closer at popular games going against the main stream were deemed to be more interesting.

	Power-Ups / New weapons	Winnable	Punishemt for failure	Reward for actions	Randomness	"Surfaceable"	Touch usage *	Casual	Saves game state or level **	Unlockables	Highscore	Player count	Co-op	Boss Monster	Levels	Multiple Game Modes
Fruit Ninja	yes	no	3	5	yes		Multi touch, Slice	yes	no	yes	yes	1	na	no	no	yes
Ninjump	yes	no	5	4	yes	Maybe	Tap	yes	no	no	yes	1	na	no	no	no
Gun Bros	yes	yes	3	2	no		Fake gamepad	no	yes	yes	no	1	na	yes	yes	no?
GT Racing		yes	2	2	no		Gyro, Hold finger	no	yes	yes	yes	1	na	no	yes	no
Hit Tennis 2	no	yes	3	1	no		Slice	maybe	no	yes	yes	1 - 2	no	no	no	no
Cut the Rope	yes	yes	2	3	no		Slice	yes	no	yes	yes	1	yes	no	yes	no
Harbor Master / Flight Control HD	no	no	4	3	yes	maybe	Drag	yes	no	no	yes	1	yes	no	yes	no
ElementZ	yes	no	1	4	yes	yes	Drag / Tap	yes	no	no	yes	1	yes	no	no	yes
PyramidZ	yes	no	1	4	yes	yes	Drag / Tap	yes	no	no	yes	1	yes	no	no	yes
Sparcle hd	yes	yes	3	4	yes		tap	yes	yes	yes	yes	1	na	no	yes	no
Smurfs	yes	no	2	3	no		tap	yes	yes	yes	no	1	na	no	no	no
Spider Man HD	no	yes	3	3	no		fake gamepad, tap	no	yes	yes	yes	1	na	yes	yes	no
Panzer class	no	yes	4	2	no	yes	drag, tap	yes	no	yes	yes	1	na	no	yes	no
Predators	no	yes	4	5	no		fake gamepad	no	yes	yes	no	1	na	yes	yes	no
Angry Birds	yes	yes	3	4	no		drag, tap	yes	no	yes	yes	1	na	no	yes	no
Deer hunter	no	no	2	2	yes		tap, drag	maybe	yes	yes	yes	1	na	no	yes	no
Dizzypad	no	no	5	1	yes	yes	tap	yes	no	yes	yes	1	na	no	no	yes
Plasma Globe	no	no	5	2	yes	yes	hold	yes	no	no	yes	1	yes	no	no	no
Halogen	yes	yes	5	5	yes	yes	hold, drag	maybe	no	yes	yes	1	na	yes	yes	no
Geared 2	no	yes	2	3	no		drag	yes	no	yes	no	1	na	no	yes	no
Mirrors Edge	no	yes	2	2	no		slice	maybe	yes	yes	yes	1	na	no	yes	yes
Friendsheep	no	yes	-	1	yes	yes	drag, tap	yes	no	no	no	2 - 6	no	no	no	yes
Zombie smash	yes	yes	4	5	yes		slice,drag,tap	yes	yes	yes	no	1	yes	yes	yes	no
My memory hd	no	yes	-	1	no	maybe	tap,	yes	no	no	no	2 - 4	no	no	no	no
Train conductor 2	no	no	2	4	yes	maybe	tap,drag	yes	no	yes	yes	1	na	no	yes	no
Can knockdown	yes	yes	4	4	no		drag	yes	no	no	yes	1	na	no	yes	no
Zentomino	no	yes	-	2	no	yes	drag , tap	yes	yes	no	no	1	yes	no	yes	no
Salt	yes	yes	5	3			fake gamepad	no	no	no	yes	1	no	no	yes	no
Cover Orange	no	yes	3	3	no	maybe	tap, drag	yes	yes	no	no	1	no	no	yes	no
Death worm	yes	yes	4	4	yes	yes	fake gamepad, tap	yes	yes	no	yes	1	no	no	yes	yes
Shift	no	yes	2	2	no		fake gamepad	yes	yes	no	no	1	no	no	yes	no
Sandstorm	yes	yes	4	3	no		fake gamepad, tap	no	yes	no	no	1	no	no	yes	no
Super Fly	yes	yes	4	4	yes	yes	fake gamepad	yes	yes	yes	yes	1	no	yes	yes	no
Xmas Planet	no	no	3	4	yes	maybe	drag, tap	yes	no	no	yes	1	no	no	no	no

* slice = quick, drag = slow ** Very hard to define what is a save/load-system

Figure 7. The iPad games analysed. For a more readable list see Appendix A – iPad Patterns and Games.

While there were no definitive "use these patterns to make money"-answers, some trends were discovered. Some major features noticed about the iPad games were:

- That the minimal time investment required was rather low.
- That highscore exists in a multitude of forms
- That several levels is the norm
- That the player can unlock different things
- That the machine saves current state upon closing the game
- That most free games are demos of priced products.

A saddening fact was that there were very few multiplayer games in the App Store meant to be played on the same machine.

It was not possible to directly translate the result of this analysis onto the Surface. For instance, the Surface lacks motion sensors, it is quite big and probably is positioned in public, so saving information locally might not be a good idea. A dislike against the common form of virtual gamepad control scheme formed in our minds, mostly because of the lack of haptic feedback and that it suffers the fat finger problem, thus getting quite imprecise.

5.1.2 iPad Game Design Patterns

The game design patterns which were used to analyse the iPad games were the following:

Power-Ups / New weapons[1]*:* The player(s) can acquire increases in power or abilities and/or new weapons
Winnable[1]: The player(s) can win the game.
Interruptibility[1]***:*** The player(s) can interrupt the game without consequences.
Punishment for failure[1]***:*** How hard the player is punished if he or she fails, high meaning the game is over and low meaning only small loss of score or such.
Reward for actions[1]***:*** How much reward the player receives for using the game, also called juiciness.
Randomness[2]***:*** If the game incorporates chance in any way.
"Surfaceable"[1]***:*** If it is possible to directly copy the game to the Microsoft Surface.
Touch usage[1]***:*** How the game uses touch input.
Casual[2]***:*** If the game is not hardcore.
Saves game state or level[1]***:*** If the game saves progress between play sessions.
Unlockables[1]***:*** If player(s) can unlock bonus features in the game.
Highscore[1]***:*** If the game has and remembers highscore between sessions.
Player count[1]***:*** The number of players supported by the game.
Co-op[2]***:*** If several players can cooperate.
Boss Monster[2]***:*** If there exists special, tougher enemies or levels in the game.
Levels[2]***:*** If the game has different levels.
Multiple Game Modes[1]***:*** If the game has multiple game modes.

Note that one pattern can be defined in many different ways. The footnotes only state how we came to know about them.

5.2 Initial Idea

The price, size and usualness of the Surface make casual games to prefer before hardcore games. The features of casual games are often used in popular iPad games, therefore this was something we wanted to follow. Since we belived, for reasons stated in 1.4 Limitations, that the Surface is most often placed in public places where people only stops for a brief moment, and does not have time nor the desire to stay for very long. Of course people can return many times or stay for several hours just to play but this is probably not desired by the Surface owner.

There where another group working in parallel with this group on the same Surface device and the fact that they where working on tabletop games made this group

[1] Defined on our own.
[2] Taken from the book Patterns in Game Design [49].

uninterested in that particular subject. Therefore all forms of tabletop helper or tabletop clone games will be left out from this project.

Our first impression of the Surface was that it is a rather impressive piece of hardware opening up some interesting possibilities, and it seemed like we had an interesting time ahead of us. The initial programs tested on the Surface did impressive things with just finger input. The future of the project looked bright.

A number of prototypes were to be created for the Surface and at the same time research and play testing were to be made. The iPad was chosen as a stand-in for the initial game analysis because there exists a much bigger number of games for the iPad than on the Surface. The iPad was also picked because it is the most popular mass market device with a large touch screen, and it has a well-equipped library with free games.

5.2.1 Internet videos

Fortunately, there exist plenty of user videos on different Internet video pages such as YouTube in the area. Videos such as game reviews or demonstrations of iPad and Smart-phone games are very common, but also many demonstrations of Tabletop games. A good inspiration for tabletop games was Microsoft's own YouTube account [51].

5.3 Development

The development process was divided into three stages; the first stage contained development of some small games to test ourselves and the Surface, this stage also contained research and play testing on the iPad. In the second stage we came up with more ideas and culled among the ideas until only a few were left. In the last stage we developed prototypes for the best ideas generated in the previous stage.

All prototypes were created using XNA 3.1 and Surface SDK 1.0. Microsoft Visual Studio 2008 was used as development environment. The Surface Simulator was used to test inputs to the games on normal computers when there was no Surface obtainable.

A more detailed description of how each game works is mentioned in 6.1 Games Developed.

5.3.1 The Dark Stage

The first step of the project was to get familiar with the Surface device and the different concepts of touch and multi-touch. A number of games are pre-installed on the Surface from the factory so trying out those was a judicious beginning. When these games were inspected and played and we were tired of them, some research in the area of big screen

multi touch games was carried out. The goal was to find inspiration from what others already have created. A development environment was created so we could start implementing something of our own, and the project was placed at Google Code. The first game to be created was a simple tower defence game. The tower defence game, also called Defence, got inspiration from a game another group of students created on a similar touch system [52]. The purpose of the Defence game was mostly so that we could use an existing design to learn the ins and outs of Surface programming. We realised that icons and images were needed for the prototypes and none of the developers were interested in drawing the images. To solve this problem the site findicons.com was used because it contained a large number of icons available for free. This quickly became our go-to place when we needed new textures for a game.

The main feature in Defence is creating and removing objects with fingers. Objects only move in a straight line towards the other side after creation, and the goal is to get the objects to the edge of the other player's side and this is the whole game. The game was fun as it was so we decided not to change it because it was not broken. The game on YouTube also uses towers to show where a player can destroy objects, but the towers seemed unnecessary for our gameplay. At this point we were unaccustomed with programming so many useful help classes in the XNA and Surface API was unused.

Our supervisor, advised us to look at games for a rather similar system, namely the iPad. More about the iPad testing can be found in chapter 5.1.1. Unfortunately, we did not have access to an iPad all the time, furthermore we did not have the economy to buy games for the iPad. Thus we borrowed an iPad and choose to analyse the games on iTunes top free games list. Furthermore, we found inspiration from many online videos where other people were playing or reviewing iPad games, or games for other multi touch devices. More about these videos can be found in chapter 5.2.1.

The concept of a popular and quite simple iPad-game was decided to be copied to practice some more Surface programming, the game copied was *Flight Control* [53] and the prototype created was Airport. The goal in Airport is to guide incoming airplanes to an airport. Airport has unlike Defence different directions objects can move in, and also stores positions towards which the plane shall move to in the future. We tried to make the planes not turn on a dime by implementing how much the planes could turn per frame. However, this endeavour was abandoned when we discovered that planes tended to get stuck circling way-points. Another problem in Airport turned out to be the randomisation of where the planes spawns, they pick an edge, a position and a direction, and then they start moving entering the screen. Sometimes the position picked is unlucky and another plane is already there, this results in a crash outside the screen and the player's game is over. We decided not to fix this because the gameplay was there and it happened only rarely.

Spurred by the Airport result, a brainstorming session was performed and two more game ideas were thought of, one using tags as primary interaction and one being a kind

of platform game. A simple and well known game which easily could make use of tags was decided to be a clone of *Asteroids* [54]. Asteroids, which we call our prototype as well, were to have the same features as the original Asteroids, but controlling the ship's position and rotation with tag.

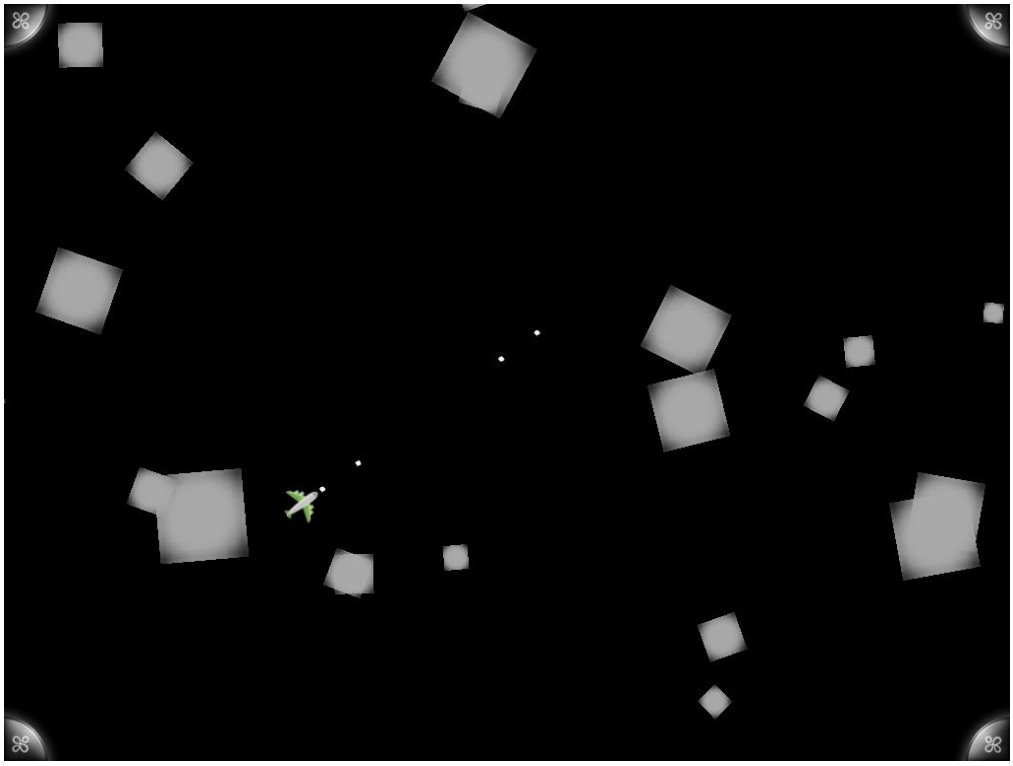

Figure 8. An image of Asteroids early in its development

Asteroids started as a single player game with only one kind of simple squares as enemies. It gradually continued to have more kinds of enemies worth different amount of points and also score multipliers. Support for several ships was added early in its life cycle while removing ships form the game was a feature added very late in the process of development. An important key feature of Asteroids was from start the ability all player ships has to, at all times, stay positioned under a tag. A player can at any time lift the tag and place it at a different location on the screen to have the ship transferred there immediately. This feature was thought to be overpowered at start so we added increasing score multipliers which were reseted when a ship jumped. After further testing and discussion, we realised that the game lost some witticism with the penalty for jumping so the penalty was removed.

A design document needed to be created for the platform game because it had quite complex features, such as multiple gravities, multiple objects and multiple player characters, see Appendix D – Design Documents. Jumping was a new feature Platform used, but it does not really have any meaning, or scope of use. At first the idea was to have bottomless pits to jump over but we decided such obstacles were not needed for the prototype. All player characters were given an attraction value, which is how much objects in the cloud are attracted to the character. The attraction increases when the

player is standing still or is shooting, and it decreases when the player is moving. This value turned out to be very hard to balance.

The main point of Platform was to force players to walk around the Surface, hunting each other. The Surface height, discussed in 6.2.2 Height, obstructed this idea though. It is not possible for a full grown person to stand upright and at the same time reach the Surface with a finger. Standing bent down while playing also leads to a problem being the player's head-butting each other.

5.3.2 The Middle Stage

To come up with more original ideas our supervisor suggested we should come up with a matrix containing a large number of ideas for different games. To do this we first noted all game genres we could come up with, then each genre received a couple of games that do exist in this particular genre, but as if they instead would be using touch or tags. Inspiration for the control mechanics on most of the games where taken from iPad and Smartphone games previously tested or seen. For instance, first person shooters would be controlled by moving the character with a finger on one side of the screen and aiming and shooting controlled with a finger on the other side of the screen, just like the game *Battlefield: Bad Company 2* for the iPhone [55].

During the idea-spawning sessions, 74 ideas were created and written down with some key features of each game, see Appendix C – Game Idea List. The ideas ranged from simple point as quick as possible-games, to more advanced city-building games. Many game ideas were considered infeasible or just plain strange, so we were compelled to remove most of them from the list. We designed a number of cull criteria by thinking where the games would be played, by whom and what we considered interesting. The criteria created are mentioned below and the reasons why we picked most of them are discussed in 1.4 Limitations. The cull criterion, "No sport games", not discussed above was created because of personal preferences. We figured that many of the game mechanics in sport games are also found in games with other themes.

