Enhancing Usability for Multi-Touch Devices

Johan Larsson

Luleå University of Technology

BSc Programmes in Engineering
BSc programme in Computer Engineering
Department of Skellefteå Campus
Division of Leisure and Entertainment
ENHANCING USABILITY OF MULTI-TOUCH DEVICES

Johan Larsson
Abstract

Interaction with computers has become far more intuitive with the introduction of multi-touch technology that allows for more of a natural way of interacting with objects.

This paper delves into how we made use multi-touch technology to create an intuitive software for holding presentations. This is something we wanted to do as holding good presentations is crucial in today’s business world. It also goes into researching and developing a piece of software for a multi-touch unit for practical use in a business setting. This is mostly as it is time that multi-touch takes the step from something “cool” to something practical in the business world.

This project also allowed us to learn more about how to do research and development working from a basic idea of a client. Especially in the second part where the application had to be modified during the project to make it fit within the constantly changing constraints of what the customer wanted.

Sammanfattning

Interaktion med datorer har blivit långt mer intuitiv i och med introduktionen av multi-touch som tillåter ett mer naturligt sätt att interagera med objekt.

Det här paper:et går in på hur vi använt multi-touch teknologi för att skapa en intuitiv mjukvara för att hålla presentationer. Det är något vi ville göra då att hålla bra presentationer är väldigt viktigt i dagens affärsvärld.

Det går också in på forskning och utveckling av en mjukvara för en multi-touch enhet som ska ha en praktisk användning i en affärsmiljö. Detta mest pga att det är dags att multi-touch tar steget från att vara något ”coolt” till att vara något praktiskt i affärsvärlden.

Det här projektet har också tillåtit oss att lära oss mer om hur man gör research och utveckling arbetandes utifrån en grundläggande idé från en klient. Speciellt under den andra delen då vi hade en applikation som behövde modifieras flera gånger under utvecklingen för att få den att uppfylla klientens konstant förändrande krav.
**Preface**

This thesis project was done at Natural User Interface, Skellefteå, during a period of about 10 weeks in the spring of 2008. The company was looking for interns to work on developing applications that they could either include in their framework or to base further development on.

We were a total of four students who got the assignment to develop different software divided into two major projects. Our part was primarily to develop an application for holding presentations mainly aimed at the business market.

Also we worked on research and development of applications to improve the usability of multi-touch, which took the form of developing a piece of software with Pawel Solyga for an external client who we cannot disclose at this time.

I would like to thank Harry Van Der Veen, Mikael Bauer, Johannes Hirsche, Peter Bomark and the rest of the staff at Natural User Interface for giving us the opportunity to do this project, guiding us along the way and helping us in so many ways both practically and with giving us ideas.

I would also like to thank the other students doing Bachelor Thesis Projects at NaturalUI at the same time as me; Fredrik Medeström, Hans Hartman, Erik Sunden and Lasse Partanen. All have contributed with ideas and inspiration for problem solving and feature improvements.
# Table of Contents

Abstract .............................................................................................................................................. i
Sammanfattning ................................................................................................................................ i
Preface ............................................................................................................................................. ii
1 Introduction ........................................................................................................................................ 1
  1.1 Goal and Purpose .................................................................................................................. 1
  1.2 Background ............................................................................................................................ 1
    1.2.1 Multi-Touch Technology .......................................................................................... 1
    1.2.2 Natural User Interface ............................................................................................... 2
    1.2.3 NUI Framework ............................................................................................................ 2
    1.2.2 Short look at existing presentation software ................................................................. 3
    1.2.3 Flow - Functionality Research ..................................................................................... 4
    1.2.4 ExtendedViewer - Specification Analysis ..................................................................... 5
    1.2.4 Abbreviations ................................................................................................................. 5
2 Methods - Design & Implementation ................................................................................................. 6
  2.1 Flow - Client Side .................................................................................................................... 6
    2.1.1 DesktopHandler ............................................................................................................. 6
    2.1.2 Entity ............................................................................................................................. 6
    2.1.3 EntityGroup .................................................................................................................. 7
    2.1.4 NetMan and Classes under NetMan ............................................................................. 8
    2.1.5 XMLVarServer .............................................................................................................. 8
    2.1.6 QuickSort ....................................................................................................................... 9
    2.1.7 SelectionManager and Selection ................................................................................ 9
    2.1.8 HRE & GuiButton ......................................................................................................... 9
    2.1.9 PDFConverter .............................................................................................................. 10
    2.1.10 Menu .......................................................................................................................... 10
  2.2 Flow - Server Side .................................................................................................................... 12
  2.3 ExtendedViewer ...................................................................................................................... 13
    2.3.1 DragAndDropArea ....................................................................................................... 13
    2.3.2 Entity extensions ............................................................................................................ 13
    2.3.2 SpeakerIcon .................................................................................................................. 14
3. Results and Future Work ............................................................................................................... 15
4. Discussion ....................................................................................................................................... 16
6 References ....................................................................................................................................... 17
Appendix 1 ......................................................................................................................................... 18
1 Introduction

