Adapting Multi-touch Systems to Capitalise on Different Display Shapes

MCNAUGHTON, JAMES,ANDREW

Adapting Multi-touch Systems to Capitalise on Different Display Shapes

James Andrew McNaughton

The use of multi-touch interaction has become more widespread. With this increase of use, the change in input technique has prompted developers to reconsider other elements of typical computer design such as the shape of the display. There is an emerging need for software to be capable of functioning correctly with different display shapes. This research asked: 'What must be considered when designing multi-touch software for use on different shaped displays?' The results of two structured literature surveys highlighted the lack of support for multi-touch software to utilise more than one display shape. From a prototype system, observations on the issues of using different display shapes were made. An evaluation framework to judge potential solutions to these issues in multi-touch software was produced and employed. Solutions highlighted as being suitable were implemented into existing multi-touch software. A structured evaluation was then used to determine the success of the design and implementation of the solutions. The hypothesis of the evaluation stated that the implemented solutions would allow the applications to be used with a range of different display shapes in such a way that did not leave visual content items unfit for purpose. The majority of the results conformed to this hypothesis despite minor deviations from the designs of solutions being discovered in the implementation. This work highlights how developers, when producing multi-touch software intended for more than one display shape, must consider the issue of visual content items being occluded. Developers must produce, or identify, solutions to resolve this issue which conform to the criteria outlined in this research. This research shows that it is possible for multi-touch software to be made display shape independent.

Adapting Multi-touch Systems to Capitalise on Different Display Shapes

James Andrew McNaughton,
Computer Science MSc,
School of Engineering and Computer Sciences,
Durham University,
2010

Contents

List of Tables

List of Figures

List of Abbreviations

Declaration

No part of the material provided has previously been submitted by the author for a higher degree in the Durham University or in any other University. All the work presented here is exclusively the work of the author and no-one else.

Copyright

Acknowledgements

The author would like to thank everyone at Durham University's Technology Enhanced Learning Research Group, One North East [1] and Ness Furniture [2] for all their help in making this research possible.

Figure 1: This project received funding from One North East. [3].

Chapter 1

Introduction

1.1 Research Overview

Multi-touch has been selected for use in an increasing number of systems. The input techniques available for systems are starting to become more diverse. With growth in availability developers have put more effort into considering how their software can handle inputs from different interaction techniques. The growth of multi-touch systems has encouraged developers to capitalise on its likely presence in future systems. To do this developers are required to make changes to the design of their software which concerns how it manages system inputs. With developers re-evaluating how their systems manage inputs, an opportunity for the reconsideration of standard elements of system outputs is provided [4]. One such element of a system's output which has long been standard is the shape of the display.

Most software systems which provide visual feedback to a user are normally designed to do so with a particular display shape. The most common of these shapes is the rectangle. A rectangular display is defined as a display shape that has four edges, which may or may not be the same lengths, and four ninety degree corners. However new technologies now allow for many different display shapes to be used in systems. Developers must now consider how to make their software flexible enough to adapt to not only displays of varying size and aspect ratios, but also to displays of varying

shape.

Multi-touch systems could benefit from the use of different display shapes. Developers may want their multi-touch system to be used with a range of different display shapes, each of which may be suited to certain scenarios of use. Due to the collaboration enabling nature of multi-touch, many of these systems may utilise a table interface where the display is horizontal. Some display shapes may have beneficial effects on the use of these systems. It is important for developers to have the choice of which display shape the system uses, but for this to be possible the system's software must be able to adapt to any shape.

So far, however, there has been little discussion about the use of software to support more than one particular display shape. Developers must reconsider the design of their systems' management of inputs, to allow multi-touch interaction. Developers doing this could also reconsider how their system manages its output. There is the opportunity now to make software display shape independent. This refers to software that can be used with any shaped visual output with no adverse effects on its visual content.

This research was begun with the objective of identifying a potential development that could benefit future multi-touch systems. An overview of the current state of research relating to multi-touch identified that there was a gap in research relating to the use of multi-touch software with different display shapes. Therefore the research documented in this thesis focused on the considerations to be made when designing multi-touch software for different display shapes. This research asked: '*What must be considered when designing multi-touch software for use on different shaped displays?*'

1.2 Proposed Outcomes of Research

To answer the main research question posed in Section 1.1 several further research questions were produced to be considered during this research. This research was intended to deliver answers to the following questions:

1. What issues occur when multi-touch software is used with different display shapes?

2. What evaluation framework can be used for judging the potential methods of allowing multi-touch software to use different display shapes?

3. What methods can be used to allow multi-touch software to use different display shapes?

4. Can multi-touch software be adapted or created to be display shape independent?

1.3 Thesis Outline

This thesis is structured as follows:

Chapter 2 details the execution and findings of a structured literature survey on the subject of multi-touch. An overview of the hardware and software used in existing and emerging multi-touch systems is given. In this overview the history of current multi-touch technology is detailed. The main groups of multi-touch technology are identified, discussed and compared. Emerging technologies, or adaptations to existing multi-touch hardware, are discussed and compared as are emerging Multi-touch Software Frameworks (MSFs). Throughout this chapter trends in the design and intended use of multi-touch systems are highlighted. Common features of both multi-touch hardware and software are discussed throughout this chapter. This chapter also highlights the gap found in research relating to the use of different display shapes with multi-touch software. The chapter then details the execution and findings of a structured literature survey on the subject of different display shapes.

Chapter 3 highlights the issue of occlusion that occurs when using multi-touch software with different display shapes. The major cause of this occlusion is identified as originating from the initial placement of content items. The identification of the issue is aided through the use of observations made on a prototype system.

Chapter 4 outlines an evaluation framework for assessing the resolution of the occlusion issue. The framework consists of criteria derived from the occlusion issue, the potential impact of attempts to resolve occlusion and the requirements of solutions to the occlusion.

Chapter 5 discusses potential solutions to the issue of occlusion. Also discussed are several methods for informing the software of the display shape.

Chapter 6 details the implementation of a selection of solutions into the SynergyNet MSF [5]. The adequate solutions chosen for implementation into the MSF are documented and their reasons for selection are discussed. Any problems encountered during implementation or deviations from the solution designs are also discussed.

Chapter 7 assesses the implementation of the proposed solutions using the criteria of the evaluation framework. Observations are made on the implementation from which suggestions of possible future improvements to the solutions are made. Any changes made during implementation to the solution methods are discussed. Any shortcomings or failures of the proposed solutions are discussed and potential corrections or alternatives are suggested where needed.

Chapter 8 provides a summary of the research carried out for this thesis. The considerations needed to be made when designing multi-touch software for use on different shaped displays are outlined and discussed. Future research relating to work done in this thesis is also proposed.

Chapter 2

Literature Survey

2.1 Chapter Introduction

The research to which this thesis relates to was begun with the intention of discovering potentially beneficial multi-touch developments. Therefore a study of current research relating to multi-touch was needed to identify where this potential development could originate from. The study was performed using a structured protocol that would allow any relevant literature relating to multi-touch to be found. The first literature survey focused on multi-touch in general to provide an overview of the current state of multi-touch research. From this overview any gaps in research could easily be identified. A second survey was then carried out using the same structured protocol. The second survey focused on an area of research which the previous survey indicated was lacking in resources. The findings of these two surveys could then be used to inform the investigation of a potential multi-touch development.

2.2 Survey Technique

A protocol was created for the surveys in this chapter. This protocol was adapted from an existing structured survey protocol [6, 7]. Stipulated by the protocol was how the collection of materials for a literature survey would be performed. This allowed for a

structured and unbiased approach to discovering and documenting resources relevant to the focus of the survey. The protocol stated that the initial collection of resources would be collected from a structured search of two, well populated, online computer science literature databases. The two databases searched were those of the Association for Computing Machinery and the Institute of Electrical and Electronics Engineers. Both were accessed using subscriptions which made the majority of resources they held available. For each database a structured search query was created around the terms relevant to the focus of the survey. Alterations were made to the search queries used. These alterations were in response to initial searches returning too many or too few results. Alterations were made until a manageable number of seemingly relevant resources were returned. A number of initial searches were made to ensure the results returned were consistent and relevant. A final search, after these initial searches, was performed from which the resulting resources were recorded. This is referred to as the primary search.

Research collated from the primary search included papers, technical reports and other grey literature. All resources found were stored in a literature database where a data extraction strategy was employed to capture relevant information. Resources which were returned by the primary search but were not accessible or deemed irrelevant on further inspection were rejected and not entered into the literature database. Any resources with material that was questionable, incomplete or of little relevance were then considered for rejection. Once all the relevant data had been extracted a survey of the resulting literature database was made. This survey was intended to find resources referenced by the collated literature which may be relevant but not already in the database. In addition to the types of literature accepted by the primary search the secondary search also included resources widely accessible through the internet, such as web-pages. This secondary search was performed by first searching the previously mentioned online databases for the titles of papers referenced in the primary resources. If not returned from these databases then alternative databases and search engines were used to try and find these resources. The resources returned from this secondary search would then be entered into the literature database. The same data extraction process was performed on these resources. The secondary search technique was repeated on

the resources returned from the previous search until only irrelevant works or resources already in the literature database were returned.

With a complete literature database a survey of all the collected resources relevant to the focus of the survey could be executed in confidence. A generic version of the protocol used in the survey can be found in Appendix A.

2.3 Multi-touch

The concept of multi-touch interactions has existed for the last three decades [8]. However it is only in recent years that the number of developers who have actively produced software which accommodates this type of capability has started to rise. New instances of multi-touch capable software, from operating systems to small applications, are now emerging. Significant numbers of consumer devices with multi-touch capabilities are starting to appear in the public marketplace. This has made multi-touch a matter of interest to end-users, developers and manufacturers.

New hardware technologies have made multi-touch interfaces cheaper, more precise and less problematic to set up. These technologies allow their interfaces to be used in a variety of scenarios from laptop touch-pads to museum installations. With the further development of these technologies more opportunities to utilise multi-touch interaction are becoming apparent to software developers. The result of these technological advances is that the flexibility and scalability of multi-touch interfaces will allow for a whole new range of multi-touch system design. Already on the market are interfaces of different sizes and features. There is a lot of choice for developers and users, not only in the features of the technologies but also in the software to be used with the interfaces. Section 2.3.3 provides an overview of the main technologies currently used in multi-touch systems, their recent developments and how their design affects their use.

Due to the differing features of each multi-touch technology there is currently no single, widely used MSF to support them all. Developers will adopt MSFs to work with based on the technology they are compatible with and the features they need. A technology can be supported by many MSFs and a single MSF can support several

different technologies. Similar to technologies, MSFs have different features and purposes. Section 2.3.4 provides an overview of a selection of the MSFs currently available for multi-touch systems, their recent developments and what features they offer.

As the availability of multi-touch devices increases the time spent by the average user interacting with such devices will increase. Until recently touch screen technologies, such as multi-touch, have been reserved for specific purposes and were not intended for everyday use. Until the launch of the Apple iPhone the only multi-touch interfaces available to end users were a handful of public area interfaces. These interfaces were used as information points which offered a limited range of functions. But with more users having access to multi-touch both at home, at work and in mobile devices the range of software which makes use of multi-touch interaction increases. As usage grows we can expect to see more applications which capitalise on interfaces with multi-touch capabilities. With the availability of multi-touch increasing, it is likely that developers will need to supply a larger range of applications which support multi-touch use. With technologies improving and allowing for more complex interactions the applications developed for multi-touch will need to be able to utilise the different features offered by each technology. Therefore it is likely that multi-touch technologies that offer a range of features will be capable of supporting many different applications. Section 2.3.5 provides an overview of the applications currently being developed for multi-touch systems.

2.3.1 Trends and Patterns in Resources

From the questions specified in Section 2.3 several search terms were produced for use with the structured survey protocol detailed in Section 2.2. These terms were 'Multi-touch' 'Design' and 'Development'. The search queries used were structured to only return results from within the last five years as the survey focuses on relatively recent innovations in the field. As discussed in Section 2.3 multi-touch has only recently been embraced by a wider range of developers. Due to this it is likely that any research before 2004 will either be referred to in more recent research and returned

by the secondary search or bears little relevance to the focus of the survey. From the resources returned by the searches several patterns and trends became apparent in the current state of literature on multi-touch.

The first observation to be made from the collected resources relates to how a multi-touch system is viewed by those working in the field. The elements of a multi-touch system can be broken down into three key groups observed by the majority of multi-touch developers. The first element of a multi-touch system is technology which relates to the physical devices used as multi-touch interfaces. The next element is entitled Multi-touch Software Frameworks referring to resources which in some way relate to the software frameworks of multi-touch systems. MSFs are software systems which interpret information provided by multi-touch interfaces. In addition to the basic function of handling multiple location based inputs, MSFs can also offer a range of additional features. The final element of a multi-touch system is the applications. These are higher level pieces of software which are usually implemented to work with MSFs and provide functionality to the user. Of these key elements, technology has the most resources relating to it, whereas MSFs are the element of multi-touch which have the smallest amount of literature relating to them. From these observations this section is divided into sub-sections which are each based on an element of multi-touch systems. In each of these sections the resources relating to the title element are discussed.

Another observation about the resources collected is that the purpose of papers relating to multi-touch can be divided into four distinct groups. The first of these groups can be labelled ‘overviews’ which refers to resources providing an overview of the current state of research and projects for a particular element of multi-touch. The second group can be entitled ‘developments’ which contains resources detailing a recent development of new multi-touch hardware or software. The next group can be labelled ‘applications and tools’which contains resources referring to the use of multi-touch interfaces in a particular scenario or tools for use with multi-touch. The final group can be referred to as ‘evaluations’ and contains resources relating to the evaluation of multi-touch systems. The literature database was modified to reflect this observation.

Another trend noted is that commercial multi-touch developments both in hardware

and software have very few features which open source alternatives do not already offer. This demonstrates a strong open-source movement in the field of multi-touch which is discussed later in this section in regard to the Natural User Interface (NUI) Group.

Most resources on the subject of MSFs found as part of the literature survey are websites and not academic papers. This could be due to the fact that a lot of the MSFs documented in these web based resources have only been released recently. This means that any research involving these recently released MSFs may still be in progress and any projects which may make use of them could as of yet be incomplete. In comparison to the topics of technology and applications it appears that MSFs are lacking in documentation. While there appear to be fewer MSFs than multi-touch technologies and applications, the difference in resources relating to each topic is not proportional. This indicates that MSFs are overlooked; this statement is backed up by several other observations. The first such observation being that most resources relating to MSFs are primarily about multi-touch technologies where the MSF is developed for the technology. In these cases the MSFs are not the main focus of the paper and frequently only mentioned briefly. The next observation is that a lot of the MSFs developed recently have had no academic write up. This is because many of the MSFs discovered by this literature survey are developed as software projects rather than research work. As a result of this most documentation on the MSFs comes from community resources such as wikis and forums.

The final observation which backs up the statement that MSFs are not as well documented as other elements of multi-touch systems is that there is little secondary research performed on MSFs. This observation is made based on the fact that a single resource rarely mentions more than one MSF. In addition to this, when multiple MSFs are mentioned in the same resource they are only briefly detailed. No in depth comparison or overview of the available multi-touch MSFs was discovered in the survey. This could be due to any existing overviews of the current field of touch MSFs not being found in the searches or more likely due to no such resource currently existing. Section 2.3.4 provides an overview and comparison of MSFs which are currently available or in production.

2.3.2 Background

There are several techniques in use that allow a surface to sense multiple contact points. The term 'multi-touch' is commonly used to describe a surface that is capable of many simultaneous contact points from fingers or hands. The sensing technique used in a multi-touch system determines the hardware needed and the requirements of the software. Most research concerning multi-touch sensing techniques concentrates on the hardware technology that allows a technique to be implemented. Systems that support true multi-touch can allow for multiple users to interact with (which means to touch the surface of) the system at the same time. This section focuses on these multiple simultaneous user supporting techniques. There are some techniques which can detect multiple user touches but only up to a low limit such as dual view, which can only detect two simultaneous inputs. This section is not concerned with techniques that cannot calculate the locations of contact points from multiple users each touching the interface in one or more places at the same time.

There are several multi-touch techniques for which there are many active developments. Each of these techniques have different attributes which can be utilised by developers in a variety of manners for numerous purposes. Therefore implementing technologies not only can be compared by parameters such as input resolution and scalability, but also by the extra functionality they can provide to users and developers. It is also important to consider that some technologies are better suited for certain designs and uses. Despite these differences multi-touch interface technologies will always have similarities as they are all designed for the same purpose of reading in multiple simultaneous inputs. Similarities also arise from the technologies having to deal with the same issues which are presented by the concept of multi-touch. Scott & Carpendale [9] outline several of these issues which current multi-touch systems must successfully overcome to be considered usable. Their research specifically considers the issues arising from technologies allowing for multi-touch input with a table interface. The basic issues originate from the software's need to manage a number of simultaneous touch inputs which allows for group interactions with a single interface. The more advanced issues outlined originate from larger displays utilised

to enable group interaction with tabletop interfaces. These advanced issues involve overcoming display quality problems and the challenge of allowing users access to regions of a display which are made distant to them by these larger displays.

Scott & Carpendale [9] also highlight the ability of some multi-touch technologies to perform user tracking where inputs can be designated as belonging to the same user. The research also highlights how the use of horizontal interfaces such as multi-touch tables differ from traditional vertical displays. It is observed in the research that horizontal interfaces encourage users to surround an interface and interact with it from different positions around the display. This research provides a general overview of the issues present in multi-touch and introduces some of the technologies currently used by developers. However only a small number of technologies are mentioned and their workings are not covered in much detail. Two of the technologies mentioned by name are DiamondTouch, which is detailed later in this section, and TViews, which is discussed in Section 2.3.3.

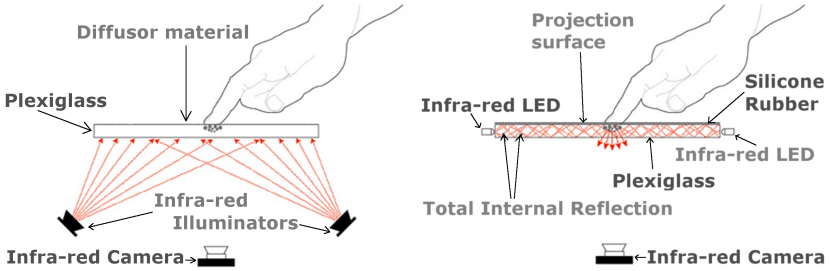


Figure 2.1: Two iterations of vision based multi-touch technologies [10].

A wider range of multi-touch technologies are detailed in depth by Rekimoto [11] who highlights three current multi-touch technologies and their workings. The first of these is a technology called HoloWall where the system's output is projected onto a transparent wall from behind and illuminated with infra-red light. A camera with an infra-red filter behind the wall is then trained on the projected display, so that when a user touches the wall the infra-red light is interrupted at that position. The camera picks up any interruptions in the reflection of the infra-red light and the software interprets

these as inputs and can derive the location of the touch. This is known as Diffused Illumination (DI) and is the basis of many developments in multi-touch technology. The workings of a typical DI system set up are shown in Figure 2.1a.

Rekimoto's [11] research also highlights that there are other similar technologies which are vision based but use different methods of providing the infra-red illumination. One such technology is known as Frustrated Total Internal Reflection (FTIR) which is employed in numerous multi-touch systems. In FTIR systems infra-red light is reflected between a layer of a transparent material and a layer of durable material onto which the output is projected. Any touches will disrupt the infra-red light and this frustrated' light will be detected by a camera positioned behind the surface as shown in Figure 2.1b. Another technology covered by Rekimoto [11] is SmartSkin which is based on the principle of capacitive sensing where the potential field generated by the human body is used to calculate the proximity and location of the user's touch. A grid shaped antenna is used in SmartSkin to measure the potential field as shown in Figure 2.2a.