Cull criteria:

- No 3D
- No board game helpers
- Nothing that requires a large investment in play time
- Nothing that cannot handle a public environment
- Nothing more than very simple AI, as in follow a player/random/simple path
- The game should support more than one player
- No gesture-recognition
- No copies of existing board games
- Combine games ideas that are very similar
- No sport games

- No game that is too complex or has a high learning curve
- No game that requires fast finger movement
- No locked downward direction to only one side of the screen
- No emulating a mouse

A list containing 20 games was derived from the old list after the culling process was finished. This list had to shrink even more for us to be satisfied so the paper prototyping method was initiated on some of the remaining ideas. This removed a ghost hunter puzzle game and a flower picking game from the list. Some ideas which turned out to have too similar interactions were combined or changed, for instance a leaf blowing game had too similar interactions with Asteroids and were removed, but also gave some new ideas to the Asteroids prototype.

The list ended up containing ten games after all culling was finished. Out of these, three games were selected to have prototypes created, for seeming interesting, fun and they also utilised different mechanics of available input. The games chosen were a racing game, an artillery game and a node battle game, the already playable Asteroids was also decided to be modified to be more similar to the leaf blowing game.

During this time, we also did a small analysis on the games we had so far. The result regarding the games was inconclusive, as they were too few and, in many cases, too underdeveloped to give any definite answer. However, the analysis brought fourth another issue. Some things that were affecting the gameplay were not being tracked by the patterns used to analyse the iPad games, and some of the iPad patterns were found to be redundant since we had designed all or none of the games to follow those patterns. This made us look for, and come up with, new patterns to use. This, along with analysing the games, continued throughout the rest of the project and is more detailed in chapter 5.3.4 Games under Analysis.

5.3.3 The Stage of Discovery

At this point, we were more confident in Surface programming, and also had some hunches in what type of games and game mechanics we liked, so the games were to be a bit more advanced. We also liked the tag steering in Asteroids, but only utilised tags in one prototype, so we decided to test it a bit more. The racing game, named Racing, and the artillery game, named Artillery, were decided to utilise tags.

In the first versions of Racing, the game only supported one car/controller and additional controllers just gave conflicting input to the car. The game quickly expanded from there to include multiple car-controller pairs and visual feedback regarding the controllers' positions. Also, a multitude of control variants were tried during the game's development. For example, while the game in the last version uses an absolute direction for a car's movement, we tried to use a more relative steering, like that of real car. But the problem with that type of steering was that it became rather confusing and unclear

regarding which direction the car was steering and what tag rotation corresponded to straight ahead. The steering wheel steering option was also disliked by testers due to its low learnability, it can still be enabled by a small change in the code though. One reason for this dislike was its lack of feedback to the user, rotating the tag more than one turn resulted in unrealistic behaviours where the car started to turn in the opposite direction. More user feedback could have been implemented to the prototype attempting to solve this, but realistic steering was instead decided to be disposed of. The car acceleration in Racing uses the distance from an anchor positioned by the player to decide how fast and in which direction the acceleration should be in. This functions somewhat like a virtual analogue stick, but without feedback.

We wanted to create a space game utilising realistic physics, where the player can feel like a real galactic emperor with a number of regulators at his or her hands. The space option was necessary due to our culling option to not have one set downward direction on the screen. The regulators would modify different settings and be represented as a number of tags in front of the user. If the regulators, or tags, were moved or rotated they would tune in a trajectory of a missile, and a big button would launch the missile when all settings felt satisfactory. At first the regulators were the only way a user could modify the missile's settings, but we decided to add one more option because we thought that the tag controls were too slow and unfamiliar. The new control became a normal arrow positioned by a finger.

An aspect that Artillery used was infinite power in the launches, players maximised the power of the missiles and it received such speed that the gravity from planets did not affect the missiles enough. This was not intended so two fixes were implemented, firstly, a maximum power and secondly, exponential population costs for launching missiles. Five percent of the own population is killed if maximum power is used; with low power, only a negligible amount of the population is killed.

Balancing Artillery proved to be quite hard, it must be possible to kill the population on the other player's planet in reasonable time and at the same time not kill too many on the own planet. Additionally, if a missile does not hit anything at first, it tends to take one big turn around the game field, outside the screen, entering the screen once more after a while. Each time the missile re-enters the screen, it most often has a slightly different speed and angle, which sometimes end up crashing into the planet which sent the missile, or with luck the enemy's planet. This feature lets players play with finding an orbit for their missiles instead of trying to kill each other thus creating a new type of game. During testing nobody achieved orbit of a missile for more than a couple of laps though.

Artillery also serves as a sandbox since a player, at any time, can add new planets to the game field with a tag. This is only possible though, if the tag used has not been used before to create a planet. These planets function just like any other orb on the game field and cannot be removed by a player. We liked this feature in gameplay because if a

perfect missile trajectory to the other player's planet is found, this player can just continue to shoot in the same direction and win, if the other player then places a planet somewhere on the screen the trajectory will change and the next missile will have a high probability to miss. We discussed to change these planets built by tags to be movable with the tag, but players would probably just had destroyed all missiles immediately they were created so we trashed the feature.

Having planets moving all the time also was a feature we thought of implementing, but this would come with two major complications. One being that it would make it almost impossible to hit the other planet and each rocket would get completely different trajectories. The other being the resolution of the screen, the planets could not possibly stay in the screen at all time with the given resolution of the Surface, and having planets outside the screen changing missile trajectories would probably become very confusing for the players.

By now we felt that we needed a two-player versus game operating in real-time. The idea called Node Battle filled these criteria and development began on the new game. The game consists of a number of nodes that constantly produce troops that the players can direct to conquer new nodes. Each player starts with one colour and node and the goal is to eliminate the other player's colour. The first iteration contained a rather severe learning curve, favouring the first one in the update queue and contained the largest positive feedback loop ever seen by us. The second iteration somewhat flatted the learning curve, removed the bias towards the first player and tried, unsuccessfully, to lessen the feedback loop by introducing more sources for mistakes.

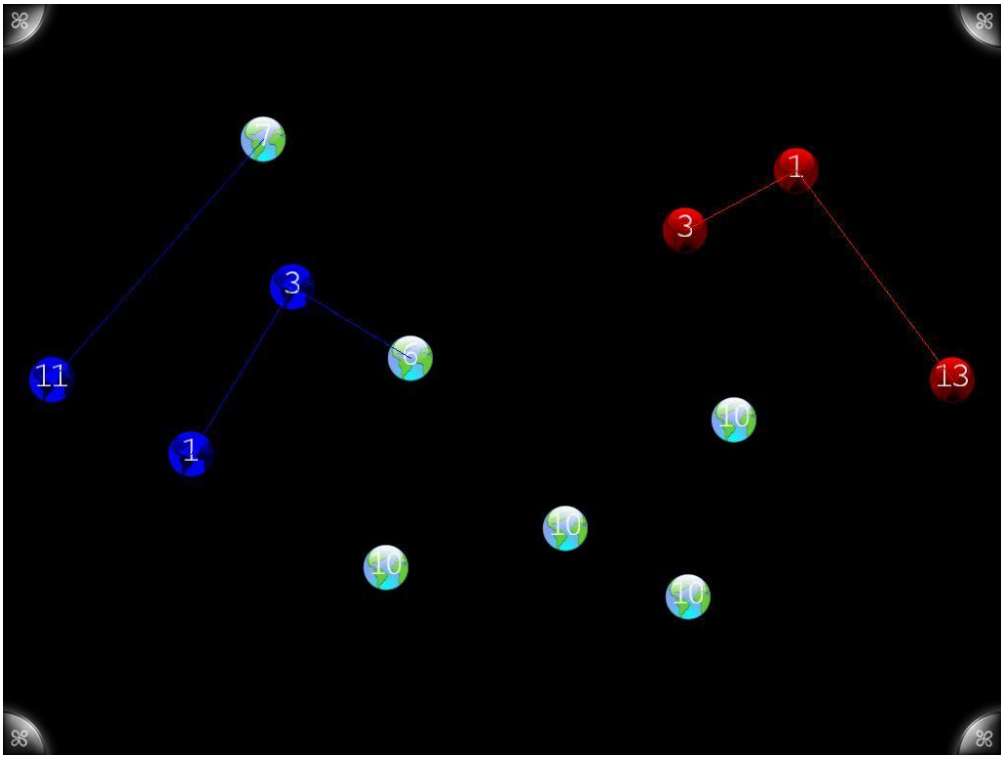

Figure 9. The first working iteration of Node Battle.

It was during a test of this iteration that we stumbled upon what we called physical play. One player started giving commands to the other player's side and the second player responded in kind. This phenomenon is further discussed in 7.1.3 Physical Multiperson play. Unsure of how to proceed and with the project coming to an end, the third and final iteration only changed game balance and learning curve. This by adding more nodes, giving them random starting troop numbers and replacing the previously used numbers on the nodes with resizing nodes.

The Asteroids changes made were necessary to let it stay unculled in the game list, because at this point it was not a multiplayer game. The multiplayer feature was added to the prototype and after some discussion we also decided to have ships destroyed in addition to loss of score when a ship touches something bad. A feature we liked was surprisingly added thanks to this, which was tags as extra life, more about this in 6.1.4 Asteroids. A problem that was found when removing ships was that players had no idea that their ship had been removed due to the fact that a ship is rarely visible for the player. To fix this we added a projectile explosion when a ship is destroyed, reaching outside the area hidden by the tag. This explosion consists of a high amount of normal bullets, which also destroys asteroids.

	Power-Ups / New weapons	Winnable	Punishment for failure	Randomness	Touch usage *	Highscore	Player count	Co-op	Non-Player Help / IRL player helper	Boss Monsters	Levels	Leming curve	mean play time for 1 one round (in min)	consecutive rounds depend on previous rounds(how)	Objects hidden by players **	other players 'in the way' ***	"game over" -able ****	Chance for Anabsis Paralysis	real-time	Late arriving players	Honor system in place *****	Snowball
Defend your side	yes	yes	3	no	tap, touch and hold	no	2	no	yes	no	no	low	2	no	low	low	yes	no	yes	yes	yes	no
Airport control	no	no	5	yes	Drag	yes	1	no	yes	no	no	low	3	yes	med	med	yes	no	yes	yes	yes	no
Asteroids with bullet hell relations	yes	no	4	yes	tag	yes	*	yes	no	yes	no	low	3	yes (highscore)	high	med	no	no	yes	yes	yes	no
Four side platform game	yes	yes	3	yes	Drag, tap, slice	no	*	no	yes	no	no	low	undef	no	med	med	no	yes	yes	yes	yes	med
Top-down Racing	no	yes	2	no	tag	yes	*	no	no	no	yes	low	3	no	low-high	low-high	yes	no	yes	yes	yes	low
Turn based Artillery game / Slingshot	no	yes	3	no	Drag, tap, tag	no	2	no	yes	no	yes	low	4	yes (nr of obst)	low	low	no	yes	no	no	yes	med
Node and troop battle	no	yes	3	yes	Drag, slice	no	2 to 4	no	yes	no	yes	low	3	no	med	med	yes	yes	yes	no	yes	high

* tag = move,rotate && presence
** Players limbs obscure game objects
*** Other players trying to be in the same physical space
**** A state defined such that it is no longer possible to proceede
***** The game dosent stop cheating, especcially not outside of the machine

Figure 10. The analysis of our games. For a more readable version, see Appendix B – Implemented Game Pattern Matrix

5.3.4 Games under Analysis

After our prototypes were elaborated enough, the same analysis done to the iPad games were done to the prototypes developed in the project. A direct comparison with the gameplay design patterns found in the iPad games was executed. It was quickly discovered that not all of the old patterns were interesting methods of comparison for our games. Some examples of this were Casual, which all the games are by design, Saves, Interruptibility and Unlockables, because there will probably be new players each time, Multiple Game Modes since that would be a new game and Visual Rewards

since the visual part was not prioritized during the implementation. Then we added and expanded upon the list of patterns for our analysis.

The added gameplay design patterns were:

Non-Player Help / IRL player helper*[1]*: Another person helping the "primary" player handling the difficulties of the game.
Learning curve*[2]*: How long it takes to learn the rules of the game.
Mean play time for one round (in minutes)*[3]*: How much time each round of the game takes, one round being a natural breakpoint in the game.
Consecutive rounds depend on previous rounds (how)*[3]*: If f(n) = f(n-1) + 1
Objects hidden by players*[3]*: How much the game is subject to the "fat arm" problem.
Other players in the way*[3]*: How much players are in the, physical, way of each other.
"Game Over"-able*[3]*: If there is an end-state were play is no longer possible.
Chance for Analysis Paralysis*[12]*: The chance that the player is given too many options and cannot decide between them.
Real-time*[12]*: If the game is not paused to wait for other players or such.
Late arriving players[12]: If the game supports that players can jump in at any time.
Honour system in place*[3]*: If the game checks for any form of cheating, outside of the game.
Snowball*[3]*: How much early advantage translates into late-game power.

It should be noted that all of the games contains several more design patterns incorporated in the prototypes but the ones above are the ones we have analysed and consider relevant to the question. For example, in several of the games it is possible to damage characters in different ways. While this is a pattern, it is not relevant when answering the question.

Analysing our Games
Just before we dedicated ourselves to writing this report, we went through and analysed our games with regards to our patterns and how well the game fit the Surface. We checked each pattern's impacts on the fun of the game and how it made the game fit the Surface. Most of the patterns were found to have little to no effect on either the fun or the games fit for the Surface. The second largest group were the patterns that had an indirect effect or only had an effect in special circumstances. And then a very small group that had a direct effect on the games fit for the Surface.

[1] Taken from the book Patterns in Game Design [49].
[2] Taken from Gameplay Design Patterns [50].
[3] Defined on our own.

5.4 Prototype Testing

The user tests which were carried out throughout the development process, as described in 4.5 Testing, resulted in a number of interesting observations. Testers were invited to the Surface and played all prototypes developed up to that point. As anticipated, much of the feedback received from the testers included juiciness, or the lack thereof, which we in many cases ignored. Because in our opinion juiciness, mentioned in 5.1.2 as the pattern *Reward for actions*, gives very little to the answer of our question.

While testing Defence we found out that the input speed discussed in 6.2.4 can be exploited by moving a finger back and forth very fast on the screen. This motion results in a big amount of taps which all results in a ball being created in the game. After further testing we found out that a player can generate many more unique contacts on the screen by quickly sliding fingers back and forth than by rapidly tapping on the screen.

While testing Asteroids we noticed that almost all deaths were caused by occlusion where the player's arm covered an important part of the screen. This issue is discussed more detailed in chapter 6.2.3 Occlusion.

The initial idea of Platform, to force players to move around the Surface, but they did not, instead they often grabbed a new character and ignored the old one. If they ended up having their character on the other side of the screen, they often stretched their arm there instead of walking there. With these observations we can draw the conclusion that players prefer stretching their arms over using their legs to reach something on the other side. Alternatively, the physical properties of the Surface prevent it. Unfortunately, we did not have access to child testers which could have been useful while testing Platform.

During one play test of Node Battle, two players started fighting over the nodes, removing each other's connections between planets and pushing away each other's hands when trying to create new connections. The game became much more violent than anticipated, but the testers had a good time and the game actually became more balanced this way than when the players would not touch each other or the other player's connections. The game also received a new objective, in addition to growing faster than the other player, also to prevent the other player from growing too fast.

Play tests of Artillery showed us that most users prefer to use fingers over tags if both options are made available. We are uncertain why it works this way but one judgement is that people are not used to and thus does not like to have objects covering the screen. One tester liked it a lot though because he really felt like the galactic emperor.

5.5 Surface Usage

While researching and working on the Surface we discovered many mechanics of it. These observations can be divided into two parts; input and output.

5.5.1 Input

While programming directly on the Surface, mouse and keyboard are necessary because the screen cannot take inputs to the operating system, it is also recommended, but not needed to have a secondary screen connected to the Surface.

When using a Surface program utilising the screen's inputs, the Surface has four input methods; fingers, byte tags, identity tags and objects. Of these only two were tested in the project; fingers and byte tags. The reason why identity tags were not tested was simply because we found none to use and also found no reason to use them in games so we gave no extra effort in finding or creating our own. That many unique values the identity tags have would be superfluous for any type of game according to us. The input method using objects was not used because none of our game ideas used it and we did not consider it interesting enough to be implemented in any of the games developed.

A sunny day while we were working on the Surface, we stumbled upon an interesting downside of it. When the light in the room was very bright the Surface started to act strange and detect inputs even though nobody was touching it. We realised that it must have been the light from the sun disturbing the machine. Shadows from users while in the bright room also created unwanted inputs. Running a calibration program included with the Surface on the machine helped but did not completely remove the problem.

