Digital X-Ray Lightbox

An alternative software solution

Chris Norval

060005662

Final Year Honours Project

Department of Computing

2009/2010

Contents

1. Abstract

Multi-touch technology has been used for some time on small devices such as mobile phones. Recently, the technology has been integrated into larger devices, such as computer monitors. This move to larger devices creates a unique opportunity to re-evaluate some of the more traditional techniques that are used in the day-to-day routine of many professionals.

There are many examples of these previously overlooked systems for performing everyday functions that would benefit from technological advancements. One industry that could potentially benefit more than others is healthcare, with the design and construction of a digital light box as an example. The traditional light box is used in hospitals around the world for viewing X-Ray and Magnetic Resonance Imaging (MRI) films. This device is nothing more than a wall-mounted box with a light inside and a diffusing screen which allows radiologists or doctors to view and evaluate an X-Ray or MRI film. Although the technology is simple, it is still widely used as it is a quick and effective way to view medical films. Although the NHS now stores all X-Rays and MRI images electronically many doctors prefer to use the wall mounted light box due to its convenience and practicality. The computers within hospitals can be used to view those exact images, however, it can be argued that the mouse and keyboard are not the most intuitive tools to navigate an image.

Figure 1: A doctor using a traditional light box to view an X-Ray film

This project is aimed at evaluating Multi-touch technology as an alternative method for the handling and manipulation of clinical images. Through the development philosophy of user-centred design, a prototype was developed, tested by medical students and updated repeatedly, in order to create an application focused around the needs and desires of doctors.

2. Introduction

2.1. Background and Context

The concept of the project came from an idea that new technology, such as Multi-touch input could revolutionise activities that previously were overlooked or impractical through keyboard and mouse input. The logic behind this idea is:

- Multi-touch is a much more intuitive and familiar way to interact with a system that was previously based on a physical object, for example a piece of paper or film
- Due to the similarities between using your fingers with a system and using fingers with a physical object, it is more likely to be adopted by people who may be less experienced computer users
- In imaging systems, Multi-touch has a significant advantage; physical alternatives allow pointing and drawing on the object using ink. However, this can damage the actual copy, whilst using a keyboard and mouse allow the user to point and draw, but do this with poor accuracy. Multi-touch combines the best of both, whilst eliminating the disadvantages.

The task that was chosen for the project was the viewing and evaluating of patients' X-Ray and Magnetic Resonance Imaging (MRI) films within hospitals. The current hospital process generally involves attaching a sheet of A4 film to a wall-mounted light box that illuminates the image of an X-Ray or MRI for viewing. In recent years a new system has become available, but is not predominantly used within hospitals. This system allows staff to view images on a computer screen using a mouse and keyboard for navigation. While all X-Ray and MRI films are now digitally available to doctors and specialists, many prefer to use the light box with the added advantage that it allows other staff to gather around the light box for diagnosis and discussion, rather than around a computer. This creates an interesting challenge for a project; to develop a system that is as similar as possible to the older light box with the addition of modern functionality. This could be used to make the jobs of the users easier, by making assessment of the images easier, allowing improved communication between specialists and potentially creating a system less prone to mistakes and confusion.

To obtain information about user requirements for this project, informal research was conducted. This took the form during the summer of 2009, of talking to fourth-year medical students who use both of the current systems, i.e. the light box and the mouse based imaging application. While these meetings took the form of verbal advice rather than a testing session, it greatly helped the process of deciding what features to attempt initially before any programming was started. The functionality decided for the project was:

- The system must move the image based on finger interaction to mimic as closely as possible using hands to move a physical piece of film or paper.
- The system must rotate the image based on finger interaction similar to using hands to rotate a physical piece of film or paper.
- The system must allow the user to zoom in and out of the image by moving two fingers in opposite directions, as if physically stretching a sheet of stretchable fabric.
- The system must allow the user to use a finger or stylus to "draw" on the image.
- The system must allow brightness and contrast adjustment, as well as inverting the colours to display the negative of the original image.

The logic behind each of these features in some cases is quite obvious, however more subtle reasoning is also used. The rotation feature is hugely beneficial as it became clear from researching this topic that certain universities, such as the University of Dundee and Cambridge University, teach their medical students to rotate chest X-Rays 90 degrees when looking for a fractured rib, as this makes the fracture much easier to notice:

"A useful trick is to rotate the image on its side; rib fractures may then appear more obvious."[8]

Feedback from medical students stated that adjustment of brightness and contrast is a beneficial feature when reviewing MRI images and X-Ray films. The draw tool allows a user to circle or outline specific areas thus highlighting regions of interest in the scan, to assist with evaluation and interpretation

Unfortunately, in the world today medical mistakes are still relatively common. While most medical documentation is still paper based this project aims to prove that emerging technology can be used to improve current ways of working, within and outside the healthcare industry. In hospitals it is an everyday occurrence for doctors to look at X-Ray or MRI scans and to write an evaluation of what they see in paper based notes. These notes may be lost, or may be subject to misinterpretation due to either poor handwriting of the scan reviewer or potentially due to lack of experience of the next person to examine the scan. The application can therefore be used to circle suspicious areas of the image for future use, as well as adding text or footnotes, which improves clarity.

2.2. Similar Products

Midway through development it was discovered that a similar product had already been designed[9], however, it is unclear whether or not development of the product has been completed. BrainLAB's Digital Lightbox is described as "a Multi-touch display that allows surgeons and physicians to instantly access and manipulate digital medical data through the power of touch". The BrainLAB system was designed by healthcare professionals and encompasses many of the features that are proposed in this project, but includes a greater range of features for MRI images. This implies that there is in fact a market for digital light boxes in the healthcare industry. BrainLAB is a German company with approximately 950 employees worldwide. While it is not proposed that this project currently stands as a worthy competitor to BrainLAB's product, it does contain a lot of the more basic and important functionality, allowing an interesting comparison of the two systems considering the differences in resources available for the development of each.

Further to the example described above, Microsoft have implemented a rather similar system at the Microsoft Medical Media Lab, located within a Washington D.C. hospital, as of August 2009[37]. This system uses a Microsoft Surface, however the aim of this application is to share the images between doctors and patients rather than for diagnostic purposes[37].

It is also worth noting that there is currently software being used within hospitals for viewing X-Rays and MRI scans, however, these applications do not have Multi-touch support. This software, which runs on a standard computer, is currently being used around the country and is in place in all hospitals with X-Ray equipment. The software, known as Pacs (Picture Archiving and Communication System) is part of a joint project to make all X-Rays digitally available within hospitals[11], cutting down on issues such as:

i. Cutting costs, by saving approximately £250,000 per NHS trust in the first year of using Pacs[11]
ii. Reducing paper and film consumption, which has an impact on the environment
iii. Reducing the risk of loss of patient records; the loss of hundreds of thousands of patient records each month has been reported by nine NHS Trusts[11]

Another commercial software package which is widely used within the scientific field of image analysis is Amira. Amira is described as "a flexible platform for your 3D visualisation requirements" which "satisfies even your most demanding needs to work with clinical or preclinical image data, nuclear data, optical or electron microscopy imagery, molecular models, vector and flow data, simulation data on finite element modules, and all types of multidimensional image, vector, tensor, and geometry data"[16]. The software, like Pacs, does not support Multi-touch input, however it contains most, if not all, of the features that would be needed for analysis of an MRI scan. The Amira system was found to be the application of choice within the Wellcome Trust Centre in Dundee, being used to build up 3-Dimensional representations of mice from MRI segments.

2.3. Scope and Objectives

The scope of this project is to create a tool for doctors and clinicians to observe easily X-Rays and MRI images.

The objective was to create an intuitive and useful application for viewing and manipulating clinical images within hospitals or health centres using commercially available and affordable hardware. The application would need to have well structured code, good use of algorithms and an easy to understand interface.

3. Multi-Touch

3.1. History

When computers were first designed for personal use in the early 70s they were text-based consoles relying on keyboard input and screen output, bearing only a slight resemblance to modern day personal computers. The Datapoint 2200 (Figure 2), hailed as the first personal computer, had no Graphical User Interface (GUI) and no means of input other than the built in keyboard[1], yet it was used as a foundation for the evolutionary development of personal computers. Shortly after the Datapoint 2200 came the release of the Xerox Alto followed by the Xerox Star: computers which were the first computers to use the Desktop metaphor, a Graphical User Interface and a mouse[2][3]. With 384KB of memory, a 10MB hard drive and a $16,595[27][28] price tag the Xerox Star is hardly comparable to any PCs used nowadays, however, back in 1981 it was cutting edge technology and helped to shape the future for personal computers.


Figure 2: Datapoint 2200

While computers were being designed for home use and becoming more affordable over the past 50 years, touch screens were also being researched and developed since the 1960s[29]. Although mostly developed for academic purposes, the HP-150, the first commercial touch screen computer, was released in 1983. The HP-150 was met with modest commercial success, selling 40,000 units, while other MS-DOS PCs were competing strongly in the market. The HP-150 was advertised as being easier to use, with screen-touching replacing many commands that would usually require a keyboard. The screen contained a series of vertical and horizontal beams of infrared light, that crossed


Figure 3: HP-150[30]

in front of the screen. When a finger was placed on the screen, some of the beams were broken and the cursor was positioned at that location[30]. Throughout the years following the release of the HP-150, touch screen systems found commercial success, being used for ATM machines, PDAs, computers inside factories and a wide range of other practical uses where a keyboard and mouse may not have been accessible. Touch screen computers are still widely used today due to low cost and accessibility, from the self-service checkouts within supermarkets to certain mobile phones and games consoles. However, in recent years an important advance has changed the way that touch screen systems are seen and used, and that is the development of Multi-touch systems.

Multi-touch systems are very similar to touch screen systems. Many use technology which is based on the older techniques, using infrared beams to detect an obstruction on screen. Multi-touch systems have one very significant improvement, which is implied in the name; Multi-touch systems can detect more than one input point at the same time. While touch screens can only detect one finger or input point, Multi-touch systems can detect up to 50 at once, based on the technology that is being used. Several companies competed to release one of the first Multi-touch products, as it became clear that this was the next big step in technology. During a TED Talk in America in February 2006, Jeff Han gave a presentation on a high resolution, low cost Multi-touch screen[31], that was met by huge interest in the media and on the internet. Han presented an image display application and World Wind, an open-source satellite imaging application, similar to Google Maps, developed by NASA. Shortly after this presentation, two of the main companies associated with Multi-touch systems today, announced their new products, featuring full Multi-touch support. In January 2007 Apple announced their immensely successful iPhone[32]. This was followed by Microsoft who announced their revolutionary Surface product in May 2007[33]. Almost three years after the release of Apple's iPhone mobile device, Multi-touch is seen as a requirement for all new smart-phones, and Multi-touch monitors are now widely available following support with Microsoft's Windows 7 operating system[34] and the recent release of the Apple iPad.

Figure 4: Jeff Han gives a real-time demonstration of his Multi-Touch screen at a TED talk[31]

3.2. Multi-Touch Tables

Originally this project was conceptualized as an application for a Multi-touch capable table computer. This opened the door to two options, a home-made system or a Microsoft Surface computer. The Microsoft Surface is a table device which is capable of projecting the display of a computer's screen onto the surface of the table from the inside, and is capable of receiving touch input points[13][10]:

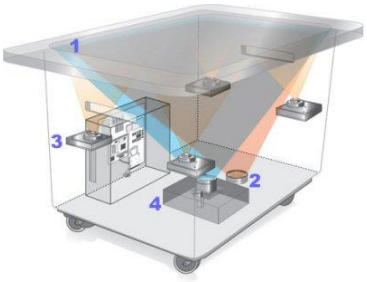


Figure 5: The internal hardware of a Microsoft Surface[10]

1) Screen - The screen is made up of an acrylic sheet which diffuses the projection, turning the top surface of the table into a large display.

2) Infrared - The input aspect of the Surface is performed by having an infra-red L.E.D. light source aimed at the screen. When an object touches the acrylic screen, the infra-red light reflects back and is picked up by several infrared cameras which can pinpoint the area being touched.

3) CPU - The internal computer within the Surface is similar to many household computers. The components consist of a Core 2 Duo processor, 2GB of RAM, and a 256MB graphics card, and the Operating System is a modified version of Windows Vista.

4) Projector - A High Definition short-throw projector allows the projection of the system onto

the Acrylic screen. This makes up the display of the device.

While the Microsoft surface was a very impressive device, and came with the software development kit allowing for immediate developments, there were some problems with the device. The main issue was that the Computing department did not have access to one of these machines, and with a price of £10,000[6] purchase of one was not seen as a realistic possibility at the start of the project.

The other option, a home-made Multi-touch system, seemed much more viable in terms of development, and there were resources available for explaining the construction of such a device[12]. The theory was much the same as that of the Microsoft Surface. The table consisted of an acrylic sheet, infrared LEDs, an infrared camera, a short-throw projector and a simple desktop PC all attached to a cabinet. This hardware, coupled with some open-source platforms, allowed for a very similar device to the Microsoft Surface on a significantly cheaper budget. This option was considered, before learning that the department already had one of these devices constructed.

The open-source platforms available allowed for an option of either using Touchlib or CCV (formerly T-Beta), two libraries which allowed the table-device to recognise more than one input point on the screen, regardless of operating system. While Touchlib was already set up on the Multi-touch table in the department, CCV is claimed to be a newer library, containing more functionality, essentially making Touchlib obsolete. After thoroughly testing both Touchlib and CCV it was determined that there was a significant problem with the calibration of the device. Even after repeatedly calibrating the software there was still a difference of about two inches from the position of a user's finger to the registered point on the surface. As accuracy was a necessary and important feature for this project, this presented a significant problem. That, combined with the lack of access (meaning that development time was restricted to working hours), forced the decision that this surface would not be practical for development of the project and that other options should be investigated.