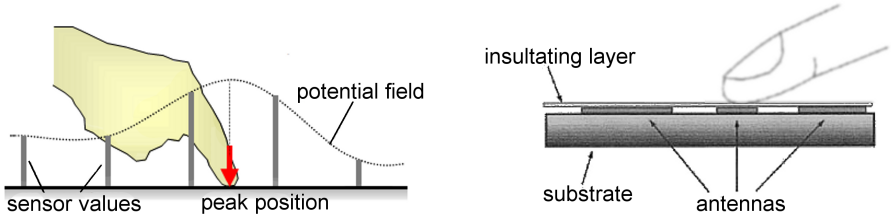


Figure 2.2: Two iterations of capacitive multi-touch technologies [11, 12].

The research details how this technology works by building an image of the input from the capacitance between the user's hand and the sensor grid. This image of the input is then filtered through a series of image processing functions to find the location of a user's touch. Rekimoto [11] also gives a brief overview of DiamondTouch which is another technology built on the principle of capacitive sensing. DiamondTouch, similar to SmartSkin, uses sensors and antennas to measure the nearby potential fields as shown in Figure 2.2b. Similar to SmartSkin in most respects DiamondTouch has

one large difference which is the ability to enable user tracking. This is because users are seated in specially designed chairs with a built-in signal receiving electrode which receives a time modulated signal from the user's touch on the table via the body of the user.

DiamondTouch is discussed in more detail by Dietz & Leigh [12] who explain the science behind the technology in great detail. In the research in question several requirements of a multi-touch system are proposed which provide a good basis for the evaluation of multi-touch technologies. The requirements specified by Dietz & Leigh [12] state that a multi-touch system should be capable of being:

- Multipoint/Multi-touch Capable: Detecting multiple simultaneous touches.
- Identifying/User Tracking Capable: Identifying touch inputs from the same user.
- Debris Tolerant: Not allowing objects left on the interface to interfere with normal operation.
- Durable: Able to withstand normal use without frequent repair or recalibration.
- Un-encumbering: No dependence on additional devices for use.
- Inexpensive: Low manufacture and running costs.

The requirements stated by Dietz & Leigh [12] are echoed by Microsoft as documented by Wang [13]. Microsoft states that the four important components of its Multi-touch interface Surface are: direct interaction, multi-touch contact, multi-user experience and object recognition. Dietz & Leigh [12] claim their technology meets all their stated requirements. However the requirement calling for multi-touch systems to be un-encumbering appears to be contradicted in their system. DiamondTouch requires users to sit on the accompanying chairs to enable user tracking which can be viewed as a dependence on an additional device for use. Several simple applications are used with a prototype of the technology. Since this research is one of the older resources returned from the two searches detailed in Section 2.2 there has been a wide range of applications developed for the DiamondTouch technology and its accompanying DiamondSpin MSF since the time of the research's publication.

More multi-touch technologies are discussed by Selker [14] who also summarises in his introduction why multi-touch is becoming popular with users and developers. The research details an observation on the sales of Apple's iPhone. Although the iPhone was inferior to many other alternative devices available at the time of release it managed to outsell them all by a significant amount. Selker [14] attributes this to the iPhone's multi-touch interface, demonstrating that the technology is in demand. Also mentioned is that as consumers' demand for the technology increase more hardware and software developers are investing in making their products compatible with multi-touch technology.

The research details a brief history of a selection of the current multi-touch technologies available. The first technology discussed is the Magic Wall developed by Jeff Han utilising FTIR multi-touch technology. A number of simple applications demonstrating the use of this technology are listed such as map navigation or manipulation of three-dimensional objects using a set of specific gestures. Another technology discussed is the Microsoft Surface which uses five cameras behind the interface (a table top) to identify user touches using a DI method, similar to that used for the HoloWall. As well as using user touches as an input the Microsoft Surface also can detect RFID chips allowing for object recognition and tracking. The final technology discussed is the UnMousePad. This is a relatively new technology and is discussed in Section 2.3.3. Selker [14] provides an in depth overview of a selection of multi-touch technologies. This can be viewed as a very brief introduction to the current state of multi-touch technologies. Also noted in the research is the importance of using real life actions as metaphors for creating gestures in multi-touch applications which bear resemblance to real world tasks. This is something discussed in section 2.3.4.

The technology of the HoloWall has already been mentioned in this section and is referenced in a number of resources detailed throughout this section. For that reason the work of Matsushita & Rekimoto [15] is included despite their research being relatively much older than the majority of other resources collected. As explained previously, the technology in the HoloWall works by detecting touch from their interruptions in the reflection of infra-red light on the interface surface. HoloWall has the ability to detect objects presented to the surface in addition to user touches. The research proposes

that special markings could be used for allowing the interface to identify objects after detecting them. The idea of a multi-touch interface being able to recognise an object appears in several of the resources collected in the survey as a method of providing a form of tactile feedback. In the research of Matsushita & Rekimoto [15] the interface used with the HoloWall FTIR technology is vertical. However, in several projects since its publication the technology has been implemented with a variety of interfaces including horizontal tabletops.

Another piece of research which fell outside of the primary survey's date range, but was included because of references to it in several other resources, was by Rekimoto [16]. This research discussed a multi-touch technology called SmartkSkin. The work explained how the potential field generated by the mesh shaped antenna can detect users' hands even when they are not touching the table. This is useful for activities where prolonged contact with the interface is necessary as this can cause discomfort to the user. This functionality is desirable in systems where certain objects need to be identified and tracked and others need to be ignored.

Schöning et al. [10] summarise multi-touch technologies as belonging to one of three groupings. These three main groups of multi-touch technologies are vision, capacitance and resistance based systems. **Vision based systems**, also known as surface wave systems, refer to where an interface uses light or infra-red based techniques such as FTIR or DI to detect users' touches. Another vision based multi-touch technology discussed by Schöning et al. [10] is Diffused Surface Illumination (DSI). Similar to the workings of FTIR, infra-red light is shone between two surfaces but instead of using reflection to frustrate the light, transparent surfaces are used so the light escapes. This light reflects off user touches and is viewed by a camera. The camera's vision is then processed using methods similar to the touch detection methods used in DI systems. It is interesting to note that vision based systems are capable of object detection. However DSI and FTIR must attach markers on the bottom of objects in a certain pattern for them to be detected and identified by the software. A possible issue arising from this method of object detection and recognition is that the pattern of markers on an object could be recreated by touches from users when the object is not present. This recreation of the object could be intentional or performed by

accident but in either situation would cause the system to function incorrectly.

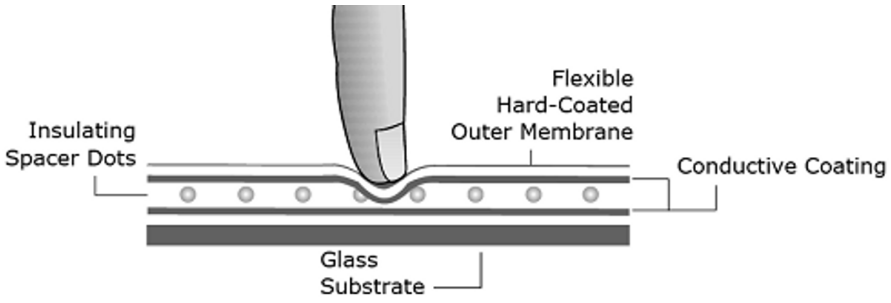


Figure 2.3: Typical workings of a resistance based multi-touch interface. [10]

DI systems can detect objects by their outline when placed on the interface though a visual pattern known as a fiducial may need to be placed on the bottom of the object for it to be identifiable. **Capacitive systems** refer to technologies such as DiamondTouch and SmartSkin which use the electric fields produced by living beings to detect touches. These systems can have a great accuracy in detecting user touches and tend to have durable interface but are expensive to manufacture. **Resistance based systems** refer to technologies which use pressure from a user's touch to connect two layers of material that will conduct a signal as shown in Figure 2.3. These interfaces are capable of using very little power but have a low resolution when detecting user touches. Schöning et al. [10] provide an in depth overview of these technologies and detail basic advantages and disadvantages for each technology. However no direct comparison of the technologies is made in the research. This is because the features and abilities of multi-touch technologies will differ depending on their implementations.

2.3.3 Technology

As detailed in Section 2.3.1 technology is the most frequently documented area of developments in multi-touch research. There are several resources which provide overviews of the current range of multi-touch technologies available such as those by Schöning et al. [10], Buxton [8], Izadi et al. [17], Rosenberg & Perkin [18], Hofer et al. [19] and Rekimoto [11]. In the wake of multi-touch becoming a popular interaction

technology, a large number of developers are adopting multi-touch for their work. Not only are commercial developers taking interest but a large number of open source multi-touch software and technologies are now becoming available. A large number of these open source developments, especially in technology, are being generated by the NUI group. This group is the origin of several pieces of research detailed throughout this survey. Multi-touch technologies are currently being developed at such a rapid rate that overviews of this field of research quickly become outdated. This section provides a current overview of emerging multi-touch technologies. Considered are a series of recent stand alone technologies and several novel adaptations which can be applied to existing multi-touch technologies. The technologies considered in sections 2.3.3 to 2.3.4 are: UnMousePad, TouchLight, FiberBoard, ThinSight, FLATIR, TViews, Wiimote gesture identification, deformable workspace and liquid displacement sensing.

UnMousePad

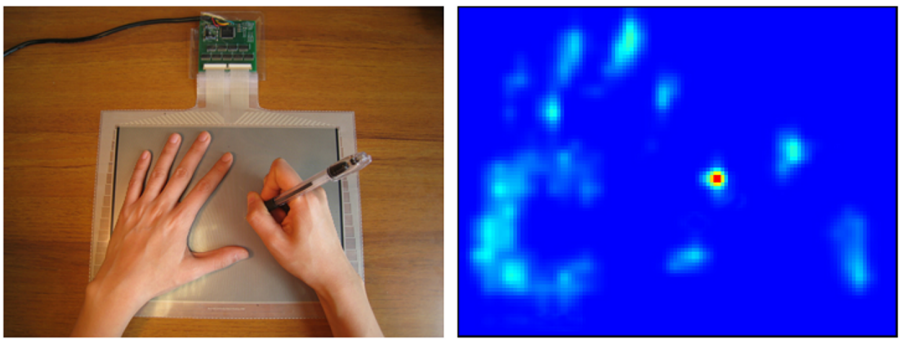


Figure 2.4: An example of the use of the UnMousePad multi-touch technology displaying user touches and object detection. [18]

The UnMousepad as documented by Rosenberg & Perlin [18] is a resistance based multi-touch technology. The technique used by this technology is called interpolating force sensitive resistance which allows for a thin and cost effective interface to be built which has a high input resolution. The research documents the workings of the technology in great detail after a brief summary of the three existing technology

groupings. Also detailed are the workings of the technology's accompanying low level software. This software is responsible for taking the image compiled from the inputs and outputting the list of touches and touch events to a user application. The interface is capable of detecting touches from objects as well as user hands as shown in Figure 2.4. Rosenberg & Perlin [18] argue that this is beneficial to the design of a multi-touch system. However other researchers, such as Dietz & Leigh [12], would disagree because this feature would result in the interface not being debris tolerant.

Though touches from objects can be read as inputs no mention is made of whether objects could be recognised by the technology without the use of markers or fiducials. Fiducials are similar to the landmarks which can be recognised by a software system as part of a visual input [20]. These visual markers can be identified by the system. Since the system should have knowledge of the fiducial, such as the shapes it displays and its size, its position relative to the camera can be calculated.

As this technology uses pressure it can detect other inputs in addition to human touches, such as styluses. The research identified several other advantages of this technology beyond its force based detection technique. These are its flexibility, its ability to detect touches through a variety of thin materials and its ability to be integrated with existing displays when implemented with a transparent material. The last advantage listed is, at the time of the literature survey's execution, only a proposal. Therefore UnMousepad currently offers only an indirect multi-touch input interface. This means that the system's output and input do not utilise the same interface. However once this feature is implemented the UnMousepad could easily be applied to existing displays with little cost.

TouchLight

TouchLight as documented by Wilson [21] is an older technology compared to some of the others detailed in this section but contains some features of interest. The majority of the research concerning TouchLight is dedicated to explaining the workings of the technology, especially the details of the image processing. Similar to the HoloWall this technology uses DI to detect touches. Using a special sheet of glass called a holoscreen

to project onto, the technology allows for transparency in its interface. Wilson [21] notes that this will enable new uses of the technology. One such use is the ability to scan objects placed on the interface as the transparent nature of the screen allows visible light cameras to see clearly what is on the other side of the interface. The transparent nature of the interface could prove beneficial to augmented reality and spatial displays. This is because computer generated images could be projected in front of real world views behind the interface. The set up is relatively low-cost; the only expensive part of the technology is the holoscreen glass.

Improving Vision Based Systems

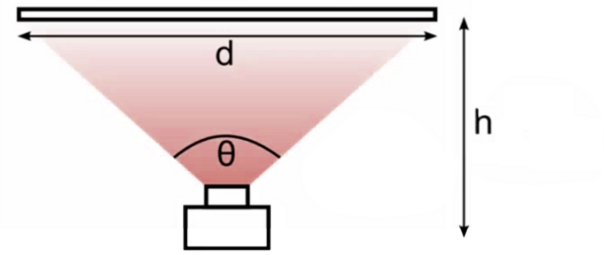


Figure 2.5: The space required for a typical single camera DI or FTIR multi-touch interface set up. [22]

Vision based systems are widely used due to the number of advantages they offer over alternative multi-touch technologies such as their affordability and durability. However a common drawback in vision based systems is the requirement for a large amount of space behind the display to be dedicated to the Vision System (VS). In DI systems space is needed for the dispersal of infra-red light but in other vision based systems such as FTIR and DSI this is not needed. However in all these systems where a typical set up is used space is needed for a camera to view the display. Equation 2.1 and Figure 2.5 demonstrate the relationship between a camera's distance from the display and the size of the field of view from the camera. If only one camera is to be used to capture the entire display the variable d in Equation 2.1 becomes the value of the longest side of the display.

A possible method to reduce the space needed by cameras in these systems is the

use of multiple cameras where each camera monitors a region of the display. This set up is used in the Microsoft Surface system [14] and allows for the cameras to be placed closer to the display. This technique of using multiple cameras is also used in the systems developed by Evoluce [23] as discussed in Section 2.3.4. Another element of vision based systems that requires space may come from the use of a projector to produce the output. In some systems mirrors are used to reduce the space needed by the projection or alternatives to a projection based display are used. There are several current developments in multi-touch technology with the objective of reducing the space required behind the display in vision based systems.

$$h = \frac{d/2}{tan(\theta/2)} \tag{2.1}$$

Fiberboard

One such effort to reduce the space required behind the display of vision based systems is presented by Jackson et al. [22] named Fiberboard. This technology uses a typical FTIR system set up with a series of infra-red LEDs surrounding a display. The display used is a LCD screen rather than a projector. This saves on the space needed by a projection so the only remaining element requiring space is the camera. Fibreboard uses a number of fibre optic cables to take the infra-red light being reflected from the display to the camera. The fibre optical cable ends are placed in a grid behind the display. The cables are bent gradually from the display to a point where they converge in front of the camera. In the system documented by Jackson et al. [22] the camera is placed at the foot of the display. However due to the nature of this technology the camera could be placed anywhere the cables can reach to. Infra-red light from the display which would usually be detected directly by the camera is instead transmitted via these fibre optic cables to the camera.

As part of this system a piece of software is used which flashes a pattern of light on the display. The software uses the input from the camera to decipher which fibre optic cable is transmitting the infra-red light from a region of the display. This software then rebuilds the image of the infra-red light from the display using this information and

passes the information on to any awaiting lower level MSF to detect the user touches. Jackson et al. [22] claim this set up allows the depth behind the display to be reduced to about one tenth of the depth required by a typical vision based system. This system provides a low cost solution to the problem of the large amount of space required in vision based systems and its accompanying software allows for any MSF used on the corresponding traditional FTIR set up to be used with this technology.

ThinSight

Documented by Izadi et al. [17] is another technology developed with the intention of reducing the space required behind the display in vision based systems. This technology, named ThinSight, works on the same principle as most vision based multi-touch technologies. However instead of using infra-red lights to illuminate an interface, a grid of infra-red emitters is placed behind the display. In place of the usual infra-red or visible light cameras a grid of infra-red detectors is also placed behind the display. When the interface is touched the light from the emitters is reflected back through the display to the detectors. The infra-red light levels across the grid can be used to build an image of the infra-red reflections over the display. This image can be interpreted using the same image processing techniques as used in DI or FTIR systems to discover touch locations. This set up has the advantage of being usable with flat displays such as LCD screens. This technology is also able to deal with severe changes in lighting conditions which usually require vision based systems to be recalibrated. Despite all its advantages ThinSight is much more complex and expensive than DI and FTIR alternatives. However Izadi et al. [17] speculate that future display technologies could incorporate this technology for a relatively low cost.

The research details other features of this technology such as its fiducial recognition and ability to use the infra-red input and output of the interface to communicated with other devices. The majority of this research focuses on explaining the workings of this technology and evaluating the prototype. This prototype was a laptop's LCD screen with the infra-red emitting and detecting grids placed behind a small section of the display. Later research by Izadi et al. [24] documents the scaling of the technology

from beyond the prototype to larger interfaces. For these interfaces in the later research touch detection is enabled for the full display. This later research focuses on advances made to the technology and shows its workings in detail. Interaction with mobile phones is documented as well as the accuracy of the touch detection. The circuitry currently used as part of the technology makes the LCD screens used significantly thicker. However this set up uses up less space behind the interface than vision based alternatives with similar display sizes. The research of Izadi et al. [24] highlights the difficulties in scaling some multi-touch technologies to different sized displays.

FLATIR

Similar to ThinSight is a new technology documented by Hofer et al. [19] called FLATIR. Similar to FTIR interfaces, a FLATIR interface is composed of two layers with infra-red light reflected between them. However detection is not performed by cameras behind the display but by an array of infra-red diodes, similar to ThinSight's infra-red detection grid. Hofer et al. [19] discuss in detail the technical aspects of implementing this technology and variables that affect its use, such as the cleanliness of the user's fingers and the amount of pressure used. This technology offers the same advantages as ThinSight at a lower cost, however it cannot allow for infra-red communication which ThinSight is capable of. The prototype built as part of this research is capable of a touch detection resolution of 1mm and as the technology is still in its early stages of development the performance may improve further.

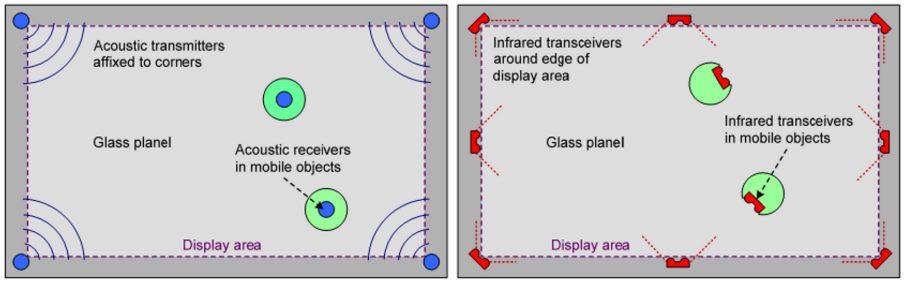


Figure 2.6: Acoustic transmitters and infra-red transceivers are used in the TViews system to provide multiple simultaneous location based inputs [25].

TViews

Mazalek et al. [25] document the development of the TViews table. The technology does not support direct touch, which means it cannot detect a user's hand or finger on the interface. However it does support multiple location based inputs which can be considered a form of multi-touch. This technology allows for multiple users to each provide multiple inputs simultaneously through the placement of objects on a display. The objects detect their position by sensing the strength and direction of acoustic signals generated from the corners of the table's display. These objects, called pucks, then communicate with the table using infra-red light to perform tasks such as updating their location as shown in Figure 2.6. The system display method used in the implementation of the technology shown in the documentation of this research was rear projection as used in some DI systems. This allows for a low cost implementation of the technology compared with other multi-touch tables.