Another calibration program included with the Surface once fixed an input problem we stumbled upon where the Surface did not recognise input in a particular area of the screen.

5.5.2 Output

We noticed that an occlusion problem is apparent in most of the games developed and also in many other games tested on the Surface. When the Surface shows images there is a big risk that a user already has or is going to put a finger or arm in that area. This results in an annoying flaw where the user cannot see this particular area of the screen because it is occluded by him or herself.

Another flaw of the Surface we found is its very noisy fan which always is running at full speed, probably to cool the projector, but still very loud and noticeable. It is easy to get used to the sound and it is not annoying in any way but at first when starting up the Surface or entering the room where it is positioned, one notices it.

The Surface does have a built in loud speaker but we never took advantage of this because we did not see it worth the time, just as with 3D graphics. Game sounds could also become disturbing in the environment the Surface is positioned in, so we decided to completely skip it.

6 Result

It is important to present the user with feedback for inputs [44]. In the general case there are many ways to do this, for example by sound or finger sensation. On the Surface however, a designer do not have any other tools than visual feedback. In the lifecycle of the prototypes, these have most often been changed to increase the amount of visual feedback. There is a clear and noticeable difference between the game that is the furthest along in this process and those that have barely begun.

6.1 Games Developed

In this chapter each game prototype will be described and in terms of playability, fun, what works well and what does not work very good at all, and such.

6.1.1 Defence

Defence is a multiplayer game where the objective is to defend a long side of the Surface. The game can be played by two or more players, where the players can form two teams, occupying opposite sides of the table and competing against the other side.

The rules are fairly simple, if a player touches the screen close to the long side edge of the screen the player will send a ball towards the other side, touching a ball which is not close to this edge will remove the ball from the game. If too many balls reaches the other side, the team on that edge loses.

Controls:

Tap	Hard press

Patterns:

Power-ups	“Game over”-able	Non-player helper
Real-time	Low learning curve	Mean play time 3 min
Winnable	Not susceptible to fat arm	Not susceptible to other players in the way
Honour system	Late arriving players	Moderate punishment for failure

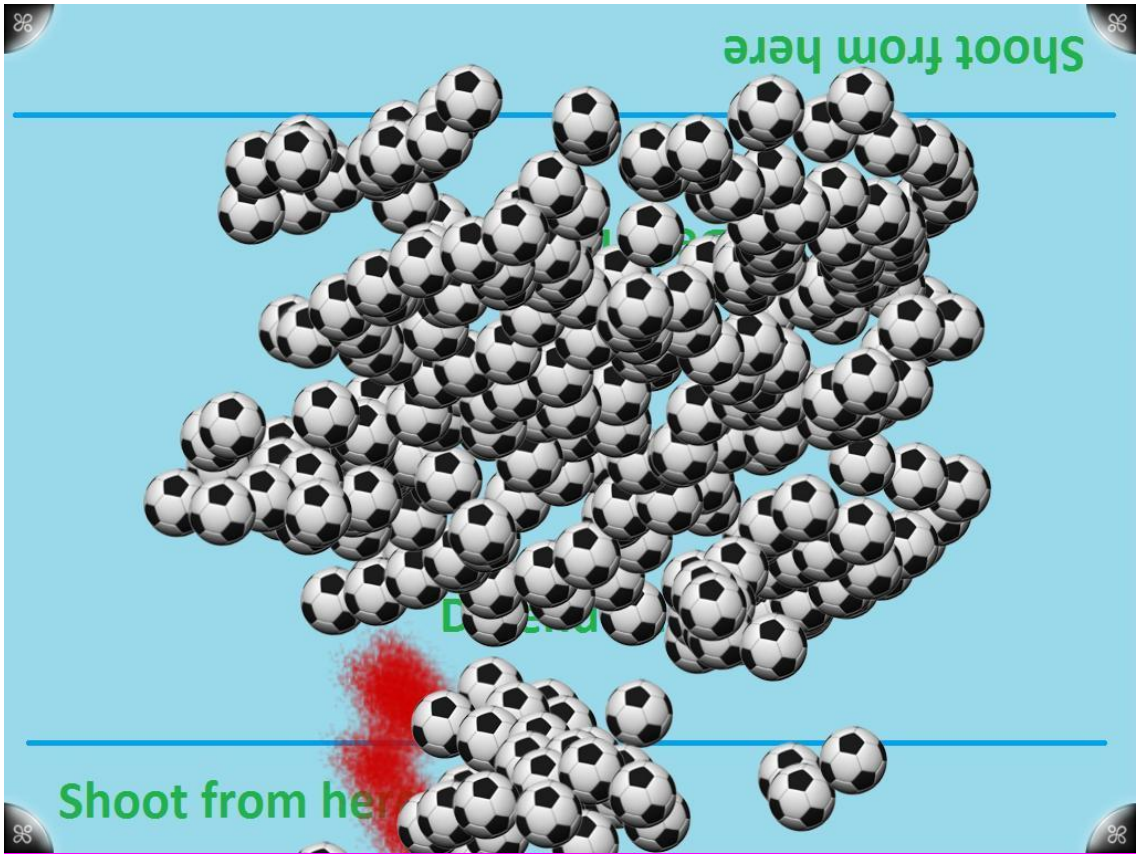

Figure 11. An image of Defence. Balls spawn when players touch the "Shoot from here" section and disappear if touched in the middle section. The blood puddles mean that some balls has gotten through.

Defence does not utilise any score or have any end conditions, the game only goes on until all players are tired of it. Instead of score, a blood puddle is created each time the players misses to catch a projectile. This puddle is added to the screen close to where the projectile left the screen. At the point when the players decide to stop playing, the only way to select a winner is to count blood puddles, or amount of blood if counting is impossible.

Strengths:

Fun for children	Many players can play together
Easy to learn	Supports wild bashing on the Surface
Interminable	No winner

Weaknesses:

Tiresome	Interminable	No score

Defence Future Work

Features in Defence which would make the game better but never were implemented might be features the source of inspiration has, such as towers, resources and score. Also, end-state and reset functionality could be needed.

6.1.2 Airport

Airport is a simpler clone of Flight Control [53] on iTunes. The objective of the game is to direct airplanes to an airport located in the centre of the screen. This is done by dragging a finger from an airplane to, preferably, the airport, and the airplane will follow the trail after the finger. If an airplane reaches the airport it will disappear and the player receives a point. The more points the player has the more airplanes will spawn and after some time the player will be overwhelmed with airplanes.

Controls:

Drag

Patterns:

Random	Late arriving players	High punishment for failure
Highscore	Low learning curve	Mean play time 3 min
Non-player help	"Game over"-able	Moderately susceptible to fat arm
Honour system	Real-time	Moderately susceptible to other players in the way

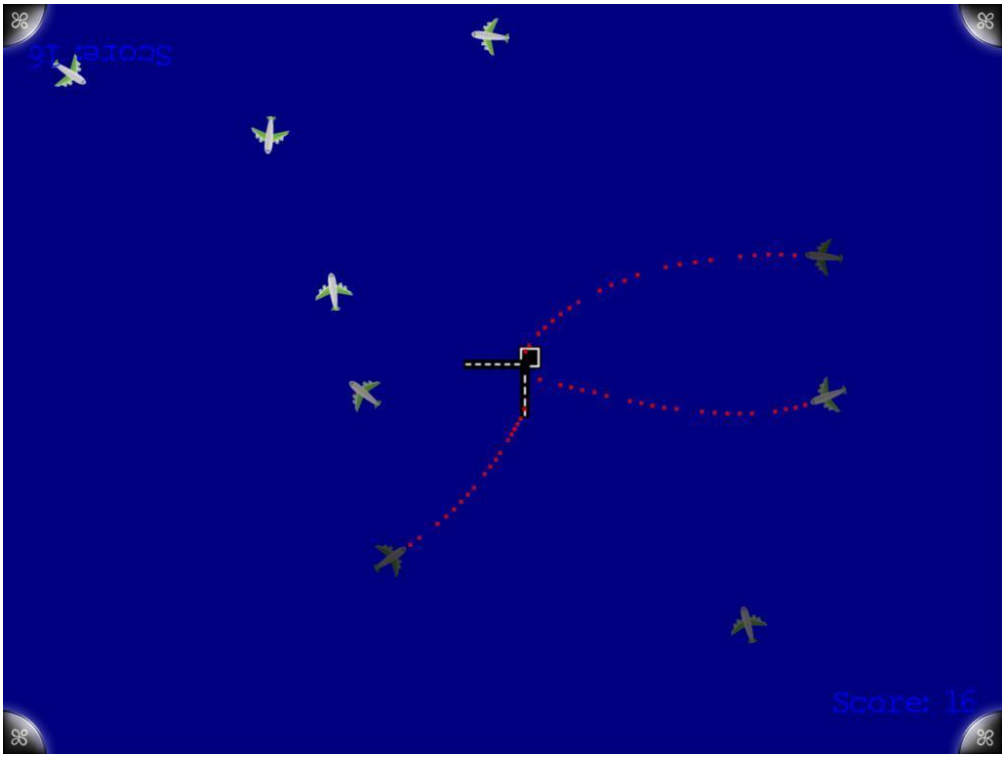

Figure 12. An image of Airport. The black and white object in the middle is the airport. Red dots are the paths a plane will move in. Planes without a path will just continue straight forward.

Airport started as another "learn how to do it" clone and never quite progressed beyond that. That said, the game is very easy to understand and play, and it supports any number of players. Problems are that the starts can be very slow, planes spawn in such a way that they can collide after a very short time and it is not very juicy. If given more

time to polish, Airport could very well be developed into a quite big time sink, but as of right now, its problems stop players from playing for more than two or three rounds.

Strengths:

Highscore	Easy to learn	Unlimited number of players

Weaknesses:

Start can be very slow	Some planes hard to see

Airport Future Work
Different kinds of airplanes would be an interesting feature, so would different airports or levels. Storms or other things that deny planes access to an area could be interesting as well. A mechanic preventing planes from spawning inside each other or steering into each other near the edge of the screen when they just have spawned should be fixed first though.

6.1.3 Platform

Platform is a multiplayer four-sided platform game where each player controls a character which can shoot, jump and run along the edges of the screen. A player character is created by touching a portal in the centre of the screen. This character is then moved by putting a finger on it and then dragging the finger in the direction the player wants to move it. Jumping works in the same way but by moving the finger upwards. A character can only shoot in one direction, which is done by tapping on it.

In the middle of the screen there is a cloud in which objects are floating. The objects are movable and throwable by any player and might also move towards characters by themselves if the character is standing still for too long. When a character is hit by an object, it can damage the character or modify the character's movement speed positively or negatively.

Controls:

Tap	Gestures	Goal oriented movement

Patterns:

Winnable	Non player helper	Low punishment for failure
Real-time	Supports late arriving players	Mean play time undefined
Honour system	Chance for analysis paralysis	Moderately susceptible to fat arm
Moderately susceptible to other players in the way		Low learning curve

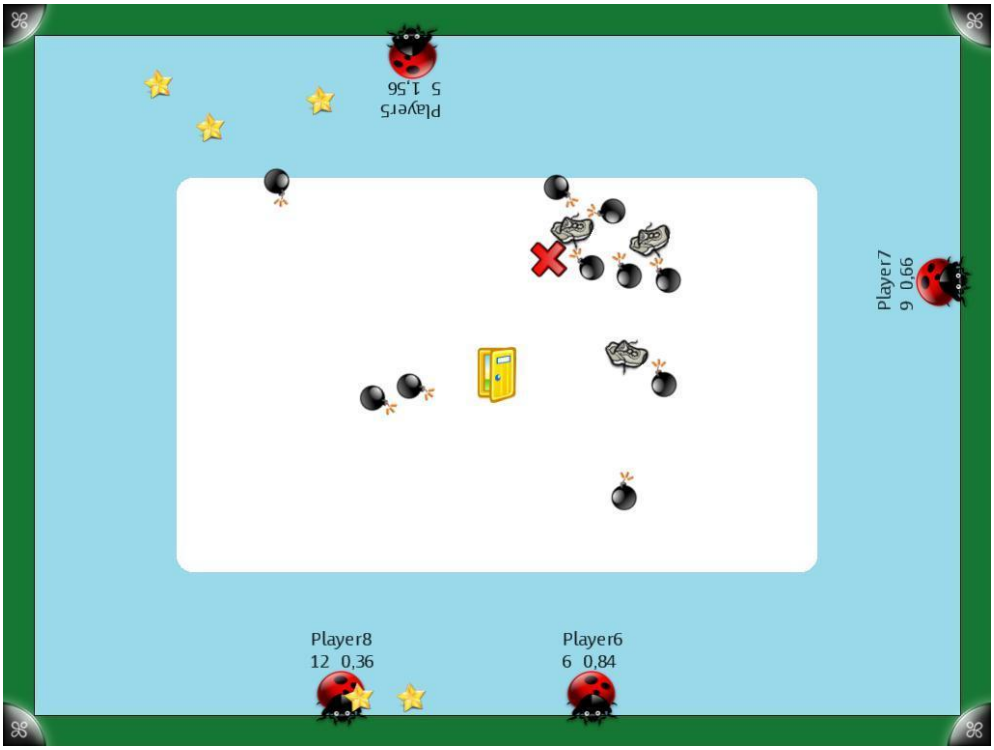

Figure 13. An image of Platform. Players spawn ladybugs from the door in the middle. The player's health and attraction are shown above the ladybugs. Bombs, shoes and crosses can be thrown towards other players.

Platform turned out to be a bit too complex and test subjects tended to not understand some features of the game. One could argue that it became too hardcore. An example of a disliked feature is that shooting is only allowed in one direction. The testers were not fond of getting up from the comfortable chair they were sitting in to walk a lap around the Surface. On the other hand, the testers quite liked to grab bombs from the middle of the screen and throw at each other.

Moving and jumping turned out to work out quite well. Testers had some problems getting through corners though, smooth corner motions with the character requires some training. Just as with Airport, Platform lacked juiciness, which probably would increase its playability and make it more fun if it was added.

Strengths:

Infinite number of players	Throwing bombs at other players

Weaknesses:

Complex	Jumping has little meaning

Platform Future Work

In this game several small improvements could be added. One very important feature which is missing is user feedback; this would be used by visualising which characters are currently selected by whom and in which direction they are moving in. Also

interfaces for jumping and shooting as well as life and attraction. And creating more obstacles in the game-world could create interesting maps.

6.1.4 Asteroids

Asteroids are a cooperative single or multiplayer game where the all players each control a space ship. The objective is to shoot down obstacles and other objects without being hit by anything. Destroying objects gives the players points but being hit by an object destroys the player's ship and removes points. Destroying objects also slowly increases a score multiplier and being hit resets the multiplier. The game is initiated without any player space ships present on the screen, but asteroids float around anyway. A space ship is added to the game if any tag is recognized by the system. This space ship will be positioned at the same position as the tag and its rotation will be the same as the tag. For each new tag positioned on the screen, a new space ship is added under it. A destroyed ship will also make its respective tag unable to create new ships on the screen, and is in a sense also dead.

All space ships in game will always position themselves straight below their respective tag on the screen and while the system detects the tag, the ship will also continuously shoot. If a tag is lifted from the screen the space ship will stay at its last position but stop shooting. If this tag is then repositioned on the screen, the space ship created by the tag will immediately jump to the new tag position. If a ship's tag is absent for too long, the ship is destroyed just as if it was hit by an object, except the point loss.

The game also includes boss monsters, which is visualised with a big object appearing at a random location which immediately starts firing dangerous objects. The boss appears when specific scores are exceeded and only one boss can be active at one time. If a ship is hit by any of the objects fired by the boss, it will immediately be destroyed, and also lose all points and multipliers. To prevent players from farming objects fired by the boss and encourage them to kill the boss, the multiplier is not used while the boss is alive. As compensation for this, the boss is worth a large amount of points and multiplier when destroyed.