This section contains some brief descriptions of the basic principles and tools used in this project.

1.1 Goal and Purpose

The goal of this project was mainly to develop the application that during the course of the project got the name Flow, a multi-touch user interface application for structuring and managing images, documents, text files, videos etc for holding presentations.

This tool is supposed to take away the complications of holding a presentation and make holding a highly professional presentation something possible to do for everyone without having to fuss around with a complicated piece of software.

The secondary goal was to do research and implementation of general ideas of where it would be possible to make use of multi touch favorably. This goal was mainly taken up by the ExtendedViewer development for an external client.

The general overall purpose was to make more practical use of multi touch technology and the more natural way of interaction that it brings in today’s technology-driven business market as that seems like the natural progression for multi-touch technology.

1.2 Background

1.2.1 Multi-Touch Technology

Frustrated Total Internal Reflection (FTIR) is the most widely spread representation of the multi-touch technology today. It is also considered to be one of the cheapest [12] techniques today. There are occurrences of other methods as well, for example [11] and [13], but FTIR seems to be the method that is most frequently referred to.

The principles of the technology are quite simple. Infra Red light is sent into the side of a pane of a semi-transparent higher-index medium (acrylic, glass, plastic) and is trapped inside this medium by the refraction index of the material. The light sources in the FTIR-case are infra-red diodes attached to the side of the pane. Total Internal Reflection occurs when the light ray is traveling inside a higher-index medium and strikes a surface boundary into a lower-index medium. When a finger touches the surface of the pane, the light is frustrated, causing it to scatter downwards where it is picked up by an IR camera.

For some deeper insight on how a multi-touch display is built I would recommend reading Visualization and Prototyping of a Multitouch Display Device [14].
1.2.2 Natural User Interface


They are passionately involved in taking the innovative multi-touch technology to the next level by creating new, natural and intuitive interfaces that enable people to interact with technology in a novel way.

They provide standard and customized multi-touch hardware and software solutions that fit the clients' exact needs and wishes.

Their corporate headquarter is located in Skellefteå, Sweden.

1.2.3 NUI Framework

It was decided from the beginning that this application was going to be written for the Natural User Interface framework, or as it is also known, the NUI framework. The NUI framework provided us with an environment where the data from the touch screen is processed and given to us as moving dots telling us how the fingers are being moved and where they are on the screen.

The NUI framework also provided us with some functions for doing math and some classes to make using the input data to move, scale and rotate objects.

We will explain a little bit about what kind of classes we used from the NUI framework and a bit about how our interface towards it looks in part 2.1, but as it is proprietary software we will not go into great detail about exactly how it works.

For some more details about the NUI framework and how it was constructed I would recommend looking at Application Development For Multi-Touch Interfaces written by Mikael Bauer [1] and the NUI Group web page and forum [2].

We did use quite a few libraries and API:s existent in the NUI framework, but as they are part of their framework I see them all as a collection and will only mention them as a single framework.
1.2.2 Short look at existing presentation software

Today’s presentation software is generally dominated by Microsoft PowerPoint and Apple Keynote. Both allow for showing of text, images and video and both allow for both making and displaying presentations.

Both of these softwares are relatively easy to learn to use basically if you are even somewhat computer-savvy, and both have come out in quite a few editions and as such have done quite a few versions of their user interface.