The research goes into great detail about the workings of this technology and gives a series of brief descriptions of the applications available for the technology. The table's hardware and software have both been developed further since the publication of the original research of Mazalek et al. [25]. These developments are covered in the later research of Mazalek et al. [26] which is discussed in Section 2.3.5. This technology offers some interesting and cost effective ideas but its inability to detect user touch could be a problem for developers who want to work with a true direct touch system. As highlighted in both of the discussed pieces of research conducted concerning the TViews table the use of objects allows for tactile feedback which can be beneficial to users.

Wiimote

Vlaming et al. [27] introduce a technology which utilises Nintendo's wiimote video-game controller. This technology does not require a surface for input which means that the user does not directly interact with the display interface. The concept of tracking hands and their gestures in open space is not new but is not usually related to multi-touch systems. This is due to the fact that gestures in open space typically utilise

three-dimensions rather than the typical two expected with multi-touch interfaces. However due to the set up of this technology only two-dimensional gestures such as those used in multi-touch can be utilised.

This technology works by using the wiimote to view infra-red lights attached to the end of the fingers on gloves worn by the user(s). From this input the location of the fingers in a two-dimensional plane can be calculated. Also by using multiplexing techniques with the lights in the gloves, detected inputs can be grouped by hand and user. Vlaming et al. [27] provide an in-depth explanation of the workings of the technology then provide a case study to evaluate the systems effectiveness when in use. Though lacking in the ability to provide a direct interaction with a typical flat display, technologies such as those used in virtual and augmented reality could be used to provide a more direct input. Future work on the development stated by Valming et al. [27] includes adapting the technology to accommodate for three-dimensional gestures. This would distance this wiimote based technology from typical multi-touch interaction.

Deformable Workspace

The concept of a deformable workspace is put forward by Watanabe et al [28]. This technology bares similarities to the set up of DI multi-touch technology but infra-red light, used to illuminate the display, is projected in a pattern rather than as uniform illumination. This requires more processing in the software but has the advantage that deformations of the surface can be detected by the software. The surface of the deformable workspace is made from a latex sheet which the user can touch and push. Their touches are picked up by the infra-red cameras and interpreted by the software, similar to the workings of other vision based systems. In addition to this the user can push their hand or finger into the latex display which deforms the interface until they release their touch.

The software used with this system can detect the deformation of the interface by the user using the infra-red pattern and can interpret the magnitude of the deformation. This allows the user to provide more information with each touch through the amount

of pressure they provide. Watanabe et al. [28] describe how this technique can be used as a method for inputting information relating to the Z-axis. This ability aids three-dimensional manipulations with multi-touch which is difficult to achieve with other technologies. Watanabe et al. [28] stipulate that in their ongoing research with the technology, better haptic feedback to the user could be made possible with the use of this technology.

Liquid Displacement Sensing

A new multi-touch technology is documented by Hilliges et al. [29] which makes use of a technique named liquid displacement sensing. The technology works with a similar set up to some instances of DI systems where a camera is placed below a horizontal interface. For the display a latex sheet surface with an acrylic surface below it is used. The space between these two surfaces is filled with black ink and sealed. The area below the acrylic pane is illuminated with uniform visible light. The output from the system is projected from above onto the surface. This may cause occlusion problems from user shadows but projection from below the display is not possible due to the black ink. Any touches on the latex interface displace the ink allowing the latex to touch the acrylic. At the position of the touch there is no black ink to absorb the light so the camera sees the latex and the software registers the location of this as touch information. This is a very cost effective set up and is capable of detecting the pressure from a user's touch.

However, a problem with this technology was highlighted by Hilliges et al. [29]. This problem is that a user's touch could cause ripples in the ink which compromises the high input accuracy of the technology. Hilliges et al. [29] in their research discuss the workings of the technology in great detail This research also highlights the fact that malleable interfaces, which utilise liquid displacement sensing, have several additional advantages, such as their ability to detect objects. Another advantage of this liquid displacement sensing technology is its provision of a soft interface which is beneficial for long periods of interaction using the display.

Comparison of Emerging Technologies

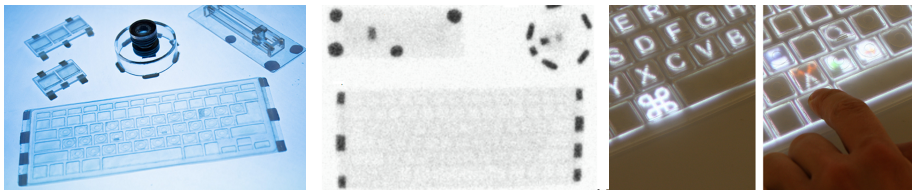
		Emerging Multi-touch Technologies								
		UnMousepad [18]	TouchLight [21]	Fiberboard [22]	ThinSight [17]	FLATIR [19]	TViews [25]	Wiimote [27]	Deformable Workspace [28]	Liquid Displacement Sensing [29]
Features	Direct Touch	N	Y	Y	Y	Y	N	N	Y	Y
	Debris Tolerant	N	N	N	N	N	Y	Y	N	N
	Unencumbering	Y	Y	Y	Y	Y	N	N	Y	Y
	Durable	Y	Y	Y	Y	Y	Y	Y	Y	N
	Proportional Cost	Y	Y	Y	N	N	Y	Y	Y	Y
	Thin Interface	Y	N	Y	Y	Y	/	Y	N	N
	Capable of User Tracking	N	N	N	N	N	Y	Y	N	N
	Capable of Object Recognition	Y	Y	N	Y	N	Y	N	Y	Y

Table 2.1: Comparison of the features of the emerging multi-touch technologies.

Table 2.1 uses the observations made on each of these stand alone technologies and the multi-touch requirements defined by Dietz & Leigh [12] to directly compare these recent multi-touch technologies. Additional fields are based on functions that the technology could support. One additional field identifies whether a technology has a proportional cost. If the cost of the technology is not proportional to the size of the display then it does not have a proportional cost. Note that '/' is used when different implementations of a technology have different features. The field Direct Touch is used in this table. This refers to a table's capability to respond to a user's finger or hand touching an interface displaying the system's output.

Enhancements to Multi-touch Technologies

Not all developments in multi-touch technology focus around stand alone technologies. Others involve adapting multi-touch for novel uses or enhancements to the technologies to enable multi-touch interfaces with new or improved features. Similar to the pucks in the TViews [26] technology Weiss et al. [30] document the use of tangible objects in multi-touch environments. This research concerns the development of a technology called Silicone Illuminated Active Peripherals (SLAP) widgets. The SLAP widgets are a collection of control objects which can be recognised when placed on a multi-touch interface as shown in Figure 2.7a. The interface used is a FTIR screen which recognises the control objects through the placement of markers in a pattern on the bottom of the objects as shown in Figure 2.7b.



(a) (b) (c)

Figure 2.7: Images of SLAP widgets [30].

The software which detects users' inputs will identify these markers as user touches and pass the information onto further software that will detect and identify the pattern of the touches. With the pattern detected the object can be identified and its placement on the interface calculated. The objects are made from a transparent material. This allows for a virtual counterpart of the objects to be displayed and updated constantly below the physical object. This virtual object can be clearly seen through the transparent physical object. In addition to this the user can interact with the virtual object through direct touches.

In their research Weiss et al. [30] demonstrate a range of control objects they have devised such as a keyboard, a slider and a dial. These devices can be used for many different functions similar to the virtual input components on other multi-touch interfaces such as the software based keyboards used by several MSFs. However

unlike their entirely software based counterparts these devices offer the advantage of tactile feedback to the user. Weiss et al. [30] hypothesise that this allows for faster and less error prone input. This is a form of prop-based interaction [31]. SLAP widgets provide a hybrid of real world and virtual components which makes them versatile and informative to use. Weiss et al. [30] cover in detail the workings of the devices and documents an experiment performed to evaluate the use of the objects. The opportunities made possible by SLAP widgets, such as enabling a tangible keyboard to display a variety of different character sets as shown in Figure 2.7c, make this technology very appealing to developers and users alike. From the methods described in the research it is apparent that most multi-touch technologies can be adapted to incorporate them or similar tangible controls.

The importance of tactile feedback in touch based interaction is highlighted by Marquardt et al. [32]. In the research in question a puck is used in conjunction with the Microsoft Surface system. The puck is tracked through the placement of a fiducial marker on its base. The puck uses three methods of providing tactile feedback to the user. The first is the use of a brake which creates friction when moving the puck across the screen. This brake can be used to convey to the user areas where they need to be precise in their positioning of the puck or areas of interest. The other methods are implemented by the use of a rod on top of the puck. The rod's height is controllable by the system and can be used to provide information on z-axis values from the display. The rod can also be used as way of interacting with the system, acting similar to the button on a mouse the user can press down to initiate an event.

Tactile feedback is also provided through the resistance of the rod when it is being pressed. This can be used to inform the user on how acceptable their current action is to the system. The pressure used by the user on the rod is communicated to the system by the puck and can be used as an additional input parameter. The puck provides an additional output to the user which is not based on vision. This 'eyes free' interaction is beneficial for many uses of interactive systems. However one draw back is that the number of users is limited by the number of pucks. The multi-touch nature of the interface does allow users to interact with the system directly without the puck but this method lacks tactile feedback. Though the puck itself is a prototype it is relatively low

cost. Its future implementations could follow the same design and also remain low cost

One drawback with the design of the prototype is that its current implementation requires a wire for power and data communication. A wireless puck would be preferable to allow the user more freedom in changing its position and orientation. Marquardt et al. [32] note how in their future work they wish to implement a smaller puck and investigate its implications. This research demonstrates the possibilities of tactile feedback in touch based interactive systems. It concludes that the combination of friction, height and pressure as forms of feedback can greatly improve user interaction.

Research by Hancock et al. [33] evaluates the use of tangible and direct touch interfaces in manipulating both two and three-dimensional information. The evaluation focuses on an experiment performed with a DI multi-touch system. A series of tasks was performed by a group of users with either direct touch or a tangible control object on information in two-dimensions, three-dimensions or a combination of both. The results from this evaluation show that direct touch was better for three-dimensional interaction. However the use of a tangible interface was shown to be beneficial for interaction with a two-dimensional environment. The tangible control object allowed for more precision in the users' interaction whereas the use of touch allowed the user to directly manipulate specific points of data. The research also highlights how for two-dimensional environments, mouse interactions which may be familiar to the user can be used with touch based interaction to improve intuitiveness. The research concludes by stating that a tangible interface allows for indirect interaction with an environment. This is highlighted as being beneficial when dealing with three-dimensional objects because a direct interaction technique would be limited to only one accessible plane. This work by Hancock et al. [33] shows that tangible objects can provide benefits beyond furnishing the user with a tactile feedback.

Echtler & Klinker [34] write about a technique to enhance multi-touch technologies with shadow tracking. While not a stand alone multi-touch technology, shadow tracking is a feature which could be used with FTIR and other vision based systems. The implementation of shadow tracking used in this research works by placing a grid of infra-red LEDs above a horizontal multi-touch interface. A camera below the interface can then detect the shadows cast in the infra-red light. This image of the shadow can be

compared with the touch input detected by the FTIR to identify any objects, such as the users' hands, which are above the display but not touching it. These objects are then identified as hovering instead of touching the interface. Information on identified inputs such as their location and hovering/touching status are then passed on to higher level software. These techniques offer a variety of uses such as the emulation of a mouse which is usually a source of problems in traditional multi-touch interfaces. This is mainly due to the difficulty of distinguishing between a gesture to move the mouse, and a gesture to drag something with the mouse. With the ability to hover, users can simply move their hand around above the interface to move the cursor rather than dragging it along the screen which can cause discomfort with prolonged use. The concept of mouse emulation is discussed in more detail in Section 2.3.5.

The research of Echtler & Klinker [34] also contains a brief evaluation which shows positive results from the use of the technology. Shadow tracking does provide a lot of advantages but also can make multi-touch interfaces more cumbersome. Worth noting is the set up of the system's display which uses a sequence of mirrors in front of the projector to reduce the space needed behind the display. This is an example of the current desire to try and reduce the space that vision based systems use. However this space saving technique does not cancel out the additional space required for the overhead lighting grid.

A novel adaptation of multi-touch technology comes from Rivière et al. [35] in the form of the CubTile. While not specifically a new type of multi-touch technology, Cubtile is an adaptation of multi-touch technology towards a particular purpose. The purpose being the manipulation of three-dimensional objects using multi-touch which is currently a source of issues for developers. The Cubtile is a cube where five out of its six surfaces are multi-touch interfaces (one surface faces the floor and therefore does not need multi-touch). The technology inside the cube uses direct lighting with a fish-eye lens performing multi-point tracking. The documentation of the research does not go into much detail into the workings of the technology but rather focuses on the possibilities it offers. The research involves an evaluation of the technology but it is too brief and informal to draw any conclusions from. Rivière et al. [35] propose further research into the gestures the technology permits. The research also suggests

that future work may involve adapting the interface in some way to allow for multiple users. This is because Cubtile's current implementation only allows for one user at a time to take control.

Butler at al. [36] introduce a new technology called SideSight which uses the sides of a multi-touch interface to extend 2D interaction. SideSight is developed with smaller multi-touch interfaces in mind, such as mobile devices. However the research presents no arguments which state that the technology could not be adapted for use with larger interfaces. The technology works by placing infra-red proximity sensors (meaning infra-red light sources and detectors) around the edges of the devices facing outwards. These sensors can then pick up objects around the devices and identify them as inputs. This allows for more accuracy in the gestures made on small interfaces as the areas around the devices can be used as an extension of the interface.

There are some obvious issues with this method such as objects in the path of the infra-red light occluding anything beyond them from the device. Another issue is the problem of debris tolerance as any objects left in close proximity to the device could be detected as user inputs. The gestures demonstrated making use of this technology in the research use only two inputs, both always on opposite sides of the devices. This could be because performing gestures this way achieves the best results from the technology. However, as a result other gestures which could be possible are not demonstrated in the documentation of this research. All the examples concerning the constructed prototype in the research, place the device on a flat plane so that the surface it is placed on acts as the interface, though there is no obvious reason why the technology could not be used in open space. Most of the research focuses on explaining the workings of the technology and speculating on the possible uses but no formal evaluation of the use of the technology is made. Future objectives for the technology as stated by Butler at al. [36] include making the technology more accurate. However no mention is made of trying to extend the technology to support true multi-touch gestures rather than just the bi-manual ones demonstrated.

As well as adding features to multi-touch interfaces, new technological developments can be used to improve existing features. Rekimoto [11], discussed in Section 2.3.2, evaluates the use of pressure in multi-touch for further improvements in

interaction. The difference between positive and negative pressures can indicate the direction of a user's input and can provide indicators to the user's position around the interface. Some multi-touch software systems make use of the knowledge of the pressure of touch inputs to collect more information from each touch. However, without a technology capable of collecting this data the information cannot be attained or used by the software. This is a limitation which can be overcome by some technologies.

However Parker et al. [37] have highlighted a possible technological limitation of any multi-touch interfaces. This is the issue of retrieving virtual objects on a remote part of the interface from the user. Though this research has little relation to multi-touch as the technology used involves tracking a single stylus, the problem of retrieving a distant virtual object on direct touch interfaces is highlighted. In the research in question the solution is implemented through software with the use of a lasso type tool. No current multi-touch technology offers a solution to this problem and with a number of multi-touch interfaces being larger than traditional interfaces so that they can accommodate multiple users, this could be a cause of frequent frustration for multi-touch users. As with the research of Parker et al. [37], solutions for any hardware shortcomings in multi-touch could implemented through developments in its accompanying software.

2.3.4 Multi-touch Software Frameworks

Despite the different implementations of multi-touch technologies the accompanying software normally performs the same functions using similar methods. Multi-touch software can usually be divided into two sub-groups; **MSFs** and **Applications** (discussed in Section 2.3.5). The term MSF in relation to multi-touch refers to software which produces meaningful information from the data supplied by the technology. Some MSFs then provide extra functionality such as additional libraries or Application Programming Interfaces (APIs) to allow other software to utilise the collected information. This section focuses on MSFs and the features provided by them to the applications they implement. Unlike multi-touch technologies, few overviews of

the current state of the field of MSF development have been produced as mentioned in Section 2.3.1. An overview and comparison of several of the multi-touch technologies currently gaining popularity amongst developers is provided in this section. Also discussed is the importance of some of the common features of multi-touch MSFs. In the resources returned by the survey a number of MSFs were referred to. All these MSFs are discussed and compared in this section. These MSFs are: Tangible Interactive Surfaces for Collaboration between Humans (TISCH), Multi-Pointer X (MPX), Touchlib, Eunomia, BBTouch, Touché, Multitouch-Framework, Snowflake, GestureWorks, Bespoke, Python Multi-touch (PyMT), reacTIVision,Community Core Vision (CCV), Evoluce Multitouch Input Management (MIM), SynergySpace and DiamondSpin.

The line between where software is classified as a MSF or an application is difficult to define. Despite the similarities in multi-touch systems there can be some serious differences between their implementations. For example, if a multi-touch system is designed for a single purpose it is likely that the MSF and application software will be inseparable. However a system designed for developers to design applications will have a clear boundary between MSF and application. The MSFs covered in this section all follow this clear division of MSF and application architecture. Beyond this we can divide MSFs into two smaller groups, lower and higher level. Lower level here refers to the part of the MSF software which deals with collecting data from the hardware and turning it into meaningful information. Normally this involves calculating touch locations from the input and transforming them so they are relative to the output. Higher level multi-touch MSFs usually supply additional features to the software such as gesture recognition and other useful libraries. An example of these two MSFs is Microsoft's latest operating system Windows 7 which supports some multi-touch functionality when connected to any compatible multi-touch hardware. The operating system itself represents the higher level MSF whereas the hardware's driver or listener service would be considered the lower level MSF.

Some MSFs divide their lower and higher level sections to allow compatibility with other MSFs whereas others will not allow this compatibility. In this section a mixture of both these types of MSFs are discussed. Lower level MSFs are usually tied to a

specific technology whereas higher level MSFs can be interchangeable. This allows the same set of features and APIs from a specific higher level MSF to be applied to different technologies. Communication between the higher level software and lower level software can be done in a number of ways. The most common method is the Tangible User Interface Objects (TUIO) protocol [38]. This protocol has allowed a number of higher level MSFs to interact with a variety of lower level MSFs on various technologies. This is an example of an effort to create standards in multi-touch software.

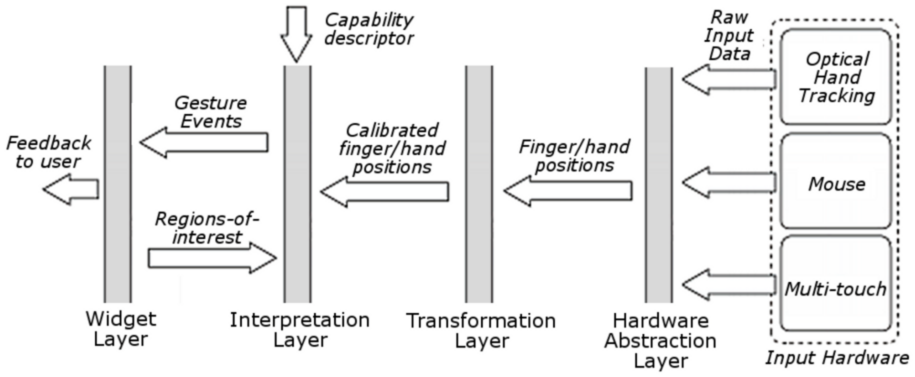


Figure 2.8: The software architecture common to most multi-touch systems. [34]

The lack of cohesion between different multi-touch developments is highlighted by Echtler et al. [34]. Their research provides an overview of the workings of software in a typical multi-touch system. The research highlights the similarities in the organisation of different implementations of multi-touch software. One common feature in the organisation of multi-touch systems is the division of software into two groupings. Multi-touch software can be described as being low-level, which primarily concerns input processing, or high-level, which entails managing responses to an input. The research proposes a standard multi-touch architecture based on these observations for all current and future multi-touch software to adhere to. Figure 2.8 shows the typical multi-touch MSF architecture which Echtler et al. [34] put forward. Each layer is described in detail as part of their research.