Controls:

Tag rotation	Tag position

Patterns:

Power-ups	Low learning curve	Supports late arriving players.
Random	Mean play time 3 min	Rounds depend on previous rounds
Cooperative	Boss-monsters	Highly susceptible to the fat arm problem
Highscore	Honour system	High punishment for failure
Real time	Moderately susceptible to other players in the way	

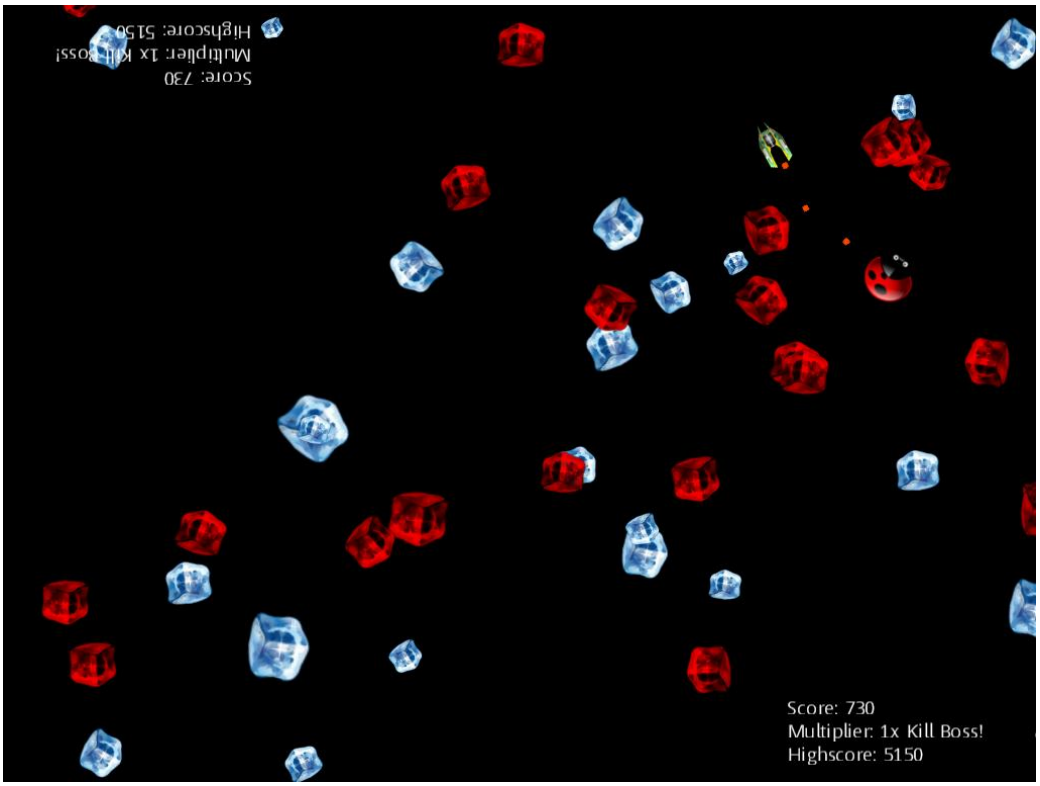

Figure 14. An image of Asteroids. The player controlled space ship is the green ship positioned in the top right corner, below the ship is the boss. The red and blue cubes are asteroids.

Asteroids has the potential to be quite a time sink for any number of players, and when several players are playing they need to communicate and work as a team to reach good scores. The game scales with the number of ships currently active. The multiplier takes longer to increase and bosses and asteroids require more damage before being destroyed if more ships are active.

Since Asteroids greatly suffers from the fat arm problem, playing and testing Asteroids on the Surface Simulator is quite a bit easier than playing it on the Surface, since the user's arm is not in the way. From this we draw many of the conclusions in 7.1.1 Fat Arm Problem. Even though Asteroids suffers from this problem, the game works well on the Surface, and if the problem could be removed, Asteroids would work even better.

Strengths:

Highscore	Boss monsters
Retro	Infinite number of players

Weaknesses:

Severely subject to fat arm problem

Asteroids Future Work

Power-ups are not in the game right now but the design calls for it, so power-ups is a given. Different weapons and other types of enemies would also be interesting.

6.1.5 Racing

Racing is a single or multiplayer game where the players control a car on a racing track. The object of the game is to get round the track as fast as possible without going of course or hitting other players. Each player controls his or her car by using a tag. Rotating the tag changes the direction that the car is heading in and moving the tag left or right accelerates or decelerates the car. Players enter the game by placing a, previously unused, tag on the Surface screen. The game changes track when any player has made a certain number of laps.

Controls:

Tag rotation	Tag position	Virtual analogue stick

Patterns:

Winnable	Mean play time 3 min	Moderate failure punishment
Highscore	Honour system Levels	Supports late arriving players
Real-time	“Game-over”-able,	Low learning curve
Potentially high number of objects hidden by players		

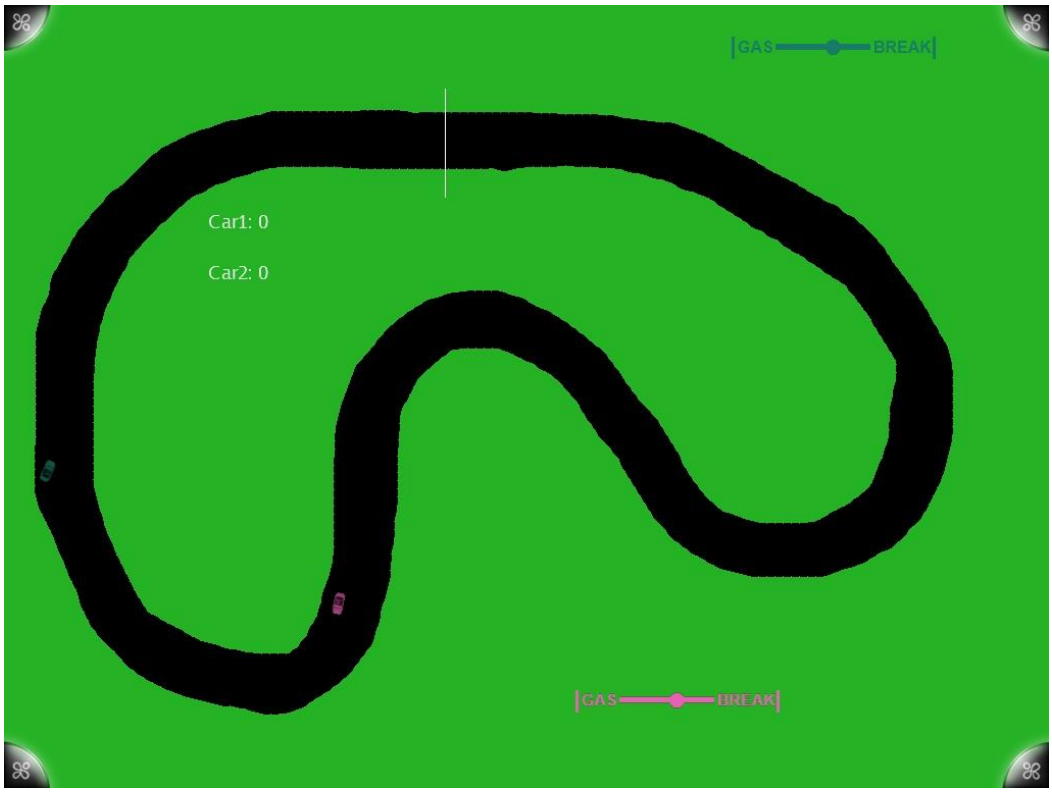

Figure 15. An image of Racing. Two cars are currently racing, one green and one purple. The lines are speed regulators for the cars, used with tags.

Racing supports multiple players, and it is possible for new players to jump right in. It has a rather simple gameplay, getting round a track as fast as possible. Despite being so simple, it has received quite a lot of attention, specifically to the steering and user feedback parts. The steering part has gone through several variations during the course of its lifecycle. The problem always being how to communicate the position of the steering wheel to the player. This was solved by using an absolute rotation system, so that the direction of the tag is the same as the direction of the car. All in all, racing works quite well on the Surface.

Strengths:

Infinite number of players	Players can drop in at any time

Weaknesses:

Some susceptibility to the fat arm problem	Lacks feedback
Can become quite crowded when all players use the non-track parts of the layout	

Racing Future Work
Making the game keep track of laps, lap times, winners and prevent players from cheating by taking shortcuts. More tracks and different cars are other thing that comes to mind. Other than that, ghost cars and track features, such as oil slicks and road spikes would be interesting.

6.1.6 Artillery

In Artillery two players receives one home planet each having the same starting population, and the goal of the game is to empty the population on the other player's planet. This is done by launching missiles at each other. A missile has a number of properties; power, which is the missile's starting speed, angle, which is the direction the missile will be launched in and weight, which is how much population it will destroy from the planet hit.

There are some obstacles in the way of the missiles; a number of orbs are floating between the planets, hindering the missiles from reaching their goals, both with their attracting force, changing the missile's trajectory and by being obstacles hindering the missile's path. The heavier a missile is, the more its trajectory is changed by orbs, but it also destroys more population when it hits. All missiles costs some population to launch, regulated by the starting power of the missile, use of more power in the missile launch results in a greater population cost. There is also a risk missiles circulates a planet and hits the player's own planet through the slingshot effect.

The players can control their missile launcher in two ways, either by dragging an arrow outwards from their planet or by rotating tags in specific areas of the screen. Players

like to control their missile in different ways. The players take turns in launching one missile each, and when one player's population reaches zero, the other player receives a point, the game field resets and a new round begins. For each round, one more planet is added to the game field. All orb positions except the player's home planets are randomised.

Controls:

Tag rotation	Tap	Gestures

Patterns:

Winnable	Medium snowball	Medium failure punishment
Levels	Low learning curve	Low susceptibility to the fat arm
Honour system	Mean play time 4 min	Low susceptibility to other players
Non-player helper	Rounds depend on previous rounds	

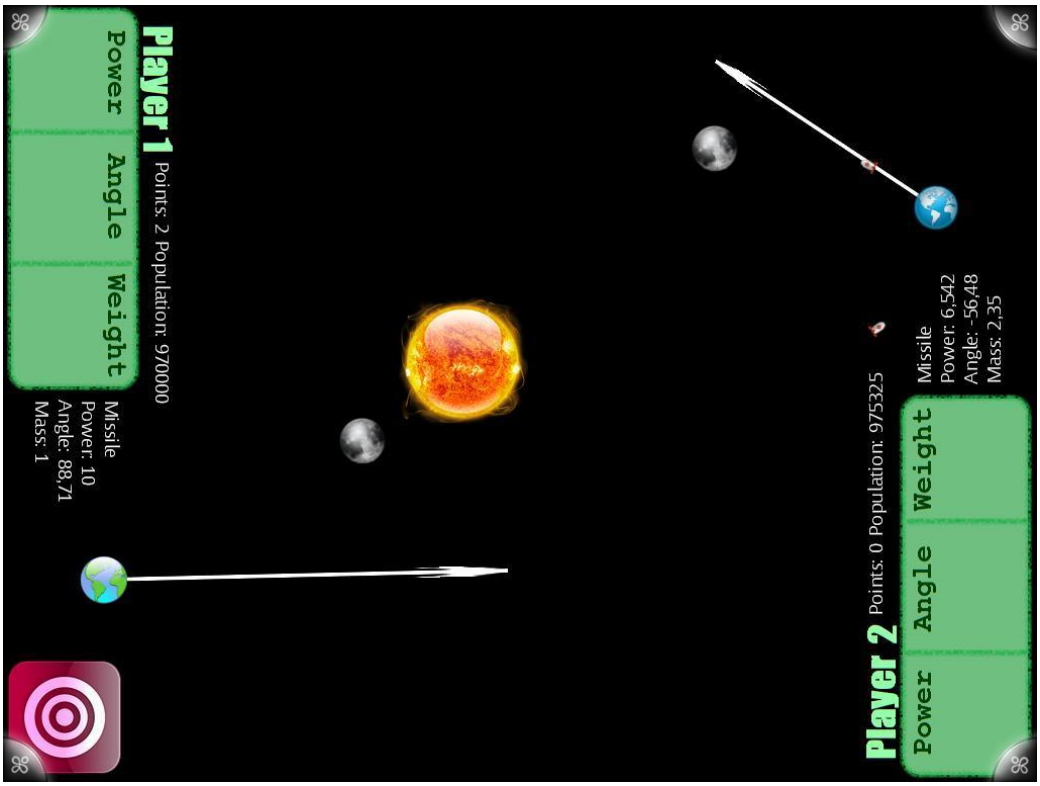

Figure 16. An image of Artillery. The player planets are the ones with arrows attached to them, a missile is fired by tapping the pink button. The sun and the two moons are orbs hindering the missiles.

When players first found out about the planet adding feature they directly started to build walls around their planets, but to their displeasure, it turns out that building this wall has more downsides than upsides for the player building it. For instance it is harder to go around the wall when it is close than when it is far away, and the extra weight nearby a player's planet makes missiles prone to taking that way. Planet positioning is a

small way of testing building mechanic on the Surface and it turned out to work quite well.

Strengths:

Addictive	Realistic physics

Weaknesses:

Players can by mistake hit themselves	No visual feedback on round ends.

Artillery Future Work

Implementing moving planets would be quite fun. Other than that, one could have objects such as spaceships that move across the board and can shoot and/or be shot down. And make sure that the game gives more visual feedback to the users.

6.1.7 Node Battle

Node Battle is a multiplayer game where two players or teams battle against each other, just like in Defence. The game screen consists of a number of planets, each having a number of troops, visualised by the size of the planet. The two teams starts with one planet each, and shall after this conqueror all other planets. Conquering planets is done by dragging a line from a currently owned planet to a, by that team, currently not owned planet. Troops then flow along the lines to either reinforce friendly planets or conquer unfriendly ones. The team who succeeds in capturing all of the opponent's planets is declared winner.

Controls:

Drag

Patterns:

Winnable	Randomness	Medium failure punishment
Levels	Low learning curve	Mean play time 3-5 min
Real-time	Fast snowball	Honour system
Moderately susceptible to other players in the way		Non-player helper
Medium susceptibility to the fat arm		"Game over"-able

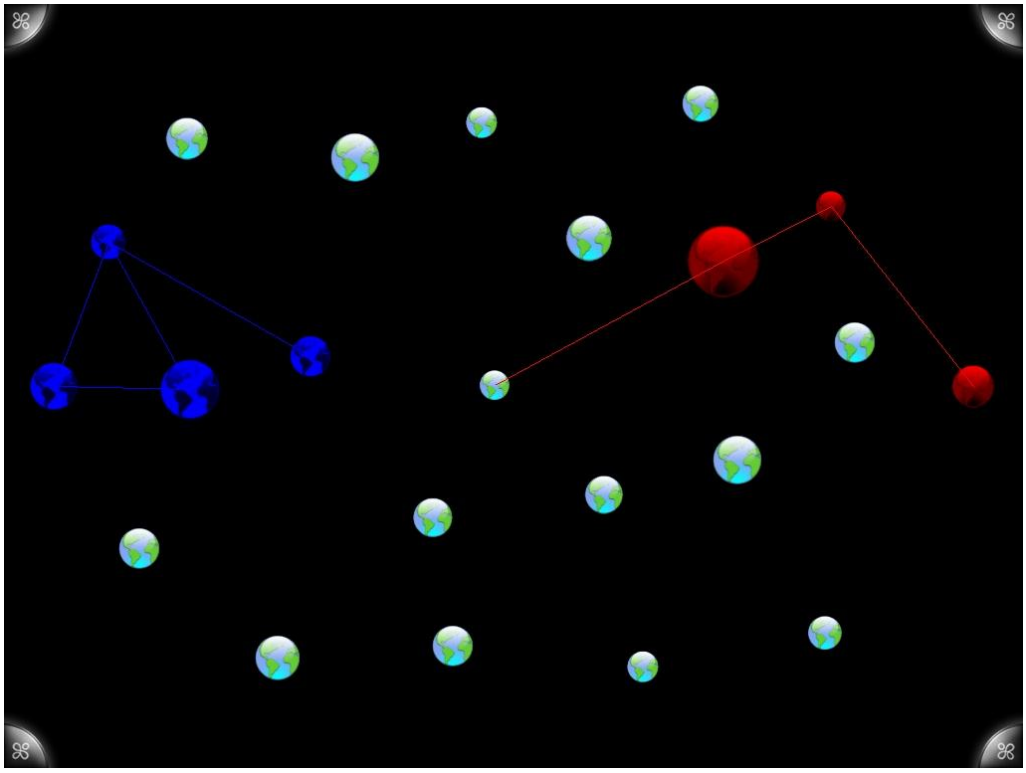

Figure 17. An image of Node Battle. The leftmost and rightmost planets are starting planets, uncoloured planets are not yet overtaken by a player. Lines are troop movements and the sizes of the planets represents the number of troops at the planet.