3.3. Multi-Touch Monitors

The next hardware option to be looked into turned into the most sensible. Multi-touch monitors were being developed on a commercial scale alongside Windows 7, with the release dates both due for October 2009. While this delayed the start of development of the project by a few months it allowed for the greatest amount of flexibility and accuracy. It allowed around the clock access to a commercially accurate device, i.e. Microsoft-made software development kits and support, as well as educational resources for learning alongside many people who would also become early adopters of this technology, through blogs, Twitter and online web forums. There were plenty of articles and reviews online about upcoming Multi-touch monitors, claiming an affordable price, and release alongside Windows 7. Examples are Packard Bell's Viseo 200T Touch Edition monitor[14] and Acer's T230H Multi-touch monitor[15]; the latter was eventually chosen as the device for development. The monitor uses optical camera technology to detect the input points. Two infrared cameras are mounted in the top two corners to scan the surface of the touchscreen. The system works out the position of the touch point based on the blockage from both cameras and registers them on the system as an input call[18].

Figure 6: Acer's T230H Multi-touch monitor

3.4. Languages and Frameworks

The next important issue that needed to be addressed was which language and frameworks would be best suited for development. After learning that Microsoft had delayed the official release of .NET 4.0 until April 12th, 2010[7], which contained the addition of Multi-touch support[17], an alternative was needed. Several alternatives were looked into, including C++ native development[19][20] and Microsoft Foundation Class (MFC) development[19] before Windows Presentation Foundation (WPF) 3.5, with the Windows 7 SDK featuring Touch beta libraries[19] was chosen as the most appropriate framework for the project. This allowed a large amount of online support and resources[22][21], as well as reasonably high level development, and did not require a

thorough understanding of advanced system calls in C++ and the writing of advanced algorithms to perform tasks that other frameworks may provide. WPF is relatively similar to the .NET library, the main difference being the interface tool uses Extensible Application Markup Language (XAML) rather than the more traditional methods. This introduces several new interface items and classes for structuring, such as positioning grids, canvases and editable control over the appearance of pre-created controls. WPF allowed the programming to be done mostly in C#, with some XAML to set up the default interface elements. The libraries provided for C# were mostly similar, however, slight differences were apparent in some of the classes, such as Image. Whereas in .NET before an Image object would be created from System.Drawing.Image, by default WPF and XAML used System.Windows.Controls.Image. The former utilised GDI+ for much greater control over the image data and the latter placed greater restrictions on the access of pixels, making colour manipulation particularly difficult.

4. Design Stage

4.1. Use Cases

Use cases are a way of detailing the functionality of a system. Each function that can be performed is isolated into a use case that specifies the flow of the behaviour. A use case allows an early understanding of how a function can be implemented as well as highlighting some preliminary problems relating to interaction or code architecture.

4.1.1. Search for CHI

To start, the user must search for a specific patient to bring up images associated with that patient. This search is done through the use of a keypad onscreen. The user is prompted to enter 10 numbers, which correspond to the user's Community Health Index (CHI) number, a ten-digit unique identifier used by the NHS consisting of three pairs of numbers indicating the patient's date of birth, followed by four pseudo-random numbers[23]. The keypad helps to eliminate any errors in the structure of the CHI as well as any foreign characters:

a) Screen presents a keypad and prompts user for a patient's CHI number
b) User enters 10 digits using the keypad
c) Once the tenth digit is entered, the application disables all buttons except the "backspace" button and enables the "Go" button
d) User can press the backspace button, which enables all buttons again but disables the "Go" button and removes the last entered number
e) Once all ten digits have been entered and the user presses the "Go" button the application checks to see if the directory for that patient exists, and if that directory contains a patient.xml file
f) If the file or directory does not exist the keypad displays a "User not found. Try again" message. If the file does exist then the xml file is read and the patient's details are obtained by the system
g) A reference to the patient's images are inserted into the drop down box, allowing images to be selected, and subsequently loaded
h) Interface is changed to show that a patient has been loaded, and the user can then choose to change image or search for a new patient

4.1.2. Free-Control Image

Users need to be able to directly manipulate the position, rotation and scale of images. To do this, they use Free Mode. Free Mode is one of the options available on the tool's control bar.

Once an image is loaded, the control bar on the right hand side becomes available. That allows the different Modes to be selectable. By default "Free Mode" is initially selected, however when on a different mode the user can change to "Free Mode" by pressing the "Free Mode" button, containing a picture of a hand.

Using Free Mode:

a) User selects "Free Mode" by pressing button on the control bar.
b) Interface changes to indicate "Free Mode" is turned on. The Free-Mode button becomes circled and the previous mode's button becomes un-circled.
c)

- User can use one finger to *move image* by placing finger over image and dragging to new location

- User can use two fingers to *rotate image* by placing fingers over image and rotating them centred on the mid-point of both fingers. Application rotates image based on angle of rotation of user's fingers
- User can use two fingers to *scale image* by placing fingers over image and dragging one in the opposite direction of the other, or both in opposite directions from the other. Application scales image based on how far the user's fingers have distanced from each other.

4.1.3. Draw on Image

Users need to be able to draw lines on top of an image. This allows them to trace around parts of the screen image and circle areas without actually writing on the images. To do this, they use Draw Mode. Draw Mode is another option available on the tool's control bar.

By pressing the "Draw Mode" button, Draw Mode is enabled and the previous mode disabled.

Using Draw Mode:

a) User selects "Draw Mode" by pressing button.
b) Interface changes to indicate that "Draw Mode" is enabled and the previous mode disabled. The Draw-Mode button becomes circled and the previous mode's button becomes un-circled.
c)

- User can drag one finger over screen.
- After user moves finger slightly, a straight line is drawn between the original point and new point onto the Drawpad element. This may happen several times a second giving the illusion that hundreds of very short straight lines are actually just one line which has curved and followed the user's finger, and that the user is drawing on to the image, whereas they are simply drawing on top of the image and all covered pixels are easily recoverable.
- When the user's finger is released from the surface of the monitor, the drawing tool stops, waiting for either the next input point to start at or for a new mode to be selected

4.1.4. Enter Text for Image

Users need to be able to place reference points on the image and then add text at the bottom. This eliminates the amount of screen real-estate consumed and makes for a much tidier way to display an image with several notable points for reporting. To do this, they use Reference Mode. Reference Mode is available on the tool's control bar.

Using Reference Mode:

a) User selects "Reference Mode" by pressing button
b) Interface changes to indicate that "Reference Mode" is enabled, and a text box appears at the bottom of the screen
c) User can place one finger on image to place a reference point ([1]) at that position. The reference number depends on how many reference points have already been placed
d) System adds a "[1] " to the text box at the bottom
e) User can either add another reference point, or select text box to enter text after the reference number, or edit a previously written reference

4.1.5. Modify Colour Matrix of Image

Users need to be able to modify the brightness and contrast levels of images, as well as to view the negative of the image. To do this, they use Colour Mode. Colour Mode is one of the options available on the tool's control bar.

The colour manipulation tool is one of the few tools that actually modify the pixel data of the original image. It allows the modification of the image's brightness and contrast levels, as well as an option to display the negative of the image by inverting the pixel colours.

Using Colour Mode:

a) User selects "Colour Mode" by pressing button
b) Interface changes to indicate that "Colour Mode" is enabled, and a box appears at the bottom of the screen containing a brightness slider, a contrast slider, and an "Inversion" checkbox

c) User can move either slider or check the checkbox
d) After every change, e.g. a slider slightly moves or checkbox is checked, the application passes the new values into a method which works out appropriate colour matrices, multiplies them accordingly, and changes the pixel values of the image, allowing for "Real Time" feedback with slider

4.1.6. Reset Image

Users need to be able to reset the image to its original state if they do not want to save or continue with their changes.. To do this, they use the Reset button. The Reset button resets the image to its original state by resetting the colour matrix, removing reference text within "Reference Mode" and deleting all Children from the Drawpad canvas, which removes all lines and reference points placed on the image.

Using Reset:

a) User selects "Reset" button
b) System resets colour matrix and "Colour Mode" sliders, deletes text data, clears Drawpad and resets positional, scale and rotational information of the image

4.1.7. Print report

Users need to be able to print a copy of the modified image along with the reference text in a report format. To do this, they use the Report Tool.

The report tool allows the user to generate a dynamic report based on the image, all modifications and all text data which is then sent to the user's printer. This allows the application to be used as a method of reporting on what was observed on the clinical image, not just for diagnosing.

Using Report Tool:

a) User selects "Print Report" button
b) Application takes modified image along with text information and generates a window based on size of an A4 sheet of paper. Image is resized to fit on the upper area of page and text is added to lower quarter and a Print Dialog box appears
c) User can either select "Print" or "Cancel"
d) If User selected "Print", the generated window is printed. System reverts back to state just before User pressed "Print Report" button

4.1.8. Save Image

Users need to be able to save their modifications to the appearance of the image. To do this they use the Save button.

The "Save" button allows the user to save the modifications to the appearance of the image to a new jpeg file for later observation. While the text and positional data are lost, this tool is useful for reloading an image which has outlined several problematic areas which need to be re-observed at a later time.

Using Save:

a) User selects "Save" button
b) Application takes modified image and generates a window based on size of original image values. Image is resized to default, with Drawpad correctly positioned, and a Save File Dialog box appears giving the ability to save to a Jpeg
c) User can either select "Save" or "Cancel"
d) If User selected "Save", the generated window is saved as a Jpeg file. System reverts back to state just before User pressed "Save" button

4.1.9. Search for New Patient

Users need to be able to search for another patient after finishing observations on the available images for a patient, and the "Search For New Patient" button on the left menu allows this to happen. This button clears all loaded data on the previous patient and reverts to the application's initial state, displaying the keypad and allowing a ten-digit CHI number to be entered and searched for.

Using Search for New Patient:

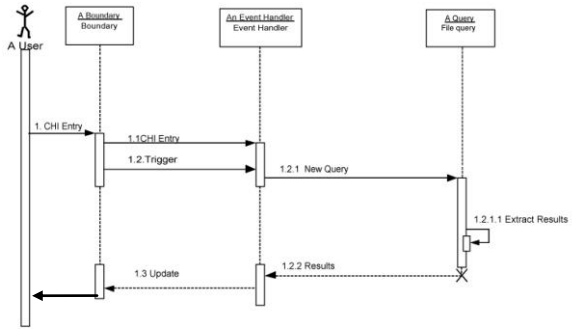
a) User selects "Search for New Patient" button
b) System resets colour matrix and Colour Mode sliders, "Reference Mode" text, deletes loaded image, resets interface by displaying Keypad and hiding all image-dependent buttons, and clears all

variables containing information on previous patient

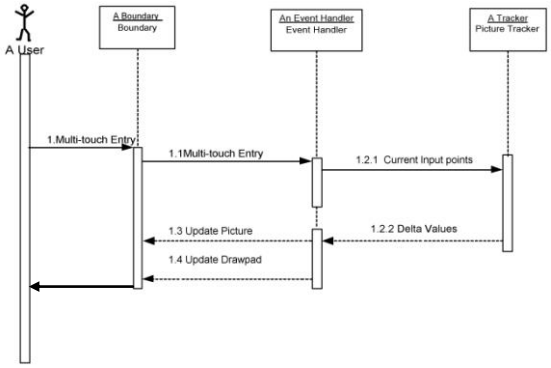
4.2. Sequence Diagrams

Sequence diagrams are provided here for functionality that was created for this project. Other functionality was provided as built-in software for the Multi-touch monitor, such as the onscreen keyboard. Sequence diagrams allow a much more detailed approach, from a developer's point of view, of how the system may handle input.

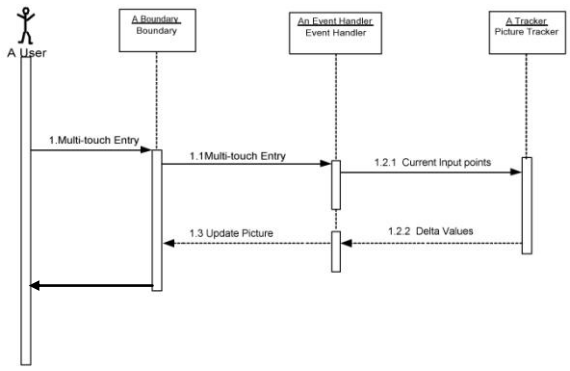
4.2.1. Search for CHI



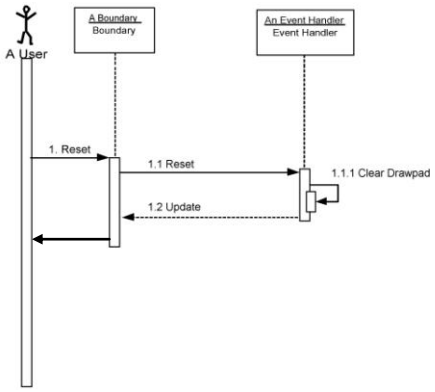
4.2.2. Free Control Image



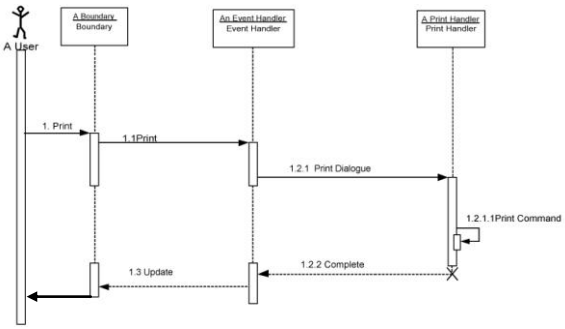
4.2.3. Draw on Image



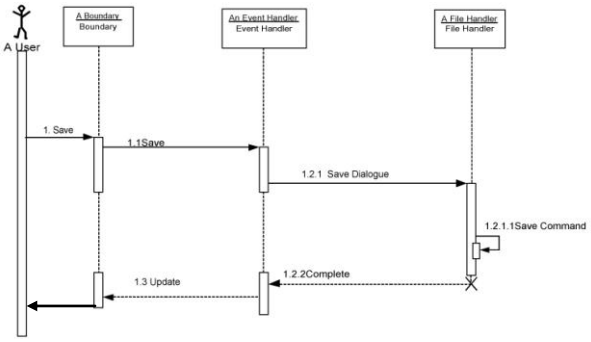
4.2.4. Reset Image



4.2.5. Print Report



4.2.6. Save Image



4.3. Planning/Gantt

The schedule for development was created based on assumptions of how long each task would take. It was acknowledged that some tasks would take longer than others, and that functionality may have changed radically after a testing session, so the development of functionality was broken down into three "Sprints". The first sprint would take approximately a month and lead straight into the first testing session, and the second and third sprints would take two weeks each, divided by the second testing session with the final testing session at the end. After the first testing session was conducted, it was decided that it would be more beneficial to the project to merge the second and third sprints into one, and ignore the second testing session. This was simply due to the fact that the basic functionality had already been tested and there were only a few new things to test, however the final testing session had all of the functionality as well as refactored code to make algorithms more secure and efficient. The Gantt chart (**Appendix 6**) had the first milestone at the start of November, however, it is worth noting that research actually started in July, when the Multi-touch table computer was initially selected as the hardware to be used. The schedule became the backbone of planning functionality, as it allowed a strong understanding of the practicalities of each feature to add, and allowed for a good understanding of what stage the application had reached.