The problem however lies in that for people that aren’t computer-savvy at all they both are quite complex and it does take a while to figure out how to format things to make them look good and to learn the interface. Also, unexpected events that causes a presenter to somehow adapt his presentation to his audience and move away from the completely linear slide format they are quite inept.

Flow does not have a tool for creating presentations, only for holding them. As such you still need some kind of tool for creating your presentations which might feel like needing extra software, but it also allows for creating presentations in any image/layout program that you feel like using.

Our application requires a piece of hardware to run the multi-touch aspect of the NUI framework, which does run with mouse support on a computer that does not have the multi-touch hardware to receive inputs from but then the whole point of using multi-touch hardware is lost. So in such a situation a software such as PowerPoint or Keynote might be a better solution.
1.2.3 Flow - Functionality Research

A large amount of research was put into how to group/sort the objects, how to move them around on the multi-touch screen and how to send them to being displayed. There were a lot of different ideas on how to do so, both from us and from outside people who gave us feedback, so the first thing we did when we started this phase of how we needed to plan out the design was to start looking at how others had done it before us.

The first thing that we did was to study 3D Desktop Environments such as Bumptop\[8\], TileUI\[9\] and Real Desktop\[10\] for inspiration. We used how they do things as a kind of reference when building our design of how to create our piece of software. We looked at all videos we could find that they have released. Having videos that show off the same type of behavior as you want for something early on really helped with thinking of the things that can cause big changes in the design to accommodate for them if they weren’t realized until later on.

After doing that we then studied the NUI framework’s ViewerApp and how it deals with displaying things and moving them around to try to get a feel for how to do basic movement to make Flow feel like a natural part of the NUI framework. We did decide at this point to step away from some of the basics of the ViewerApp. It was decided this early on that having to move around the viewport was far too complicated. It was decided to keep the screen stationary without the viewport moving and that you see the entire desktop at once. It was also decided that we wanted to make the items draggable but not floating as they are in ViewerApp since that just felt like an element that could add confusion.

Then finally we got a demonstration of the flash presentation software prototype produced by Adithya Ananth and Divesh Jaiswal while studying National Institute of Design in India. In many ways this software did represent the way we had already envisioned the software, but the unique perspective of someone who is a specialist in design was still quite refreshing when it came to organizing the ideas for development. We got a whole lot of ideas for possible improvements on the design we already had at that point and we added the idea of somehow being able to draw on the pictures and that that drawing would also show up on the display.

After taking all of these into account it all lead us to one big realization more than any other: Functionality takes precedence over anything else. Especially in the case of Bumptop it felt as if they had decided to do a lot of things that look cool even though they were quite meaningless or in some cases even hindering functionality-wise. Things like being able to do special gestures to accomplish various visual effects are things that we decided weren’t needed and more likely to be confusing than help with making the software intuitive. As such we decided that less really is more and keep it simple.

The basic concept eventually boiled down to that employing the KISS principle as much as possible is definitely preferable, especially in a high-pressure situation such as when holding a presentation remembering complex gestures can be unnecessarily complicated. The basic functionality was as such limited to handling of pictures, video and PDF-documents in a viewer environment combined with a way of sending them to a display server application that shows it on a projector and some way of drawing on them. The fact that we wanted to keep it simple caused us to dismiss the idea that we had initially of actually “throwing” the pictures onto the projector as that is something that may very well be too hard to execute flawlessly in a pressured situation. Which led us to realize that just dropping the pictures somewhere is enough to make it intuitive.

Another major thing that we decided in the early analysis stage was that we wanted to try to do the presentation over a network connection as this brings some flexibility in how to hold presentations and it means that one can hold them over great distances. Dividing them also means that multiple presenters can share a display, even if they are using different clients as long as they follow the
protocol, and it means that one presenter can present on any number of different displays as they can add any amount of displays to their list of recipients of their data.

### 1.2.4 ExtendedViewer - Specification Analysis

The primary difference is that it isn’t designed for displaying entities for anyone else, it will just deal with displaying the pictures on one screen. A rather special one at that as the format differs quite a bit from most of the multi-touch screens built by Natural User Interface in format as it is about 50:1 in aspect ratio.