Node battle can easily be modified to support more than two teams. If the players play the game by only interacting with the Surface, the player who makes the first mistake loses unless the other player makes an even bigger mistake. However, players that interact with each other, in the form of severing enemy node connections and defending the player's own connections, will have a much more even and difficult game. One could also have "traitors" or "saboteurs", non-players that help one side, followed by switching sides, or torpedo the efforts of one or both players. This variant of the game leads to some quite intense gaming but benefits from players that know each other, to avoid unnecessary rage. It was quite surprising to discover this part of the gameplay since it was not intentionally put in the game, but it was nonetheless quite pleasing to see it emerge. This aggressive and physical type of gameplay fits quite nicely on the Surface and other types of large touch interfaces stable enough handle the multiple players. Because of this, Node Battle was a useful experience and works quite well on the Surface, although it needs more development to really shine.

Strengths:

Competition	Quite intense gameplay
Easy to grasp	Outside help/torpedo effort

Weaknesses:

Snowball effect	Requires players to know each other well to play

Node Battle Future Work
An interface that is easier to learn and gives more visual feedback to the user. Also, the amount of troops in the nodes could be used to regulate how far the user is allowed to draw connections to other nodes. An ability to add additional teams in the beginning of the game and a way to reset the game after it is done might also be appreciated.

6.2 Human Surface Interaction

When we first laid eyes on the Surface we thought that it is quite impressive, and in our experience, all first time users find it interesting, impressive and a little strange. These were also our own first thoughts when starting this project. The opinions might be a little biased though, since most of the test subjects have been people interested in technology and computers.

Playing games on the Microsoft Surface clearly comes with a number of problems. The complications found were far more to the number than the advantages, and some of them are quite irritating.

6.2.1 Multi-touch

The biggest advantage of the Surface is its ability to cope with a multitude of input, over fifty fingers and tags, at the same time, and also having the physical space required to do so. In a blog-post from December 16:th, 2007 Robert Levy states that the benchmark for how many inputs the machine has to handle is from a scenario where four players use ten fingers and three game pieces each on the Surface [56]. He also states that it is because of games the Surface can handle this many inputs, so the machine is clearly built with playing multiplayer games in mind. Physically, six persons should have room to stand around the table without hustle playing at the same time. So the Surface has potential to be a great multiplayer platform.

Also, one part we decided not to approach in this thesis is computer aided games. For the sake of fairness, it should be mentioned that there exists some rather impressive projects developed or under development in this department. Games such as Vectorform's The Settlers and the Surfacescapes project, mentioned in 2 Background, have shown that the Surface can handle computer aided games very well.

6.2.2 Height

The Microsoft Surface is set at a rather annoying height, too low for most full grown persons to use with comfort, and hindering extended use. Its lack of space below the screen also induces problems in the decision of where to put ones legs while using the Surface.

6.2.3 Occlusion

While testing many of the prototypes, a problem unknown to the group before the project began was discovered. We named the problem “the fat arm problem” after its close relativity to “the fat finger problem” discussed by H.Benko and D.Wigdor [44]. The diferences between them is that the amount of screen space covered is greater and what body part does the covering.

The fat arm problem arises when a player touches the screen with a finger or a tag and his or her arm covers the line of sight to a part to the screen. In many of the prototypes, especially in Asteroids, this becomes a big problem and the player will not notice objects flying towards the player controlled character.

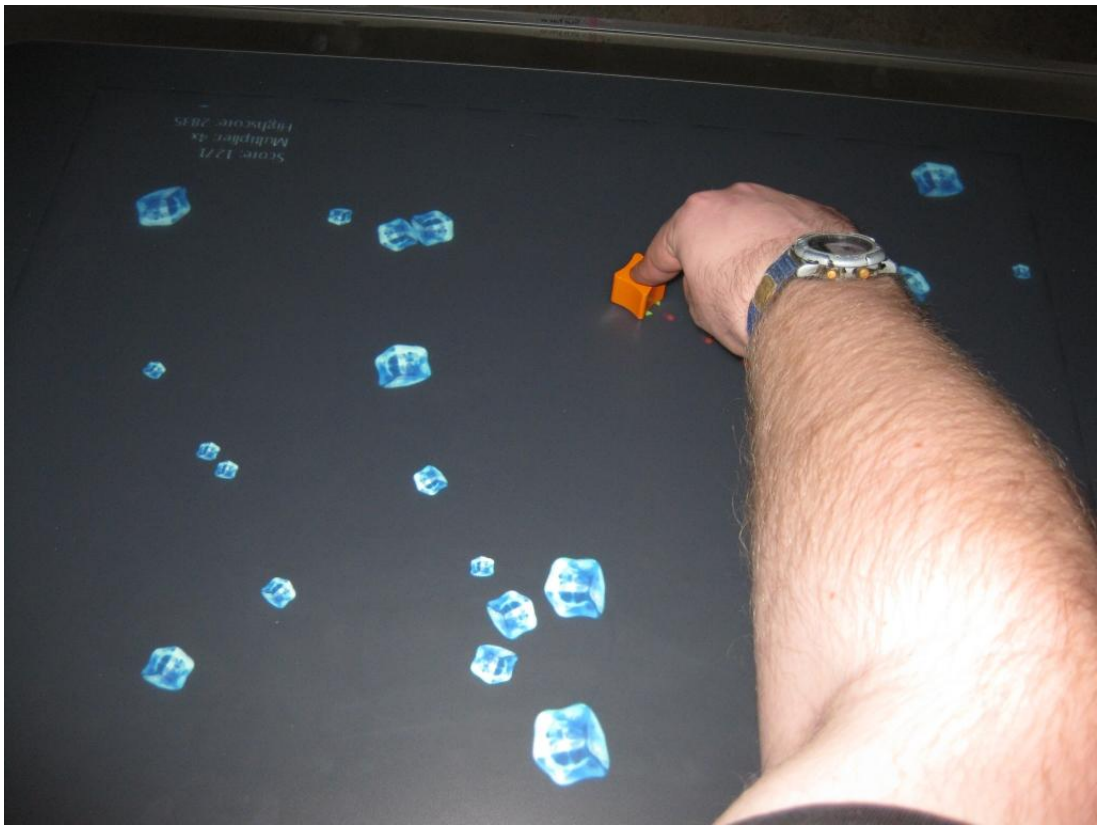

Figure 18. An image of one of the developer’s arms. Note the large part of the screen occluded.

The image ovan shows the player’s view in the Asteroids game and the arm clearly covers a big chunk of the screen. A solution could be to have an offset distance between the tag position and the player character position. But new problems arise such as incapability to position the character close to the edge and the position might be harder to predict.

A solution to the fat arm problem is hard to find since arms must be used by all players, and replacing the arm with for instance a stick removes many of the reasons of using touch screens. Another way the problem could be solved is by moving all important information to a place where the players do not have their arms. The problem then

becomes “were exactly is that?”. On a Surface with multiple users at different positions, the designer can either box the interaction area to a specified part of the Surface, i.e. a virtual gamepad, accept that the information might be covered or design so that fast reactions to events are not needed like in most turn based games.

6.2.4 Input speed

The Surface’s Infrared cameras, and the software analysing the images, have problems handling objects moving at high speed across the screen. This is most notable with fingers since the system might interpret fast moving fingers as a new contact and believe the finger was lifted and pressed down again. With byte tags, it is less of a problem, since the same tags always has the same unique value. However, for the larger and more complex identity tags, Microsoft recommends that they are used stationary or near stationary [57]. More about how the tags work can be found in chapter 2.1 Surface Technology.

6.2.5 Light Sensitivity

Since the input is collected through cameras, the input is quite sensitive to uneven lightning in the room. For example, if the Surface is placed close to a window which partially lights up the screen, it might interpret moving curtains or the sun’s movement during the day as input. If the Surface is placed in a well-lit environment, it has a tendency to interpret the users’ shadows as input. A solution to this, in terms of minimizing false input, is to use the Surface in a darker room, and during daytime have some distance to any uncurtained window. However, this fits very poorly with the assumption that the Surface is primarily used in public spaces which tend to be well lit.

This problem could be exploited by using a strong light source such as a flashlight or laser pointers pointing it towards the screen, fooling the Surface into believing that it is valid input. The possibilities of this have not been explored in the project though.

6.2.6 Resolution

The Surface’s fixed resolution of 1024 by 768 pixels is rather low considering the size of the screen. This leads to a hard limit on the amount of content a designer can fit onto the screen without cluttering. It also prevents developers from showing advanced graphics when the pixels are quite big. The ideal solution would be to allow the designer to set the resolution so it fits his or her application. This low resolution also limits use of split screen games where two players have one half of the screen each.

Compared to the Apple iPad, which has exactly the same screen resolution as the Surface but with a screen measuring 9.7 inches [60], the Surface resolution appears quite low with its much bigger screen measuring 30 inches.

6.2.7 User Feedback

One of the largest problems turned out to be how to give a user direct feedback on their actions. On a keyboard or a mouse, the user's fingers receive immediate feedback from pressed buttons and the user knows that he or she has pressed a button. On the Surface, the only haptic feedback is the feel of stationary plastic on the user's fingertips. So some mechanic that visually gave feedback about actions, preferably one that was not covered by various body parts. The prototypes vary on how well they accomplish this. For example, in Asteroids it was quite hard to see that the player's ship had been destroyed until an explosion was added with debris flying everywhere.

6.3 General Game Mechanics

The control schemes used by the games differ only slightly between certain games. For example, both Airport and Node Battle uses a basic drag control. The main difference in control between these games is that Airport is played as a cooperative game while Node Battle is a more versus type of game.

The tags and much of their use were primarily used in the same way as fingers, but making them uniquely identifiable by the Surface. This allowed in-game objects to be bound to specific tags, allowing things like jumping across the field in Asteroids. In many ways, tags therefore came to represent a game object, be they ship or car or knob, and allowed players to interact with that element in a more concrete way. However, tags suffer from the same problems that fingers do and therefore tags turned out to be interesting but not ground breaking in these cases.

In Node Battle, the player is given no feedback before an action is completed and accepted as input, confusing users about what to do and what is acceptable input. Platform suffers the same problem where a user might have problems knowing which character he or she currently is controlling and even if any character is controlled at all.

For game mechanics, the biggest result was that speed kills. Unexpected game objects appearing and being occluded from sight by player's arms are a problem. The faster these objects move, the higher the probability that players will not notice and react in time. This leads to player frustration and may cause a player to walk away in anger.

The Surface input speed also causes some interesting and unexpected effects on game mechanics. For example, if a player moves his or her fingers rapidly back and forth over the Surface, the machine interprets the movement as several fingers more than there actually is. The result in the games vary depending on the control scheme of the game, from lost contacts in all games that rely on swipe and a lot of extra accepted input in games relying on tap.

Game mechanics other than those mentioned were found to have a negligible effect on how fun a game is.

6.3.1 "New" Game Design Patterns

These are the patterns that were defined internally during the course of the project. Some, like Power-Ups / New weapons and Highscore, are modified variants on other patterns discussed in Patterns in Game Design and Gameplay Design Patterns. Others, such as Touch and Objects hidden by players, are defined internally.

***Power-Ups / New weapons**:* The player(s) can acquire increases in power or abilities and/or new weapons
Winnable: The player(s) can win the game.
Interruptibility: The player(s) can interrupt the game without consequences.
Punishment for failure: How hard the player is punished if he or she fails, high meaning the game is over and low meaning only small loss of score or such.
Reward for actions: How much reward the player receives for using the game, also called juiciness.
"Surfaceable": If it is possible to directly copy the game to the Microsoft Surface.
Touch usage: How the game uses touch input.
Saves game state or level: If the game saves progress between play sessions.
Unlockables: If player(s) can unlock bonus features in the game.
Highscore: If the game has and remembers highscore between sessions.
Player count: The number of players supported by the game.
Multiple Game Modes: If the game has multiple game modes.
Mean play time for one round (in minutes): How much time each round of the game takes, one round being a natural breakpoint in the game.
Consecutive rounds depend on previous rounds (how): If $f(n) = f(n-1) + 1$
Objects hidden by players: How much the game is subject to the "fat arm" problem.
Other players in the way: How much players are in the, physical, way of each other.
"Game Over"-able: If there is an end-state were play is no longer possible.
Honour system in place: If the game checks for any form of cheating, outside of the game.
Snowball: How much early advantage translates into late-game power.

6.3.2 Game Design Patterns Analysis Result

The result of the analysis showed several things. First of all, several of the patterns had no discernible effect upon whether the game fit the Surface or not or if the game is fun or not. *Highscore*, *Power-Ups*, *Winnable*, *Boss Monsters*, *Consecutive rounds depend on previous rounds (how)*, *Chance for Analysis Paralysis*, *Honour system in place*, *Levels*, *Snowball*, *"game over"-able* and *Punishment for failure* showed to have a negligible impact on how fit a game was for the Surface and personal taste played a much larger part, than these patterns, in determining if a game was fun or not.

Other patterns had a more indirect affect. *Randomness*, *Touch usage*, *Player count*, *Co-op*, *Non-Player Help / IRL player helper*, *Learning curve* and M*ean play time for one round (in min)* all had indirect effect on how well a game fits the Surface. Either by decreasing or increasing the impact of other patterns or in combination to create an impact. The exact way how will be discussed in chapter 7.4.1.

The last and smallest group contained the patterns which had direct impact on game mechanics fit for the Surface. *Objects hidden by players*, O*ther players in the way*, *Real-time* and *Late arriving players* all showed a direct impact on a game mechanics fit for the Surface and will also be discussed in chapter 7.4.1.

7 Discussion

In this chapter we will discuss thoughts about the interactions with the Surface, as well as associated issues that arise when developing for the Surface. Furthermore, there are sections about the developed games, methods used and the quirks of the games as well as future work.

Overall, we are quite satisfied with all the prototypes, the result and our way of working. Of course, there is always room for improvement but if we were to do this once again we would do it in much the same way.

7.1 Surface Interaction

Interacting with a Microsoft Surface is at first a rather cool and interesting experience. Touching and swiping your fingers and hands over the screen generates a powerful feeling. However, after some time playing games, a couple of negative properties appear. But as most other problems, one learns how to mitigate and work around them. The solution, such as the ones mentioned in the fat arm section below, is often contra productive to the desired gameplay though.

7.1.1 Fat Arm Problem

The “fat arm problem” which arises when using a Surface, as noted in 6.2.3 Occlusion, or any other bigger touch screen, severely limits which kind of games are feasible to play on a tabletop computer. If the player’s extremities cover large parts of the game area from view and at the same time the game relies on fast reaction to events, players may experience severe frustration when important objects are hidden from view. This since failure to react to events can lead to anything from a lower score to complete game over. If the players try to mitigate the problem by only interacting in a small area in front of them, they limit their own “play-area”, and could just as well use a keyboard, and make themselves susceptible to objects appearing from the border of the screen.

Figure 19. An image of gameplay from the game League of Legends. The map in the bottom right part of the screen would be covered by someone's arm if played with touch.

The game League of Legends [58] is a good example of the problems that may occur by the "fat arm problem". In League of Legends, two teams battle against each other in a virtual environment and the objective is to destroy the other team's main building. A way to make this task easier is to kill the other team's player characters when they try to hinder this. One of the more effective ways to accomplice the latter is to set up ambushes. To survive an ambush, a player must either react quickly enough to get away or to know where the enemy are, thus avoiding the situation. If a player has a large part of the screen covered by his or her arm, it may take several seconds longer to detect and react to the ambush, which increases the possibility of death. A good way to survive ambushes is to keep track of all other player's positions by using the in-game mini-map that is situated in the lower right corner of the screen. For a player using his or her right arm to control the character, that arm will cover the mini-map to a large degree and thus hinder the players overview of the game.

Among the prototypes, Asteroids suffers from the fat arm problem the most, with asteroids covered by player hands and arms being the prime cause behind destroyed ships. Airport, Platform, Racing and Node battle experience this problem as well but the players can generally handle it without too much problems by lifting their hands a little. Artillery does not require fast actions and in Defend, the player's shots move in a straight line from the player's fingertip, so they are never covered by the player's arms. This shows that one solution to this problem is to avoid things that requires fast reaction to occurring events or make sure that those events only happens where there is a assurance that it will be seen by the players.

7.1.2 Virtual Gamepad

A virtual gamepad is when the game projects the controls, often in the form of a directional pad and some buttons, onto the screen. The players then interact with the gamepad in the same way that they would do with a physical gamepad, often using thumbs. This control pattern is quite common on the iPhone and iPad. The benefit of this pattern is that the designer gains more input-states and can have the same interactions, i.e. finger touches screen, to produce different result depending on context. The developer also has full control of where the player's fingers are to be positioned and can design the interface accordingly.