5. Implementation

5.1. Features

The features were an important aspect of the project, which had the overall aim of creating the most user-friendly application incorporating the most relevant features possible. Initially, informal interviews were conducted with medical students, as well as testing midway through development, to ensure that these potential users had a say in what features were useful and which they would like to see incorporated. Throughout development the plan of features changed significantly, with some being drastically changed, some being added, and some being dropped from the schedule. Originally, while in research stages of the project, the application was set to have the following features:

Function	Description	Action
Move/Translate	Move the image up, down, left or right	Place one finger on the image and drag to another location
Rotate	Rotate the image around the point of first contact to the angle created by the movement of the second contact point	Place one finger on the image followed by a second finger. Drag the second finger to another location
Scale	Scale the image up or down while maintaining proportions	Place two fingers on the image and drag them in opposite directions to zoom in and together to zoom out
Draw	Draw a thin coloured line on the image	Select the Draw tool, place one finger on the image and use it to draw on the screen
Crop	Cut off part of the image	Select the crop tool, place one finger on the image and use a second finger to draw a line where the image should be cropped
Reset	Reset all changes and revert to first state	Select Reset button
Change Brightness, Contrast and Filter	Increase or decrease brightness, change contrast, or choose which shades of white are rendered	Select button and use one finger to adjust slider bars accordingly

These features allowed for the basic functionality that was required. The Move feature and Scale feature allowed the user greater control of what they would like to see; the Rotate feature allowed the users to turn images for a better, or a more unique perspective; the Draw tool allowed areas to be traced, circled or highlighted; the Crop feature allowed the user to remove irrelevant areas, such as normal scans on an MRI film, borders etc; the Colour control allowed the user to adjust the

brightness and contrast of the image as well as an experimental algorithm planned for basic edge detection for fractures; the Reset feature allowed the image to be defaulted to its first state.

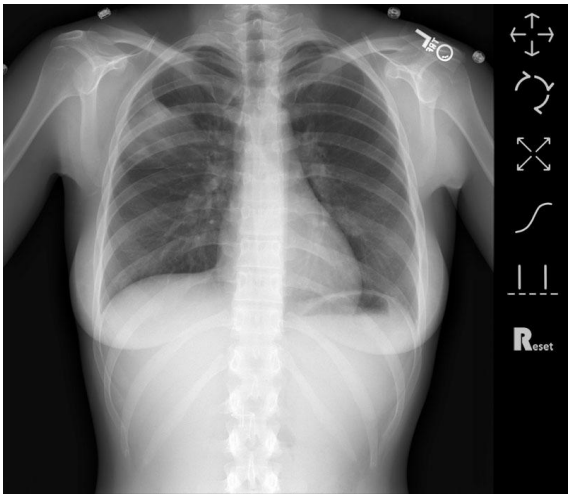


Figure 7: An early conceptual interface for the DAL application

The first conceptual design of the interface (Figure 7) contained a simple layout with all of the tools originally planned, however, this was to change drastically throughout development:

- The Move, Rotate and Scale tools were to be amalgamated into a single "mode" which allowed all positional adjustments to be done together, and at the same time, rather than having to switch tools. "Free Mode" was a much more intuitive and fun solution for controlling the image, allowing fluid adjustments to be done, as if the scan was a piece of elastic material in front of the user.

- A text tool was required to add information to the image. Originally it was planned that a text box could be placed on the image after selecting the chosen position. However, with editing and rotations being difficult and the cost of real estate on-screen being high it was decided that a better method was needed. This brought about the concept of the referencing tool. When a user selects the Text button a text box appears at the bottom. The user can then touch anywhere onscreen and a red reference point would be added, for example "[1]". This would have a transparent background onscreen and take up very little screen space, and a [1] would appear in the text box at the bottom. The user could then select the text box and use a keyboard or the onscreen keyboard to add text to the textbox. The idea was that another user could look at the reference point on the image and then go to the text tool to view the text box at the bottom. This would contain more information about why the reference was added, e.g. "[1] - Fractured rib" or "[6] - Discolouration of liver". This solved rotational issues and the problem of text taking up a lot of room while keeping the image tidy and creating a report of the problem.

- The Text tool did not really change throughout development. The idea was simply to be able to trace lines with a finger onto the image, which could be removed if needed. This was why a separate Drawpad class was created, allowing for an editable and clearable canvas to be put over the picture, but not allowing any modifications apart from colour matrices. The draw lines and reference labels were added to this Drawpad and the Drawpad would have its contents cleared if the Reset button was pushed.

- The colour control changed to having a brightness slider, a contrast slider, and instead of an edge detecting algorithm it contained a checkbox for inverting the pixels of the image. These calculations were performed through the use of colour matrices, which affected the image. When one of the sliders or the checkbox changed values it passed these values into an update method, which controlled the colour manipulation algorithms.

- The reset concept stayed the same, however as implied above the way it worked was adapted to the Drawpad system. When the user presses reset, the method sets the colour properties of the Picture back to default and updates, before clearing all children of the Drawpad Canvas.

- Print was a feature that was conceived midway through development as an alternative method of saving. It was soon adapted to create a unique and powerful tool for users. When it looked as though Save was not possible there was a need for users to be able to store their modified images, and the print concept was created. When pressing print, a print dialog appears and the user can send the scan to a picture. The unique aspect of this tool is that it prints to A4, retains all colour modifications, all lines and reference points, as well as the reference text, which is placed at the bottom of the page. The image adjusts itself to fit onto the top half of the page and the page can be placed directly into the patients notes, with all information needed. This allows the application to be used for diagnostics,

which removes the amount of time required to write up findings, as well as also including a copy of the X-Ray into the appropriate section of the patient's notes. This reduces the risk of medical mistakes as it is much easier to articulate problems on an image with a picture, with lines highlighting the issue, rather than simply using words. (An example of this printed page can be seen on **Appendix 1** at the end)

- The save tool was implemented towards the end of the development after it was requested by end users. This allows the user to save the modified image, with all lines and colour modifications, for later use. The application saves the modified image as a jpeg, so although the user can open the image with the adjustments it will not retain the text information and resetting the image will take it back to the saved state, not a clean version. **Note: This tool does not overwrite the saved image, it is meant to be a shortcut to viewing an already modified version.**

- The crop tool was dropped from the development schedule since it was thought to have a limited, and relatively unnecessary purpose. While so many other features were selected to be incorporated, it was agreed that the schedule should be re-prioritised and Crop was never implemented.

5.2. Code Structure

The structure of the code was based loosely on the Model-View-Controller (MVC) design architecture. This was chosen to provide the highest level of flexibility throughout development, providing high cohesion and low coupling as the classes are not strongly related to each other unless derived, and dependency on other classes is very small. Classes are created for a specific purpose and attempt to communicate only through appropriate methods via parameters. Keeping closely to the MVC architecture proved to be quite unrealistic due to the nature of WPF, and some slight adjustments had to be made. For example, the Picture "window" class could only be changed from the interface, due to it being an interface component, so the Interface class required some code which simply bridged the gap between the Model class and the lower levels of the Interface components. While this may not be seen as ideal for MVC purists, it was the most appropriate way for passing modified versions of the images from classes containing the colour algorithms to the objects which display the image as part of the interface.

5.2.1. The Interface class

The interface class was essentially a combination of the "View" and "Controller" of the MVC architecture as it was responsible for maintaining the interface of the application as well as containing the event handlers which controlled the calls generated by pressing a button on the screen. The Interface class was also responsible for tracking the user's input points, and contained the Picture and Drawpad object as an interface component, moving them when appropriate. It was intended that since the MVC architecture was used, all of the code within the Interface class, and subsequently the Drawpad, Picture, PictureTrackerManager and PictureTracker classes would only relate to changes within the interface itself, and would contain no algorithms for anything else. For example, the colour manipulation algorithms, XML reader, image loading and directory/file maintenance would be part of the "Model", separate from the "View", which was maintained throughout development. However, due to the high level of interaction with the interface components that obviously comes with this kind of project, the "View" and "Controller" aspects grew to be quite large. The following classes were part of this grouping:

- Interface: as specified above, bridges interface and Model components and displays the interface onscreen. This is essentially the main class

- Picture: A simple "UserControl" within WPF which represented the interface component to display the image. When an image is loaded the file path is passed into an instance of Picture, which becomes an "Image" on the interface, the difference being that Picture is a WPF component, so it cannot be passed around while attached to the interface. It supports Multi-touch input, and is essentially just like any other object you would find in the WPF/.NET Toolbox therefore containing very little code.

- Drawpad: Very similar to Picture, a "UserControl" which holds many of the same advantages and disadvantages of any other WPF

component. The main difference between Drawpad and Picture is the way that they can be transformed graphically. Picture contains an Image object, so to scale the component it is often a requirement to resize the Width and Height values, by using ScaleX and ScaleY when wanting to increase the size of the image the area of the Canvas will increase, but the picture will not. Drawpad is the opposite. By changing the height and width values to match that of a Picture, the two objects will not align. This is because by changing the height and width of the Drawpad only the Canvas is resized, not the components inside. Scale counters this problem. Although it may seem logical to combine Picture and Drawpad into a single object, this cannot be done due to the way the contents of the two objects maintain their proportions.

- ColourControl: This is a simple "UserControl" which combines the interface components to specify colour properties. It simply contains a slider for brightness, a slider for contrast, and an inversion checkbox. The only code it holds returns the values of the three variables to the Interface class, to be passed into the Model in order to perform these colour adjustment algorithms based on the values selected through this interface component. It would have been possible to simply make ColourControl part of the Interface class, however to maintain good object oriented design, and to not use unneeded code within the application, this was derived into its own component.

- Numpad: Like ColourControl, Numpad is simply an interface component for gathering user input onscreen. It contains buttons, similar to those that may be found on the front of an Automatic Teller Machine (ATM) to enter pin details. This particular method was chosen to eliminate input errors as a result of misunderstanding. The user presses one of the buttons, which adds the number to the display panel. When all ten digits are entered the user can then press the "Go" button, which returns the double value to the interface class, which then passes it to the Model class to be checked and the patient details loaded.

- Print: This class appropriately scales the Picture, Drawpad and Reference Text information onto an A4-sized Window, which is then passed into a PrintDialog. Due to its heavy interface connection (using Picture, Drawpad, mainly resizing functionality), this is the most appropriate place for the class.

- Save: Like Print, this class simply scales Picture and Drawpad onto a Window before saving the contents of the window to a jpeg image. It's heavy use of Picture and Drawpad, as well as it's resizing code warrants a position within the "View" grouping of code.

- PictureTrackerManager: A modified version of a class from a Multi-touch tutorial obtained from the Microsoft Developer Network (MSDN)[24], this class contains a stack of PictureTracker objects. This handles the Multi-touch input points and allows for an input point to be assigned to a Picture Tracker instance. For example, if a user places a finger on an image and then quickly drags the finger off the image, the collection of input points generated will be recognised as a single drag, and will associate the input points with a single PictureTracker instance. This allows for a large level of control within Multi-touch functionality.

- PictureTracker: Again, PictureTracker is an extended version of a class which was written based on a Multi-touch tutorial. It can be viewed from a URL within the references section[24]. PictureTracker has several features. First of all, a single instance is created when a user places their finger on the screen, and the instance is only released when the user's finger is released. This allows a single instance to track the movement of the user's finger, which is beneficial for tracking input points. This class also contains the code which moves, rotates and scales the Picture and Drawpad components, draws onto Drawpad, adds reference labels to Drawpad and handles inertia for both. Inertia is a simple feature for continuing movement after an instance of PictureTracker is released, although with quick deceleration. This means that if a user quickly drags an image and lets go, for a short time the picture will continue moving onscreen until coming to a stop. This seemingly simple feature for aesthetic value actually creates a much more interactive and realistic feeling to dragging an image, and without inertia the application doesn't feel quite as realistic. Inertia was a component that was specified in the tutorial and slightly modified to decrease the extent of movement after release.