The second difference is that we are doing it for an external client that will want it made to their specifications and as such the entire design was not up to us.

It also meant adding support for a drag and drop area for copying a file to a USB memory from the program. The area was to fade in when a USB memory was attached, disappear when it was disconnected, pulsate glows and display status messages and progress bars about the status of the copying.

It also required that there was a drag-able speaker icon object that controlled what video displayed that currently has audible sound.

### 1.2.4 Abbreviations

- **NUI, NaturalUI** - Natural User Interface
- **XML** – Extensible Markup Language
- **USB** – Universal Serial Bus
- **PDF** – Portable Document Format
- **KISS** – Keep It Simple Stupid
2 Methods - Design & Implementation

2.1 Flow - Client Side

See Appendix 1 for a diagram over the main classes of Flow excluding HRE and XMLVarServer that fit in to too many places. This is also a rough estimate of what the original design model looked like with only the unneeded AudioHandler taken out and SelectionManager added. In some classes the amount of data that can be manipulated is perhaps a bit exaggerated, but the application was built for flexibility as well as parts of it may later on be re-used for other applications created within the NUI Framework.

2.1.1 DesktopHandler

The basis for the application is a class called DesktopHandler. Having everything plug into DesktopHandler was mainly a way to bring a certain level of portability and to be able to test elements that did not depend on touch inputs without being dependent on the NUI framework for absolutely all parts of testing.

The DesktopHandler gets fed inputs by the NUI framework and it is the entry point and it as such the “main” class for Flow as far as the NUI framework is concerned. In our application it works as a wrapper for DesktopHandler.

2.1.2 Entity

Entity is one of the items in the Flow application. It has one or several pictures on it and can be scaled, rotated and moved around. It can also be drawn on and stores notes in a .XML-file that the presenter can display by pressing a button with the text details on it that is in the bottom right corner of the image/video.

Using two fingers on an entity allows for scaling and rotating it. Holding down three fingers on it locks it in place and allows the user to draw lines on it using a fourth finger, this was mainly designed with using three fingers to lock it rather than allowing the user to draw on it using his third finger on it since that might make the user unintentionally scale or rotate the image while drawing leading to inaccurate drawing. We chose to only allow the user to draw with one color, one brush thickness and that there is no way to selectively erase parts of the drawing. This was to simplify implementation, most specifically the network implementation of the drawing, but also general handling of the lines.

The parts of the Entity are HRE:s and GUIButtons (see 2.1.8), with a PDF Entity having multiple HRE:s with one for each page. In retrospect being able to break up a PDF into multiple entities might have been a good idea, but in designing the Entity class we decided to keep them grouped as one entity since the entities are kept in a more one entity equals one file fashion.

An Entity has a button named “Details” which slides out a black area from behind the Entity with notes that the presenter has written about the Entity. These notes scroll if they do not fit the area visually and there is no limit as to how many notes one can write. The lines of the notes also get added line-breaks if they do not fit the width of the area.
The notes are stored in `Content/Notes/<entity filename>.XML` and are structured as:

```xml
<entity filename>
    <note value="foo"/>
    <note value="bar"/>
</entity filename>
```

Entity is by far the biggest class as it has multiple functions for setting and getting values as there is quite a few ways that entity can be modified. Some are a bit redundant however as we stepped away from the original design at some point and made entities inherit from Interactable, a NUI Framework class for helping with the creation of movable, rotateable, scalable objects optimized for multi-touch. We already had entities that did all that before we switched, but it was not as optimized for use with multi-touch, so they didn’t “feel” entirely right.

### 2.1.3 EntityGroup

Entity Groups are groups of entities ordered in either a stack or a grid fashion. They can be sorted based on Filename, File Type, Date/Time last changed and File Size (selectable from the menu).

When the bottom entity of a stack is moved the entire stack is moved and when another entity in the stack is moved it’s pulled out of the stack.

If an entity is tapped it’s picked up and sent to the back of the stack when the stack is moved.

An EntityGroup contains a XMLVarServer that contains data for grouping such as the offset of two elements in a stack in y, the maximum number of entities to put in width in a grid and so on.
2.1.4 NetMan and Classes under NetMan

Netman is the main class for network management under which the subclasses for file transfer and transfer of blob data for blob drawing exist.