The problem with this is that the interface, and the player's fingers, covers an area of the screen that cannot be used to display important information, noted in 6.2.3 Occlusion. That coupled with the low resolution noted in 6.2.6 Resolution leads to the problem that a game designer has a rather small screen area to display the actual game. The buttons also does not give any haptic feedback, see 6.2.7 User Feedback, and thus the player might have a hard time determining where origo, i.e. the position that is used as origin when determining what input the user is giving, is and how far it is to origo. Another problem is the fact that the point of using a large touch screen is disrespected if all controls are moved to a small area alongside the edge of the screen.

Of the prototypes, Artillery and Racing are the ones closest to using a virtual gamepad of some sort. In racing, players control the throttle by moving a tag relative to the position where it first had contact with the Surface and in Artillery, tags can be used as control nobs and to fire shots, the player has to press a big onscreen button.

7.1.3 Physical Multiperson Play

One major difference between playing multi-player games with mouse and keyboard with a gamepad or on a shared surface is the ability players have to mess up for each other. On a keyboard, players often sit far away from each other with separate screens, which almost completely removes the ability to disturb other players. All players also have both hands occupied on their own mouse and keyboard and thus, it is a great sacrifice to get to, and disturb, other players. While playing video games, players are often closer to each other and pushing other players work but is often disliked and discouraged. For board games, interfering with other players and their game pieces is very easy but strongly frowned upon. On a tabletop it is not tabooed to hit, shove and in other ways affect other players so games can take advantage of this "feature". And since the Surface can handle several players at once, as mentioned in 6.2.1 Multi-touch, we feel that they should use this "feature", although the Surface's height, noted in 6.2.2 Height, might dampen the fun.

7.1.4 Tags

Tags are an interesting game mechanic but they should not be over used. Tags tend to over-complicate things when different tags have different functions and it might even get crowded with them. All tags also need to be present at the Surface when playing and if different tags have different functions and one tag is missing, the game might become unplayable. Therefore we have decided not to take advantage of a high number of unique tags in any of our games.

Tags add a new layer of abstraction between the user and the application. Tags can be used as extra lives, a resource or even buildings in the game; this is a somewhat new feature though, which users might not have encountered before in video games. Asteroids uses tags as extra ships in the game, if a ship is destroyed, the tag which was connected to the destroyed ship can no longer be used to spawn a new ship. In artillery, a player can build new planets to the game field by using tags. But just as in Asteroids, only one planet can be built for each unique tag value.

Using a tag for rotation with a non-flat object is different compared to the standard rotation using two fingers touching the screen. Rotating objects on the screen with two fingers is intuitive in the way that it works like rotating a real flat object on any surface, like a paper, and the response is almost exactly the same as in the real world. Rotating a tag is intuitive in a different way where the user screws the tag to rotate the object instead, which can also be directly applicable to the real world.

The prototypes developed only supports byte tags as we see no reason why any game would need to use identity tags for the gameplay. As mentioned in chapter 2 Background, there can be 256 unique values on byte tags, more than enough for any gameplay mechanic. In comparison, an identity tag contains 8 bytes, so it could be used as user identification, replacing username and password allowing users to fetch their game progress from a server, which would remove the need of a virtual keyboard.

7.2 Surface Development

The Microsoft Surface was quite fit for games from the beginning. Since it is running Windows Vista and uses the XNA and .Net frameworks, both freely available from Microsoft's homepage and designed to ease development, it is very easy to get going and programming for a Surface is very much like programming for a normal computer. It is, however, not quite as easy to finish and get somewhere that is actually fun. While game programming on the Surface has all the normal pitfalls of software and game development it also has some rather interesting problems of its own, and most of them arise from its user interaction.

An interesting aspect of software development for a multi touch device is the multiple targeting points used and this was the first time stumbling upon this area for both

developers. When programming for a normal computer, it is always known where the mouse pointer is and when a mouse button is pressed. With multi touch, fingers do not have unique identities, so when a new contact is detected one does not know which finger it is and if several users are using the Surface, who the finger belongs to. There can also be large amount of contacts at the same time, and all of them have to be handled simultaneously. The identity problem was solved in Asteroids and in Racing by the use of tags since a tag has a unique identifier. A tag is directly paired with a player character and the finger identity problem is avoided. In fact, one of the more interesting features of Asteroids; that a ship can jump over the whole screen instantly, came from the fact that when the user lifts a tag from the Surface and puts it somewhere else on the Surface, it is intercepted as instantaneous by the machine.

Another thing to think of when using tags is what piece they are fastened to. If the object is larger than the tag the Surface has a tendency to interpret it as more contacts. Users also have to understand that they can manipulate this physical object to make things happen on the screen.

7.3 Method discussion

Overall the methods worked well. Would the result have been better if we had used other methods? Maybe, but overall the used methods took us to a rather pleasant place. Since we were familiar with the methods before the start of the project, we knew when and how to use them and saved a lot of time by not having to learn something new. While it is always nice to learn new things, we wanted to spend as much time as possible implementing, testing and analysing.

Brainstorming got us 74 ideas over several sessions, with the occasional “hat method” thrown into the mix to break mental blocks. While the output could be higher, the spread of ideas was nice and after culling away most based on our limitations, a paper prototyping session showed what was actually doable and what was not. Paper prototyping was discovered to share similarities with the Surface way of interaction. In both, the user uses his or her fingers to move stuff around.

Our programming variant also worked well, allowing speedy parallel development, and none of the headaches that come from two developers working on the same code. Together with quick and dirty prototyping, it allowed us to get the prototypes testable in a very short amount of time. Testing followed the same pattern while analysing was less clear-cut. This was mostly because of our unfamiliarity with thinking in terms of game design patterns. It still provided some nice results, it just took more time to get something useful out of it.

7.4 Game Prototype Reflection

The ideas behind the prototypes were first decided upon to train us in Surface programming and later designed to explore on of the interaction niches left yet untouched. While this method may not have been the most efficient, it fit the two, initially clueless, designers and allowed empirical testing of the interaction. The prototypes follow the quick and dirty pattern of design, and were thus designed and implemented in a few days' time each and often two or more in parallel. This worked well and allowed us to quickly get results, and provided grounds for further testing.

The hardest part was the fine-tuning of the prototypes in relation to the input given by the Surface and to balance this so that a user could accurately execute the actions intended. This proved somewhat tricky and several of the prototypes still has inaccuracies in input. These have not been fixed for the most part because it was not deemed a significant enough obstacle to invalidate the results. The finger part of the problem often stem from the fat finger problem, i.e. that the player cannot see what he or she is touching, combined with the fact that humans slightly change the touch between each interaction leading to that the centre point of the finger being slightly different between each touch. The clearest example of this exists in Airport were if two planes are close together and the player attempts to separate them by selecting one and leading it away from the other plane, the wrong plane is selected and given a path into the other plane instead of away from it.

Fine-tuning tags have similar problems when it comes to rotation. For example, Racing tested different types of steering, one of which used relativistic steering. The problem was: How much should the wheels turn when the player turns the tag X degrees and how to communicate this to the player. This was not helped by the friction between tag and surface having a high static component and a low kinetic one gives a tendency to exaggerate results. This particular problem was solved by implementing absolute steering were the direction of the tag was equal to that of the car.

The prototypes have all run afoul of at least one of the limitations of touch or of the Surface. And in all cases we have managed to either mitigate the problem or find a way around it. However, many of the found solutions to touch limitations and problems require that the user is alone, or that the pace of the game is so slow the limitations do not matter. Since some styles of gameplay relies on high speed, the second solution is not a silver bullet and the first one do not mix very well with our desire to create multiplayer games.

Defence is best described as a "screen masher", i.e. the player presses the screen as fast as possible, and is quite fun in its simplicity. It does not suffer particularly much from any of the mentioned Surface problems and is a good example on how to avoid the fat arm problem. We are pleased with how Defence turned out.

Airport suffers more from the various problems associated with the Surface, but is still quite fun after a rather slow start. As a project primary decided upon to teach Surface programming, it did its job very well and served as a good lesson for the rest of the games. All in all, we are pleased with how Airport turned out.

Asteroids was the first attempt to use tags and it demonstrate, together with Racing and Artillery, quite nicely what can be done with tags. It also demonstrated the fat arm problem to us. Despite this problem, we are quite pleased with how Asteroids turned out and if we could fix its flaws, we would be very pleased with ourselves.

Platform was our first attempt to not copy an existing game. The idea was to get participants to chase each other around the table. This idea somewhat fell apart when it was discovered that players preferred, for reasons probably regarding the Surfaces height, to be stationary while throwing bombs at each other. It was then abandoned to allow focus on other games and at the time of writing, we have mixed feelings towards Platform. On the one hand, it is quite fun to throw bombs at targets, not quite as fun to move that target around the board.

Racing was the next attempt to use tags as controllers, this time as a wheel. It tested different steering variants during development and displayed both how hard it can be for a player to discern the direction of a tag and a possible way to get around the fat arm problem. It could use added features like a dessert could use rain, yet we are pleased with how it turned out.

Artillery is another attempt to design something which is not a straight copy, although there exist many games with artillery as its main theme. It single-handily showed that turn-based games are almost completely immune to problems regarding fat arms. It is also a rather addicting game and we are thus very pleased with it.

Node Battle is the runt of the litter and has received the least amount of time. It was also on its way to become a complete failure, regarding explored things and fun to play, until it decided to show us what we have described in chapter 7.1.3 Physical Multiperson Play. If this feature was somehow removed, Node Battle would go back to being an extreme snowball game where the first person to make a mistake loses. After the testing session that showed Physical Multiperson Play, we refined the Node Battle game somewhat, but we lacked the time to do very much. Because of this, it needs features like a politician needs votes yet we are pleased with the result.

7.4.1 Game Design Patterns Reflection

Game design patterns allowed us a way to analyse games and come up with new ideas. However, with the power of hindsight we can see that many of the patterns had very little effect on a game regarding game mechanics fit for the Surface. For example, *highscore* affects replayability and invite players to try and beat the currently highest

score. But it affects games to different platforms equally and does not change a game's fitness for the Surface in any particular way.

Other patterns, such as *Randomness*, proved to have an indirect effect when in combination with other patterns, such as the ones regarding occlusion. If game objects appear at random, the player can do nothing to negate the bad effects from the fat arm problem. On the other hand, if game objects appear in a defined pattern, players can learn and predict where something will happen and take steps beforehand.

Yet other patterns, such as *Objects hidden by players*, have a very large effect on how suitable a game is for the Surface, as has been discussed in the parts regarding the fat arm problem.

The effects of the different patterns that were deemed to have an indirect effect:

- *Randomness*: Showed to have an indirect effect when in combination with other patterns, such as the ones regarding occlusion. If game objects appear at random, the player can do nothing to negate the bad effects from the fat arm problem. On the other hand, if game objects appear in a defined pattern, players can learn and predict where something will happen and take steps beforehand.
- *Touch usage*: The adversity of several of the problems mentioned in chapter 6.2 depends upon how the game is controlled. Simply touching the table showed to have less severe variants of the problems then, for example, moving a tag around.
- *Player count*: With less supported players came less occlusion and clutter, less of a problem if each player had an interface as small as possible. Also more fun with more players.
- *Co-op*: Raises the level of fun in a game, but too dependent on other things, like available players, to be deemed a direct effector.
- *Non-Player Help / IRL player helper*: If supported then the same as *Player count* on the occlusion parts and fun parts but without the clutter part.
- *Learning curve*: All the games where designed to have as low a curve as possible, intended to increase the fun and accessibility of the game. Other things had more of an effect.
- *Mean play time for one round (in min)*: All our games designed to have between three and five minutes of play. Since there is no comparison available, we cannot say how much of an effect a longer game-time would have. We believe that a longer game time would have a negative effect on fun and spine health.

The effects of the different patterns that were deemed to have a direct effect:

- *Objects hidden by players*: The primer cause of player frustration, since occluded game objects are a big problem, as discussed in chapter 6.2. The more affected a game is, the less fun and suitable for the Surface it is.
- *Other players in the way*: Less severe than the one above but still a source of player frustration. The more affected a game is, the less fun and suitable for the Surface it is.
- *Real-time*: A pattern that can single-handily negate the two patterns above, since a turn- based game does not suffer at all from those two. Also, if a game is in real-time but progressing slowly, it lessens the impact from those two patterns.

- *Late arriving players*: Adds a lot of fun, especially for the late player. Also adds a new layer of strategy since existing players has to take new players into account.

7.4.2 Discussion of New Game Design Patterns

A specific game mechanic can be defined in many ways and it is thus questionable how new some of the patterns mentioned in chapter 6.3.1 "New" Game Patterns, are. For example, we have defined a pattern called *Winnable*, while Patterns in Game Design [49] has a pattern called Unwinnable. Both patterns describe the same thing but from different point of views. The reason we kept our name on this game mechanic is because we defined it before we knew it was already defined with a different name, and we saw no reason to change it since it is only a difference in semantics.

Other patterns are variations upon patterns from other sources. For example, Game Design Patterns [50] lists Highscore Lists as a pattern. Our *Highscore* only stores one position. This specific change were decided upon when we discovered that the tested iPad games tended to have only one highscore, not a list of them.

Then we have the patterns defined entirely internally. They have come from our interaction with the iPad games tested, books and papers read and the Surface games we developed. This group of patterns include ones such as *Objects hidden by player* and *Other players in the way*, both of which turned out to be very important for the result, as is discussed in the chapter ovan. Had we not been searching for and defining new patterns, the conclusion of this thesis would be very different.

7.5 Control Types

The control types mentioned in Theory in 3.2 are built for normal computer use, but some of them are also applicable for games. For instance the flick action, which normally works as keyboard shortcuts, could be used for many games where shortcuts are widely used. Rotation on the other hand is harder because it is often hard to get accurate angles and a hand has a hard time rotating more than half a lap. In Asteroids and Racing we wanted to quickly be able to rotate the player-controlled character several laps, with two finger rotation this would be impossible and therefore we picked the tag rotation instead.

Many games which would have tested different control types were culled from our game ideas list due to their requirement of a mouse. Also their lack of multiplayer options culled some of them. For instance, real time strategy-games are often single player games. If we would have made any prototypes having close similarities to mouse controls, using actions such as double tap, hard press and press and tap, we could have tested touch interactions using virtual mouse in games too. But we did not have time for this and also were more interested in games that did not use a mouse.

Tags could be added to all devices using touch but require that the device is of a certain size to be effective. If the device is small and the tag large, only a small number of tags

fit on the device at the same time. If both device and tag is small, there is room for more tags on the device but each tag may become very hard to handle.

7.6 Future Work

The Microsoft Surface 2.0 is to be released in the near future [59]. With its more advanced hardware, hopefully this will fix many of the problems noted in this report. If Microsoft has made significant changes to the API, then that might warrant a reiteration of our question on the new hardware.

Other work that might be interesting is gestures with one or more fingers and tags and games utilizing several networked Surfaces. Or one could try to port an existing XNA game, preferably something more advanced, onto the machine to try more a traditional type of computer game on a new interface. Or the route we decided to cull away, computer aided games, which already has several interesting on-going projects.

The prototypes developed are all in various states of completeness. It would be quite interesting and fun to finish them. We also realise that the naming of the games are quite unimaginative and renaming them to something more innovative should be considered. Also the graphics and many of the icons used in the games should be updated to a more juicy style. Sound is another feature the prototypes are missing.

Games only utilising single user interaction should also be considered to be developed, games such as League of Legends [58] and/or other Strategy games would be interesting to see how to handle.

8 Conclusion

The objective of this thesis was to explore the possibilities of Microsoft Surface, and define what kind of gameplay suits it. The focus is more on the Surface's way of interacting with the players rather than on graphics and complex gameplay. To achieve this goal, similar systems have been analysed, some prototypes designed, implemented and their feasibility on the Surface tested. To further help our efforts game design patterns have been defined, collected and used to compare games.

The seven quick and dirty prototypes developed, each explore a niche in Surface interaction and together they cover a large part of the available interaction space that the Surface allows and allows us to be confident in our answer. The prototypes have been a useful vessel in exploring what is possible and to map the larger problems and a possible way around them.

8.1 Question answer

The question as stated in chapter 1.3 Question Formulation:

Which types of game mechanisms are well suited for the Microsoft Surface?

Considering the Surface height limitations, games that require long continuous amount of time are out of the question. So are games that require fast reaction to events, due to the fat arm problem. In contrast, slower games and games where game objects move in a very predictable way negate the problem and work well or very well on the Surface. Other game mechanics depend more on design and implementation than on the Surface. Tags give some rather interesting possibilities, most of which lies in controlling game objects, but from a game mechanics point of view, they have not shown to either prohibit or encourage any specific kind. However, it should be noted that we have chosen not to look at computer aided games and this could be an area where tags really shine.