The hardware abstraction layer is where the raw information from hardware is used

to calculate the positions of touches and other input data. The transformation MSF maps input co-ordinates to their corresponding locations in the system environment. Between this layer and the interpretation layer is where the TUIO protocol is usually employed by the MSFs which use it. The interpretation layer then uses this modified input information to calculate what its effect should be on the current system environment. The widget layer then produces the output of the environment after the effects of the inputs have been applied.

The divide for most MSFs which differentiate between the lower and higher level occurs between the transformation layer and the interpretation layer. However some other MSFs incorporate the interpretation layer as part of the lower level MSF. The widget layer can be considered the application part of the software though some of the features of higher level MSFs are sometimes incorporated within it. As part of the research in question a MSF is being developed as a result of the observations made on the similarities between multi-touch MSFs which is detailed later in this section. This research, though short, provides an excellent over view of the current state of multi-touch MSFs. Varcholik et al. [39] in their research provide a more detailed overview of the hardware abstraction and transformation layers as shown in Figure 2.9. The process can be compared to a data pipeline such as those found in graphics processing. The example given is specific to a vision based multi-touch system but the process in other multi-touch technologies is similar.

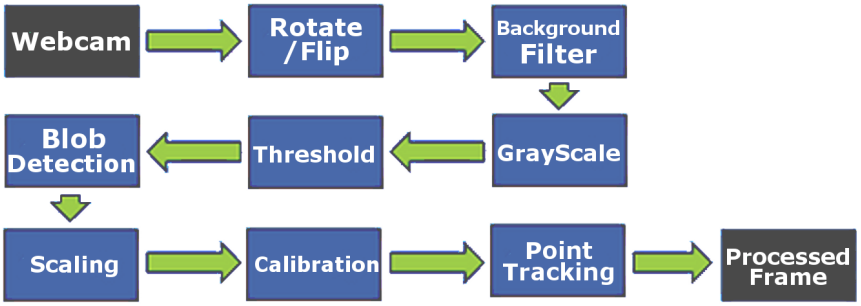


Figure 2.9: The image processing pipeline found in a typical low level MSF [39].

TISCH

The MSF developed from the work of Echtler et al. [34] is named TISCH. Echtler, [40] describes this MSF as an abstract architecture intended for use on any operating system with most multi-touch technologies. This MSF includes both low level and high level elements of a MSF, though either can be used with other MSFs. The lower level MSF provides methods and drivers to work with a range of hardware including cameras for vision based systems and DiamondTouch. TISCH can output with TUIO allowing higher level MSFs that use this protocol to use any hardware that TISCH can use. The higher level MSF offers several object tools for use in applications and the ability to recognise several common gestures. TISCH also enables shadow tracking and is used in the shadow tracking project documented by Echtler & Klinker [41] as discussed in Section 2.3.3. This MSF is only mentioned briefly in the original research by Echtler et al. [34]. However more recent research by Echtler [40] offers a lot more information on the MSF's continuous development. Part of recent TISCH developments is a patch which allows the MSF with interact with MPX.

MPX

MPX as mentioned by Echtler et al. [34] is discussed in more detail by Hutterer & Thomas [42]. The X in MPX refers to x.org, an implementation of the X window system used in several operating systems. MPX is a modification to x.org which allows for the window system to detect more than one input simultaneously. Originally designed to allow for multiple mouse inputs to control several cursors on screen, the MSF can now be used for any multiple simultaneous location based inputs such as multi-touch. MPX can be considered to be a higher level architecture and application as it provides an intermediate step between some low level MSFs and higher level MSFs. It provides an API which allows other software to access the information it has amassed from the lower level MSFs. The research by Hutterer & Thomas [42] does not go into much detail about the workings of MPX. However the research does evaluate the collaboration enabled by an application which utilises the MSF. In the evaluation multiple mouse pointers are used as the input, though due to the nature of MPX the

application could have used a multi-touch input.

Touchlib

Another MSF mentioned in the research of Echtler et al. [34] is the Touchlib library. This is a low level MSF built for vision based multi-touch systems intended for developers. The MSF provides a TUIO output and an API for use with applications built in the C++ programming language. Highlighted by the developers of Touchlib is the fact that TUIO is a protocol which can be sent across networks. This allows for one computer to collect and process the data from an interface. The calculated information can then be sent via TUIO to another computer which handles the higher level software and output. Currently Touchlib only functions on the Microsoft Windows operating system but it is a current objective of the developers to produce a cross platform version. Wallin et al. [43] provide details on the workings of Touchlib as well as executable and source code versions of the MSF itself.

Eunomia

Eunomia, as detailed by Cuypers et al. [44], is a multi-touch MSF engineered with use in public multi-touch displays in mind. The research provides a brief summary of existing multi-touch systems before explaining the Eumonia MSF in detail. Designed for use with FTIR technology the MSF contains elements of both a lower and higher level MSF. There is no separation between these lower and higher level software elements diminishing the ability to use Eumonia with other multi-touch MSFs. The higher level software in the Eumonia MSF offers several features to the applications it supports such as a gesture library. Also provided by Eumonia is a multi-media library which allows multi-media objects such as videos and images to react to gestures from the user(s). The purpose of this MSF is very specific as represented by its limited features and the small number of supported applications showcased in the research. This MSF is only intended for developers creating multi-touch system for use in public spaces. Since there are other MSFs which include the features of Eumonia among other desirable features which the MSF lacks it is doubtful Eumonia will be widely adopted

by developers in the future.

BBTouch

Varcholik et al. [39] provide information on a range of MSFs in their research which are also discussed in this section. Their research provides an overview of the important and useful components for a typical multi-touch system including details on where evaluation of the system's use is required. A MSF mentioned in the research of Varcholik et al. [39] is BBTouch, a new variation of the previously popular lower level multi-touch MSF called openTouch. BBTouch is a lower level MSF which collects information from cameras in a FTIR set up. Many different higher level multi-touch MSFs currently being developed utilise this MSF. BBTouch is designed to work with cocoa, a Mac OS software framework.

Touché

Cocoa is currently the software framework used by several separate developers creating multi-touch MSFs. One of these MSFs called Touché is documented by Kaindl [45] who provides an in depth description of the MSF as well as its source code. Touché consists of both higher and lower level MSFs and is designed to work with vision based multi-touch technologies. The higher level MSF does not include any features beyond supplying applications with transformed touch locations; however since a TUIO output is supported by Touché's lower level software other MSFs can be interchanged with the higher level software of Touché.

Multitouch-Framework

Multitouch-Framework is another MSF which makes use of the cocoa software framework and is discussed by Borchers et al. [46]. Providing both lower and higher level MSF components, Multitouch-Framework supplies input handlers for several different technologies. These supported technologies range from vision based system, such as FTIR, to capacitance based systems, such as the Apple iPhone. The higher level MSF supplies little more than transformed touch location information. However

since a TUIO output is supported by Multitouch-Framework's lower level software other higher level MSFs could be used instead. Though similar in many aspects to the Touché MSF [45] Multitouch-Framework differs in its support of multiple multi-touch technologies rather than just vision based systems.

Snowflake

The Snowflake suite is a commercially available MSF developed by Natural User Interface Technologies [47]. NUI Technologies is a commercial company who are not to be confused with the NUI Group mentioned in Section 2.3.2. Snowflake was developed as a suite of software for use on the Microsoft Windows operating system with vision based multi-touch interfaces. The suite supplies a lower level MSF which is referred to by Natural User Interface Technologies [47] as a tracking system called Touchcore. This lower software acts as a bridge taking the TUIO output from other low level MSFs and transforming the data to a format used by other pieces of software in the Snowflake Suite. Also in the Snowflake suite is additional software which enables inputs collected by the MSF to be used by Microsoft's Windows 7. Other software in the suite allows the low level MSF to collect input data from some instances of single and dual touch hardware. The documentation provided by Natural User Interface Technologies [47] provides little detail on the workings of the system. This is likely to be due to the Snowflake suite's commercial nature.

Gestureworks

Spadaccini et al. [48] document Gestureworks, a flash based higher level multi-touch MSF which works using information transmitted in the TUIO protocol. Gestureworks supplies a gesture library which any of the applications developed for it can access. Spadaccini et al. [48] make note of how their flash based MSF offers a better handling of the inputs resource-wise than other higher level MSFs. As Gestureworks is a commercial product few informed comments on the workings of the MSF are freely available.

Bespoke

Bespoke, a MSF for vision based systems containing both lower and higher level software, is discussed by Varcholik [49]. Point information is transmitted via Open Sound Protocol (OSC) between the lower and higher level MSFs allowing for other high level MSFs to be used with the lower level software. For the same reason the higher level MSF of Bespoke can be used with any low level multi-touch MSF which outputs its touch location information via OSC. The MSF currently offers a range of features, tools and applications which are made available to developers under the Berkeley Software Distribution software licence. Bespoke is well documented by Varcholik [49] who notes that the MSF is currently designed for use with Microsoft Windows operating systems only.

PyMT

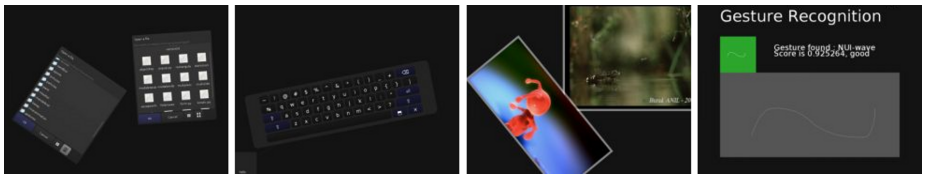


Figure 2.10: A series of applications for the PyMT MSF including file navigation, virtual keyboard, media manipulation and a tool for practicing gestures [50].

A MSF for multi-touch written in the Python programming language is documented by Hansen et al. [50]. Named PyMT, this MSF contains both higher and lower level multi-touch software. The lower level software manages a number of specific device listeners such as those for mouse or stylus inputs. Due to the project being coded in python the MSF can be run on Microsoft Windows, Apple OS and Linux making it cross platform. The lower level software supplies data directly to the higher level software which does not allow for the lower level MSF to be interchanged with elements from other MSFs. However the higher level MSF element of PyMT supports TUIO inputs so could be used with several lower level MSFs. The higher level MSF offers a number of features to the applications it supports including an animation

library and video playback. Hansen et al. [50] provide a great deal of information on the workings of the MSF and constantly provide information on the progress of the current developments of PyMT. A number of applications are available for this MSF, a selection of which is shown in Figure 2.10.

reacTIVision

A MSF gaining popularity among multi-touch developers at the time of the survey is reacTIVision. Documented by Kaltenbrunner & Bencina [51] reacTIVision is a lower level MSF designed for use on vision based systems, FTIR in particular. The MSF supplies touch location information via TUIO to any compatible higher level MSFs and also supports symbol recognition. This means that higher level software could use this information to recognise objects with specific symbols placed on them. Kaltenbrunner & Bencina's [51] research provides a large amount of information on the workings of the technologies compatible with reaTIVision. Also provided by the research are details of the MSF's architecture and features. Originally reacTIVision required users to place fiducial markers on their fingers for their touches to be recognised by the software. However recent releases of the MSF are now capable of recognising user touches without the need for these encumbering identifiers.

CCV

CCV, as documented by Ramos et al. [52], is quickly becoming very popular for use in vision based systems. Also known as tbeta, CCV is a low level MSF built for multiple platforms using vision based technologies as a multi-touch interface. Despite being a lower level MSF CCV does provide some features which would normally be associated with higher level software such as basic gesture recognition. The MSF outputs the calculated touch information via various outputs such as TUIO and customised Extensible Markup Language (XML) based protocols. This allows for a wide range of higher level MSFs and multi-touch applications to be supported by CCV. Ramos et al. [52] provide a large amount of information on the workings of the MSF along with examples of source code for applications to use with the MSF. Future

work on the MSF involves enabling CCV to support object recognition and to calculate momentum produced from movement of the input rather than just the touch positions.

Evoluce Multitouch Input Management

Another feature designated for future implementation in the CCV project [52] is the support of multiple cameras which is a feature that many MSFs lack. This feature allows for more than one camera in a vision based system to be trained on an interface which allows for the distance between the camera and interface to be reduced. This is useful for larger interfaces such as those used in tabletop displays like Microsoft's Surface [14]. A reason for this lack of implementation could be due to the issues which arise from areas of the interface on the edges of a camera's field of view. A possible workaround for this lack of support is to combine the inputs from the cameras with additional video editing software. The resulting video stream from the editing software can then be fed into the lower level multi-touch MSF. However if the fields of view of the cameras are spaced too far apart there could be areas of the interface which are not monitored by any camera meaning any user touches here would not be identified. To correct this cameras must be placed so that there are no blind spots. However this setup results in regions of the interface being monitored by one or more camera. These overlap regions can cause issues as any touches in these areas will be seen by more than one camera and will therefore be identified as two separate touches by the software. For this reason simply combining the two or more camera inputs is not enough.

Reducing the overlap area is difficult without creating blind spots. The curvature of camera lenses which mean that the edges of the fields of view are not straight enough to tessellate the camera views without any overlaps or gaps. Evoluce [23] detail the development of a lower level MSF that can handle inputs from several cameras simultaneously. The software detects when two simultaneous touches are viewed by two cameras in the same location and identifies that both are seeing the same touch and outputs the single touch. This allows for user touches to be tracked across the boundaries between camera view points. This means that the output of touch locations

from this software can be used effectively by any higher level MSFs no matter how many cameras are being used. Note that this is only possible if there are no gaps between the fields of view of the cameras. The Evoluce MIM, similar to many other VSs, can provide its output in the TUIO protocol but also has the ability to output its calculated data through Java RMI [53]. This allows the lower level MSF to interact with higher level MSFs that utilise the Java programming language with reduced latency compared to when TUIO is used. The reduction of latency is important to improve users' interaction with the interface. A delay between the user performing a gesture and the system reaction can lead the user to feel like they are not in direct control.

SynergySpace

SynergySpace is a multi-touch MSF designed for use in education. Higgins et al. [54] document the project which SynergySpace is intended for. There is little information on the technical workings of the MSF in this original proposal but the purpose of the software and its goals are outlined. The project intends to develop and evaluate the use of multi-touch in a classroom environment. SynergySpace is a higher level MSF built in Java allowing the software to be run across different operating systems. The MSF accepts a range of inputs including TUIO. Though designed for DI multi-touch technology in particular the TUIO support allows the MSF to support a multitude of multi-touch technologies.

The SynergySpace MSF manages the data supplied from any compatible lower level MSF. This data is then used to interact with elements in a virtual environment. This environment is rendered by the Java Monkey Engine (JME) [55] games engine. This games engine environment, which can utilise two or three-dimensions and employ realistic physics, is one of the many higher level features provided by this MSF. Applications can be designed to use this MSF if a developer requires direct access to the features of JME as several examples from the resource show. Though intended for a classroom environment the MSF could support applications designed for other purposes. More information on the current developments within the SynergySpace project is provided by Burd et al. [56].

DiamondSpin

Vernier et al. [57] document a MSF developed for the DiamondTouch table. DiamondSpin offers features which enable multiple users to interact with an interface in an orientation-less way. This means that users can use the interface from any direction or side of the display. One of the features of this MSF is a series of menu-bars where one is made available for each user. Thanks to DiamondTouch allowing for user tracking, users can each have their own menus which provide buttons labelled with symbols for a variety of different activities. These buttons act as an alternative to gestures and are similar to the icons used in most graphical operating systems. An example of one of these buttons is the 'magnetiser' which orientates all objects nearby towards the user as shown in Figure 2.11a. Another feature of the DiamondSpin MSF of interest is the interface current which moves icons used for accessing applications and functions around the edges of the interface. This is useful for large interfaces intended for multiple users as this allows the icons to be accessible to users without the need for them to reach across the table or into other users' workspaces. The features of DiamondSpin are also described in the research of Piper & Hollan [58] and Rick et al [59].

Comparison of MSFs

Multi-touch MSFs can be compared based on their features such as gesture support or platform compatibility. Table 2.2 compares the common features of the lower level MSFs discussed in this section so far.

The common features of the higher level MSFs discussed are compared in Table 2.3 . Some MSFs appear in both tables as they incorporate features of both lower and higher level multi-touch software. In lower level MSFs the features which are the basis for comparison are the technologies they support, the operating systems they support, TUIO output, its accessibility to developers, support of multiple cameras and the ability to recognise objects. Multiple Cameras refers to the ability of the low level MSFs to effectively make use of more than one camera in vision based systems. For the higher level MSFs their accessibility to developers, operating system support, TUIO support

		Lower Level MSFs												
		TISCH [34]	touchlib [34]	Eumomia [44]	BBTouch [39]	touché [45]	Multitouch.Framework [46]	Touchcore (Snowflake) [47]	Bespoke [49]	PyMT [50]	reacTIVision [51]	CCV [52]	Evoluce MIM [23]	DiamondSpin [57]
Features	Cross Platform	Y	N	Y	N	N	N	N	N	Y	Y	Y	N	Y
	TUIO output	Y	Y	N	N	Y	Y	Y	N	N	Y	Y	/	N
	Open Source	Y	Y	N	Y	Y	Y	N	Y	Y	Y	Y	N	Y
	Object Recognition	N	N	N	N	N	N	N	N	N	Y	/	N	N
	Vision Based	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
	Capacitive	Y	N	N	N	N	Y	N	N	N	N	N	N	Y
	Resistance Based	Y	N	N	N	N	Y	N	N	N	N	N	N	N
	Multiple Camera	N	N	N	N	N	N	Y	N	N	N	/	Y	X

Table 2.2: Comparison of the features of the emerging lower level MSFs.

and object recognition are again the features which the comparisons between the MSFs consider. In addition to these features are also a range of higher level additional features which some of the MSFs offer. TUIO support is one of the features assessed for the lower and higher level MSFs as this allows for interaction between different MSFs.

It is important to note the difference between the definition of object recognition for lower and higher level MSFs. For lower level MSFs object recognition relates to the ability of a system to identify an object from its outline on the interface or the pattern from a visual fiducial marker in the input from the technology used. In higher level MSFs the term object recognition refers to the identification of patterns of identified user touches (most likely processed by a lower level MSF before hand) and linking this pattern with an associated object. It is important to note that some MSFs which only feature in one table may be capable of supporting features only listed in the other table. As the features not listed in a MSF's table which it does fulfil are not deemed relevant to the comparison these features will not be noted in the table. '/' indicates where the support of a feature depends on the implementation of a MSF and 'X' indicates where

		Higher Level MSFs										
		TISCH [34]	MPX [42]	Eumomia [44]	touché [45]	Multitouch.Framework [46]	Snowflake [47]	Gestureworks [48]	Bespoke [49]	PyMT [50]	SynergySpace [54]	DiamondSpin [57]
Features	Cross Platform	Y	N	Y	N	N	N	Y	N	Y	Y	Y
	TUIO input	Y	Y	N	N	Y	Y	Y	N	N	Y	N
	Open Source	Y	Y	N	Y	Y	N	N	Y	Y	Y	Y
	Object Recognition	N	N	N	N	N	N	N	N	N	N	N
	Gesture Library	Y	N	Y	N	N	Y	Y	Y	N	Y	Y

Table 2.3: Comparison of the features of the emerging higher level MSFs.

the inclusion of a feature is not applicable for a MSF. For example the ability to use multiple cameras does not apply to capacitative and resistive systems.