5.2.2. The Model Class

The Model class corresponds quite simply to the Model aspect of the MVC architecture. This class contains the code and sub-classes to perform algorithms that do not rely heavily on the interface components. The Model class contains the code to load the image for the Picture component, as well as reading the Patient.xml file before returning the information to the Interface class. Another function of the model class is to store variables that relate to the directory and file name of the current patient and current image loaded for reference. The sub-class to Model, ColourManipulation, contains the algorithms for calculating the ColorMatrix used for adjusting the brightness, contrast and pixel inversion of the image. When the ColourControl UserControl on the interface is edited, the values are passed to Interface, which are then passed into Model. Model creates a ColourManipulation instance and passes these values in as a constructor, and can then call getFinalColourMatrix() to obtain the final ColorMatrix variable. Inside ColourManipulation the brightness and contrast variables are normalised to between 0 and 1 and are each added to their own ColorMatrix. These matrices are then combined using a matrix multiplication algorithm to create the final matrix. The matrices are all 5x5 (RGBA) and can then be applied to the current image within Model, before being passed back through to Interface to be reapplied to the current Picture instance.

5.3. Methods and Algorithms Explained

The algorithms used were designed to carry out the tasks of the function efficiently. However, due to the MVC architecture at times this requires multiple classes to contain parts of the algorithms. The description of some of the more complex algorithms, and the reason for such complexity, are described here.

5.3.1. Search

The Search feature allows for the user to search for a patient by their Community Health Index (CHI) number, a unique a ten-digit unique identifier used by the NHS consisting of three pairs of numbers indicating the patient's date of birth, followed by four pseudo-random numbers[23]. A patient's CHI number is unique and is used throughout the NHS in the UK to eliminate any ambiguities brought on in the event that two patients have the same date of birth and the same name. Once the ten digits have been entered the user can then press the "Go" button, which takes the number and checks it in the folder containing all patients. If the CHI number searched for is "0404884520" then the application will check for a directory in the storage area and then check if a patient.xml file is present inside. For example, after entering this CHI number the application will check "D:\DALPatients\0404884520\patient.xml" to see if the file exists. If the file or directory does not exist then the directory is not recognised as a patient's storage area for scans and images, and the keypad returns the message that a patient with that CHI number could not be found. If a patient.xml file is found then the xml file containing patient data is loaded, references to the images within that folder are added to the application and the interface is changed to indicate that a patient has been loaded. This method of input allows for a very high level of validation and error recovery, discarding the need for significant input handling of this number, and allowing full control over the appearance of the keypad which is much more suitable than the on-screen keyboard for entering a patient's CHI number.

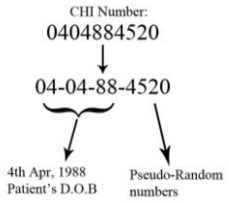


Figure 8: How a CHI number is made up[23]

5.3.2. Colour Manipulation

The Colour Manipulation tool is responsible for changing the brightness and contrast values, as well as giving the ability for the application to invert each pixel colour, essentially making the picture a negative of itself. In order to do this a User Control was created, with a simple interface. This "window" contained the two sliders and a check box, and had been modified to be more sensitive to touch input by increasing selectable

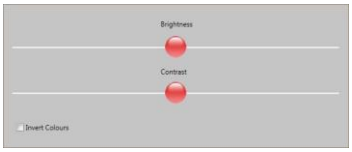


Figure 9: The User Control Window, showing Brightness and Contrast sliders with Inversion checkbox

areas. When this static window is constructed, a reference to the parent class is also passed in so that a method of the parent class could be triggered by an event of the child. Initially this was seen to be potentially poor structuring of code. However, after further research it was apparent that this was the best way to accomplish the task, and it was the safest way to make the event handler work how it was supposed to. The window was set to Visible and Hidden when "Colour Mode" was toggled on and off respectively. In order to give as close to real-time feedback as possible (approximately 1 second) the "valueChanged" event handler of each slider, as well as the "Click" event for the Check Box were used.

When either of the slider values are changed, or if the check box is toggled, the event handlers call a method which sends three variables to the parent object, the value of the brightness slider, the value of the contrast slider, and the Boolean variable of the inversion checkbox. From here the variables are then passed into the Model class, where the values are passed into the ColourManipulation class and after some matrix multiplication calculations are performed a 5x5 ColorMatrix object is returned with the correct colour matrix. This colour matrix is then applied to the current image before being returned to the Interface class and set as the new image of the current Picture component. The structure of this thread of execution is used as it separates getting the interface data, and setting the updated image from the code that actually performs the algorithm, thus lowering coupling and increasing cohesion.

5.3.3. Numpad

The Numpad user control was created to actively prevent confusion and error input when searching for a patient's images via their CHI number. It prevents keyboard input, eliminating illegal characters and too many digits (if the user was to copy and paste a string of 15 digits into a similar control the system might break) and allows only the correct number of digits to be searched for. This is performed by disabling buttons when they are not ready to be pressed, for example:

Figure 10: The Numpad component while a CHI is being entered

- If the user has no digits entered, they cannot press the "Backspace" button as it is disabled
- If the user has less than ten digits entered, they cannot press the "Go" button as it is disabled
- If the user has exactly ten digits entered, they cannot press any of the numerical buttons, only "Go" and "Backspace" as the rest are disabled
- If the user had exactly ten digits entered and pressed backspace, the "Go" button would become disabled and all numerical buttons would be re-enabled
- If the user had one or more digits and repeatedly pressed backspace until there were no digits entered, the "Backspace" button would become disabled as there is nothing more to delete

Arguably this could be seen as a negative way of taking input, as it immediately dismisses the use of a keyboard to enter the CHI numbers. However, considering Multi-touch functionality is the central theme within the project and time constraints disallowed certain features to be fully made accessible for all scenarios, it was decided that this was an acceptable loss and this could be added in future development. Preventing errors through user input was a highly important factor while developing this application.

5.4. Interface

The interface changed quite significantly throughout development, from the early conceptual interface design using Photoshop to the final product, the interface changed as a result of functionality adjustments, implementation of new features and user feedback through testing. The conceptual image within the Features section (Figure 7, Page 14) was the starting point for the design, alongside a handful of paper-based sketches, which covered the initial functionality. Throughout development the interface was adjusted and optimised in several different ways to create the second main interface design.

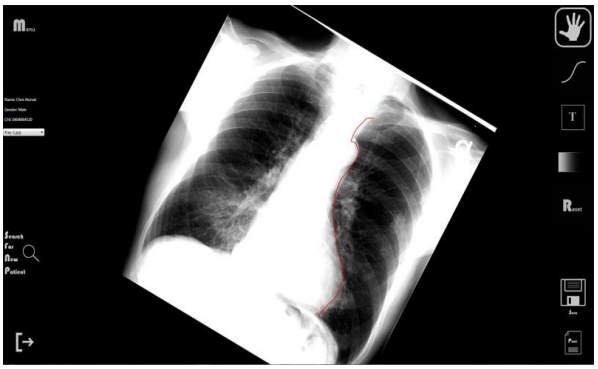


Figure 11: The next step of the interface, with all system functionality

This had all of the functionality built in, and was the interface used throughout testing. While this interface design seemed fine throughout development, user testing indicated that the buttons were not intuitive and difficult to understand. For example, on Figure 11 the method of exiting the application can be seen on the lower left hand corner of the screen, half of a square with an arrow pointing away from it. While this seemed easy to understand at the time of design, representing leaving a box, a doorway, or another symbolic object representing the application, it was suggested during testing that these buttons be modified so all contained text, greatly increasing the clarity of what the button would do if pushed. The interface was modified towards the end of development to reflect this feedback, and a much more intuitive interface was the result.

The final interface can be seen in Figure 12 and contains text information on all of the buttons.

The interface was designed to reflect the application itself, a simplistic tool with a handful of features, but very powerful when used in a specific task. Since the application was designed to be running constantly and essentially a built in aspect of the hardware, the interface was designed not to conform to the standard Microsoft Windows layout. It was the intention for the Windows desktop and other applications not to be accessible in the final system, rather the digital lightbox to be a standalone system. The "Exit" button was used in the prototype to allow a quick exit from the application. Had the prototype been a true representation of the final product there would be no "Exit" button and no way to access the computer desktop. One example of an existing system with this restriction in place is an Automatic Teller Machine (ATM) which may run Windows XP, however, the desktop should not be accessible to the general users. The interface was designed to be clean, non-intimidating for inexperienced users, and to prevent user errors wherever possible. There are no hidden buttons and every function can be accessed with one press of the finger, ensuring precision and accessibility.

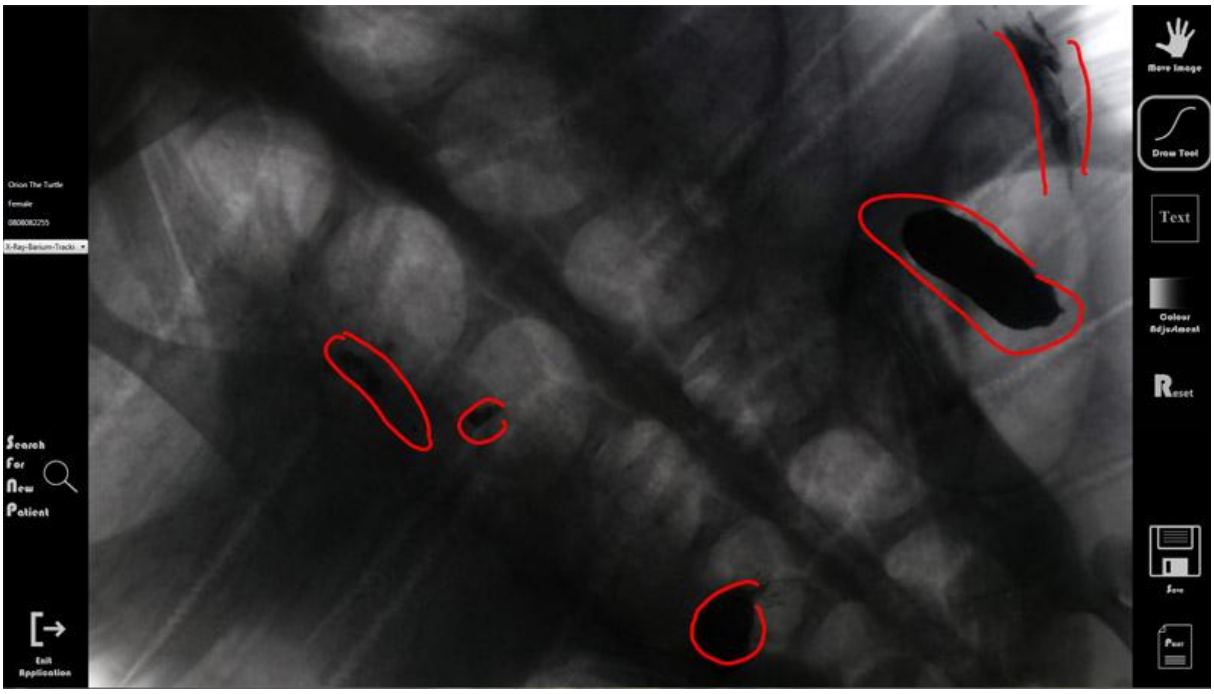


Figure 12: The final interface design

The black and light grey theme was chosen for several reasons. These two colours complement each other when used together, they reflect simplicity and match the colour scheme of many of the images which will be manipulated through the application (black and white X-Rays, MRIs etc.) and in the occasion where a colour picture is used, such as highlighted brain activity on an MRI image, the interface acts as a non-distracting border and allows for easier detection of subtle colour changes within the image itself.

5.5. Error Prevention

Error trapping is designed into the application in order to ensure stable operation even if the user attempts to input invalid data. The application is also designed to prevent incompatible modes of operation, by disabling menu selections that may cause errors or a system crash.

5.5.1. No Multi-Touch Hardware

The first method of error prevention ensures that a Multi-touch monitor, or some kind of Multi-touch hardware is connected and enabled in order for the application to start. As the project was an attempt to simulate a standalone system where Multi-touch is the main method of user input, it seemed sensible to prevent the application from loading and waiting for "Stylus" (touch) inputs to perform any tasks. If the application is started and there is no Multi-touch hardware installed then an error message appears, telling the user that there was no Multi-touch compatible hardware found and asks the user to contact the "Administrator" before the application is terminated.

5.5.2. No Image Loaded

If a user was to attempt to change one of the sliders, brightness or contrast, of the ColourControl component when an image was not loaded, this would cause the application to crash as it would attempt to apply a "colorMatrix" to an object which does not exist. This is why the interface was designed to prevent all graphical functions of the application to be hidden and inaccessible when there is no image loaded. When a patient is found, the right menu bar is hidden until the user chooses which image to load, and that image is successfully loaded. This also prevents errors from other "Modes", such as Draw, attempting to add lines to something that isn't there, and the reference tool adding a label. By simply hiding the interface bar when an image is not loaded maintains the integrity and reliability of the application.

5.5.3. Wrongly Entered Patient CHI Number

The "Numpad" component was designed to prevent wrongly entering a patient's CHI number. The "Numpad" disables buttons which should not be accessible, such as "Go" if there are not exactly ten numbers entered, or any of the numbers if there are already ten digits entered. This ensures that any queries for a patient are done through a ten-digit CHI number rather than, for example nine digits.

The Numpad also prevents user input from the keyboard (although the user can enter numbers through pushing the buttons with a mouse if desired). This prevents illegal characters being entered, such as letters or foreign symbols.

If the user enters ten digits and searches for a patient with that CHI number, and that CHI number does not match a patient, then the label on the Numpad will change from "Please enter the patient's 10-digit CHI number" to "Error, no patient could be found with that CHI number, please try again" before allowing the user to re-adjust the CHI number entered.

5.5.4. No images available

If a patient folder is created and contains a patient.xml file but with no images inside then the patient can be searched for, and the patient information will, as usual, be displayed on the left interface bar. The right interface will remain hidden, as no image is loaded (preventing crashes) and the combo box will inform the user that there are no images available. This prevents any functions from being performed while no images are available, which could potentially jeopardise the stability of the application.