It has a XML-file that is named DefaultClients.xml which stores the Clients which it sends data to.

It has a function named sendToAllClients that sends an image to all clients for displaying an entity on the DisplayServer and a function named SendBlob for sending a finger position to the display server for displaying drawing on a picture.

The classes directly under NetMan are DrawCli and NetworkHandler where NetworkHandler handles the file transfers and DrawCli is the drawing client. The reason that they are split is due to that it allows for far more flexible drawing on entities and file sending with both parts more optimized for it’s part.

All network transfers are handled through using Winsock2. This would of course have to be changed if one was to run the app on another platform to accommodate that platform’s way of using sockets.

2.1.5 XMLVarServer

The XMLVarServer takes in an xml-file whose path is given in the constructor. It then gives the user access to the values within it using getFloatVariable(<name>), getIntVariable(<name>), getStringVariable(<name>), getCharVariable(<name>) and getBoolVariable(<name>). It also has a Exists(<name>) function that tells if the variable got loaded correctly from the xml.

The XML files are structured as such:

```
<filename>
    <name> value="<value>"/>
    <name2> value="<value2>"/>
</filename>
```

So as such, taking a small part of the file DesktopSettings.xml as an example:

```
<desktopsettings>
```
This means that loading this file and doing a get(StringVariable("wallpaper")) would return "wallpaper.png". This class is based upon XMLVarServer that I wrote in C# for the Region Wars [7] game development project.

This version, unlike the previous one, uses a library for loading XMLs, namely tinyxml [5]. This allowed for me to more quickly build a fast and light implementation without having to spend a lot of time on parsing the actual xml-files rather than just taking the data tinyxml gave me and building up data structures for storage and access based on that.

### 2.1.6 QuickSort

This is a class that actually goes beyond just QuickSort, it takes in a Entity list and sorts it according to the parameters set. It is generally used instantiated inside a EntityGroup to allow for code such as group->sort() with the parameters set earlier.

QuickSort uses nodes named QSNode to sort. Potentially this could allow for extending it to sorting objects of any type if they inherit from entity.

### 2.1.7 SelectionManager and Selection

SelectionManager is the overlying selection manager. All it does is manage the selections as we are dealing with multitouch and want to allow people to create multiple selections in different locations at the same time. Due to the way our Selection Manager works we allow for any number of selections being created at any given time.

A selection is a trail of blobs of a finger moved that make up a line. When the finger is released the line closes and makes a “circle” within which all Entities are selected and turned into a group.

The lines stay alive for 5 seconds after being released and then they are cleared. The inputs are based on that all fingers have unique id:s in the NUI framework and as such their updates are tracked uniquely and fed to the selections.

### 2.1.8 HRE & GUIButton

HRE or Hirarchial Rendering Element is a class that has been designed and re-designed a couple of times during the project. It has been designed to create a way of simplifying rendering and especially hirarchial rendering during Flow's development.
A HRE can be set to be Pre or Post, which means to be rendered before or after it’s parent, it will always be rendered when it’s parent is unless set to disabled, but it will not be deleted along with it’s parent.

You can set which part of it’s parent a HRE will be based upon based on a simple 0,1,2 system. The left edge of it’s parent is 0, the middle is 1, and the right is 2. Same goes for down edge being 0, middle being 1 and top edge being 2. So a setBasePosition(1,1) which is the default will mean that it’s position is based upon the middle and setBasePosition(0,2) will mean it is based on the top left corner.

The base position is not the same thing as the position, the position is how long from the base position an item will be placed. If you for example set it to have a base position of 2.0 and a position of -0.02, 0.0 it will be placed 0.02 to the left of the bottom right corner.

There is also a way of setting it to follow scale in x or y or both. This to simplify for creating elements that need to scale with their parent objects such as titlebars or frames. Allowing to turn it on and off for each axis separately allows a much greater freedom than just being able to turn it on or off entirely.

GUIButton is an extension of HRE that allows for creating a HRE that is “clickable” and will return if a finger is set down inside it.