Features of MSFs

Beyond providing applications with the location information of user touches, multi-touch MSFs can offer a range of features as shown by the MSFs discussed previously in this section One of the most common features is object recognition which is desirable for several reasons such as providing tactile feedback to users. Weiss et al. [30] highlighted this in their research on SLAP widgets as covered in Section 2.3.3. Some multi-touch technologies cannot detect objects without the use of a pattern of markers on the bottom of the objects as discussed in Section 2.3.2. This is an example of where software can overcome the shortcomings of the hardware used. This is beneficial for higher level MSFs that are intended to provide the same functionality, no matter what technology they are used with. Another example of MSFs overcoming a shortcoming of a technology is the implementation of methods for user tracking. Few technologies support this, and none can provide user tracking without encumbering the user without some sort of identification device. However some MSFs, such as CCV

[52], at the time of the survey being conducted are developing methods in which user's hands can be tracked using vision based technologies. This new development would allow for touch inputs to be grouped by the user.

The research of Hilliges et al. [60] concerns a multi-touch software system called Photohelix, a MSF built to support a selection of media orientated applications. This software supports the common MSF feature of object recognition with a set of control items, such as the pen object. These can be beneficial to users by providing a more tangible interaction with the system. Photohelix includes a gesture library which is a common feature among some multi-touch MSFs. The work of Hilliges et al. [60] highlights the importance of gesture design in multi-touch applications. Some MSFs will implement and define gestures whereas other MSFs rely on their supported applications to provide the gesture recognition implementation. With a set of gestures common to all applications supported by a MSF, users can gain familiarity with these gestures quickly allowing for better interaction.

Hilliges et al. [60] provide an overview of the features of Photohelix before stating objectives for its development. An evaluation of the MSF is documented in which a series of comments from participants are noted along with the evaluation's quantitative and qualitative results. Several observations on issues that need to be considered in gesture design are derived from these results by Hilliges et al. [60]. One such observation is the need for gestures not to require too much display space to perform. If a gesture is too large a user may not have enough interface real estate to perform it. The research also observes that gestures must be precise for manipulation of small objects or screen regions. Another observation on gestures was that users may confuse similar gestures or may perform an unwanted gesture when intending to perform another. The gestures implemented in Photohelix are all based around the control objects rather than user touches but the observations made in the research are still relevant to direct touch inputs.

Gesture support is an important consideration when deciding on a multi-touch MSF during the development of multi-touch system elements. Gestures in multi-touch are similar to commands in text based operating systems and keyboard short-cuts in graphical applications. This similarity is that they provide a method for users to invoke

commands directly without having to navigate a menu system. An important issue for consideration in MSF development is how intuitive a gesture is. Users interacting with a system will at some point want to perform a new action which they have not performed before. This may be as part of a task which the user has no experience of. The task could also be a familiar activity but in a scenario which is new to the user. With a growing number of multi-touch applications and MSFs it is important for a system to inform the user of the available gestures and their functions. This is important due to the possibility of different MSFs having their own gestures for different events and interactions. As with any interaction system a user will need to learn the interaction techniques somehow. Intuition alone cannot be relied on by developers as a method of informing users how to interact with the system.

Vanacken et al. [61] document the development of a tool to aid users in learning multi-touch gestures. Their research shows how a tool was implemented which informs users about the available gestures while an interface is in use. The research details the features and workings of the software but no evaluation of the tool had been performed at the time of its publication as the tool documented is only a prototype. The developed tool in its current implementation is an application which allows users to interact with a selection of multi-media objects using gestures from a MSF. The user's interaction with the media triggers events informing the user of the gestures available for their current activity. It is possible that this tool could be implemented to be part of a MSF allowing any application to make use of these tips when users show signs of hesitating. This tool is comparable with an application for the PyMT MSF [50] shown in Figure 2.10 which allows users to practice gestures.

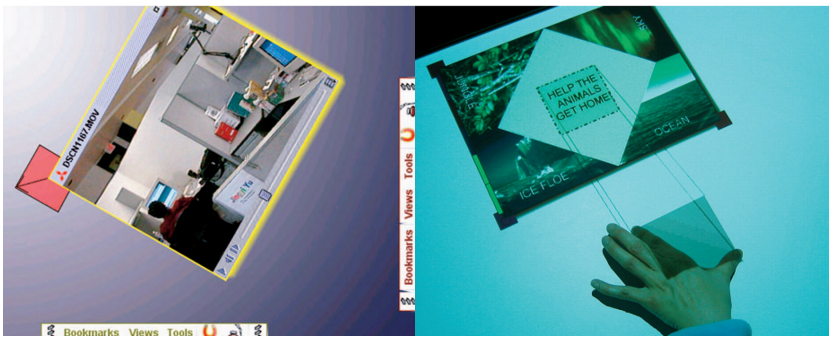
Rather than using instructions to inform users of the available gestures, allowing users to figure out the gestures themselves can be considered more intuitive. Wang [13] highlights the importance of mapping real world experiences to human-computer interaction techniques as metaphors. Wang [13] outlines how the gestures used for certain tasks in an intuitive system should mimic the real world actions performed to accomplish similar tasks. This allows users to draw on their own real-world experience if they need to perform a gesture for a task they have no experience of. For example, the gesture commonly used to flick objects on multi-touch interfaces can be traced back

to its real world counterpart activity of throwing an object. For both the gesture and real world action the user moves the object with the appropriate speed in the desired direction before releasing the object to allow it to travel with its own momentum.

A possible method to make gestures more intuitive is to create a common set of gestures which are used to perform similar tasks in different systems. There is currently a handful of these gestures, such as those commonly used for manipulating the transformation of objects (meaning the position, scale and rotation) of objects, in multi-touch. A series of these multi-touch gestures and techniques are covered by Shen et al. [62]. Their paper evaluates the collaboration enabled by applications on DiamondSpin as discussed in Section 2.3.4. Shen et al. [62], similar to Wang [13], make light of the importance in making sure gestures relate to the activity they are intended for. The applications evaluated as part of this investigation are covered in some detail in the research but the focus is on the observations derived from the evaluations. Shen et al. [62] detail DiamondSpin's ability to rotate everything on the interface in response to a dragging gesture performed on the interface. This is similar to the real life workings of a 'lazy Susan'.

Another feature the MSF offers is the use of proxy objects to control other objects. This is a solution offered to the problem of a user's hand obscuring the object they wish to manipulate as shown in Figure 2.11b. This solution is compared to the real life activity of controlling string puppets. While this feature does undermine the direct touch capability of the technology it is shown to be an effective solution by the research's evaluation. This feature could also be seen to resolve the issue detailed by Parker et al. [37] of manipulating objects on remote parts of a display as discussed in Section 2.3.3.

As highlighted by Shen et al. [62], hands on a direct touch interface can obscure the objects being manipulated beneath. This is an issue considered by Moscovich & Hughes [63]. As part of their research an investigation into whether inputs from one hand or from separate hands are better for particular gestures was conducted. The research details the experiment that was performed to evaluate both bi-manual (two hands) and uni-manual (one hand) gestures. The results from the experiment are discussed in detail with any observations made backed up by the data collected. The



(a) (b)

Figure 2.11: DiamondSpin MSF used by Shen et al. [62].

ultimate observation from this research is that bi-manual and uni-manual gestures are suited for different tasks. Gestures which require precision around specific points are suited to being performed by two hands as are gestures which require separate control of two points. Gestures which require a single hand are betted suited for tasks for which users have little reliance on visual feedback, such as scaling and rotating around the centre of an object. The research states that in the future developers and users must consider the number of hands that would be best suited for a task before hand. Also proposed for further work is an investigation into the reasons why a specific number of hands are suited for particular tasks.

Kin et al. [64] evaluate a number of different input techniques and compare them to the traditional mouse and keyboard based input in their research. Their work highlights the different techniques currently used in single touch, dual touch and multi-touch systems. Also detailed are the findings of previous works performed to compare direct touch technologies with indirect input techniques. The research explains the experiment set up to evaluate the different input techniques and provides an in depth analysis of the results. The speed and rate of errors performed by each input technique is recorded from the experiments as well as the use of different fingers for the touch inputs. Touch based techniques were much quicker at providing input than the indirect input techniques used though there was a higher rate of error when multiple fingers were used. From the observations derived from the results several guidelines for future

developments of multi-touch interfaces are suggested by Kin et al. [64]. One such guideline is the statement that single finger inputs should be used whenever a precise input is required due to their lower error rate as shown from the experiment. Kin et al. [64] conclude by stating that further research evaluating input gestures could lead to more guidelines for improving the development of gestures used in multi-touch.

Schöning et al. [65] investigate expanding the inputs a user can provide to a system by looking at the use of feet as well as hands in multi-touch. Using a Nintendo Wii Fit balance board, an evaluation was performed to explore how the use of feet can aid multi-touch inputs. Similar to multi-touch hand gestures, a number of foot gestures are used which the board can recognise. One such gesture is where users balance on the outer edges of their feet which returns the system to the main menu. In the research in question a map navigation application was used with these hand and foot gestures. Schöning et al. [65] explain the possible advantages of utilising feet as well as hands, such as the provision of more simultaneous inputs.

The research details the evaluation of the developed system and the qualitative data collected from the participants. This data showed that use of the foot and hand gestures together produced less fatigue than hand only gestures with increased input speed.. However the results also showed that the accuracy was worse with both hand and foot gestures than with hand gestures only. Participants also indicated that the foot gestures were less intuitive than hand gestures. Since foot gestures are new to both users and developers in relation to hand gestures their design and use could improve with future research. The MSF used at the time of the survey only supports a single application. Future work suggested by Schöning et al. [65] involves adapting the MSF to support more applications. Other future work outlined includes developing a foot sensor which allows users to roam more freely rather than being constrained to the balance board. As is evident from the research discussed in this section, gestures can be an important feature of multi-touch MSFs. In combination with other features these gesture libraries provide a metric for developers to use when deciding which MSFs to adopt for their own multi-touch systems..

2.3.5 Applications

The division between the different elements of multi-touch software can be difficult to define due to the lack of similarities between implementations. For example, several higher level MSFs perform one or more of the lower level MSF duties and some applications provide the functions of a higher level MSF. In this section applications which build on top of existing feature rich MSFs and those which partially incorporate their own MSF features are discussed. There is currently a slew of applications being released for multi-touch devices in the wake of the Apple iPhone. With multi-touch technology becoming more and more common the number of applications designed to incorporate this interaction technique looks likely to rise dramatically. The majority of recent applications developed for multi-touch fall into two categories. The first category consists of applications developed for multi-touch interfaces used in a particular scenario for a specific use. The second category consists of applications built as tools to aid general multi-touch use. Section 2.3.5 provides an overview of a selection of examples from the category of applications developed for a particular purpose.

Applications for a Particular Purpose

As discussed in Section 2.3.3, the TViews system discussed by Mazalek et al. [26] supports several multi-media applications. These applications are intended to provide leisure activities, such as viewing and sharing media, to multiple users simultaneously. The applications documented by Mazalek et al. [26] include an image sorter, map browser and a game called pente. Another application documented is springlets, which has no definitive objectives but offers users several objects which react in different ways to their inputs. Similar applications are also supported by other multi-touch systems such as Photohelix [60] discussed in Section 2.3.4 and PyMT [50] discussed in Section 2.3.4. These types of media orientated applications are common in multi-touch systems. This is likely to be due to the fact that they clearly demonstrate supported features of a MSF and the advantages of multi-touch over traditional input techniques.

Mazalek et al. [26] provide an overview of the applications available for the

TViews system and document an evaluation of the use of these applications. In the findings of this evaluation Mazalek et al. [26] highlight the possible causes of issues that could arise in everyday use of a multi-touch system. Such causes including the dissipation of a multi-touch interface's novelty and a system's placement in a user's home. Another cause of issue originates from the input detection by a system where prolonged use with inaccurate sensing can cause the user to become frustrated. Several of the findings provide important points of information for multi-touch developers to consider. One such point of information to consider is the importance to multi-touch developers and users alike of practical single-user activities. Mazalek et al. [26] state that these applications should be as critical to developers as 'leisure' applications which are intended for multiple users. Similar to Wang [13] as detailed in Section 2.3.4, the research highlights the importance of mapping real world activities to applications. The research of Mazalek et al. [26] concludes in specifying future objectives for the TViews system. These objectives include implementing more applications for the MSF and further evaluation of its use in real world environments rather than in lab based experiments.

Piper & Hollan [58] discuss the use of multi-touch tables and their effectiveness when used in undergraduate study groups. As part of their research an application was used to facilitate these study groups. This application features images which users can write on top of simultaneously The user can also erase text they or other users have written on the images previously. This is similar to the real world task of group study with the use of lectures notes. Once the users are satisfied with the text they have added to the images they can choose to display a layer of relevant notes which shows the text added to the image by a lecturer. These notes can include the correct answers to questions included in the original images or can highlight points of interest. The application is built for the DiamondSpin MSF, which was discussed in Section 2.3.4, and is used with DiamondTouch Tables.

Piper & Hollan [58], in their research, explain the features of the application and evaluate how student groups use the software and tabletop. In their research the results from groups using the tabletop are compared with the results from groups performing the same tasks with pen and paper. The results showed that the groups using the

multi-touch tabletop were able to complete most tasks in less time than the groups who used pen and paper. The tabletop groups were also more likely to choose to repeat a task which is beneficial for study. An observation derived from the evaluation of the application use is that for a text heavy application it is beneficial for all text to be orientated the same way. This orientation should be with users sitting along one edge of the interface so that the text can be clearly read. Piper & Hollan [58] provide further analysis of the results and conclude that even small and simple tabletop applications can be beneficial for educational activities. Educational multi-touch applications are a common trend in research. Applications that aid study can be seen to be becoming more common. Examples of this trend are the SynergySpace Project [56] and the OurSpace MSF used by Rick et al. [59] for their evaluation of how children collaborate with multi-touch

(a) (b)

Figure 2.12: The VPlay Application System [66].

Taylor et al. [66] document the use of multi-touch tables for advanced on the fly video editing. Known as 'VJing' this form of video performance art is becoming increasingly more common in venues such as nightclubs and at music events. The application, called VPlay, allows the tasks involved in video performance art to be executed using a multi-touch interface. The application functions by displaying events in video editing such as the transitions between clips as objects. Video editors can then quickly splice, cut and reorder clips as well as apply effects to them. The application also allows for the creation of new clips by capturing input from a video stream. Taylor et al. [66] provide an in depth explanation of the application after a brief explanation

of the scenario and technology. Another example of developers wanting to reduce the space used by vision based systems is present in this research as shown by the setup of DI tabletop system used. The table is designed to be partly collapsible so that its height can be reduced when it is not in use as shown in Figure 2.12b. Observations are made on the initial deployment of the application in a scenario similar to its intended use in the real world. Highlighted is the fact that the multi-touch interface allows multiple video editors to collaborate with each other simultaneously as shown in Figure 2.12a.

Noted in the research is the observation of how the large interface attracted other users, who were not directly involved with the video editing, and encouraged them to try interacting with the application. These lay users could build their own small sequence of events on the interface without affecting the simultaneous work of the video editors. Another observation made was that the video editors would use multiple fingers for the quick assembly of a series of events then use a single finger to fine tune it. This is similar to the observations of Kin et al. [64] in Section 2.3.4 of how a user can be more precise with the use of a single finger. An issue highlighted by observation of Taylor et al. [66] is that without tactile feedback the video editors needed to be continuously looking at the interface during its use. This meant that their vision was required by the application at all times which did not allow them to view the output footage in the venue and the audience's reaction to it. Future work proposed for this application is the support of physical devices. This is made possible as the technology currently used for the system is DI which can allow for object recognition. This is intended to allow the video editors to perform eyes free interaction with the table which is beneficial for uses of technology such as this.

The single purpose multi-touch applications covered so far in this section have all been shown to provide a possible advantage to users over traditional alternatives. Another single purpose application is documented by Dixon & Sherwood [67] . However, rather than being specifically multi-touch, the application is intended for use with any touch based interactive system. The focus of this application is to provide interactive whiteboard users with more functionality from the interface. Of interest is the use of a handwriting recognition tool within the application. The benefits of a system being able to understand a user's natural writing as an input are highlighted by

Dixon & Sherwood [67]. Also provided in their works is an overview and evaluation of the use of the tool. Hand writing recognition is a tool becoming more common in multi-touch software. Examples of this trend are the MathPad application developed for the SynergySpace [5] MSF and PyMT's [50] recognition of some user drawn characters. This is an example of where a tool can be implemented to benefit other software in multi-touch systems.

Application Tools

So far the applications covered in this section are developed for specific purposes, however Dixon & Sherwood's [67] inclusion of a handwriting recognition tool highlights the growing trend of developing tools for multi-touch either as part of applications or as stand alone applications. These tools could be implemented as part of future multi-touch MSFs but are currently developed as separate applications either for versatility or prototyping purposes. One example of these multi-touch tools is presented by Bailly et al. [68] who document the creation of a menu system designed specifically for use with multi-touch. The objective of this tool is to provide a menu system for multi-touch MSFs which is intuitive to access and use. The menu makes use of the entirety of a user's hand rather than using one point of contact like traditional menu systems. The multi-touch menu application is activated and sustained by identifying when the heel of the user's hand (the area of the palm closest to the wrist) is present on the interface. This makes the activation easy to perform, simple to remember and difficult to initiate accidentally which are all features desirable for an application such as this. Users then navigate and issue command to the menu through the use of gestures with their fingers and thumb on the same hand. These gestures can be combined to provide quick access to the elements of the menu.

Another advantage of this tool is that once the user knows the workings of the software they can access and navigate the menu with little reliance on the visual feedback from the interface. However an issue that arises from this application is that the user will need time to adapt to the new menu methods as they differ greatly from traditional menu systems. Another issue is that the application needs the lower

level software MSF to support recognition of the palm of a user's palm. This is considered an issue because many lower level MSFs will not supply information on the size or shape of a possible touch. This is information that would be needed by the application to identify the user's palm. Since the palm covers a larger area than a typical touch input some lower level inputs will not identify it as an input meaning the information will not be passed on to the higher level software. The lower level MSF could supply information that could be used for identifying the palm of a user on the interface in additional to touch data. However protocols used for communicating between MSFs may not support the transmission of this additional information. Bailly et al. [68] document the features of this application in detail and explain design choices determined by the physiological characteristics of a typical user's hand. The research proposes optimisation of the tool by adapting the menu for use by left handed users and evaluating the application in comparison to other menu techniques.

Benko et al. [69] introduce the idea of enhancing multi-touch input with a technique called muscle sensing. The system which this research uses is the Microsoft Surface in conjunction with an electromyogram which monitors the electric current associated with the contraction of muscles. Using this information the system is able to identify which finger of the user is performing which touch. This is beneficial in that it allows for tracking of users' hands and fingers. This system also offers improved debris tolerance over a typical Microsoft Surface set up. This is because it will be able to distinguish between touches that come from a user and other touches. However a drawback of this set up is that the technology is encumbering due to the need to wear the electromyogram device sensors. This can be seen as undesirable in a multi-touch system using Dietz & Leigh's [12] criteria. Benko et al. [69] also highlight other benefits of the system such as its ability to detect the pressure from a user's touch.

Another benefit highlighted is the ability to perform certain gestures above the display which can be detected by the electromyogram. They also state that in their future research they hope to make the electromyogram sensors less encumbering. The ability to distinguish fingers allows for many different uses, including the use of a specific finger to perform a specific action or initiate a certain event. This means that the fingers of the user can be used as a method of menu initiation and navigation

without the need of a menu which is displayed the whole time during use. Similar to the work Bailly et al. [68], individual fingers can be used to access features quickly at any time. A benefit of the system produced by Benko et al. [69] is that it can correctly identify a finger touching the interface without the need to make assumptions. This means there is less possibility of error and less of the display needs to be used up by the menu which are both desirable features of a menu system. However, whether these benefits are worth the encumbering nature of the electromyogram sensors is an issue to be considered by developers.