5.6. Setup and Configuration

In order to set up the application on a completely new computer, a few tasks will need to be completed before the application can run. This may, on occasion, require a change in the code, since it searches for "D:\DALPatient". If the new computer is not set up to have a D-drive partition then it will needed to be changed accordingly. Here are some of the other things that will need to be ensured to run the application:

- The operating system must be Windows 7 (due to Multi-touch support)
- The Windows 7 SDK must be installed in order to compile, as well as WPF 3.5
- If no compiling is needed, the system requires the WPF 3.5 redistributable files installed
- A Multi-touch monitor must be set up and plugged in. (A few compatible monitors were tested and are listed in Section 6.2.3.)

In order for the search function to find patients a new folder should be created on D:\ (or wherever specified on edited code - see above). The folder should be called DALPatients and this folder contains all patient folders. To create a patient folder inside DALPatients create a new directory with the patient's 10 digit CHI number as the name, for example "1234567890" and add any images associated with that patient to this folder. Once this is done create a new XML file called "Patient.XML" to the patient's file. This XML file must be in the following structure:

```
<?xml version="1.0" encoding="utf-8"?>
<Patient>
    <christianNames>**FN**</christianNames>
    <surname>**SN**</surname>
    <CHI>**CHI**</CHI>
    <gender>**GEN**</gender>
</Patient>
```

where:
FN is the first name of the patient
SN is the surname of the patient
CHI is the patient's CHI number
GEN is the patient's gender (Male or Female)

so for example....

```
<?xml version="1.0" encoding="utf-8"?>
<Patient>
    <christianNames>Chris</christianNames>
    <surname>Norval</surname>
    <CHI>0404884520</CHI>
    <gender>Male</gender>
</Patient>
```

would be a male patient called Chris Norval, with the CHI number: 0404884520

6. Testing

Two types of testing were performed: User testing and Functionality testing.

6.1. User Testing

Testing the application turned out to be quite a challenging aspect. The targeted users of the system were doctors, clinicians, vets and medical students, although the application could be extended for any job which required intuitive control over images with some minor adjustments to modify the CHI aspect of searching. While the project was focused more towards the use within hospitals it posed a problem managing to obtain feedback from consultants and professionals in the healthcare industry due to their busy schedules. It was decided that medical students were possibly the best candidates for testing the prototype throughout the project due to their eagerness to participate and experiment with potentially futuristic medical tools. However, this could be seen as slightly atypical group when it comes to testing the practicality of the system due to the high percentage of highly computer literate people within the current generation of students', with older generations of staff possibly favouring the older method. Three user tests were conducted: Initial requirements gathering and two usability tests.

6.1.1. Requirements Gathering

The first interaction with medical students came in the form of informal meetings with friends. The project was talked about and some ideas gathered. This was not strictly considered to be testing, and the feedback was not structured and documented in any formal way, however this requirements gathering helped shape the initial

structure of the application. The participants were all fourth year medical students at Dundee University and had not yet specialised in any one specific topic, such as radiology.

6.1.2.1. Usability Testing

Four participants were recruited from the medical department of Dundee University. All four students were on their fourth year of studying medicine, which ensured clinical experience, and none of the students had yet specialised in any one area of medicine. Gender was not taken into consideration and all participants were over the age of eighteen. These four participants agreed to test the system twice and were used to gain a better understanding of how the application may be used within the industry.

6.1.2.2. Procedure for Testing

The first stage in testing the application with medical students was to create the appropriate information sheets and questionnaire, structuring the techniques used for testing, before applying for ethical approval. A document explaining the recruitment process, procedure and conditions of the study was created (see Appendix 2) The test was broken down into four stages; *consent stage*, *interview*, *testing stage* and *questionnaire*. The participants were individually tested and each session lasted no longer than thirty minutes.

The consent process contained a consent form which required a signature (see Appendix 3), as well as an information sheet which provided a background to the project (see Appendix 4). These pages were given to each tester to read through and to sign the consent form, which confirmed that they were happy to participate, and also informing the tester that they could leave the study, or refuse to answer a question without reason and at any time.

After consent was obtained the researcher would discuss the idea with the participant in an attempt to gather any concerns or thoughts that the participant may have had in the form of a conversation. This interview stage allowed the participant to ask any questions and discuss the benefits and drawbacks of a digital system in place. The conversation was very informal to encourage honesty and a relaxing environment for the participant.

The participant was then introduced to the prototype and weas encouraged to experiment with all of the features and buttons. The participant would test the functions of the prototype one by one, and was observed in this stage so that the ease of use of the application to new users could be documented, noting the participant's ease of navigation, understanding of how each function works, as well as if they came across any faults or problems.

Once the prototype was tested the participant would answer the questions on a questionnaire (see Appendix 5) before being debriefed.

Ethical approval was given and this testing technique was conducted on the participants twice throughout development, once at a midway point, and again when the application was finished. It was originally planned to have three stages of testing throughout the development stage, however, feedback from the first testing session indicated that numerous additions would be required to the application. The schedule was re-prioritised and it was decided that the time spent rounding off another iteration of the program and holding testing sessions would be better spent continuing development.

6.1.4. Results of Testing Sessions (Midway)

The feedback received during the first testing session was very useful, with three of the four testers asking for easier control over tools, such as the brightness and contrast sliders, which resulted in the customisation of these controls to increase the selection area and make it easier to select with little or no past practice on a Multi-touch system.

Another aspect of the testing sessions was the ability to observe and take notes of how the user interacted with the prototype while using it. With little practice and understanding it was useful to see how quickly each of the participants got confident of moving, rotating and scaling the image in order to look at areas where they thought a

problem could occur, which greatly varied between each participant. As implied above it was very beneficial to see some of the participants struggle to use their finger to select the "Thumb" of the sliders, or selecting the check box for colour inversion, which led to the interface being updated with larger selection areas.

The questionnaire provided the best feedback throughout testing, with the following results being gathered after the first session:

(Where 1 means Strongly Disagree and 5 means Strongly Agree)

Question	Average Answer
I am heavily involved in the use of X-Ray or MRI viewing equipment	2.5
I can see the benefits in a digital alternative to this system	4.5
I found the digital alternative system to be better than the system currently in use in hospitals	4.25
I would prefer to use the digital alternative if it was available at the hospital or place where I work	4.5
I found the digital alternative system to be intuitive and easy to use	3.5
I found the rotate feature to be useful	3.25
I found the scale feature useful	4.25
I found the move feature to be useful	3.25
I found the draw feature to be useful	5
I found the brightness feature to be useful	3.5
I found the contrast feature to be useful	3.5
I found the colour invert feature to be useful	3.25

The open questions within the questionnaire pointed out that the controls were not particularly easy to use when first testing the prototype. It was requested that some of the buttons be made larger, such as the slider "thumbs" and make the move tool easier to use. All participants found the draw feature particularly useful. The two-finger issue with Multi-touch monitors (full discussion of this problem is given in Section 7.1.) was mentioned by three of the four testers who asked for this issue to be resolved as it made the rotate and scale functions difficult to use.

6.1.5. Results of Testing Sessions (Final)

The second testing session used the same procedure for the study as the first, and the four medical students were used again for testing the application at the end of development. The questionnaire was slightly changed (Section 1 and 2, see below) to allow for adjustments within the application.

Again the participants were observed and this time had the ability to test some of the newer features which were not available during the first testing session, such as Print and Save, as well as being able to use the new interface. It was observed that the participants all found the sliders easier to use, and the interface much clearer to understand, and unexpectedly they all found the free-control feature much more straightforward to use despite no major changes being made to the way the position of the image could be manipulated.

The results to the updated questionnaire were as follows:

(Where 1 means Strongly Disagree and 5 means Strongly Agree)

Question	Average Answer
I found the digital alternative system to be better than the system currently in use in hospitals	4.75
I would prefer to use the digital alternative if it was available at the hospital or place where I work	5
I found the digital alternative system to be intuitive and easy to use	4.25

I found the reference feature to be useful	5
I found the save feature to be useful	3.5
I found the print report feature to be useful	4.75
I found the Search feature to be intuitive and straight forward	4.5
I found the interface to be clear and consistent	4.75

The open questions within the questionnaire provided the feedback that controlling the image was better, but still difficult when rotating or scaling. The print feature could be particularly useful and the reference system was a good way to point out anything that needed a complex explanation.

6.1.6. Other (informal) Testing Sessions

Another informal testing session was done by a visitor to the department, Professor Sara Czaja. With medical software and clinical past experience the information learned from this five-minute session was very important in realising some of the more ambiguous interface buttons. At this point some of the buttons contained text, some contained text and a picture, but most just contained a picture which wasn't overly informative. Professor Czaja advised that the interface should be changed to specify the function of a tool in a more clear and concise way, which was implemented for the final prototype.

Towards the end of the development a meeting was also set up with Suzanne Duce, an MRI technician within the Drug Discovery unit of the Wellcome Trust Research Centre. The meeting allowed for a more in-depth understanding of MRI scanners for scientific use, mainly used for analysis of the effect of different drugs on the physiological functions of mice, as well as seeing how the MRI data are obtained and used to interpret the results of the test. The department used commercial software, known as Amira, for viewing the MRI scans, which allowed the construction of a computerised 3D model of the mouse from the slices[16]. The meeting was useful for understanding the needs of the scientific community, rather than in healthcare, for analysing clinical data. Although it was too late, and too much of a challenge, to implement some of the features that would have been beneficial for the scientific community, the meeting was very informative and provided several points of interest for future development.

6.2. Functionality Testing

6.2.1. White Box Testing

Functionality testing attempts to test cases based on the use cases and an understanding of the code. Test cases can be created and the expected output and correct path of execution can be followed to ensure that the code is working as intended. The tests were based on the Use Cases, listed above.

Test: Search for CHI number **Expected Result**: Allows user to enter CHI number and search for patient, obtains information and then populates combo-box with list of available images **Actual Result**: The user can enter a CHI number and after pressing the "Go" button the interface changes so that patient details are listed on the left and a combo-box appears with a list of images to load **Pass/Fail**: Pass
Test: Free Control Image **Expected Result**: Allows user to move, rotate or scale image on screen based on touch input **Actual Result**: The user can move one finger on the screen to move the image, rotate two fingers onscreen to rotate the image, or place two fingers on screen and move away from each other to stretch the image. While rotating the image, the image can occasionally rotate in the wrong direction or do unexpected things **Pass/Fail**: Problem with monitor - please see Problems section
Test: Draw on Image **Expected Result**: Allows user to "draw" on the image by using a finger

Actual Result: The user can use a finger as a sketch tool by dragging a finger along the monitor. This adds a red line to the image
Pass/Fail: Pass

Test: Enter Text For Image

Expected Result: Allows the user to add a reference point to the image by touching somewhere, or add text by touching the text box and using the onscreen keyboard

Actual Result: The user can touch the image to add a reference point as well as adding text to the text box by touching it and adding it to the onscreen keyboard

Pass/Fail: Pass

Test: Modify Colour Matrix of Image

Expected Result: Allows the user to manipulate the brightness, contrast and pixel inversion based on two sliders and a check box. This updates the image in real-time

Actual Result: The user can modify the brightness, contrast or invert pixels by using the onscreen slider to change the values. These update in real-time

Pass/Fail: Pass

Test: Reset Image

Expected Result: Allows the user to reset the image by defaulting the colour matrix and clearing the Drawpad

Actual Result: The user can reset the image. This causes the colour balance to go back to normal, the lines and reference points on the image disappear, and the text information entered is cleared

Pass/Fail: Pass

Test: Print Report

Expected Result: Allows the user to print a report version of the image

Actual Result: A Print Dialog box appears, and if chosen to print then an A4 page is printed featuring the modified image along with the text information at the bottom

Pass/Fail: Pass

Test: Save Report

Expected Result: Allows the user to save a modified version of the image

Actual Result: A Save dialog box appears, and the image can be saved to the folder

Pass/Fail: Pass

Test: Search For New Patient

Expected Result: Allows the user to search for a new patient, clearing all patient information and modifications to the image

Actual Result: Keypad appears and a new CHI number can be searched for

Pass/Fail: Pass

Test: Add Invalid Input To Numpad While Searching

Expected Result: Prevents the user from entering more than ten digits, less than zero digits, invalid characters (symbols and letters) and prevents the "Go" button being pressed if less than ten digits are entered

Actual Result: "Backspace" button is disabled if no digits entered, "Go" button is disabled if less than ten digits are entered, number buttons are disabled if ten digits have already been entered

Pass/Fail: Pass

Test: Search for CHI with no Patient.xml File at Location

Expected Result: Returns negative search

Actual Result: After pressing "Go" the label is changed to inform the user that there was no patient found

Pass/Fail: Pass

Test: Search For Patient With No Images

Expected Result: Interface remains hidden, application does not crash

Actual Result: The user cannot access any buttons which would cause the application to crash

Pass/Fail: Pass

6.2.3. Compatibility Testing

Compatibility Testing is a method of performing non-functional tests on an application in order to understand what systems will be able to handle the application and under what conditions. The following Compatibility tests were performed:

- Screen Resolution: Originally developed for 1920 x 1080 screen resolutions, the following resolutions were also tested: (1768 x 992), (1680 x 1050), (1600 x 1024), (1600 x 900), (1440 x 900), (1360 x 768), (1280 x 1024)*, (1280 x 960)*, (1280 x 800)*, (1280 x 768)*, (1280 x 720)*, (1176 x 664)*, (1152 x 864)*, (1024 x 768)*, (800 x 600)* all with success (*Note: Although the interface supports all resolutions listed above, the lower resolutions can occasionally warp the Picture image while rotating or scaling if the image source is much higher. Although no functionality is limited by this it can be quite noticeable)

- Operating system: The system Operating System was looked into, and it was clear from the start that the application would not run on any other platform than Windows 7 due to development within WPF and Multi-touch. Future releases of Mono, for Linux, may allow the application to run on various Linux distributions but currently there are no alternatives available

- Hardware (Monitor): Tested on an Acer T230H Multi-touch monitor[25] and an Ilyama Prolite T2250MTS monitor[26], both with equal success

Unfortunately due to a limitation with hardware the following compatibility tests were not achieved, however, with more resources these tests would ideally be conducted in the future:

- Hardware (RAM, CPU, GPU)

- Patch versions of Windows 7 (only version 6.1 (Build 7600) tested)

7. Problems

7.1. Hardware

Throughout the course of the project several problems became apparent, ranging from limitations of the Windows Presentation Foundation (WPF) framework to performance issues, however the majority of these were relatively easy to fix or to bypass in some way. The biggest problem was caused by the driver issue with the Acer T230H Multi-Touch Display monitor. The monitor which was used for development came with a fundamental flaw, an apparently documented issue with 2-camera displays[4]. This caused the input points to become confused when more than one input point is registered. The two input points would distance themselves from the actual point of contact from the finger, and go in completely different directions. The problem was eventually discovered to be that when one finger is detected by a camera, and a second finger goes into the "Shadow" of the first finger the monitor fails to successfully track what direction each finger should then go, as seen in Figure 13 and Figure 14.