### 2.1.9 PDFConverter

The PDFConverter class allows for converting a PDF to a series of JPG images in a subfolder named as the PDF. It allows for loading PDF’s as Images into entities and sending them to the display server for displaying.

It uses two pieces of open-source code, pdf2ppm that is a part of ImageMagick [3] and ppm2jpg that is a part of Netpbm [4]. It can also take existing images for PDF files that have already been converted once to keep down loading time and processing power needed.

The PDF converter is something that we had on our list of critical minimum requirements as pdf is an open widely supported standard for documents with multiple pages of text and images.

### 2.1.10 Menu

The Menu class handles the menu for setting up sorting and grouping parameters. It slides out from the right on the push of a button. This has been redesigned a couple of times to improve ease of use. It consists of one HRE and a few buttons that are GUIButtons.

It requires that it gets the new AspectRatio of the window whenever it changes. This is due to that it...
needs it to be able to stay on the right side at all times. This is due to Flow not being resolution-
sensitive and as such possible to run at any resolution the user may want to have.

Then there is a function named Draw() that handles drawing of the menu and fingerDown which needs
to be called when a finger is put down within the menu area for the menu to react to finger input.
2.2 Flow - Server Side

The server side application is quite simplified and scaled down to the bare essentials. This is mainly for two reasons:

1. We wanted something light that can be run on the same computer as the client if needed. Them running on the same computer is more or less standard operation as in most cases the presentation will be held by the person in place and the network part allowing one wants to hold presentations over great distances is mostly an additional feature.

2. There is no need to complicate it, and that the DisplayServer we currently have can work as a development platform for anyone who wishes to write their own display server. As long as it can receive the network data properly anyone can write their own that would better fit their own purposes such as supporting other platforms since our DisplayServer is at the moment Windows-only.

It at the moment lacks any kind of sound support as this wasn’t in our minimum specification since it was clear from the start that the allotted time frame for developing the server wasn’t going to be enough for it. Adding on something to allow for streaming audio would be a good idea for improvement if one wants to make the present over distances functionality greater as that would take out the requirement to use a third-party software for that.

As I’ve said earlier the server has no limitation as to how many clients can connect to it just as Flow’s client doesn’t have any limitation as to how many servers it can send data to. As such an arbitrary amount of clients can connect to the server at the same time without any conflicts at all.

As the network part of the display server is built so that it can quite easily be ported to any platform that supports tcp/udp sockets it would not even be that much of an issue to port it to some mobile devices purely technically speaking. It would require modifying the bottom layer of the network handler to change how the socket connection is set up. As the calls used for sending files and drawing data to/within the Network Handler are on higher abstraction levels for everything but the bottom-level socket handling it would only require altering the bottom layer.
2.3 ExtendedViewer

ExtendedViewer uses some of the classes from Flow such as DesktopHandler, Entity and XMLVarServer. Basing it on code from Flow was our own choice as we didn’t feel that re-coding everything brought any advantages compared to just extending code we already had to suit the purpose.

Hence this part will only detail the classes and design choices that were specific to this application and not the ones that carried over from Flow.

We did do most of the code design on our own and only the desired functionality was something that was set. The desired functionality was changed and expanded as the project progressed and we tried to alter our design to accommodate the new wishes of the client.

2.3.1 DragAndDropArea

The drag and drop area is designed to deal with copying of files to a USB memory based on Entities being dropped on it.

The main functions for it are ConnectUSB() that is called when a USB memory is connected and causes the area to fade in, DisconnectUSB() that is called when the USB memory is disconnected and causes the area to fade out, IsInside() that detects if an Entity is inside the area and getLastDropped() that returns the name of the last Entity dropped on it.

It also has an array of different statuses, each with a different descriptive text and a glow pulsating in a different color. These all appear in certain situations, and they automatically switch back to the base status when the others are done. It is designed for being simple to use with an interface that is very simple and it handles most of the state switching on it’s own as long as it gets events that tells it when the copying is initialized, how it’s going, when it’s done and so on.

2.3.2 Entity extensions

The most visually obvious thing is that the Entities have been extended with a frame, a sound indicator to notify the user if the sound is on or not and a button for playing/pausing the video(if it is a video).