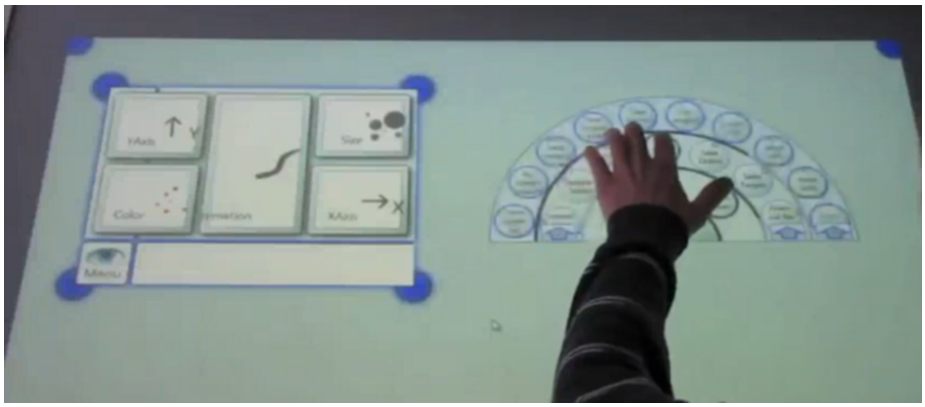


Figure 2.13: The Tap application system menu in use [70].

Another piece of research which implements a method of intuitively navigating commands and data in multi-touch is presented by Flöring & Hesselmann [70]. As shown in Figure 2.13, the application developed as part of this research contains a menu system. This menu is initialised by the use of the user's palm touching the interface as with the work of Bailly et al. [68]. The menu system is shown by Flöring & Hesselmann [70] to work in semi circular layers. The user on their current layer will choose an option which will then show another layer with more options. This is a method of implementing the tree structure of most menus in a way which is intuitive to use with multi-touch. The user can rotate the current layer so that the option they desire is accessible. The menu system does require a user to view the display so occlusion from the hand could be problematic. However this approach allows for the options of the menu to change depending on the scenario of its usage, rather than relying on a set

of static commands for all uses.

The application comes with other features beyond the menu. One such feature is the use of borders around visual items which can be used to resize and move the items whereas within the borders a user's touch will interact with the object itself. This allows for the typical multi-touch item transformation actions which are common to be applied to all objects which have their own associated actions and gestures. Another feature of the application comes from tapping a corner of the display which brings up a gesture layer that then performs an action based on the user's next gestures. For example the user drawing a circle with a single finger will bring up a help system which will inform the user on how to use the application. This is useful as without the gesture layer these actions could be performed by accident in normal use. For example moving an object in a circle would activate the help system when it was not wanted.

A problem with the method of initialising the gesture layer is that it could be performed by accident by actions such as a user leaning on the corners of the interface. Also the gesture layer requires the user to know the actions to be performed. Though the help system does provide information on these it does require the user to learn and retain a potentially large amount of information about the library of gestures. Also to access the help system the user would already need to know at least one gesture. The use of a palm to activate the menu is simple, easy to remember and is unlikely to be performed by accident in comparison to the other initiation gestures showcased in this research.

Despite the growing number of applications being developed for multi-touch there is a much larger range of applications available for traditional input systems. For this reason the topic of adapting these legacy applications to work with multi-touch is common in resources discussing the applications of multi-touch systems. As with any new human-computer interaction technique, problems can arise from multi-touch's compatibly with existing software applications. New interaction techniques and their enabling technologies benefit from being compatible with existing software or by providing methods of adapting legacy applications to work with them. Normally the time taken for software developers to adapt their applications to be compatible with any new technology will be proportional to the magnitude in the change between

technologies. With multi-touch being fundamentally different to current computer interaction techniques the time taken for developers to adapt their software for multi-touch could be substantial. This is important to consider since slow development of multi-touch systems could lead to users and developers adopting the technology slowly, or not at all.

Rick & Rogers [71] document the adaptation of a legacy application to be compatible with multi-touch interaction in their research. The application they adapt is a construction kit allowing younger users to learn about maths and art named DigiQuilt. The application is adapted to use the DiamondTouch multi-touch interface with the resulting software named DigiTile. Rick & Rogers [71] provide an overview of DigiQuilt and detail a case study of its modification to DigiTile. In the case study several different issues in the process of adapting a legacy application to be multi-touch compatible are listed. Also detailed are the actions taken to resolve them. The foremost issue was that the software was designed to take its input from a mouse device driver and not a multi-touch device input. The application needed to be modified to handle a multi-touch input and manage multiple points of simultaneous interaction. Rick & Rogers [71] highlight that these issues are not just specific to DigiQuilt but to any application being adapted for multi-touch.

When adapting a legacy application to handle multi-touch inputs consideration needs to be made of all possible concurrent activities in the application which could now occur. The software is likely not have been designed to manage concurrency between these actives before as it was not possible to perform them simultaneously with a single location input. Some applications may not be suited for adaptation due to this limitation. Therefore it is important to consider the concurrency of activities within the software when embarking on adapting an application to use a multi-touch input. The nature of the interface needs to be considered beforehand also. If the interface is a tabletop display then the application should not assume a fixed orientation as with a vertical display which most legacy applications are typically designed for. Also the display size should be considered as tabletop interfaces intended for group work will be a lot larger than typical computer monitors.

As well as the software changes to the application the use of the application was

noted to have changed as well. By allowing for multiple simultaneous inputs the application now allowed for collaboration between multiple users. Rick & Rogers [71] highlight this as a benefit, especially for use of the DigiTile application. However for other applications this could be seen as a disadvantage such as for any application where users are intended to work separately. Time needs to be invested during adaptation of an application to multi-touch in developing techniques to stop any of the functions enabled by multi-touch from undermining the purposes of an application. To conclude the case study an evaluation of the system's use was conducted. However the results are not present in the published documentation at the time of the survey being compiled. Rick & Rogers [71] note how quick it was to create DigiTile in comparison to the development time of DigiQuilt. They also highlight how the nature of the application changed to become more collaboration oriented.

The use of single touch interfaces for agile planning meetings are the focus of the system documented by Ghanam et al. [72]. This research does not specifically focus on the use of multi-touch, but the features detailed and the purpose of the application could be easily provided by a multi-touch system. The agile planning tools used in this research were not developed to support multi-touch, due to the limitations of the technology used with the software. However, throughout the documentation, references are made to how multiple simultaneous inputs could be utilised by future implementations of the software. In the research Ghanem et al. [72] state their intention to adapt the application for use with multi-touch systems. This is an example of developers adapting their own software specifically for multi-touch. Using the approach used by Rick & Rogers [71] this objective appears to be achievable.

The agile planning tool software features the ability to easily create and remove cards which can be drawn on. This is similar to the real world task of agile planning in which cards are used to write down ideas and pass them around. This is an example of an application mimicking a real world task. As detailed by Wang [13] in Section 2.3.4, this is a technique which helps users gain familiarity with an interaction technique.

The software implementation makes modifying and storing the cards simpler and also uses voice recognition for commands as an alternative input method. This is highlighted in the research as a useful feature of large touch interfaces. This is because

it provides an alternative input method which is not dependent on locations which could be beyond a user's comfortable reach. Ghanam et al. [72] provide an in depth overview of the features and typical use of the application. Their research also documents an evaluation of the use of the application in comparison to using traditional paper-based agile planning meeting methods. The method of conducting the evaluation appears to provide a wealth of useful information and details evaluation techniques that could be employed in the evaluation of other multi-touch systems. From this evaluation a number of qualitative results and observations are stated showing the touch interface to be favourable over traditional methods. One of the observations derived from the results is that participants state that they think the application would be better with a multi-touch supporting interface.

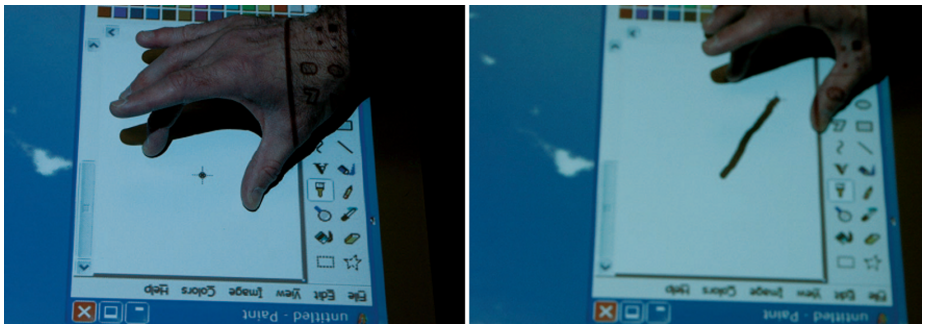


Figure 2.14: The gestures used for moving the cursor and clicking with the DTMouse [62].

An alternative to adapting legacy applications to become multi-touch compatible is to emulate traditional input techniques with multi-touch. Mouse emulation is used in several pieces of research as a method of supporting legacy applications. The research of Shen et al. [62] discusses the idea of emulating the classical input of the mouse. By emulating a mouse, any legacy software designed for a mouse input will be made compatible. In the research in question the gestures implemented for their mouse emulation are discussed. Shadow tracking, as detailed in Section 2.3.3, is not used in the system. Therefore the tool is implemented with separate specific gestures for moving the cursor and performing a dragging operation. The point between two fingers touching the interface is identified as the cursor location as shown in Figure 2.14. This

is useful as the user's hand will not occlude the cursor.

When a third simultaneous touch is made this is interpreted as click. Whether it is interpreted as a left click or right click depends on the location of a click in relation to the two fingers being used to control the mouse location. If the touch is detected between the fingers on the y axis then this is interpreted as a left click and if a touch occurs the right of the rightmost finger this is interpreted as a right click. This allows for all the same basic interactions that can be performed with a mouse, to be performed with the emulation tool on multi-touch. However several issues arise from this arrangement. For example a gesture could cause fatigue in the user's hand after long use as there is no physical mouse to rest the palm of the hand on.

Two other issues with this method of mouse emulation arise from the same causes of issues highlighted by Rick & Rogers [71] concerning the adaptation of legacy applications to multi-touch. The first issue is caused by orientation, as some multi-touch interfaces provide an orientation-less interface. If a user is interacting with an application one hundred and eighty degrees from the mouse emulations intended orientation then left clicks would be perceived as right clicks and vice versa. This mouse emulation tool is called the DTMouse and is built for the DiamondSpin MSF as discussed in Section 2.3.4. DTMouse is also used in the work of Wigdor et al. [73] who conducted an evaluation of day to day use of multi-touch. As part of this long term evaluation standard legacy applications needed to be supported which the single participant would have otherwise used on their traditional input computer. Providing a mouse emulation tool was the method chosen for providing a way to interact with these applications rather than adapting all the applications to utilise multi-touch inputs. It should be noted that closed source software can only be modified by developers with the correct privileges.

Varcholik et al. [39] provide an overview of the gestures utilised in their implementation of a mouse emulator for multi-touch. A different set of gestures from DTMouse are used with this mouse emulator built for the Bespoke MSF discussed in Section 2.3.4. The first difference to note is the simpler method of using one touch to move the mouse and a brief tap at a location to simulate a left click. Though capable of being performed by accident and causing occlusion of the cursor by the user's finger

this gesture is simpler than the DTMouse alternative. It is also similar to the commonly used gestures for track pads which provide mouse inputs to some laptops.

A number of gestures which require two touches are available for this tool including alternative gestures for moving the cursor and performing a left click. The original click and cursor movements could be disabled in favour of these alternatives as the two touch gestures are less likely to be executed by accident. Holding one touch on the interface and tapping to the right initiates a right click while performing a dragging motion the right of a touch causes scrolling. Also the task of 'tabbing' between windows can be performed by dragging a touch left and right above another touch. This is not a typical mouse input but an implementation of the alt-tab command used on a keyboard interface. It is also worth noting that under a change in orientation the gestures which rely on one touch will be unaffected whereas those which use two or more will function incorrectly.

Another mouse emulation tool is presented by Lo & Khoshabeh [74] called TUIOmouse. This tool provides mouse emulation to several operating systems using input from any TUIO protocol outputting devices or MSFs. This emulator uses a set of gestures which are similar to those used in both the DTMouse and the Bespoke mouse emulation application. Users move the mouse with a single touch then use an additional touch to the left or right to simulate a left or right click respectively. Also three simultaneous touches can be used with the direction of the middle touch being used to initiate scrolling events. Again, similar to the DTMouse, this emulator suffers from the problem of only working correctly with a single orientation. However, since the application only works in operating systems which have only one orientation of all their components, this is not a major issue.

Similar to the concept of mouse emulation is the virtual keyboard used in several of the MSFs and applications discussed in this chapter [5, 62, 30]. These virtual keyboards create a representation of a keyboard on the multi-touch interface. These representations can be used in the same way as a real world keyboard thanks to the ability to accommodate simultaneous inputs. However in the research of Wigdor et al. [73] the mouse emulation tool is used to provide input to the virtual keyboard. This was because the keyboard used in the evaluation was legacy software designed for a

mouse input. Therefore only one input could be provided to the virtual keyboard at a time. This was highlighted as a cause for the participant in their multi-touch system evaluation preferring to use a physical keyboard for text input. Both keyboard and mouse emulation can be seen as methods of using real world activates as metaphors for multi-touch interactions. As mentioned by Wang [13] this can be beneficial to users.

Mouse and keyboard emulation appears to be a quick and simple method of supporting older applications with multi-touch inputs. This may make older applications compatible with multi-touch but it will not allow them to take advantage of some of the features it has to offer such as simultaneous inputs from multiple users. New versions of applications built with multi-touch compatibility in mind can take advantage of these features. However some issues can arise from incorporating these new features that older software, built for traditional interaction techniques and input technologies, would not have needed to deal with. One such issue is caused by the multi-touch's allowance for multiple users to simultaneously use the same workspace. The interface can become cluttered with work from multiple users. Objects at different orientations can be inconvenient as users will need to rotate each individually to view them correctly. Pinelle et al. [75] address these problems and several others with a tool called TableTrays.

This tool creates areas on screen in which users can place objects. These areas act as trays which can be dragged and rotated around the screen keeping their objects contained within them in the same relative configuration. Pinelle et al. [75] in their work detail the workings of the tool and highlight the advantages of each feature of the trays. Also documented is an evaluation of the use of the system. From this evaluation several observations are made. One such observation is how the TableTrays help in the separation of work in collaboration. The trays are also noted as helping in filing away unwanted objects and in transferring large groups of objects at once.

The technology used in this research does not permit direct touch, but tracks multiple user held styluses around the interface. This allows the system to utilise user tracking in managing control of the trays. For direct touch multi-touch interfaces user tracking could be used with compatible technologies or alternative methods of tray access control could be investigated and implemented. The rotation and scaling

gestures used for the system trays rely on single inputs used in conjunction with regions of the trays. These could simply be removed and their functions replaced with the appropriate typical multi-touch gestures. TableTrays offer a solution to the issue of clutter in multi-user interfaces and could be implemented effectively as a tool for direct input multi-touch systems.

The applications in this sub-section show a selection of the many studies of adapting multi-touch towards particular uses The tools discussed in the latter part of this section provide a basis for multi-touch supporting general computer use. As detailed previously, these general tools could in the future be implemented into multi-touch MSFs as additional features. The ability to use legacy software on multi-touch interfaces through emulation, adaptation or another method allows for multi-touch systems to support the current everyday uses of computers. If these legacy applications are correctly supported on multi-touch systems the appeal of multi-touch to users wanting systems for both specific and general purposes could be broadened. New multi-touch uses could therefore be supported by the creation of new applications, the adaptation of existing software or the use of current applications with the emulation of traditional inputs.

2.3.6 Multi-touch Survey Summary

The findings from this survey show that multi-touch systems do have some common features. One such common feature between most multi-touch systems is the software architecture they utilise [34]. Another common feature is the intended use of multi-touch systems. Many of the systems produced are intended to support multi-media management and educational software. However the most common feature of the multi-touch systems encountered in this survey was the shape of the interfaces. All interfaces, except the 'Puddle of Life' application featured in the PyMT project [50], discussed in the survey are rectangular in nature. A current shortcoming of multi-touch systems is that no MSFs appear to facilitate the possibility of allowing its contents to be used with a range of interface shapes.

Few standards or protocols have been proposed for the mapping of multi-touch

software to the design of their interfaces. In the resources covered in the literature survey no work highlighted any direct regulation in mapping multi-touch software to the design of the interface. With common features such as support of the TUIO protocol some software attempts to become abstract enough to work on any interface. However this has the negative effect of this software then not utilising some specific features of particular multi-touch technologies. Rick & Rogers [71] do highlight the importance of considering the size and orientation of multi-touch interfaces when adapting software to be compatible with them. This could be expanded on to include other elements of physical interface design such as its shape. New research could focus on finding correlations between design elements and their effects on use and software development. Observations resulting from this research could be used to propose standards for future multi-touch developments.

2.4 Different Display Shapes

The results of the first survey on the current state of multi-touch technologies identified an area of research which could benefit from further investigation. How the shape of the display can influence the use and design of multi-touch software was identified as a gap in research by the survey. The first survey returned no resources which directly related to this area of research. This indicated that further investigation of this area would be required to evaluate if it would benefit from future developments. A structured literature survey was therefore conducted using the technique detailed in Section 2.2 to build a collection of resources relating to the use of different display shapes with multi-touch systems.

2.4.1 Trends and Patterns

The initial search was conducted using variations on the terms ‘multi-touch’, ‘software’, ‘non-rectangular’ and ‘display shape’ . These terms were chosen to represent the gap in research indicated by the first literature survey. On the initial search using these terms however very few related works were returned. The search terms used

were generalised and made less specific for when the search was repeated. This search returned more resources when the 'multi-touch' term and its variations were removed. However the total number of resources found by this survey after making the search terms less specific is relatively small compared to the number of resources returned by the survey relating to multi-touch research.

The small number of returned resources may be due to research on this topic using an unexpected terminology which differs from the terms used in the search. A series of structured searches were made using a number of possible terms which could relate to different display shapes. These searches however returned the same set of resources returned by the initial search so that it is unlikely that research exists on this subject which uses different terminology. Another possible reason for the small number of returned resources is that there are only a small number of academic sources relating to the topic of different display shapes. This possibility is much more likely due to past technological constraints causing the employment of non-rectangular displays to be limited. The resulting returned resources can be placed into two categories. Either the resource relates to a technology based development that allows for a new shape of visual output or it relates to software designed for a non-rectangular display. Therefore the findings from this survey are divided into two sections. Research relating to technology based developments is discussed in Section 2.4.2 and software based developments are discussed in Section 2.4.3.

2.4.2 Technology

The past technological constraints on display shapes are now starting to be removed by innovations in hardware design and adaptation. The vast majority of displays used in computer systems are rectangular. However there are several examples of non-rectangular displays which are now becoming available. One such example is Toshiba Matsushita's circular TFT LCD display [76] as shown in Figure 2.15. This monitor is employed in vehicles to display typical dashboard information via a liquid crystal display the shape and size of a typical car dashboard dial. Another example of a non-rectangular display is the Motorola Aura [77] which features a circular display.

The typical features of mobile phone software are displayed through this display. These displays are showcased with some supporting software but how the software handles the non-rectangular displays was not disclosed in the resources reviewed. It is likely that the software for these displays is designed for a specific output shape. Also since the displays discussed so far are tailored for specific purposes it is likely that the software will only offer a limited amount of functions. The growth of the availability of non-rectangular displays is shown by the number of patents filed in the last seven years for technologies which permit non-rectangular displays [78, 79, 80].