After uploading a demonstrative video of the problem[5] and conducting research it was discovered that this is a driver problem and Acer are responsible for releasing updated drivers which would fix the issue. However, after contacting Acer the manufacturer would not reply or comment on the issue. This issue caused severe problems while developing and testing, turning all Multi-touch features unreliable and therefore making the fundamental functionality, rotation and scaling of the image, unpleasant to use, which was clearly reflected during the first testing sessions. With little that could be done to counter this issue the only realistic option was to ignore it throughout development and wait for a fix from Acer which had no estimated date of release. Other Multi-touch monitors using the two-camera system also suffer from this problem, however the effect that this has varies by the monitor's drivers.

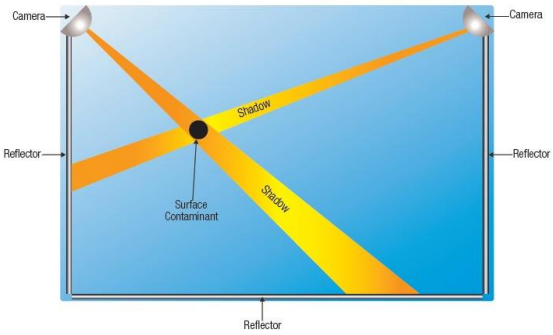


Figure 13: One Finger Input with a two-camera display Multi-touch monitor, highlighting shadows.

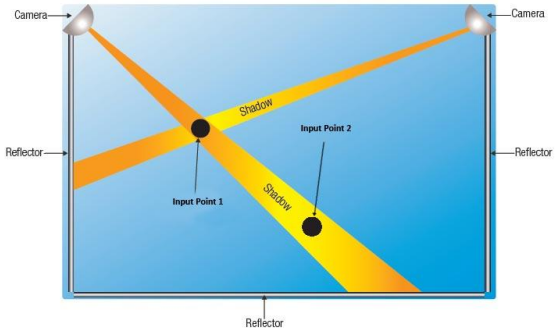


Figure 14: Two Finger Input with a two-camera display Multi-touch monitor where one input point is in the "shadow" of the other input point, causing loss of accuracy.

As well as the driver issue with the Acer T230H Multi-touch Monitor, hardware consistently remained a headache throughout the entire development cycle. Initially development was planned using the "home-made" Multi-touch table within the department, however this had problems of its own relating to calibration and application issues. After changing the Multi-touch platform from "Touchlib" to "Tbeta" and the problem consisting it was decided that the table was simply not fit for the needs and accuracy of the application. This left the project with two semi-realistic alternatives. The first being to wait for the release of Windows 7, which was due in October and included full Multi-touch support. The second was to attempt access to a Microsoft Surface system, but with a steep price of £10,000[6] and the university with no obligation to purchase one it seemed like this would be out of the question, which ironically it wasn't as the university had one donated to it over Christmas by NCR. This left waiting for Windows 7 and a Multi-Touch monitor as the best alternative, however after the release of Windows 7 on October 22nd 2009 it was discovered that all commercial Multi-touch monitors had been delayed for release until mid-December, pushing the development back to mid-January due to the festive holidays.

Another minor problem with the hardware, especially with testing, was the issue of calibration and practice. When users tested the system they sometimes found difficulty in getting an input point exactly where they wanted it to go, such as grabbing a slider or circling a specific area. This was simply found to be a lack of practice and understanding of how the Multi-touch monitor gathers an input point. When a user places their finger on the screen the monitor attempts to average the entire area that is being touched and registers the input point as the centre of here. In Figure 16 an illustration demonstrates this. When a users finger presses the screen there is a large area touching (shown as the black circle). This is averaged to find the centre point, which is used as the point of input. This problem could be solved by improving the way in which the Multi-touch drivers recognise the input point from the users touch.

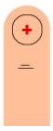


Figure 15: How the Acer T230H Multi-touch monitor calculates an input point based on a finger

One problem, which was not seen during testing but is well documented for all forms of touch-screen monitors, is repetitive strain injury (RSI), a condition caused by repeatedly inducing unnatural stress into an area of the body. This could be caused by continually lifting an arm to touch the monitor, and can easily be prevented by using the system in moderation.

7.2. Software

With the semi-reliable hardware obtained the development of the prototype started, however the development was not without its own problems. The framework which would allow for Multi-touch development was not in a supported release and would not be until April 12th 2010[7], however a few alternative options were available, the best one being the use of Windows Presentation Foundation (WPF) 3.5 alongside the developer beta release of the Windows 7 SDK.

7.3. WPF

WPF with the Windows 7 SDK allowed for Multi-touch development on the recommended WPF subsystem which simplified handling Multi-touch input but at the cost of losing the GDI+ component which allowed for very powerful control over 2-Dimensional images and their content. While a work-around was eventually found for converting WPF images to GDI+ images, and vice versa, during initial development it was very difficult to gain access to some important image information, such as the height, width, and pixel data. This caused alternative algorithms to be created for seemingly simple features such as drawing to the screen due to the lack of the Graphics class, although lines could be drawn by placing a canvas element with a translucent background over the image, which would rotate, scale and move with the X-Ray or MRI image.

As stated above a work-around was eventually discovered which would allow for the conversion of WPF images to GDI+ images, which was used in the colour manipulation algorithm. This seemingly worked well apart from a significant performance issue with the method Graphics.DrawImage(), which, while used with the slider to determine what colour matrix to apply to the image, created a significant latency delay of about 1.5 seconds. This delay was significant and completely removed the idea of real-time feedback. A solution to this, although not an ideal one, was to remove the real-time aspect altogether and have the brightness and contrast only update when an Update button is pressed. The solution may seem like a step backwards but allowed for smooth feedback of the slider handle, which kept the application's flow intact.

The slider element was the next problem encountered. Due to the high resolution of the screen the slider handle, or "thumb", was too small to easily select with a finger. With no easy way of controlling the scale or appearance of the slider element a way of controlling the appearance was eventually discovered through the use of templates and Style code inside the XAML file. This allowed for the overloading of the default appearance and therefore the thumb of the slider. A similar problem was found with the checkbox, but considering the simple functionality of it this could be fixed by placing an invisible button over the top of the entire area, which would toggle the checkbox's value.

One issue raised through testing was that the images were scaling and rotating from the centre point, rather than based on where the input points are on the image. This was documented on countless websites and is a downfall with the beta touch library within WPF. While a solution can be put in place, it was decided that this issue should simply be put down to "Future Development" due to time constraints and the unusually large amount of code required to fix this issue.

WPF had one, quite significant flaw which was very noticeable through development. This bug, which caused the system to bombard the user with dialog error messages and print/save windows was a problem with the way WPF handles these windows when called by a touch input point. When a button's event caller is set to "MouseLeftButtonDown" it accepts both mouse input (by left clicking) and touch input via the user's finger. The problem occurs when a print dialog, save dialog or dialog message box is called up after the user presses the button by touch. After the method is called and finished it returns back to an idle state, however the next touch point that the user makes, whether it is on the Picture, on one of the buttons, or even on the exit icon, will result on the wrong method being called, calling the method with the dialog box. This means that if a user tries to print an image and then tries to perform any other task, no matter where they touch on the screen will again call the Print event and bring up the print dialog box until the application is terminated. "StylusButtonDown" was a solution to this. When used instead of "MouseLeftButtonDown" the problem is no longer seen, however the trade-off is that this only accepts touch input, and therefore the button cannot be pressed by mouse. It was decided that the actual functionality was initially more important than giving the user options, especially when the application is designed for Multi-touch, and with that in mind "StylusButtonDown" was used and the WPF bug documented. This again highlights the problem of working with new technologies and working with beta development libraries.

8. Conclusion

8.1. Summary

The original aim of the project was to develop a Multi-touch application, that would act as a stand-alone device for viewing X-Ray and MRI images. The application was created to experiment with completely different and new technologies to prove the point that such new systems are still missing from the day-to-day lives of healthcare specialists. With healthcare being such an important part of everyone's lives it would be expected that doctors have the very best equipment available. Sadly, this is not the case. The main three goals of the project were that the system was to be intuitive, useful and commercially affordable. The Digital Lightbox surpassed these requirements.

The project expanded greatly throughout development with the addition and refinement of many features, which evolved the application from simply an observational tool to a diagnostics station with report and sharing functionality. The original aims were all met, proving that new technology, such as Multi-touch monitors can change the way that people use computers. This opens the door to updating an almost unlimited number of hands-on tasks in the daily lives of professionals, which can be improved by this kind of technology. The system was tested by medical students who universally agreed that they would prefer to have a system like this in place throughout hospitals. They found the interface to be easy to understand, the features to be self-explanatory and the concept to be novel. Another interesting point to note is that it was realised midway through development the wide range of tasks that this tool could be used for. Veterinarians could also benefit from this application, as could lecturers within universities who teach medical students how to diagnose from clinical images. In fact, any task that required the analysis of images could use the system, from town planning to architects, artists to product designers, and from forensics to biology, the possibilities are almost limitless.

With the requirement to control costs in hospitals, it was interesting to compare the cost of an existing wall-mounted lightbox with the equipment used for this project. Clearly, it is not a perfect comparison, due to the additional expenditure incurred with the digital lightbox, such as networking and providing a capable Windows 7 computer. However, for a rough idea of the cost difference a wall mounted light box can cost around $430[35] whereas the monitor used for development, the Acer T230H costs around $380[36], proving that even with the additional expenditure of setting up a viable network and server solution it is commercially viable. With future development of the application, alongside training of healthcare professionals and experts in its use, it may be more expensive not to use the digital lightbox when taking into account the errors that may be prevented and potentially the lives that could be saved.

8.2. Evaluation

Overall, the project is a good solution to the problems faced with medical imaging and diagnostics. The functionality, although somewhat basic, allows for a wide variety of tasks to be completed and different kinds of images to be observed. The software is designed to maximise error prevention in order to maintain reliability and simplicity for both technologically experienced and inexperienced users. Many of the functions allow for mouse and keyboard input as an alternative to just Multi-touch, catering to as wide an audience as possible. This is important for designing an application for a wide range of industries. It was attempted to conform to the Model View Controller architecture wherever possible, although due to the high number of interface-related operations relating to the view and the controller these sections of the architecture are still larger than a computer scientist may initially expect. Throughout development of the project several problems occurred due to the use of new hardware or because of the use of WPF, which caused difficult decisions to be made from a developers perspective, a few of which are listed below.

One criticism of the application is the lack of full keyboard and mouse support. While in order to maintain a maximum level of accessibility it would be beneficial to provide an alternative to the Multi-touch monitor unfortunately that was not possible with this project. Adding both keyboard and mouse support as well as touch input would allow the user to have a greater choice in the way that the system operates. All features work with

only Multi-touch input, and a keyboard and mouse are not needed, however full support for these input methods would ensure that the application can be used the way that a user wants to interact with it.

As listed in the Problems section there is a flaw with Multi-touch and WPF that causes event handlers to loop infinitely on occasion when a dialog window or box appears. Normally when using the "MouseLeftButtonDown" event caller both Multi-touch input and mouse input generate a call to the event. However, in the event of an error message, print dialog or save dialog, using Multi-touch to call the class results in a WPF bug which repeatedly re-calls the event after completing the intended task, such as saving or printing. This causes infinite numbers of print or save dialogs to appear after each other. The solution to this was to change the event caller from "MouseLeftButtonDown" to "StylusButtonDown" which is similar in nature to "MouseLeftButtonDown" but with the significant difference that a mouse cannot call the event. While this means that a user cannot use a mouse to generate a report or save to a file it does prevent against the loop error. This is in part the reason why keyboard and mouse input are not as widely supported as Multi-touch input. Due to WPF's current flaws it was a straight choice between the two.

Another criticism is that due to the lack of availability throughout development of an in-place NHS system, and due to the lack of knowledge of these systems, it was not possible to design accurately the application to have a similar network layout or file storage model in place. This was the reason that the search function is designed the way it is, searching for example for "Patient.xml" within "D:\DALPatients\0404884520\". Part of the research involved coming up with a technique to simultaneously check if the directory is that of a compatible patient, as well as loading the patient information, such as name and gender, and this seemed a suitable, although nowhere near perfect, way of presenting how the system might operate within the industry. Another limitation of the application is that it would not open an actual X-Ray or MRI data file due to a lack of understanding how these files are constructed and not having access to some of these files. It was decided that the application would load Jpeg images, simply to demonstrate the functionality and applicability of the project using a prototype, rather than a perfect solution.