Entities were also extended to include a path to the file which would be connected to the entity and as such be the file that was copied to the USB memory when the Entity was dropped on the drag and drop area. This was done by adding another line to their notes xml-file with the name contentpath (see entity, 2.1.2)

They were also expanded to have a set title in their titlebars and not the filename which is what they have in Flow. This was yet another line in their notes xml-file named title.

The two things added are not necessary for the entities to load properly and if one is missing it will quite simply not have something to actually copy or not have a title and just have a blank title bar.

However, a complete xml file for the extended entites look like:
When an Entity is dropped on the drag and drop area it is set to fade out, move to a random location and then to fade in again. This was accomplished by adding on a function for setting target alpha and target position of the Entity. When an entity is in target alpha mode it will continuously slowly fade it's alpha towards that value and when an entity is in target position mode it will set it's target alpha mode to 0, and when the target alpha is reached it will move and set it’s target alpha mode to 1.

This can be described in pseudocode more clearly as:

```plaintext
if Entity->getTargetPosition().x < 0 && Entity->getTargetPosition().y < 0
    Entity->setTargetAlpha(0)
if Entity->alpha == 0
    move entity to random location
    Entity->setTargetAlpha(1)
    Entity->setTargetPosition(-1,-1)
```

With a constant fading towards target alpha regardless of what it is.

This works well as negative positions aren't allowed, and as such giving it a target position less than 0 is a flag to tell it that it shouldn’t fade out and move.

### 2.3.2 SpeakerIcon

The SpeakerIcon inherits from the entity class and only differs from it in a few ways. It can also be dragged around, but not rotated or scaled. Initially we had an idea that it would have a scalable outer circle that could make the area it influences bigger or smaller, but this was cut due to that it only made using it confusing and strange.

It adds on a function that tells it if it is on an video or not and if it is it turns on the sound for that video and turns off the sound for all other videos. It only turns off the sound of the other videos whenever it is dragged onto an video to allow for that if a user drags it onto an video and off it for it to not obstruct his/her view the video shouldn’t go silent.

It is a quite simple object, but it makes quite a bit of a difference in implementation as you have to do a separate way of drawing it and a separate means of handling it not along with the entities as it isn't supposed to be affected by the same things as the entities in some cases such as rotating, scaling or being copied to USB, but it is in some cases as in dragging around.
3. Results and Future Work

Both flow and the video displaying program work as they should and as such I find this project to be a great success. We have created business applications for the NUI framework and have as such extended the usability of multi-touch in a business setting.

ExtendedViewer, the video application has received positive feedback when it was showed, I cannot go further into that as there is no press release yet but I am sure the reader of this document will know what the application in question is when it is released.

Future expanding on the application in both the server and the client is possible. The framework is rather open and does not have any real restrictions on what is and isn’t possible to do and as such it isn’t impossible to do any kind of expansion one can wish to implement.

The directly useful one would be some kind of implementation of loading Microsoft .ppt-files as these are quite a standard when it comes to presentations today and being able to load them would be quite an advantage.

Also, improving the server with adding on support for sound would be a useful additional feature that would especially mean that one can do more vivid presentations across long distances using the network feature.

I also had the idea that it might be interesting to create Flow Micro, a version of Flow that allows for running presentations on your iPhone. This is so far outside the span of this project that it almost isn’t worth mentioning under further work, but it would be a quite nice addition for doing small presentations that don’t require any big special hardware. However, as I don’t have an iPhone it is nothing I will be attempting.

Rebuilding parts of the network handling in the DisplayServer to use something such as SDL where socket handling becomes more abstract, which allows the network handling to be compiled on any platform that supports SDL could also be a good idea as it allows for even easier porting of the DisplayServer. The DisplayServer network protocol could also be extended with zooming in on specific parts of an image and displaying multiple images side-by-side as that may be useful for presentations.
4. Discussion

I am generally satisfied with my work flow and we did absolutely everything that was in our minimum requirement and we managed our time quite well and to a great extent we followed our original plan. The only deviation from it was that the wishes of the client on the ExtendedViewer application demanded that we spent a little more time on that than what was planned.

I feel as if this report would have felt more complete if I would have gotten to tell all about the other project and not just Flow. Sadly there is no press release out for it yet and as such it is still confidential.