The large size of the Surface's screen allows for a virtual game-pad solution that does not cover that much of the screen. However, if it is supposed to be a multiplayer game and all participants should have their own controls, the available screen space shrinks for each player. While this is easily negated by using minimalistic interface or limiting the number of players, care should be taken by a game designer before following this route. And because of the Surface low resolution compared to the screens size, it is rather hard to get impressive graphics. While graphics is not a game mechanic, it is often used as a selling point. Also, on the one hand, unless several players are cooperating, one player cannot affect that large of an area at once. On the other hand, a game designer could transform this into something fun, requiring several players to cooperate to reach a goal. So, as long as the designer is aware of the Surface's

limitations, and can find a way to design around those limitations, all kinds of gameplay are possible on the Microsoft Surface.

More generally speaking, any interactive tabletop have a hard time avoiding the fat arm problem, but the height problem is very easy to remove. For a multi-touch device it is also rather hard to avoid the occlusion problems arising from user's appendices, but if the devices are portable, a game designer can design a game that require more time to learn and play.

Despite the fact that our question is about game mechanisms, no game mechanism is mentioned in the answer. Instead we state interaction patterns. This is because we found that the game mechanism which fit/does not fit the Surface were dependent on the interaction between the user and the Surface.

8.2 General Conclusions

Apart from tags, there is not much new under the sun. The various commercial touch and multi-touch devices have explored games with various variants of touch input and the Surface add tags and size to the mix. While the size makes some things easier, such as text readability and physical area available for interface, it does not really affect the game mechanics.

Tags however, can affect game mechanics. The tag's unique values allows the Surface to track a specific tag, even if it is lifted from the Surface and put back someplace else. This allows the Surface to track specific users, as long as nobody is cheating, and provide some interesting opportunities. Since they can be put on any object, they allow for game objects that exist outside of the computer. That can be cards that affect a game when placed on the Surface or representations of digital assets that control those assets when placed upon the Surface.

Interaction Design had a much larger impact then the team initially realized and a large part of the answer to the fit/do not fit question came to rely on how players could interact with the Surface. This fact might not have surprised a dedicated interaction designer, but for two game designers it was a strange realization.

The release of Surface 2.0 will hopefully fix some of the more annoying "features" the Surface has, specifically its height and reliance on Windows Vista. This release might invalidate our answer, but we believe that that possibility is quite small.

To conclude, it has been a fun and interesting journey and we have grown as designers and as developers. While we both have designed and developed games in the past, never before so many during so little time and on such different hardware, and it really made us better understand what it takes to develop games for a living. And to answer the question most asked by other programmers when they hear what we are doing:

“How does it differ from a normal PC?”. Answer: The only user input you have is 40 mice.

9 References

[1] Brookhaven History - The First Video Game?
http://www.bnl.gov/bnlweb/history/higinbotham.asp
[Accessed 27:th of May 2011]

[2] Wolf, Mark J.P. (2008). The Video Game Explosion. Greenwood Publishing Group. pp. ISBN10: 031333868X

[3] Wikipedia - Home computer
http://en.wikipedia.org/wiki/Home_computer
[Accessed 27:th of May 2011]

[4] The INQUIRER - Touch screen is the future of computing, says Microsoft
http://www.theinquirer.net/inquirer/news/1051547/touch-screen-future-computing-microsoft
[Accessed 27:th of May 2011]

[5] Johnson, E.A. (1965). "Touch Display - A novel input/output device for computers". Electronics Letters 1 (8): 219–220.

[6] Bill Buxton - Multi-Touch Systems that I Have Known and Loved
http://www.billbuxton.com/multitouchOverview.html
[Accessed 27:th of May 2011]

[7] phpkb v1.5 - Microsoft Surface Computer – Muli Tuch Technology
http://www.knowledgebase-script.com/demo/article-420.html
[Accessed 27:th of May 2011]

[8] Samsung SUR40: Specs, Price and Release Date For Microsoft Surface 2.0
http://pinoytutorial.com/techtorial/samsung-sur40-specs-price-and-release-date-for-microsoft-surface-2/
[Accessed 27:th of May 2011]

[9] ZDNet - CES: New Microsoft Surface to be priced at $7,600
http://www.zdnet.com/blog/microsoft/ces-new-microsoft-surface-to-be-priced-at-7600/8368
[Accessed 27:th of May 2011]

[10] Microsoft - Developing Microsoft Surface Applications on a Separate Workstation
http://msdn.microsoft.com/en-us/library/ee804897%28v=surface.10%29.aspx
[Accessed 27:th of May 2011]

[11] Fast Company - 11 Killer Apps for Microsoft Surface
http://www.fastcompany.com/blog/chris-dannen/techwatch/11-killer-apps-microsoft-surface-videos
[Accessed 27:th of May 2011]

[12] Softpedia - Microsoft Surface Settlers of Catan Blends Analog and Digital Gameplay
http://news.softpedia.com/news/Microsoft-Surface-Settlers-of-Catan-Blends-Analog-and-Digital-Gameplay-150434.shtml [Accessed 27:th of May 2011]

[13] Microsoft Surface Blog - Games Pack available to our customers
http://blogs.msdn.com/b/surface/archive/2009/04/06/games-pack-available-to-our-customers.aspx
[Accessed 27:th of May 2011]

[14] Vectorform - Vectorform App Store
http://apps.vectorform.com/
[Accessed 27:th of May 2011]

[15] Surfacescapes - What is Surfacescapes
https://www.etc.cmu.edu/projects/surfacescapes/index.html
[Accessed 27:th of May 2011]

[16] The Jelly Reef
http://www.thejellyreef.com/
[Accessed 27:th of May 2011]

[17] YouTube - Microsoft Surface in the Seattle Sheraton Hotel
http://www.youtube.com/watch?v=LKKGoysjhhc&feature=player_embedded
[Accessed 27:th of May 2011]

[18] YouTube - Microsoft Surface Application- Barclay's
http://www.youtube.com/watch?v=cBF5BI5H7vs
[Accessed 27:th of May 2011]

[19] YouTube - Microsoft Surface at the Rio in Las Vegas
http://www.youtube.com/watch?v=xeWdy6eCqDc&feature=player_embedded
[Accessed 27:th of May 2011]

[20] D' Technology Weblog - Download Microsoft Surface Games Pack
http://www.ditii.com/2009/04/08/download-microsoft-surface-games-pack/
[Accessed 27:th of May 2011]

[21] YouTube - Walkthrough of The Settlers of Catan on Microsoft Surface
http://www.youtube.com/watch?v=sdUFT01vzyI
[Accessed 28:th of May 2011]

[22] Rovio Mobile - Angry Birds Walkthrough Videos
http://www.rovio.com/index.php?page=angry-birds-walkthrough-videos
[Accessed 28:th of May 2011]

[23] Rovio Mobile (2009) Angry Birds. Published by: Chillingo/Clickgamer.

[24] RetroCom - BellSouth - IBM Simon
http://www.retrocom.com/bellsouth_ibm_simon.htm
[Accessed 28:th of May 2011]

[25] Apple - iPhone 4 Technical Specifications
http://www.apple.com/iphone/specs.html
[Accessed 28:th of May 2011]

[26] ConsoleDatabase - Nintendo DS Console Information
http://www.consoledatabase.com/consoleinfo/nintendods/
[Accessed 28:th of May 2011]

[27] iNiS (2006) Elite Beat Agents. Published by: Nintendo.

[28] Apple - iPad Technical Specifications
http://www.apple.com/ipad/specs/
[Accessed 28:th of May 2011]

[29] Reactable
http://www.reactable.com/
[Accessed 28:th of May 2011]

[30] C. Ganser Schwab, A. Steinemann, R. Hofer, A. Kunz, InfrActables: Supporting Collocated Group Work by Combining Pen-Based and Tangible Interaction, Proceedings of Tabletop 2007, 2007, Newport, Rhode Island, USA

[31] Tech-FAQ - Pointing Devices
http://www.tech-faq.com/pointing-devices.html
[Accessed 28:th of May 2011]

[32] Microsoft - Parts of a computer
http://windows.microsoft.com/en-US/windows-vista/Parts-of-a-computer
[Accessed 28:th of May 2011]

[33] Microsoft – Touch
http://msdn.microsoft.com/en-us/library/cc872774.aspx
[Accessed 28:th of May 2011]

[34] Clifton Forlines , Daniel Wigdor , Chia Shen , Ravin Balakrishnan, Direct-touch vs. mouse input for tabletop displays, Proceedings of the SIGCHI conference on Human factors in computing systems, April 28-May 03, 2007, San Jose, California, USA

[35] Robert Mack, Kathy Lang, A Benchmark Comparison of Mouse and Touch Interface Techniques for an Intelligent Workstation Windowing Environment. PROCEEDINGS of the HUMAN FACTORS SOCIETY 33rd ANNUAL MEETlNG - 1989

[36] MacKenzie, S., Sellen, A., and Buxton, W. (1991). A comparison of input devices in elemental pointing and dragging tasks. ACM CHI Conference on Human Factors in Computing Systems. p. 161-166.

[37] Clark, Josh (2010) Tapworthy: Designing Great IPhone Apps. O'Reilly Media. ISBN10: 1449381650

[38] Android Academy 101 - Speaking Android- Tap, Pinch, Swipe, and Hard Pres
http://www.androidacademy101.com/2011/02/speaking-android-tap-pinch-swipe-and.html
[Accessed 28:th of May 2011]

[39] Multi-Touch Technology - Multi-touch trackpad gestures
http://www.multitouchtechnology.com/mac/multi-touch-trackpad-gestures/
[Accessed 28:th of May 2011]

[40] 36peas - 5 iPhone game control design pattern cheat sheets - gestures, tilts, d-pads, sticks and buttons fight it out
http://www.36peas.com/blog/2010/10/27/5-iphone-game-control-design-pattern-cheat-sheets-gestures-t.html
[Accessed 28:th of May 2011]

[41] Hrvoje Benko , Andrew D. Wilson , Patrick Baudisch, Precise selection techniques for multi-touch screens, Proceedings of the SIGCHI conference on Human Factors in computing systems, April 22-27, 2006, Montréal, Québec, Canada

[42] Juul, Jesper (2009) A casual revolution. MIT Press ISBN-10: 0262013371

[43] William Buxton, Ralph Hill, Peter Rowley, Issues and Techniques in Touch-Sensitive Tablet Input (1985) SIGGRAPH '85 Proceedings of the 12th annual conference on Computer graphics and interactive techniques ISBN:0-89791-166-0

[44] Hrvoje Benko and Daniel Wigdor Chapter 11
Imprecision, Inaccuracy, and Frustration: The Tale of Touch Input
Tabletops - Horizontal Interactive Displays
Human-Computer Interaction Series, 2010, Part 2, 249-275, DOI:
10.1007/978-1-84996-113-4_11

[45] Otmar Hilliges, Andreas Butz, Shahram Izadi, and Andrew D. Wilson
Interaction on the Tabletop: Bringing the Physical to the Digital
Tabletops - Horizontal Interactive Displays
Human-Computer Interaction Series, 2010, Part 2, 249-275, DOI:
10.1007/978-1-84996-113-4_11

[46] Jussi Holopainen & Staffan Björk, Game Design Patterns
Lecture Notes for GDC 2003 Lecture,
2003. Available Online:
http://www.gents.it/FILES/ebooks/Game_Design_Patterns.pdf
[Accessed 28:th of May 2011]

[47] Jones, J. C. (1992). Design methods, second edition. John Wiley & Sons. ISBN10: 0471284963

[48] C. Snyder (2001) Paper Prototyping
http://www.cim.mcgill.ca/~jer/courses/hci/ref/snyder.pdf
[Accessed 28:th of May 2011]

[49] Jussi Holopainen & Staffan Björk. (2005). Patterns in Game Design. Charles River Media, Inc. ISBN: 1-58450-354-8

[50] Gameplay Design Patterns
http://62.142.53.59/daisy/patterns/3-DSY.html
[Accessed 28:th of May 2011]

[51] YouTube - Microsoft Surface's Channel
http://www.youtube.com/user/mssurface
[Accessed 28:th of May 2011]

[52] YouTube - Multitouch Tower Defense
http://www.youtube.com/watch?v=71BfgXZVBzM
[Accessed 28:th of May 2011]

[53] Firemint (2009) Flight Control. Published by: Firemint.

[54] Atari (1979) Asteroids. Published by: Atari.

[55] DICE (2010) Battlefield: Bad Company 2. Published by: EA.

[56] Microsoft Surface Blog - Why 52?
http://blogs.msdn.com/b/surface/archive/2007/12/16/why-52.aspx
[Accessed 28:th of May 2011]

[57] Microsoft - Designing Microsoft Surface Applications for Tagged Objects
http://msdn.microsoft.com/en-us/library/ee804762%28v=surface.10%29.aspx
[Accessed 28:th of May 2011]

[58] Riot Games (2009) League of Legends. Published by: Riot Games.

[59] Microsoft Surface - How has the Surface Changed?
http://www.microsoft.com/surface/en/us/WhatsNew.aspx
[Accessed 28:th of May 2011]

[60] Apple iPad - Technical Specifications
http://www.apple.com/ipad/specs/
[Accessed 28:th of May 2011]

Appendix A – iPad Patterns and Games

	Power-Ups / New weapons	Winnable	Punishemt for failure	Reward for actions	Randomness	"Surfaceable"	Touch usage *	Casual	Saves game state or level **	Unlockables	Highscore	Player count	Co-op	Boss Monster	Levels	Multiple Game Modes
Fruit Ninja	yes	no	3	5	yes		Multi touch, Slice	yes	no	yes	yes	1	no	no	no	yes
Ninjump	yes	no	5	4	yes	Maybe	Tap	yes	no	no	yes	1	na	no	no	no
Gun Bros	yes	yes	3	2	no		Fake gamepad	no	yes	yes	no	1	na	yes	yes	no?
GT Racing		yes	2	2	no		Gyro, Hold finger	no	yes	yes	yes	1	na	no	yes	no
Hit Tennis 2	no	yes	3	1	no		Slice	maybe	no	yes	yes	1 - 2	no	no	no	no
Cut the Rope	yes	yes	2	3	no		Slice	yes	no	yes	yes	1	yes	no	yes	no
Harbor Master / Flight Control HD	no	no	4	3	yes	maybe	Drag	yes	no	no	yes	1	yes	no	yes	no
ElementZ	yes	no	1	4	yes	yes	Drag / Tap	yes	no	no	yes	1	yes	no	no	yes
PyramidZ	yes	no	1	4	yes	yes	Drag / Tap	yes	no	no	yes	1	yes	no	no	yes
Sparcle hd	yes	yes	3	4	yes		tap	yes	yes	yes	yes	1	na	no	yes	no
Smurfs	yes	no	2	3	no		tap	yes	yes	yes	no	1	na	no	no	no
Spider Man HD	no	yes	3	3	no		fake gamepad, tap	no	yes	yes	yes	1	na	yes	yes	no
Panzer class	no	yes	4	2	no	yes	drag, tap	yes	no	yes	yes	1	na	no	yes	no
Predators	no	yes	4	5	no		fake gamepad	no	yes	yes	no	1	na	yes	yes	no
Angry Birds	yes	yes	3	4	no		drag, tap	yes	no	yes	yes	1	na	no	yes	no
Deer hunter	no	no	2	2	yes		tap, drag	maybe	yes	yes	yes	1	na	no	yes	no
Dizzypad	no	no	5	1	yes	yes	tap	yes	no	yes	yes	1	na	no	no	yes
Plasma Globe	no	no	5	2	yes	yes	hold	yes	no	no	yes	1	yes	no	no	no
Halogen	yes	yes	5	5	yes	yes	hold, drag	maybe	no	yes	yes	1	na	yes	yes	no
Geared 2	no	yes	2	3	no		drag	yes	no	yes	no	1	na	no	yes	no
Mirrors Edge	no	yes	2	2	no		slice	maybe	yes	yes	yes	1	na	no	yes	yes
Friendsheep	no	yes	-	1	yes	yes	drag, tap	yes	no	no	no	2 - 6	no	no	no	yes
Zombie smash	yes	yes	4	5	yes		slice,drag,tap	yes	yes	yes	no	1	yes	yes	yes	no
My memory hd	no	yes	-	1	no	maybe	tap,	yes	no	no	no	2 - 4	no	no	no	no
Train conductor 2	no	no	2	4	yes	maybe	tap,drag	yes	no	yes	yes	1	na	no	yes	no
Can knockdown	yes	yes	4	4	no		drag	yes	no	no	yes	1	na	no	yes	no
Zentomino	no	yes	-	2	no	yes	drag , tap	yes	yes	no	no	1	yes	no	yes	no
Salt	yes	yes	5	3			fake gamepad	no	no	no	yes	1	no	no	yes	no
Cover Orange	no	yes	3	3	no	maybe	tap, drag	yes	yes	no	no	1	no	no	no	no
Death worm	yes	yes	4	4	yes	yes	fake gamepad, tap	yes	yes	no	yes	1	no	no	yes	yes
Shift	no	yes	2	2	no		fake gamepad	yes	yes	no	no	1	no	no	yes	no
Sandstorm	yes	yes	4	3	no		fake gamepad, tap	no	yes	no	no	1	no	no	yes	no
Super Fly	yes	yes	4	4	yes	yes	fake gamepad	yes	yes	yes	yes	1	no	yes	yes	no
Xmas Planet	no	no	3	4	yes	maybe	drag, tap	yes	no	no	yes	1	no	no	no	no