Figure 2.15: Toshiba Matsushita's Circular TFT LCD Display [76].

Discussed in the works of Coelho et al. [81] and Holman & Vertegaal [82] is the concept of deformable displays. An implementation of this concept is showcased by Lee et al. [83]. Their work details the technological development of displays which can be folded by the user. Several example surfaces which can be used with the software documented in this research include a sheet of paper, a scroll, an umbrella and a fold-away fan. The output from the system is projected onto these surfaces which can be folded.

Sensing techniques which make use of infra-red markers on the surfaces are employed to detect the positioning, orientation and deformation of the displays. Also

additional infra-red emitters can be used as an input allowing for a direct input. The sensing techniques can detect when the display is folded, how it is folded and then adapt the output of the system to fit the new display shape. This is an example of a system adapting its output to fit different display shapes in a dynamic manner. However the software currently used is built to adapt to a small range of potential display shapes. The layout of content is specifically designed for each significantly unique shape that can be achieved through deformation of the display. This means that the software is unlikely to be dynamic enough to adapt to a display shape without prior configuration of its visual contents for the specific shape in question.

The D20 device discussed by Poupyrev et al. [84] is an example of where a system can utilise non-rectangular displays. This device is intended to be a hand held interface which displays an output on all its sides. The device's visual output is provided through displays which each cover a side of the object. As the object is not expected to always be a cube, the shape used for the device in the research's documentation is a regular icosahedron; the displays are not expected to be rectangular. In the research carried out with the regular icosahedron each side is triangular.

There is an added level of complexity in the display of the output from this device as it is intended to use several of its displays simultaneously. For example the device may show a menu which stretches across the sides of the object and therefore appears on several displays at once. Not only are the individual displays unlikely to be rectangular but the combination of several displays can create a combined visual output which may not be rectangular. With the combination of displays on different sides of the device this combined visual output will also be over three-dimensions. The shapes of the individual displays and the possible combined output of the displays to form an encompassing visual output are known to the developers before hand. Therefore the software could be tailored specifically to the limited range of potential shapes the device could use as an output. However the software will require some flexibility in its ability to adapt to different display shapes, a concept discussed in more depth in Section 2.4.3.

Benko et al. [85] discuss another device which utilises a three-dimensional output for its display. This device is called sphere and projects its output onto a three

dimensional globe. The sphere device utilises multi-touch interaction performed on the surface of the globe to receive input from the user. It is noted by Benko et al. [85] that the software used with the display was adapted specifically for use with the device. The software is not dependant on a rectangular display, but instead requires the use of a spherical display. Some features of the software are entirely dependant on the spherical nature of the display. For example, the menu system which the devices uses is arranged around the top of the sphere so that it can be viewed from most perspectives that a user may be stood in around the device. Due to the curved nature of the display a user can only ever see, at most, half of the full visual output at one time. The software is designed to accommodate this by ensuring any content shown to a user is capable of being orientated and scaled to fit the region of the display which the user can see.

Toshiba Matsushita's circular TFT LCD display [76], deformable interfaces [81, 82], the D20 device [84] and the sphere device [85] are all technologies designed with the intention of supporting ubiquitous computing. Ubiquitous computing is a term applied to computer systems which blend into a user's environment so that their presence is not noticeable [86]. Greenfield [87] argues that an era of ubiquitous computing is dawning where more systems will be designed to blend into their environment. Ubiquitous computing becomes easier to achieve as emerging input technologies, such as multi-touch, and more complex types of computer communications, such as RFID tags, are made more widely available. For computer systems to be ubiquitous they are required to be designed around typical user environments, rather than having user environments tailored around them as has previously been the case. The implication of this is that many previous standard elements of typical computing systems will need to be reconsidered. One of these typical standard elements is the shape of a system's interface.

Weiser's discussion of ubiquitous systems [86, 88] highlights how these systems will augment current every-day activities using natural interactions. This work discusses how a well designed system should become unnoticeable when not in use. This requires the system and its interfaces to be capable of fitting into any environment. Due to the costs involved in designing ubiquitous hardware, and the software to support it, for each environment, it is important for a system to be capable of dynamically

adapting to its environment and usage. The deformable interfaces discussed by Coelho et al. [81] and Holman & Vertegaal [82] are examples of ubiquitous interfaces which can be used in a variety of scenarios. It is likely that the ability for interfaces to be easily reshaped to fit their current environment and usage will become a common feature of ubiquitous computing in the future.

In Weiser's work [86, 88] a prototype tablet was designed and used as part of a demonstration of ubiquitous computing. The current popularity of similar tablet devices [89] demonstrates how there is now a growing usage of ubiquitous designs in modern computers. The main reason for the current increase in the adoption of ubiquitous system designs are the advantages they offer over more conspicuous systems. Van Dam [90] highlights how ubiquitous computing helps in reducing the separation between users and computers. This allows for computer interaction to be much more natural which aids computer interaction.

The work of Steimle et al. [91] identifies a major problem that ubiquitous computing may encounter. This is the issue of occlusion. In particular Steimle et al. [91] discuss the issue of a user occluding parts of a vertical display. In the work it is discussed how a user may for different periods of time occlude visual information shown on a display. The system discussed identifies the areas which are occluded most often. This work identifies how occlusion can affect system use and how ubiquitous interfaces are susceptible to this issue.

Due to the varying designs of potential ubiquitous interfaces their visual contents will need to be adapted to fit different displays. The work of Weiser [86, 88] and Steimle et al. [91] show how ubiquitous computing environments contain displays of many varying sizes. These varying sizes may lead to the need for visual content of the systems to be extremely scaled up or down to fit a display. Having visual information scaled too large or small can be problematic for users. For example, text made too small can be impossible for some users to read. This is a problem common to many interfaces, not just those in ubiquitous systems. As there is a wide range of displays currently available to users which have different sizes and resolutions, developers have had to ensure their software is capable of adapting to different display sizes [92]. This is usually done through using ratios based on a display's resolution for positioning

visual contents, rather than using absolute values. Since there are existing solutions for managing visual content on displays of different sizes, the scaling of the visual outputs of a system is not such a large issue for ubiquitous interfaces.

The work of Weiser [86, 88] also highlights how in an environment where ubiquitous computing systems are used, the interfaces of a system could be positioned in various orientations. In the environment described in the work, there are a number of vertical displays used, along side tablet interfaces which can be used both vertically and horizontally. An additional orientation issue with tablets is that they can be rotated along a number of axes. For example the tablet's display can still be rotated while being kept vertical. This could be problematic in some scenarios where visual content could be placed so that it is orientated in such a way that makes it difficult to convey information to the user. For example, a rectangular tablet can be kept vertical and rotated by the user from a portrait to a landscape orientation. Without corrective software any text which was originally aligned correctly will now be orientated ninety degrees from the users perspective and will be difficult to read. This is a problem common to many interfaces, such as those found in mobile phones and the current generation of tablet devices. As these types of ubiquitous interfaces are common, developers have already provided solutions to this issue. The most common is the use of a device's accelerometer to detect which way it is currently being orientated [93]. The visual contents are then rotated to fit this orientation. Since there exist solutions for managing visual content on displays of different orientation this, in addition to scale, is not such a large issue for ubiquitous interfaces.

As mentioned by Steimle et al. [91], occlusion can be a major issue for ubiquitous interfaces. Due to the nature of their design and use, some regions of the display may be made unreachable due to occlusion. This may be due to a user's interaction with a the system where direct touch is involved. Occlusion may also be caused by the design of the interface. Features of the environment, or the system itself, could cover areas of a display. Occlusion is a major issue as visual information can be lost when hidden from a user's sight. Input is also impaired when the necessary visual components for an interaction are occluded.

For displays which can easily have their shape changed, such as the deformable

interfaces discussed by Coelho et al. [81] and Holman & Vertegaal [82], parts of the display may not be visible to the user. This is not because part of the display is being covered but because it is facing in a different direction to the interface's main area. This is another form of occlusion. Despite the fact that these areas of the display are not covered, they are still not visible to a user viewing the interface's main area. None of the works returned in this survey identified potential solutions for resolving occlusion in ubiquitous computing. Therefore, out of the three issues which can affect ubiquitous interfaces of (scale, rotation and occlusion), occlusion poses the largest threat to the use of the system. This is because, unlike the other two issues, there are currently no standard solutions to this issue of occlusion.

The work of Kammer et al. [94] highlights how the most common gestures for multi-touch systems include those used for rotation and scaling. This means that in addition to existing solutions regarding the rotation and scaling of visual content items, multi-touch gestures could also be used for correcting these issues. If a visual content item does not have a fixed position a user could use simple movement commands and gestures to resolve occlusion. But this is not always possible, especially when a content item is fixed in a specific position. In these circumstances it becomes the software's responsibility to correct the occlusion.

For a system to be ubiquitous it may be required that the system's interfaces are built around its surroundings. This may require changing the size, orientation and shape of an interface. The deformable interfaces [81, 82] and Toshiba Matsushita's Circular TFT LCD Display [76] are both examples of this. The work of Coelho et al. [81], Holman & Vertegaal [82] and Poupyrev et al. [84] have shown that there are technologies capable of adapting systems to accept inputs from interfaces of different shapes. The research discussed has also shown the ability of technologies to produce outputs tailored for the predicted potential shapes of the interfaces used. Developments in ubiquitous computing have shown how technologies are able to create and manage interfaces of various shapes. However for these interfaces to be supported, software which is able to utilise different display shapes is required.

2.4.3 Software

Though there are new technologies emerging which allow for non-rectangular displays there have always been methods for giving displays different shapes. For example, by covering regions of a visual output, such as a monitor or projection, the shape of the visual output can be easily changed. This means that most current technology can be adapted to allow a system to use a non-rectangular display. However there is little call for these display shape changing techniques due to the reliance of the majority of existing software on rectangular visual outputs. This section first looks at common user-interface design practices used in the majority of current software and identifies why many may not be suitable to non-rectangular visual outputs. Several instances of software that is not reliant on rectangular visual outputs is then discussed.

Current common graphical software design practices are inappropriate for non-rectangular displays. In the work of van Dam [90] it is noted that graphical interface components are all designed for use on traditional rectangular interfaces. The work discusses how common user interface features, such as windows, icons and menus, are all tailored for a rectangular output. The individual elements of the visual output of a piece of software are referred to in this work as visual content items. Each element of a display which can be positioned separately by the software is classed as an individual content item. In operating systems visual content items can include icons, windows and menus.

Van Dam [90] describes how the visual elements of many software systems would need to be redesigned for ubiquitous computing. Many are required to be redesigned to adapt to the orientation of a display. For example it is common practice for icons to be accompanied by text. On a horizontal display which may be viewed from various perspectives the inability to rotate these icons and their accompanying text would make them difficult for users to view correctly.

Many of the user interface elements of current software would not be suited to ubiquitous interfaces where the shape of the display may vary. For example context menus used by many graphical software systems are rectangular. One of the main reasons for this is so that their edges can align with the sides of a rectangular display.

Calculations for checking that the menu will not extend beyond the monitor are easy to carry out when both the menu and display are rectangular. If either of these elements were to be potentially non-rectangular, the calculations involved would become much more complex.

Many visual content items, such as the windows commonly used in graphical operating systems, are usually rectangular. This may be due to technological constraints or restrictions of the software a system is built on. Another reason for the rectangular nature of visual content items is because of their capability to be scaled fill a display of the same shape. An example of this is the maximization of windows in most graphical operating systems. Due to the rectangular nature of the majority of visual content items many functions of software systems only accommodate rectangular items. For example the maximisation of a window could be problematic for current methods if a non-rectangular window was used on a rectangular display [95]. These types of methods would be required to be modified to accommodate the possibility of non-rectangular displays.

There are instances where some graphical systems have been adapted to make more use of non-rectangular content items. The X window system used in some Linux systems is an example of an operating system component with the ability to modify its visual contents. This window system makes it possible to create non-rectangular windows [96] which could be beneficial for the use of Linux operating systems with non-rectangular displays. The Compiz window manager [97] used in some Linux operating systems can be used to modify the appearance and behaviour of visual operating system elements. Therefore this window manager could be used to change the shape of the visual contents of an operating system to fit a different display shape. These methods of altering the visual components of operating systems highlight how existing software can have their reliance on the display shapes they were originally intended for removed.

It is not only the visual content items of a system which are designed assuming a rectangular display. As mentioned before in relation to the management of non-rectangular windows, many common user-interface functions are reliant on a rectangular display. A common graphical operating system function is the alignment

of the windows on a display so that they make the best use of the display's real estate do not overlap. This can be done using very simple calculations when rectangular displays and visual content items are involved. If either of these elements may possibly be non-rectangular the calculations involves become much more complex.

Shortcomings in existing software with regard to accommodating different shaped displays have the implication that software cannot rely purely on current interface design practices. For software to utilise different shaped displays additional effort must be put into the design of its visual components. The software used with the non-rectangular displays discussed in Section 2.4.2 can be assumed to be applications or frameworks specifically tailored to the one specific output shape. This is true for the PyMT project's 'Puddle of Life' [50], the only resource returned in this survey that directly relates to multi-touch. This application is designed specifically for a circular display shape and is not intended for use with any other visual output shapes such as the typical rectangular display. In addition to this circular display-dependent application the PyMT multi-touch framework showcases several applications which utilise a number of rectangular displays ' stitched' together. This means that several visual outputs are seamlessly joined together to create one large display. This large display must always be rectangle but the applications can function correctly with the unusual aspect ratios this set up allows.

Dietz et al. [98] discuss a system in which software is needed that can dynamically adapt to different display shapes. In this system a series of projectors are set up to display the output from a system onto several surfaces of differing shapes and sizes. From the projectors several separate visual outputs are produced. The outputs are not separated by projector however but by the surface they project onto. The projects provide one continuous output by using a 'stitching' technique similar to that used in PyMT. However unlike PyMT the encompassing output resulting from the 'stitching' of these projected outputs does not need to form a rectangular shape. The research in question highlights how readily available technology, such as the projectors used, can produce non-rectangular displays by 'stitching' multiple visual outputs. The resulting encompassing output is then divided into separate visual outputs which are the same shape as a corresponding surface that they are to be projected onto. This projecting onto

different shaped surfaces is another example of a method of producing non-rectangular displays with widely available current technology.

The output from the system is tailored to fit the surface it is projected onto. This is achieved by creating a virtual model of the surfaces the projectors will use which the software then adapts the visual output to. This requires some user involvement for the design of the virtual surface models but requires that the software be somewhat adaptable. The software used in this work for both case studies detailed appears to be relatively simple. The visual content items are large and non-transformable. This means that once the content is positioned by the software it cannot be transformed in anyway by user interaction. Large images and videos which fill displays are cropped so that they are projected only onto the displays and not the surrounding areas. This shows how the software adapts the visual output to the display shape. Despite the simple content of the system this work shows that software can be made capable of adapting its visual output to different display shapes dynamically.

Research which highlights the need for software to be adaptable to different display shapes is presented by Holman & Vertegaal [82]. This work highlights that system outputs are no longer needed to be constrained to the typical rigid structure of current display technologies. The main focus of this work is the consideration of new technologies which could enable future interactive devices to be any shape or size. Examples of interface devices given by Holman & Vertegaal [82] include soft drink cans, globes and paper. This highlights that systems may need to not only adapt their output to displays of differing two-dimensional shapes but also of differing three-dimensional shapes. The ability to do this can be made possible by projecting onto the surface of an object. For the examples given in this work the system has prior knowledge of the possible objects it may be projected onto. The software is then informed, either by the user or through a sensing technology, which object is being used.

It is apparent that the software used in the research discussed is tailored specifically for the objects used in the examples. The discussion of this work focuses on the possible ability to deform an interface of an interactive device to aid input. This ability to be deformed may also require the software to adapt the output to the deformation

of direct-touch interfaces. Developers may want the same software to run on different interface objects to improve interaction with the system. If a user has gained familiarity with the software on one interface device then the use of the same software on a different device with a different interface shape will ease the transition between devices. This highlights that the need for software to be able to adapt to different shaped interfaces may become a real necessity in the near future.

2.4.4 Display Shape Survey Summary

Section 2.4.2 highlights how non-rectangular displays are technologically feasible. These non-rectangular displays may take the form of specific shapes or of deformable interfaces which uses can reshape at will. Ubiquitous computing, where computer systems become part of the environment, was identified as likely to benefit from the use of non-rectangular displays. With ubiquitous computing becoming more common the need for non-rectangular displays is likely to grow.

It is important to note that there are issues which must be overcome when designing software for use with ubiquitous interfaces. These issues include the scaling of visual content items due to the wide range of potential display sizes and the rotation of an item due to the various orientations a ubiquitous screen may be placed in. However both these issues have existing solutions which are common in many software systems. The major issue of occlusion was also identified and was highlighted as having no common solution unlike the other issues. Occlusion was identified as likely to occur in interfaces where different shaped displays are used and must therefore be considered when producing software for non-rectangular displays.

An important focus of ubiquitous computing development is the adaptation of software so that it can utilise a wide range of displays with differing designs. The literature survey returned a number of resources discussing ubiquitous computing. These resources detailed pieces of work on ubiquitous systems which were concerned with ensuring that software would work on displays of differing orientations [93] and sizes [86, 88]. However, for some interfaces to be truly ubiquitous, non-rectangular displays may be needed. Some systems may be intended to blend in with a number

of different environments. It is likely that each environment will require a system to use a different shape for its interfaces. Therefore it is important for software used in ubiquitous computing to be capable of adapting its visual contents to different display shapes.

With current hardware able to support a range of display shapes, the feasibility of software which can utilise multiple display shapes must be considered. Section 2.4.3 describes several pieces of research which involve software which can utilise a range of different display shapes [83, 84, 98]. It is interesting to note that large corporations are now investing in the development of non-rectangular displays, particularly circular interfaces. This growth in interest and use may lead to a situation soon where end users could potentially be using one of several display shapes available on the market. With developers unaware of which display shapes a user may have access to their software must be able to adapt to different display shapes. This implies that software must not rely on a specific display shape.

A major issue in implementing systems which utilise different display shapes is the adaptation of the visual contents of the software to fit these new display shapes. Existing frameworks and operating systems are usually designed for specific display shapes. This is because many of the graphical elements common to most software systems, such as menus, windows and icons, are designed to work specifically with non-rectangular displays. Not only are the visual content items of many systems reliant on a rectangular display but the functions of many user interfaces are also designed on the assumption that they will only be used with rectangular displays. Once a software framework is designed to be display shape independent, then it is possible for the software it supports to become display shape independent as well.

Many common software systems have a large dependency on rectangular displays but can have parts of their visual content adapted. For example some adaptations may allow graphical operating systems to be made compatible with non-rectangular displays. The rectangular shape of window frames common to many operating systems is an example of modern graphical operating systems' reliance on rectangular displays. However in some operating systems the appearance of these windows can be modified through several techniques. The window management components of an operating

system can be modified to change the shape of operating system elements including the windows used. Of course changing the design of visual components alone is not enough. Many of the functions which manage the visual elements common to many current user interface designs are dependant on a rectangular display. This holds true for user interfaces designed for ubiquitous interfaces. Functions may be implemented to allow software to use displays of different orientations and sizes [93], but these functions are designed under the assumption that they will be used with a rectangular display. Therefore when designing software for non-rectangular displays it is important to consider the design of the visual content items and, more importantly, the functions which manage them.

The resources reviewed in this section show that there is a growing need for different shaped displays in modern systems and that current technologies are capable of supporting them. These resources also show that there are methods of configuring the content of a system to fit a display, or series of displays dynamically. For this to be done information on the shape that the visual output will utilise needs to be provided to the software. Though most of the software discussed was designed for either a specific set of output shapes or to display simple content they reveal that it is possible for a system to adapt its visual output dynamically to different display shapes.