I wish we would have had more time to develop Flow, I think that with some more design guidelines from an interface designer it is a piece of software that could be sold to big corporations the world over. It may require special hardware, but the hardware keeps getting cheaper every day.

However, for the software to sell I think it would require a way to actually create the slides of the presentations and not just rely on people to use some random external tool even though they could pretty much use any tool.

A big issue we have had is that due to other projects NaturalUI have been working on during the same time we haven’t had the opportunity to test everything on a multi-touch unit at all times during the development and have had to depend on mouse inputs to test. This has lead to that we’ve had some issues finding problems that involve multiple fingers moving at the same time.

In the code we have messed up the usage of the words client and server in a lot of cases. Mostly due to bad specifications of what server and client means in this project. The DisplayServer is quite often referred to as a client in the Flow client application and that causes unnecessary confusion. This is something I think we would have gotten past with a more clear specification of the meaning of server and client in this project.

I must say that I’m especially satisfied with the XMLVarServer which, even if it wasn’t optimal, works fast, smooth and performs really nice under stress. At some point the other project group was creating XMLVarServers and parsing in data from XML multiple times each frame and there still wasn’t much of a performance hit. It may still need a little tweaking but I think it can possibly be a great addition to any future projects I work on.
6 References


```cpp
@EntityGroup

- n_qs: void
- n_groupType: int
- n_type: int
- n_EntVect: std::vector<Entity>*
- n_XVarServ: XMLVarServer

// <create>-EntityGroup()
// <destroy>-EntityGroup()
+ setType(i_type: int): void
+ getType(): int
+ addEntity(i_Entity: Entity): void
+ removeEntity(i_Entity: Entity): void
+ getEntityVector(): std::vector<Entity>*
+ setEntityVector(i_EntVect: std::vector<Entity>*): void
+ updateGroupPos(): void
+ first(): Entity
+ last(): Entity
+ printList(): void
+ sort(i_type: int): void
+ setBackwards(i_backwards: bool): void
+ setGroupType(i_type: int): void
+ setGroupType(): int
+ sendToFront(i_entity: Entity): void
```
InputHandler

<<create>>= InputHandler()
<<destroy>>= InputHandler()

+CheckVaEntities(i_entlist: std::vector<Entity*>, inData: BlobData): Entity
+FingerUp(i_entlist: std::vector<Entity*>, inData: BlobData): void
+FingerUpdate(i_entlist: std::vector<Entity*>, inData: BlobData): void
```cpp
Menu

- _n_ButtonArray: GUIButton
- _n_SortType: int
- _n_Grid: GUIButton
- _n_Dock: GUIButton
- _n_Date: GUIButton
- _n_Size: GUIButton
- _n_File: GUIButton
- _n_Ext: GUIButton
- _n_voidEn: void
- _n_Vle: HRE
- _n_aspect: float
- _n_slideSpeed: float
- _n_slideCheck: int

<<create>>-Menu()
<<destroy>>-Menu()
+fingerDown(_n_inData: Kohana:: KohanaData): bool
+setSortingType(_n_inType: int): void
+getSortingType(): int
+setVoidEn(_n_inDR: void): void
+slideMenu(): void
+AspectF()_aspect: float): void
```
NetMan

m_clients: std::vector<char> *

create() -> NetMan()
create() -> NetMan(i_defaults: bool)
destroy() -> NetMan()
+addClient(i_ip: char): void
+addClient(i_ip: std::string): void
+sendToAllClients(i_path: char): void
+parseXML(): void
+send2Dobj(i_x: float, i_y: float): void
+floatTostr(i_value: float): std::string
PresentationApp

- nFont: Blob::TrueTypeFont
- n_dHr: DesktopHandler

<<create>>-PresentationApp()
<<destroy>>-PresentationApp()

-OnInitialize(): bool
-OnDoInitialize(): void
-OnReload(): void
-OnUpdate(inFrameTime: float): void
-OnDraw(): void
-OnMouseEvent(inEvent: SDL_Event): void
-OnFingerDown(inData: BlobData): void
-OnFingerUp(inData: BlobData): void
-OnFingerUpdate(inData: BlobData): void
-OnFingerExit(inData: BlobData): void