The structure of the code is based loosely on the Model View Controller architecture in order to increase cohesion and limit coupling by reducing dependencies within the classes. While some functions conform well to this structure, e.g. Colour Manipulation, which takes the values from the Controller and passes them to the Model, which performs the appropriate algorithms before passing the adjusted image back to the View, others do not specifically rely on the Model at all. This is due to the fact that the application is heavily based on graphical input and interface components, and it seemed appropriate to simply take data from the controller and use it to manipulate the view in some way, such as the addition of lines to the Drawpad for Draw, the addition of labels via the referencing tool or the manipulation of the image constraints when using Print. It is accepted that the code is not strict in relation to the MVC architecture, and the structure of code would not be perfect in a purist's eyes. However, time constraints and a lack of experience developing for such a real-time graphical application based on Multi-touch are the reasons for an imperfect code architecture. The difficulty of developing for a new technology was highlighted from the start, and the lack of resources available for developing a pure and tidy application for new hardware, on a relatively new framework, resulted in the lack of a strictly conformed-to architecture.

8.3. Future Work

One of the interesting aspects of developing this system was that due to its potential size and a lack of resources it has allowed for some very creative suggestions for future work. While it was simply not possible to reach a fully "completed" prototype within the timescale of this project due to the near infinite number of features that could be added or tested, the prototype was designed to contain only basic functionality, although additional features could be added in the future. With unlimited time and resources the application could become more powerful and useful, and some of the ideas for future development are listed below.

A relatively simple feature would allow multiple images to be loaded at once, for direct comparison. This would allow for example, different angles of an X-Ray of a patient to be compared, which could be extremely useful for doctors using the system.

One idea that was considered at the start, but which was not applicable for the first prototype, was to have a barcode scanner built into the side of the wall-mounted monitor. Patient files in hospitals contain barcodes, containing the patient's CHI number. If this could be used to search for a patient's images by simply scanning the CHI, rather than entering it via the "Numpad" component, this would save time and make the application more flexible and reliable.

Auto labelling was an idea that stemmed from current computer vision research. The idea was that a chest X-Ray, for example, could be loaded into the application and labels could be toggled on and off, which named different parts of the body within the X-Ray, such as the patient's heart. This could also be extended to look into auto-diagnosing based on edge detection or colour properties, for example the system could inform the doctor straight away that the patient's 5th rib on the left side is fractured, or if there is some discolouration or unusual shading in the patient's liver.

Automatic brightness and contrast adjustment would be an interesting feature to develop as it would load the X-Ray and attempt to adjust the contrast and brightness to make the image as clear as possible. Applications like Adobe Photoshop can perform this, so there are algorithms available for this task.

Simple adjustments to the interface, controlled by a configuration file, would make the application more flexible. The application could be set to display a patient's date of birth rather than their CHI number, change the directory that contains the patient files, change the size and layout of interface buttons, change the colour of the draw tool and reference tool, as well as perhaps an undo and redo feature.

Speech recognition could be used to speed up the process of a doctor using the reference tool. When a reference point is placed they could simply press the text box, hold it down and say "Fifth rib fractured on the left side" into a microphone, which would convert the sound to text and add it to the text box.

It would be beneficial to actually read in raw X-Ray and MRI images. This would make the application much more realistic in the way it would potentially work in the industry, loading these files rather than Jpeg images. This could also be extended to look into creating and displaying 3-Dimentional models from a series of MRI slides. There are applications to do this task, such as Amira, so it could be implemented.

Full support of keyboard and mouse, as well as Multi-touch input would be a beneficial feature, increasing the number of options that a user would have if they did not want to use the Multi-touch feature.

Possibly the most exciting and intriguing idea for future development is to redevelop a version for tablet PCs. Tablet PCs are set to become the next technology to have widespread use, and the flexibility of the application being portable within the hospital would have several beneficial implications. Each doctor could carry around a tablet computer and when talking to a patient they could simply load the application, find the patient's details and then instantly diagnose the images. This would also have implications for cost as it would be much cheaper to use a tablet PC than a Multi-touch monitor and computer, costing approximately $400 per tablet computer, without the need for anything else.

If more time and resources were available some of these ideas would make for a potentially interesting commercial product. The intention would be to re-develop a tablet pc version of the application, with a built in camera which would function as a barcode scanner and image gathering device, as well as many of the adjustments listed above. The project would attempt to modernize many of the techniques used within healthcare today, to make doctors' jobs quicker and easier, with everyone benefiting from the change.

9. References

[1] Lamont Wood, 1994. The Man Who Invented the PC. [Online] Available at: http://www.americanheritage.com/articles/magazine/it/1994/2/1994_2_64.shtml [Accessed 18 April 2010].

[2] Blinkenlights Archaeological Institute, 1999. What was the first personal computer? [Online] Available at: http://www.blinkenlights.com/pc.shtml [Accessed 18 April 2010]

[3] D.Hailey. Macintosh and the first GUI. [Online] Available at: http://imrl.usu.edu/OSLO/technology_writing/004_003.htm [Accessed 18 April 2010]

[3] Brad A. Myers, 1996. A Brief History of Human Computer Interaction Technology. School of Computer Science, Carnegie Mellon University. [Online] Available at: http://dimetic.dime-eu.org/dimetic_files/Lect%2012%20to%20Windrum%20-%20MalerbaEtAl.pdf [Accessed 18 April 2010]

[4] Microsoft Developer Network Forums, 2010. Problem with Windows 7, Hardware Drivers or Acer T230H Multitouch Monitor? [Online] Available at: http://social.msdn.microsoft.com/Forums/en-US/tabletandtouch/thread/9f2b2508-a80a-4b5d-828d-96cfad883404 [Accessed 18 April 2010]

[5] Youtube, 2010. Demonstration of Multi Touch Input problem on Acer T230H. [Online] Available at: http://www.youtube.com/watch?v=qR23EOTkZBk [Accessed 18 April 2010]

[6] Microsoft, 2009. Microsoft Surface Order Form. [Online] Available at: http://go.microsoft.com/?linkid=9686818 [Accessed 18 April 2010]

[7] Rob Caron, 2010. Visual Studio 2010 and .NET Framework 4 Launch Date. [Online] Available at: http://blogs.msdn.com/robcaron/archive/2010/01/13/9948172.aspx [Accessed 18 April 2010]

[8] R. Hopkins, C. Peden and S. Gandhi, 2010. Radiology for Anaesthesia and Intensive Care, Second Edition. [Online] Available at: http://assets.cambridge.org/97805217/35636/excerpt/9780521735636_excerpt.pdf [Accessed 18 April 2010]

[9] BrainLAB AG, 2009. Discover Digital Lightbox. [Online] Available at: http://www.brainlab.com/scripts/website_english.asp?menuDeactivate=1&articleID=2514&articleTypeID=276&pageTypeID=6&article_short_headline=Discover%20Digital%20Lightbox© [Accessed 18 April 2010]

[10] CheapLaptops, 2007. Microsoft Surface Diagram: How it all works. [Online] Available at: http://www.cheaplaptops.org.uk/20070601/microsoft-surface-diagram-how-it-all-works/ [Accessed 18 April 2010]

[11] A.Mccue, 2007. NHS completes Pacs digital x-ray project. [Online] Available at: http://www.zdnetasia.com/nhs-completes-pacs-digital-x-ray-project-62035884.htm [Accessed 18 April 2010]

[12] A. Castle, 2009. Build your own multitouch surface computer. [Online] Available at: http://www.maximumpc.com/article/features/maximum_pc_builds_a_multitouch_surface_computer [Accessed 18 April 2010]

[13] Microsoft, 2009. What is Microsoft Surface. [Online] Available at: http://www.microsoft.com/surface/en/us/Pages/Product/WhatIs.aspx [Accessed 18 April 2010]

[14] D. Robinson, 2009. Packard Bell reveals Windows 7 multi-touch products. [Online] Available at: http://www.v3.co.uk/v3/news/2249322/packard-bell-reveals-windows [Accessed 18 April 2010]

[15] P. Mann, 2009. Acer announces multi-touch T230H monitor. [Online] Available at: http://www.hexus.net/content/item.php?item=20946 [Accessed 18 April 2010]

[16] Amira, 2010. Product Overview. [Online] Available at: http://www.amira.com/amira.html [Accessed 18 April 2010]

[17] Jaime Rodriguez, 2009. On WPF, Windows 7 and Silverlight. [Online] Available at: http://blogs.msdn.com/jaimer/archive/2009/05/27/wpf-4-and-net-framework-4-beta-1-list-of-features-totrack.aspx [Accessed 18 April 2010]

[18] 3M, 2010. Optical, Surface Acoustic Wave, and Bending Light. [Online] Available at: http://solutions.3m.com/wps/portal/3M/en_US/TouchSystems/TouchScreen/Information/TechandAppBriefs/ [Accessed 18 April 2010]

[19] Microsoft, 2009. Online Training, Multi-touch. [Online] Available at: http://channel9.msdn.com/learn/courses/Windows7/Multitouch/ [Accessed 18 April 2010]

[20] MSDN, 2008. Windows Touch: Developer Resources. [Online] Available at: http://code.msdn.microsoft.com/WindowsTouch [Accessed 18 April 2010]

[21] R. Townsend, A. Tsao, 2008. Windows 7: Developing Multi-touch Applications. [Online] Available at: http://channel9.msdn.com/pdc2008/PC03/ [Accessed 18 April 2010]

[22] MSDN, 2010. Ink, Windows Touch, and Tablet PC Development Forum. [Online] Available at: http://social.msdn.microsoft.com/Forums/en-US/tabletandtouch/threads [Accessed 18 April 2010]

[23] NHS, 2008. CHI. [Online] Available at: http://www.dwfchp.scot.nhs.uk/article/uploaded/DunfWFCHPWebsite-CHI.pdf [Accessed 18 April 2010]

[24] Microsoft Training, 2009. Build a Multi-Touch Picture-Handling Application. [Online] Available at: http://channel9.msdn.com/learn/courses/Windows7/Multitouch/Win7MultiTouchManaged/Exercise-1-Build-a-Multi-Touch-Picture-Handling-Application/ [Accessed 18 April 2010]

[25] Acer, 2010. T230H [Online] Available at: http://www.acer.co.uk/acer/productv.do?LanguageISOCtxParam=en&kcond61e.c2att101=68966&sp=page16e&ctx2.c2att1=17&link=ln438e&CountryISOCtxParam=UK&ctx1g.c2att92=174&ctx1.att21k=1&CRC=590984170 [Accessed 18 April 2010]

[26] Ilyama, 2010. ProLite T2250MTS-1. [Online] Available at: http://www.iiyama.com/ms_GL/Product/category/5/product/182 [Accessed 18 April 2010]

[27] Nathan Lineback, 1996. Xerox Star. [Online] Available at: http://toastytech.com/guis/star.html [Accessed 18 April 2010]

[28] Bruce Damer, 2000. The Xerox Star 8010 "Dandelion". [Online] Available at: http://www.digibarn.com/collections/systems/xerox-8010/index.html [Accessed 18 April 2010]

[29] J. Broz, T. Dimiropoulos, A. Schallmo, M. Younus, 2009. Touch Screen Technologies. [Online] Available at: http://courses.ece.illinois.edu/ECE317/presentations/Touch_Screen_Pres.pdf [Accessed 18 April 2010]

[30] HP Computer Museum. HP-150. [Online] Available at: http://www.hpmuseum.net/display_item.php?hw=43 [Accessed 18 April 2010]

[31] TED, 2006. Jeff Han demos his breakthrough touchscreen. [Online] Available at: http://www.ted.com/talks/jeff_han_demos_his_breakthrough_touchscreen.html [Accessed 18 April 2010]

[32] [Online] Available at: http://www.macworld.com/article/54769/2007/01/iphone.html [Accessed 18 April 2010]

[33] M. Honan, 2007. Apple unveils iPhone. [Online] Available at: http://d5.allthingsd.com/20070530/microsoft-surface/ [Accessed 18 April 2010]