* slice = quick, drag = slow

** Very hard to define what is a save/load-system

Appendix B – Implemented Game Pattern Matrix

	Power-Ups / New weapons	Winnable	Punishment for failure	Randomness	Touch usage *	Highscore	Player count	Co-op	Non-Player Help / IRL player helper	Boss Monsters	Levels	Lerning curve	mean play time for 1 one round (in min)	consecutive rounds depend on previous rounds(how)	Objects hidden by players **	other players in the way ***	"game over" -able ***	Chance for Analysis Paralysis	real-time	Late arriving players	Honor system in place ****	Snowball
Defend your side	yes	yes	3	no	tap, touch and hold	no	2	no	yes	no	no	low	2	no	low	yes	yes	no	yes	yes	yes	no
Airport control	no	no	5	yes	Drag	yes	1	no	yes	no	no	low	3	yes	med	med	yes	no	yes	yes	yes	no
Asteroids with bullet hell relations	yes	no	4	yes	tag	yes	*	yes	no	yes	no	low	3	yes (highscore)	high	med	no	no	yes	yes	yes	no
Four side platform game	yes	yes	3	yes	Drag, tap, slice	no	*	no	yes	no	no	low	undef	no	med	med	no	yes	yes	yes	yes	med
Top-down Racing	no	yes	2	no	tag	yes	*	no	no	no	yes	low	3	no	low-high	low-high	yes	no	yes	yes	yes	low
Turn based Artillery game / Slingshot	no	yes	3	no	Drag, tap, tag	no	2	no	yes	no	yes	low	4	yes (nr of obst)	low	low	no	yes	no	no	yes	med
Node and troop battle	no	yes	3	yes	Drag, slice	no	2 to 4	no	yes	no	yes	low	3	no	med	med	yes	yes	yes	no	yes	high

* tag = move,rotate && presence
** Players limbs obscure game objects
*** Other players trying to be in the same physical space
**** A state defined such that it is no longer possible to proceede
***** The game dosent stop cheating, especcially not outside of the machine

Appendix C – Game Idea List

1. Defend your side
    a. Create objects which float towards the other side
    b. Destroy objects by touching them
    c. Prevent objects from entering your side
2. Defend all edges of the surface
    a. Cooperative game to avoid letting anything leave the area
    b. Enemies spawn from the centre and must be killed with towers
3. Slingshot game
    a. Shooting at other players
4. Eat and grow (Snake)
    a. Avoid self and other snakes
    b. Control by rotating a tag
5. Block stacking (Tetris)
6. Tennis game (Pong)
7. Paddle control
    a. A ball bounces around, the ball is not allowed to leave the area
    b. Control a paddle with a finger to prevent the ball from leaving the screen
    c. The objective is to make the ball hit things
8. Airport control
    a. Airplanes enter the screen randomly.
    b. The player shall lead the planes to an airport by dragging a finger from the plane to the airport
    c. The game ends if two planes crash.
9. Asteroids with bullet hell relations
    a. Players controls space ships with tags
    b. The players shall destroy everything by shooting on it
10. Scrolling platform game
    a. Two players, one on each long side of the screen
    b. Players needs to help each other by modifying obstacles on the other side
11. Four side platform game
    a. Each edge of the Surface is a floor with gravity, the middle has no gravity
    b. No limit in players, players runs along the floors and around corners
    c. Players lose by getting hit by obstacles falling from the cloud
12. Resource transport
    a. Two players and a “bank”
    b. Goal is to make more money than the other player
    c. Buy and build material producing factories and mines, sell or sublime material
    d. Buying materials from the bank is more expensive than from the other player
13. Three in a row game
    a. Switch places on blocks to create more than three in a row
    b. Three in a row removes the three blocks, four and five removes more blocks
14. Labyrinth
    a. One or two players shall get to the middle of the labyrinth fast

15. Follow the Leader
    a. The leader uses a number of colours, figures and gestures. All players shall do the same, failing to do the correct combinations removes one life out of three.
16. Lasers and Mirrors / Particles and Magnets
    a. A laser light or particles comes from a position on the screen
    b. The player places tags on the screen which redirects the light or particles.
    c. The goal is to get most of the light/particles to a specific place on the screen.
17. Jigsaw puzzle
    a. The player moves puzzle pieces on the screen to create a full image.
18. Fruit Slicer
    a. Lots of fruits and bombs fly up on the screen and back down again
    b. The player shall split the fruits in half by slicing over them with a finger
    c. Missing fruits or touching a bomb is bad
19. Waiting game
    a. The game plan is a restaurant where customers comes to eat.
    b. The player shall take orders, cook, serve food and remove dishes from tables.
    c. Everything is done by touching things, in time and in the correct order.
20. Progressquest
    a. Player starts the game by choosing race, religion and class.
    b. Game then starts to fight/kill enemies, do quests and level-up, all without any input from the player.
21. Rougelike
    a. Player controls a character which needs to survive as long as possible in a dungeon.
    b. The deeper in the dungeon the player is, the harder the game becomes.
    c. Player has magic, equipment and skills to help his decent into the darkness.
22. Pen and Paper game (computer help for PnP RPG-games)
    a. Draws maps and keeps track of monster and player positions.
23. Role playing game
    a. A long epic quest to rid the world of evil.
    b. Uses rules from the "Pathfinder" rule set.
    c. Controlled by onscreen interfaces
24. Healing game
    a. A number of characters, a number of healing spells.
    b. Characters lose health on a regular and irregular basis.
    c. Keep everyone alive as long as possible by healing them fast and effective.
25. Split screen Shooter
    a. 3D first person shooter which splits the screen in half
    b. The player controls the character's walking with the left hand and aiming and shooting with the right hand
26. Building a city
    a. Player is given a piece of land, and has to develop that into a fabulous city.
    b. Drag buildings from a menu and drop them to build.
27. Growing a creature

    a. The player receives a creature which has to be fed and trained to become faster, stronger and gain new special powers
    b. The player then battles other creatures
28. Flight Simulator
    a. Simulates an airplane.
    b. Uses on-screen interface for the less common controls, and tags to simulate the main stick and other common controls.
29. Music instrument simulator
    a. Given strings, keys or such, lets the player press or draw to simulate an instrument to make music.
30. Split screen racing
    a. 3D
    b. Controlled by on-screen interface, and using a tag as a steering-wheel.
31. Top-down Racing
    a. Controlled with a tag placed on the surface
    b. The tag is rotated as the steering wheel in the car
    c. Gas and breaks is controlled as the position of the tag
32. Fighting game
    a. On screen interfaces and gestures
33. Ice hockey
    a. Players point/tags where they want the puck-controlling player to go, tap to shoot, quick draw towards other players to pass.
34. War control
    a. Turn-based strategy
    b. Supervise troops
    c. Putting a tag on a unit shows an interface, point in it to give orders.
35. Turn based Artillery game
    a. The game plan is a solar system with a number of planets, rocks and a sun.
    b. Players adjust angle and power of a rocket, to try and hit other player's planets
    c. All planets and the sun has gravity, some planets and rocks moves.
36. Card game
    a. A number of cards with tags with effects. Players play a card, the surface register the card and plays the effect
37. Solitaire
    a. Various single-player card games.
38. Negotiation game
    a. Each player determines her move in conjunction with the other players.
    b. Each player then enters her moves, in secret, into the surface, which then plays all moves.
39. Economy game (Monopoly)
    a. As the board game.
40. Find hidden secrets (Battleship)
    a. As the board game of the same name
    b. Two players, one on each short side with a big sheet in the middle preventing the players from seeing each other ships
41. Find a special object in an image (Where's Wally?)
    a. The player is asked to find and tap on a specific object in a large detailed picture.
42. Press and pop balloons as fast as possible (Whack a mole)

    a. Balloons appear on the surface, the player has a number of seconds to touch them to gain points.
43. Keep finger away from obstacles
    a. Player controls a dot by holding and dragging a finger, goal is to not let the dot touch anything on the screen
    b. Walls and other dangerous things moves in a predetermined path in each level
44. Pick flowers
    a. Each player is given a tag and a dark screen
    b. The objective is to pick more flowers than other players.
    c. The tags are placed on the screen and removes black fog around the tag
    d. Each player can only pick flower close to their own tag. Most flowers win.
45. Throw balls in holes
    a. Player is given a number of balls and the goal is to throw them into holes.
    b. Controls balls by giving them a speed and direction, by flicking.
46. Build and balance towers out of objects
    a. Blocks, balloons and other strange stuff must be balanced on small areas
    b. Player wins when all objects are balancing without falling
47. Find the pirate treasure
    a. 3D world and a map over an island with a treasure chest marked
    b. Player navigates the island without GPS positioning
    c. Mission is to find the treasure as fast as possible
48. Mind control game
    a. Convert enemies to become your minions, and make them kill each other.
49. Staff training and management
    a. Make a company thrive by employing, training and combining staff
50. Master thief attempting to burgle a palace
51. Steer a bouncing rubber ball through a course
    a. Control the ball by making it bounce towards a finger.
    b. Every level has different holes and ramps to get past.
52. Puzzle involving the theory from finite automata's
    a. Things like regular expressions.
53. Placing flashlights to banish ghosts
    a. One dark house, full of ghost.
    b. Player controls a number of flashlight, ghost are afraid of the light
    c. Goal is to scare the ghosts to a specific area.
54. Quiz game
    a. A combination of “Jeopardy” and “Who wants to be a millionaire”.
    b. Pick a subject, get four alternatives,
55. Paint and add physics to the drawing
    a. Player paints objects, game adds physics
    b. Goal is to get a ball to the end
56. Disaster creator and controller
    a. Control natural disasters
    b. Player is evil and shall kill as many as possible
57. Plumbing game
    a. Direct fluid in pipes
    b. Pipes are organized
58. Control a soldier and his moves on a battlefield
    a. Make gestures to make the soldier execute different moves

59. Farm simulator
    a. Plant crops, harvest and develop the land.
    b. Uses tags to bring up context menus for fields.
60. Game for cats
    a. Chase mouse or laser pointer
    b. Draws a picture along a route for a cat to chase.
61. Colour Battle
    a. Each player is a colour and colours are spread by tags, goal to paint 50% of the area in the players colour
    b. Each player has only a limited amount of paint.
    c. Players can convert rival player's paint
62. Node and troop battle
    a. Every node owned by players produces troops
    b. Troops can be sent to other nodes to conquer them, at ten the node is taken
    c. If troops in a node reaches zero the node is lost
    d. Troop losses are always one to one.
63. Decryption
    a. Player is given a text that contains a hidden message, and some clues
    b. goal is to find the hidden text
    c. clues might be situational
64. Hero wars (Dota)
    a. Each player has a hero and a side
    b. Objective is to destroy the other side
65. Snowflake game
    a. Goal is to stop the maximum number of trains with the fewest number of snowflakes...
66. Sniping game
    a. The player has a view where enemies and friends are moving around
    b. The goal is to protect the friends and not be seen and killed by enemies
67. Throw as far as possible
    a. The angle and speed of the first throw is decided by two moving bars which the player must touch at the correct time to get a maximum starting distance.
    b. The object thrown will bounce a couple of times but the player can increase the power of some bounces by touching the screen at correct place and time.
68. Jumping game
    a. Controls a bunny to jump on floating objects, getting higher for each jump.
    b. Bunny moves towards the finger.
    c. More points for each jump,
69. Herding cats
    a. Trying to get a group of cats to do the same thing, by tapping, dragging or such.
    b. When all cats does the same thing the player receives points.
70. Leaf blowing game
    a. Clear a lawn of leaves, using a tag as the leaf-blower.
    b. The wind, dogs and children are enemies messing up the leaves
    c. Blowing on dogs makes them go away, blowing on children makes their evil parents appear and chase and punch you

71. Human body defence game
    a. Cooperative game where the players control a human body's defence system
    b. The systems controlled are immune-system and such.
72. Math game
    a. A challenge consisting of different counting and solving is given to the players
    b. The answer to the challenge shall be returned to the surface on time
73. Bingo
    a. Each player picks a number of cards and then numbers are called out
    b. First player to get a row wins, if a player misses a number that player cannot win on that row
74. Fluid control
    a. Move floating objects by moving the fluid they are floating on
    b. A certain number of objects need to reach the goal to complete a level

Appendix D – Design Documents

Surface Airport

The game is very similar to the popular IPhone-game, Flight Control. The player's mission is to land airplanes on an airport and the game ends when two airplanes crashes into each other.

Surface Asteroids

The game takes place in outer space containing several asteroids and dangerous space ships. The player takes on the role of a space ship and the missions is to survive as long as possible destroying as many obstacles and enemies as possible.

The ship and its controls

The ship controlled by the player is always firing its weapons.

The player controls the ship by putting a tag on the surface and moving it around, the space ship will always be located directly beneath the tag's position no matter how fast the player moves the tag. The rotations of the ship will also always be the same as the tag. If the tag is lifted from the surface, the space ship will enter warp mode, in warp mode the ship cannot be hit by anything and will not take any damage, but it cannot fire its weapons either. Warp mode can only last a few seconds and when the player puts the tag back on the surface the ship reappears on the new position, but will not start firing its weapons again before a few seconds has passed

Upgrades

The game will be paused from time to time to change level and to let the players rest a bit. During this time upgrades can be bought by the players using cash earned by destroying enemies and collecting loot. Upgrades are things like more damaging weapons, more solid space ship, longer warp windows and shorter weapon pause after warp.

Several players

More players can join in on an on-going game by putting a new tag on the surface, this adds a new space ship with exactly the same features as existing ships on the surface. Up to four players can play at the same time.

Surface Platform

The game is a platform-like game for one or more players (with the space around the table being the only limiter), where each edge of the screen works as a floor with normal gravity towards that edge. To create a character, a gate is positioned in the middle of the screen from which a player can grab a new character and drag this to any edge, this character will be added to the game and will be playable like any other character, this can be done at any moment in the game. Around this gate a large gravity-

less cloud consisting of different powerups and powerdowns is positioned, a player can at any point in the game touch these objects and either drag them towards themselves or drag them towards an opposing player.

Characters

Characters are controlled with one finger, touching the character and moving it outside the character makes it move in this direction, moving the finger slightly upwards makes the character jump. The player has to keep the finger pressed to make the character continue walking, otherwise it will stop. By standing still with the character holding the finger on it, a new button is added above the character, shoot. Shooting creates a projectile which slowly moves straight to the right seen from the closest edge of the screen. If a character is hit by any other player's projectile, it will instantly be killed.

Each character has three attributes, one for how many points it has, one for its attraction to objects in the cloud, and one for its speed. The attraction is defined by how much the character is moving, moving decreases attraction and standing still increases it. Having high attraction makes negative objects in the cloud move towards the attracting player, a very high attraction can even make the objects leave the cloud, falling directly towards the player.

The cloud

Inside the cloud which is located in the centre of the screen there are a number of items in the form of powerups or powerdowns floating around. The powerups can increase your speed, decrease the speed of everyone else or similar. The powerdowns on the other hand makes you move slower, can stun you or might even kill you. The space between the cloud and the ground has normal gravity, except if a character with high attraction is apparent, in this case the gravity is directed more towards this character.

Death

Dying puts the killed character in the edge of the cloud slowly moving towards the middle, and giving all other players currently on the game field one point, when the dead character reaches the gate in the centre of the cloud, it is removed from the game. The player can before this happens choose to continue playing by dragging the character back to any position of the game plan.

End and defining a winner

The game automatically ends when all characters are removed from the game by dying and reaching the gate in the centre of the cloud. This automatically will happen if no character has been moving for a while, so in a sense the players decides when to stop. The winning player is the one having the most points at this point.