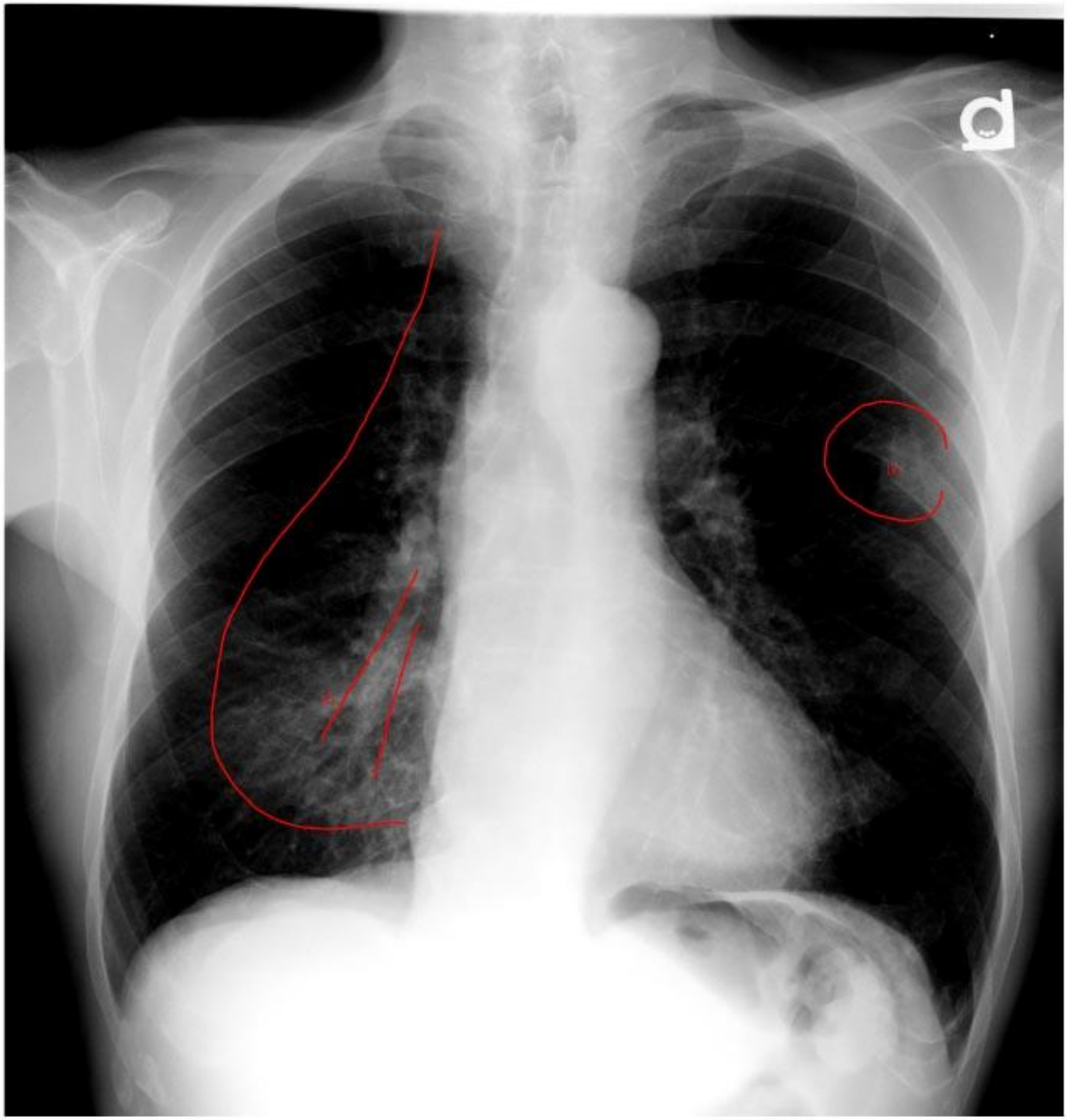
[34] Y. Kiriaty, 2009. MultiTouch Capabilities in Windows 7. [Online] Available at: http://msdn.microsoft.com/en-us/magazine/ee336016.aspx [Accessed 18 April 2010]

[35] MSEC, 2010. Deluxe X-Ray Illuminators. [Online] Available at: http://diagnostic-supplies.medical-supplies-equipment-company.com/product/ppf/id/17595/new_prod_full.asp [Accessed 18 April 2010]

[36] Amazon, 2010. Acer T230H bmidh 23-Inch Wide Touch Screen Display. [Online] Available at: http://www.amazon.com/Acer-T230H-bmidh-23-Inch-Display/dp/B002TLTEAA/ref=sr_1_1?ie=UTF8&s=electronics&qid=1271250855&sr=8-1 [Accessed 18 April 2010]

[37] A. M. Piper, R. Campbell & J. D. Hollan, 2010, "Exploring the Accessibility and Appeal of Surface Computing for Older Adult Health Care Support", Human-Computer Interaction Lab, University of California. [Online] Available at: http://hci.ucsd.edu/ampiper/pap0124-piper.pdf [accessed 24 April 2010]

[38] NUI Group Authors, 2009. Multi-Touch Technologies. [Online] Available at: http://nuicode.com/attachments/download/115/Multi-Touch_Technologies_v1.01.pdf [accessed 24 April 2010]



Name: Chris Norval, Gender: Male, CHI: 0404884520
Reference information:

[1] fractured rib
[2] discolouration

Appendix 1: An example of a page generated by the Print tool

The Digital Alternative to Lightbox (DAL) project attempts to look into how applicable commonly available technology can be for doctors. This is done through attempting to create a computerised version of the existing method for viewing X-Ray images using a wall mounted light-screen. The goal is to create an intuitive and useful system which can do things that cannot currently be done using this pre-existing system, with functionality which doctors may find beneficial in the analysis of an X-Ray or Magnetic Resonance Image. DAL is an Honours project from a fourth year student at the University of Dundee, and runs from July 2009 to June 2010. Throughout the project, feedback will be obtained from medical students at the university as well as healthcare professionals, which will shape the design and functionality of the system to create a user-oriented and easy to use device.

Recruitment: Participants will be recruited from the medical department of Dundee University and Ninewells Hospital to test and give input into the system. All students will be on the fourth year of their course, or above, to ensure clinical experience. This shall be done through contacts within the Dundee University School of Medicine to recruit students, and through contacts, if possible, through the University of Dundee's Computing staff to recruit healthcare professionals. If the numbers are not sufficient, an email shall be placed on Hermes, an automated email sent out every week to staff and students at the University, appealing for volunteers. All volunteers will be over 18 and gender is not taken into consideration.

Procedure: The procedure will involve a brief conversation about the participant's thoughts on such a system, and whether or not the participant would find such a system useful in their day-to-day working schedule at the hospital. This will be a strictly informal interview to encourage honesty and a relaxing environment for the participant. The next stage of the procedure will involve the participant trying out the prototype system, and attempting to carry out common activities on the system, however they shall be informed that the system is a prototype, and may not be fully functional. The final stage will involve the participant filling out a questionnaire with open and closed questions relating to the workings of the system. The procedure should take approximately 30 minutes or less.

Conditions and Consent: Each participant will be given an informed consent form to sign. The participant will be given the right to choose not to do either of the last two stages, or for notes to not be taken during the first stage of the procedure, as well as the right to unquestionably, and without punishment, withdraw from the study at any time. All data obtained from the study will be strictly confidential and stored on an encrypted, secure digital hard drive, and will only be used to further develop the system and to present statistics to peers at the University of Dundee. Participants will be stored as code for reference, for example the first participant shall be P1, the second shall be P2, and so on.

Contact Information: During the procedure the interviewer will attempt to answer any questions that the participant has, however if any other questions emerge after the procedure has taken place, the participant is encouraged to contact Chris Norval at c.norval@dundee.ac.uk or on:
0795 8798 730.

Appendix 2: The procedure document for the testing sessions

Dear Participant

Thank you for your interest in this project. This document explains the structure of the study, and what it will involve. Several tick boxes are provided in this document and can be ticked if you do not wish for a certain aspect of the study to be included with the other results. Please read this document clearly and sign at the bottom to state that you understand and accept the conditions of this study. If you have any questions, feel free to ask the interviewer/researcher. Participation is voluntary and participants can withdraw at any time without giving a reason and without penalty.

Structure of study

The researcher will begin by briefing you on the purpose of the study, and the project in question. An information sheet will be provided and you will be asked to read through this, which briefly explains the functions and instructions for using the software. At this point an informal interview will take place, where you are encouraged to talk about the concept of the system with the interviewer, and share any thoughts or ideas you have regarding the project. For example, you may think it would be worthwhile for a certain feature to be implemented due to either positive or negative past experiences you may have had. The interviewer will be taking notes throughout the conversation, which are strictly confidential.

If you do not wish for these notes to be taken during the conversation, please tick here ☐

The next stage of the study will involve you trying out the prototype system, and experimenting with what the system can do. The system will be on a computer screen, however may not be fully functional, and may consist of pictures if at an early stage. During this process, the interviewer will observe how comfortably you navigate through the system, and any difficulties you may have.

If you do not wish for this stage to be included in the study, please tick here ☐

The final stage of the study will involve you filling out a questionnaire relating to the system. The questionnaire will consist of open and closed questions. The open questions will ask you to tick a box on a scale, based on how much you agree on the statement. The closed questions will ask you to write a very brief description of your thoughts on the subject. You can choose to miss out any of the questions on the questionnaire without reason.

☐

If you do not wish for this stage to be included in the study, please tick here

Points to consider

- All data is strictly confidential, and will be kept only as long as the project requires it on an encrypted, secure hard drive
- You are helping to evaluate a new system. You are not being tested
- There are no right or wrong answers, and honesty is encouraged
- There is no risk involved with participating in this study, however if you have any medical conditions which you feel may be relevant e.g. arthritis or epilepsy, you are encouraged to inform the interviewer or leave the study

Your participation in this study is greatly appreciated, and any feedback, suggestions or data will be used to improve the current system.

Please date and sign below to confirm that you understand and accept the conditions of this study, you are over 18 years of age, and that you are happy for the data provided to be used in future implementations of the system.

Name of participant: _____

Signature: _____

Date: ___/___/_____

Witness of Researcher: _____
(not to be completed by participant)

If any questions arise during the procedure, please feel free to ask, however if any questions arise before or after this period, please contact the interviewer at c.norval@dundee.ac.uk or on 0795 8798 730 .

Appendix 3: The consent form for the testing sessions

A Digital Alternative to Lightboxes

Information

The Digital Alternative to Lightboxes project is developing a software system to research a possible alternative to light boxes, currently used in NHS hospitals, for viewing X-Ray and Magnetic Resonance Imaging data. It is an Honours Project within the University of Dundee and runs between July 2009 and June 2010.

The system will be built to allow for use on a Multi-Touch Monitor, a touch screen monitor which can sense more than one input point, or multiple fingers at any one time. This shall allow for the following functions of the software (subject to change):

Function	Method
Rotate the image	Place two fingers on the image, and rotate wrist in desired direction
Scale the image	Place two fingers on the image, and move one finger further away from the other, or both away from each other at the same time
Move the image	Place one or more finger on the image, and move to the desired location
Crop the image	Place two or more fingers on the image, and use another finger to draw a line horizontally or vertically across the screen. This will give a crop window that can be resized, moved, or cancelled before being confirmed
Draw on the image	Select the pen tool from the sidebar, and use one or more fingers on the screen to tell the system where to draw
Increase/Decrease image brightness	Select Image Brightness tool from the sidebar, and adjust slider before pressing Confirm
Invert Colours	Select Invert Colours button from sidebar once to toggle inverted colours, or again to go back

The feedback obtained from participants will be used primarily for making the system more intuitive, easier to use and more functional, and secondary for statistical comparisons between the system and the current method of analysing X-Ray or Magnetic Resonance Imaging data. The first stage will be an informal interview, based on thoughts and opinions. The next stage will involve the participant testing the prototype system before filling out a questionnaire, consisting of open and closed questions. Participants will be asked to sign a consent form saying that they are willing to participate in the study, and that they understand the conditions of the procedure. The consent form will

explain that the participant can withdraw at any time, choose to have certain stages of the study omitted, and what the data obtained shall be used for, and how it shall be stored.

If you would like to know more about this project, or any questions arise, please contact Chris Norval at c.norval@dundee.ac.uk or 0795 8798 730 .

Appendix 4: The information sheet for the testing sessions

A Digital Alternative to Lightboxes

Please tick the appropriate boxes with an X depending on how much you agree with the statements, with 1 meaning strongly disagree
2 meaning somewhat disagree
3 meaning no opinion
4 meaning somewhat agree
5 meaning strongly agree

1.1. I am heavily involved in the use of X-Ray or MRI viewing equipment	[1 | 2 | 3 | 4 | 5]
1.2. I like the current system which is implemented	[1 | 2 | 3 | 4 | 5]
1.3. I can see the benefits in a digital alternative to this system	[1 | 2 | 3 | 4 | 5]
1.4. I found the prototype system to be better than the current system	[1 | 2 | 3 | 4 | 5]
1.5. I would prefer to use the digital alternative if it was available at the hospital or place where I work	[1 | 2 | 3 | 4 | 5]
2.1. I found the prototype system to be intuitive and easy to use	[1 | 2 | 3 | 4 | 5]
2.2. I found the prototype system to be as functional as I would like it to be	[1 | 2 | 3 | 4 | 5]
2.3. I found the rotate feature useful	[1 | 2 | 3 | 4 | 5]
2.4. I found the scale feature useful	[1 | 2 | 3 | 4 | 5]
2.5. I found the move feature to be useful	[1 | 2 | 3 | 4 | 5]
2.6. I found the crop feature to be useful	[1 | 2 | 3 | 4 | 5]
2.7. I found the draw feature to be useful	[1 | 2 | 3 | 4 | 5]
2.8. I found the brightness feature to be useful	[1 | 2 | 3 | 4 | 5]
2.9. I found the colour invert feature to be useful	[1 | 2 | 3 | 4 | 5]

3.1. With the answers to part 2 in mind, please explain briefly what you would like to see improved within the next iteration of development

3.2. Please explain briefly what you liked about the system, and what you would find useful on a daily basis

3.3. Is there anything further you would like to mention about the project?

Appendix 5: The questionnaire for the testing sessions

A digital alternative to X-Ray Light Boxes

Number	Task	Resource	Start	End	Duration	% Complete	Q4 - 2009			Q1 - 2010			Q2 - 2010		
							October	November	December	January	February	March	April	May	June
1	Start Project		1/11/2009	1/11/2009											
2	Research		1/11/2009	29/1/2010	62										
3	Ethical Review Form		20/11/2009	26/11/2009	5										
4	Proposed Questionnaire		24/11/2009	25/11/2009	2										
5	Conceptual Sketches		28/12/2009	31/12/2009	4										
6	Sprint 1 - First Prototype		18/1/2010	26/2/2010	30										
7	Interview and questionnaire with doctors		1/3/2010	5/3/2010	5										
8	Sprint 2 - Updated Prototype (1)		1/3/2010	19/3/2010	15										
9	Re-Evaluation with Doctors		22/3/2010	26/3/2010	5										
10	Sprint 3 - Updated Prototype (2)		22/3/2010	9/4/2010	12										
11	Final Evaluation with Doctors		12/4/2010	16/4/2010	5										
12	Sprint 4 - Final Adjustments and Refactoring		12/4/2010	16/4/2010	5										
13	Testing		19/4/2010	23/4/2010	5										
14	Report		15/2/2010	23/4/2010	47										
15	End Project		24/4/2010	24/4/2010											

Appendix 6: The Gantt chart for the project