2.5 Chapter Summary

In this chapter two structured literature surveys were discussed. The first provided an overview of the current state of research relating to multi-touch. In this overview a gap in research was identified concerning how the design of a multi-touch interface, specifically the shape of its visual output, effects interaction. This area of research was identified as being likely to benefit from further investigation and potential future developments. A second literature survey was therefore carried out which investigated this gap in research. This second survey initially focused on multi-touch systems that used different shaped displays. However relatively few resources were returned compared to the number returned by the first survey. Therefore the scope of the second survey was widened to focus on any technologies or software systems which enabled

the use of different display shapes. This second survey highlighted how most software is built for a single specific display shape, usually a rectangle.

Also highlighted were software systems that could adapt their visual content to a small range of display shapes. However these software systems usually required configurations of their content for each display shape to be defined prior to the use. The combined findings of these two surveys showed that the use of different display shapes with multi-touch could be beneficial and that there is the technology to make it feasible. However currently there is no multi-touch software which can dynamically adapt its content to different display shapes. The methods used in the software systems discussed in Section 2.4.3 to adapt their visual contents to different display shapes are unlikely to be directly applicable to multi-touch software. This is because the visual contents of multi-touch systems are much more complex than the software used in the systems discussed due to their potential ability to be transformed by the user. Also visual content items in the multiple-display shape supporting systems discussed were cropped which may be detrimental to the visual contents of multi-touch systems which may contain visual information that would becoming meaningless or incorrect if parts of it were to become missing. These findings were used to inform the decision that the research documented in this thesis should focus on investigating methods which could allow multi-touch software to become display shape independent. This line of investigation was considered to be likely to lead to delivering the initial objective of the research, a development that would be beneficial to multi-touch systems.

Chapter 3

Problem Statement

3.1 Chapter Introduction

The feasibility and implications of creating software to utilise different display shapes were considered in Chapter 2. Prior research highlights that software needs to be tailored for a display shape or else issues may be encountered. Some of these issues may only be encountered in specific circumstances whereas others are common to all instances of software that use different display shapes. This chapter identifies the major issues that were observed when using software which is dependent on a particular display shape with a differently shaped display. The causes and implications of these issues are also discussed. These issues were identified from observations on an experiment.

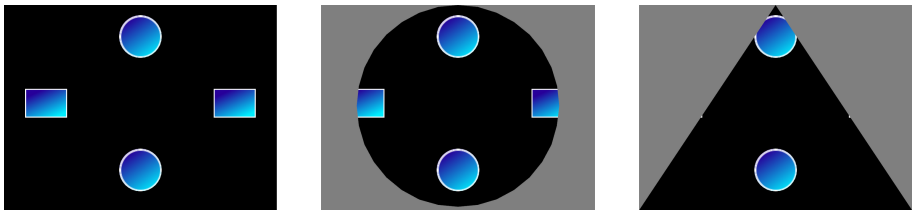
3.2 Experiment

Several observations were made on the impact of using different display shapes with software designed for a particular visual output shape. Using a structured approach, observations on the issues incurred by different display shapes on display shape dependent software could be made. From these observations the issues likely to affect software could be identified.

3.2.1 Method

A prototype system was developed for this experiment. This system utilised multi-touch software which was dependent on a particular display shape. To the visual output of this display a border would be applied to change the shape of the interface. It was decided that the border should be implemented in software to avoid the cost of a hardware implementation of a different display shape. The software was used and any issues observed would be noted and recorded. The contents of the software were made transformable. This meant that the participants in this experiment could manipulate the content items directly. This was so that attempts could be made to try and correct the observed issues. Observations on these attempts to resolve issues are discussed in Section 4.2.

3.2.2 Implementation

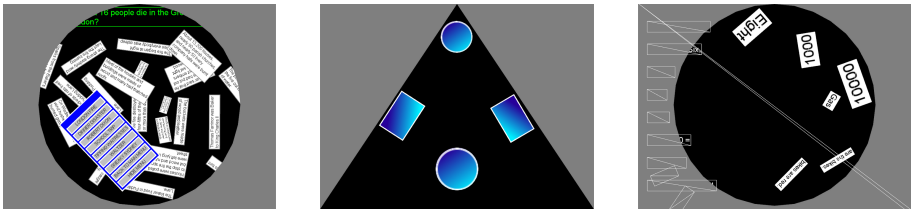


(a) (b) (c)

Figure 3.1: The borders used in the prototype.

The prototype system was implemented into SynergyNet, a MSF built on the SynergySpace MSF [5] discussed in Section 2.3.4. The SynergyNet MSF itself is discussed in more detail in Section 6.2. The implementation of the MSF used in the experiment was dependent on the visual output being rectangular. The prototype functioned by placing borders, rendered in the software, around the MSF's visual output. The software borders were rendered as JME geometry mesh objects which had their vertex, face and edge values defined directly in MSF code. Rendered by this prototype were a rectangular border, a circular border and a triangular border. The SynergyNet MSF supports a number of applications, a selection of which were

chosen for use in this experiment. The applications selected were used with each of the software rendered display borders. The use of these different borders with one of the MSF's applications is shown in Figure 3.1.



(a) (b) (c)

Figure 3.2: A selection of attempts to resolve occlusion in the prototype software.

For each application a series of tasks was outlined which were attempted with the different display borders. From the attempts to complete these tasks observations were made. Issues that made the tasks difficult or impossible to complete were noted. Whenever an issue was observed the circumstances causing it were noted and screen-shots of its consequences were made. A selection of these screen shots is shown in Figure 3.2.

The ability of users to move the contents of the applications was used in attempts to resolve the initial issues observed through manual manipulation of the items. The success, failure or further impact of these attempts were also noted and are discussed in Section 4.2. Whenever an issue was observed attempts were made to recreate it with other applications or display shapes. This was to evaluate whether the issue was reproducible. It also demonstrated whether the issue would affect any multi-touch software used with different display shapes or be specific to certain combinations of applications and display shapes.

3.2.3 Findings

From this experiment several observations on issues from applying a different display shape to multi-touch software were made. The use of a different display border was observed to initially have one single consequence, the occlusion of regions of the visual

output. This can be seen in Figures 3.1 and 3.2 where visual content items can be seen to be occluded by the display borders. This can have an undesirable effect on objects being display in these regions. These occluded visual content items can become unfit for purpose when occluded. This is because the visual information they convey may not be legible to the user when partially or totally occluded. Also users may not be able to directly manipulate these occluded visual content items. The direct manipulation of these content items, such as moving the item or scaling it, may be required for certain tasks in the software.

Occlusion caused by the placement of content items can potentially occur in any software using different display shapes. This is due to the display shape occluding regions of the original software environment. Any content items initially placed in these regions will be occluded, potentially to an extent that leaves them unfit for their desired purpose. It is possible that a software environment may fit within a display shape without any occlusion. However, unless the software environment and the display are the same shape, this implies there are regions of the display which are not occupied by the software environment. This is undesirable as these regions are unused. In these circumstances the software environment can simply be enlarged until it occupies all regions of the display. This results in regions of the software environment being occluded by the display shape. This again leads to the potential issue of visual content items being occluded.

A second issue was also observed. This issue concerned SynergyNet's ability to allow users to flick content items. Usually a user would move a content item through performing a dragging gesture with one finger directly on a visual content item. On releasing their touch the item would continue along the vector that the item was last moved in with decreasing momentum till it stops. If the item encounters a border defined in the software before stopping it bounces away from the edge according to Snell's law [99]. However, SynergyNet's software borders are implemented as a static rectangle. Therefore when a flicked content item reaches the border of a non-rectangular display shape it does not react to it and continues along its current vector. This is problematic in that items are potentially unfit for purpose once they travel beyond the border as they become occluded.

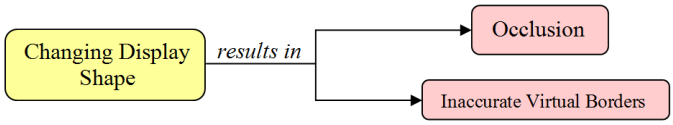


Figure 3.3: Issues resulting from using different display shapes.

Figure 3.3 shows the two observed issues resulting from using the prototype software with different display shapes. From these observations it becomes apparent that issues arise due to a piece of software's assumption of a particular display shape. The experiment showed that the dependence of software on a particular display shape leads to the potential occlusion of content items which is undesirable.

3.3 Chapter Summary

The issue of occlusion caused by the initial placement of content items is potentially present in any software system that may be used with different display shapes. However, occlusion resulting from inaccurate software defined borders is limited to software that utilise a flick gesture which has a similar implementation to SynergyNet's flick gesture. This thesis focuses on resolving the issue of occlusion caused by the initial placement of visual content items. This is because occlusion caused by the initial placement of content items could potentially affect a wider range of software than occlusion resulting from an inaccurately defined software border.

Chapter 4

Evaluation Framework

4.1 Chapter Introduction

To allow multi-touch software intended for horizontal displays to utilise different display shapes the resolution of the occlusion caused by the initial positioning of content items is required. To assess the suitability of potential solutions to the problem of occlusion an evaluation framework is required. In this chapter observations on the experiment detailed in Section 3.2.1 regarding the attempts to resolve this occlusion through direct manipulations are detailed. These observations are used to identify the likely impact of potential solutions on visual content items. Also discussed are the requirements of any solution required to resolve the issue of occlusion caused by the position of content items when different display shapes are used. Using the identified possible impact and requirements of potential solutions an evaluation framework is outlined comprising of a collection of criteria. This framework can be used to identify whether a solution is adequate for resolving occlusion caused by content item positioning in multi-touch software.

4.2 Observations on Attempts to Resolve Occlusion

To identify the impact of potential solutions, the prototype system discussed in Section 3.2 was used. From the previous observations on this prototype, the circumstances which resulted in the occurrence of occlusion caused by the placement of content items were known. These circumstances were specific combinations of a display shape and a SynergyNet Application. When the occlusion of a content item on its initial placement was encountered, attempts were made to resolve the occlusion. This was done through direct manipulation of the content items.

The participants in the experiment could directly transform and deform the content items through gestures made on the multi-touch interface. When an item was completely occluded the JME [55] environment, used in the SynergyNet MSF, was switched to a different render-state. This render-state showed the visual objects' bounding boxes which allowed the occluded object to be seen and manipulated. The use of this bounding box render-state can be seen in Figure 3.2c. When occlusion from the display border was removed from a content item, notes on the manipulations performed were made. These notes were used to inform the design of the solutions detailed in Section 5.2.

When a content item was manipulated in such a way that resulted in the occlusion from the display border being removed observations on the impact of the manipulations could be made. Potential solutions are likely to utilise the transformations and deformations used in the participants' manipulations. Therefore the solutions would have the same impact as the manipulations performed in this experiment. The impact of potential solutions is important to consider. This is because the impact may result in content items becoming unfit for purpose. Each impact of the transformations and deformations which resolved the initial occlusion of the content items were noted. With this information criteria can be developed to assess whether solutions correctly manage their impact.

From the observations, each impact of the attempts to resolve the initial occlusion was noted. These manipulations were observed to have resulted in content items being occluded by other content items, being rotated, being excessively scaled and

losing their layout. Though technically a transformation, scaling is considered as a deformation in the context of this research due to its similarity to other deformations. Each of these outcomes from the attempts to reduce occlusion can result in content items being unfit for purpose. These outcomes therefore constitute the potential impacts of solutions to the initial occlusion of content items. Each issue emanating from the impact of potential solutions is clearly stated and discussed in more detail in Section 4.3.

4.3 The Potential Impact of Solutions

Using the observations on the impact of attempts to resolve content item occlusion detailed in Section 4.2 the major issues which solutions will need to manage can be clearly stated. There are many possible solutions that could be applied to resolve occlusion but all will likely involve transforming the content items in some way. The transformation of the content items can potentially lead to several issues. The first of which is again the occlusion of visual content items. Some solutions to the original occlusion could exacerbate matters by transforming content items to positions where they are less visible. This solution inflicted occlusion is caused by items being moved to positions where they are overlapped by other content items.

Impact A: As a consequence of transforming a visual content item so that it is no longer occluded by display borders, the item may become occluded by other content items.

As mentioned previously, methods of combating the initial occlusion may involve the transformation of content items. One of the transformations used as part of these methods could be rotation. On some displays content may be preferably viewed from specific perspectives around the display. For example vertical displays will be expected to be viewed from one perspective. With the wrong orientation visual information displayed by content items, such as text, can become difficult for users at certain perspectives to comprehend. For example if a content item display text is rotated one hundred and eighty degrees from its preferred orientation on a vertical display

it would appear upside down to the user and would be difficult to read. The likely perspectives at which users view the display could differ between different displays. An issue arises if content, which can only be viewed correctly from specific perspectives, can be potentially rotated away from its intended orientation.

Impact B: As a consequence of transforming a visual content item so that it is no longer unacceptably occluded by a different display shape the item may become rotated to an undesirable angle.

The transformation of content items to combat occlusion can result in other issues. This issue originates from potential changes to the positioning of content items. Content items are often placed in a specific layout such as in grids or in circles. Transformation of the visual content items can have an affect on their layout. A non-constant transformation applied to the content items (meaning a transformation which may differ between visual content items) may alter this layout. For some systems the original layout, or some variation of it, may be required. Therefore the relationship between the positions of content items is another issue to be considered in the design of solutions.

Impact C: As a consequence of transforming a visual content item so that it is no longer unacceptably occluded by a different display shape the relationship between the locations of the items may be lost or modified to an unacceptable extent.

Another relationship involving the content item positions which some systems may require to be maintained is between the locations of content items and features of the display shape the software was originally designed for. An example of this is common in many Graphical User Interfaces (GUIs) designed for rectangular displays where a visual content item may be designed to fit into one of the display's ninety degree corners. In a display shape with no ninety degree corners these content items would no longer be able to maintain this relationship. This may be an issue for some software, but for multi-touch software intended for horizontal interfaces it is not a cause for concern. Due to multi-touch software often being designed to be orientation-independent, as

discussed in Section 3.3, there are unlikely to be any content items positioned in relation to features of the display shape. Also the software cannot be expected to align content items appropriately with potentially non-existent features of the display shape. Therefore no specific issues relating to this particular impact of potential solutions are outlined regarding this loss of relationship between content item positions and features of the display shape.

Some methods of resolving the initial occlusion may result in content items becoming deformed in some way. Deformation is where the shape and size of the content item is altered in a non-uniform manner. This includes actions such as stretching or squashing where the content item's size is changed by different values along different axes. Some deformations may be acceptable when the magnitude of their influence is minimal but they could become problematic if performed to extremes. For example if a content item containing text is squashed to an extreme, the text may become unreadable to users. Other types of deformation can make content items unfit for their intended purposes even when they are performed to small magnitudes. The deformation of visual content items is therefore an issue to be considered in the design of solutions.

Impact D: As a consequence of modifying a visual content item so that it is no longer unacceptably occluded by a different display shape the item may be deformed in an undesirable manner or to an undesirable extent.

Solutions must be capable of managing their potential impact appropriately. Different solutions may have a different impact. Therefore an evaluation framework is required to assess whether a solution's impact is acceptable for a multi-touch software system intended for horizontal interfaces. To assess whether a solution correctly manages its impact the evaluation framework instated needs to make use of the potential issues identified. However, when selecting a solution not only does its impact need to be evaluated, but also its requirements.

4.4 Requirements of Potential Solutions

Solutions to the issue of occlusion caused by the initial placement of content items are all likely to share the same set of issues potentially resulting from their impact. These solutions also have requirements, at least one of which is common to all. Requirements which are specific to a small number of solutions can be satisfied as part of the solution design. However, a requirement for all solutions would benefit from being treated separate to the solutions. This way several methods of fulfilling the requirement can be produced. This is beneficial as the different methods could take advantage of certain features of a system. If a feature of a system that some of these methods make use of is not present then an alternative method can be found. Also this approach can increase the speed of the method of designing new solutions by providing a number of already available methods of fulfilling solution requirements to developers.

The one requirement common to all potential solutions, identified from observations on the use and design of the prototype, is the need to be informed about the display shape. A solution will not be able to resolve occlusion without knowing the display shape that it must fit the visual content items into. As the visual output of a system could take the form of many different shapes hard coding the display shape information into the software is not always an admissible option. The shape of the display acts as a variable that the software must dynamically adapt to. The input of this variable is therefore a requirement of any software needing to dynamically adapt to different display shapes.

Requirement A: The software must be informed of the display shape being used as the system's visual output.

Therefore any potential solution to the occlusion resulting from the initial placement of content items requires to be informed of the display shape and must manage its potential impact on content items. This was the only requirement common to all potential solutions observed, From the issues resulting from the impact of resolving occlusion and the requirement of informing software of the display shape a framework for evaluating potential solutions can be created.

4.5 Defining the Evaluation Framework

From the impact and requirements of resolving occlusion defined in Section 4.3 and Section 4.4 respectively, a framework can be derived to judge the adequacy of potential solutions. For these issues and requirements criteria can be used to judge the suitability of possible solutions. The evaluation framework produced in this work for multi-touch software intended for horizontal interfaces comprises of a number of criteria. These criteria were engineered to give a binary response on whether a solution is suitable or not. If a solution successfully fulfilled all the framework's criteria it would be deemed adequate for use in resolving the occlusion issue. When producing criteria for different systems, developers will need to consider which consequences of potential solutions their software can tolerate. When a developer is aware of the issues which will affect their software negatively they can then outline specific criteria with which to judge the potential solutions. The evaluation criteria can not only be used for evaluating the implementation of solutions but can also be used to inform their design. If a developer is aware of the criteria when designing a solution to occlusion in a particular system they will have a clear idea on the impact their solution is allowed to have.

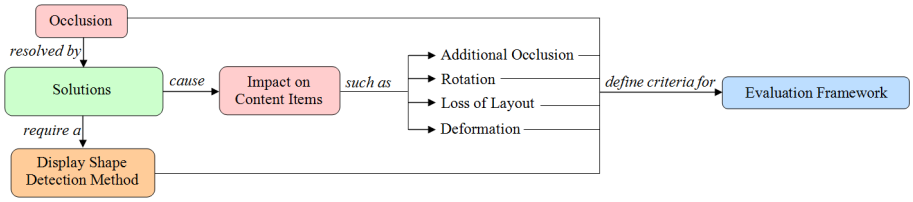


Figure 4.1: Hierarchy of factors which define the evaluation criteria.

As Figure 4.1 shows, the criteria for the framework are derived from the initial issue of occlusion, the potential impact and requirements of attempts to resolve the occlusion. In this section criteria for judging that a solution successfully resolves occlusion in such a way that leaves content items fit for purpose in systems which use horizontal multi-touch interfaces are considered.

4.5.1 Approach to Identifying Evaluation Criteria

The specifics of the criteria relating to solution impact and requirements were derived using the observations discussed in Section 4.2. Additional observations and notes were also made using the prototype system discussed in Section 3.2. Theses additional notes involved precise measurements made through the software and physically by hand on the display with rulers and protractors. When an issue which made a content item unfit for purpose was present, measurements were made of the relevant factors which made the item unusable.

For example when an item was occluded, the amount of occlusion taking place was noted. Attempts to resolve the issue through manual manipulation of the content items were made. When the content items were deemed fit for purpose the relevant details were then recorded again. The typical values at which content items would become fit for purpose in each circumstance were recorded. These threshold values and notes were compared between each circumstance the issues occurred in. The most extreme threshold values were used to define criteria. For example the smallest amount of occlusion in any of the circumstances noted to cause a content item to become unfit for purpose, was used as criteria.

The terms ‘transformable’ and ‘non-transformable’are used throughout the remainder of this thesis to describe content items. These terms relate to whether a user can transform an item in a system’s visual output in some way. If an item is non-transformable the user will not be able to transform or deform the item directly through multi-touch gestures after the system has finished initialising and placing the item. The majority of issues resulting from the impact of occlusion solutions were observed to arise when non-transformable content items were present in software. If a transformable content item is set up in such a way that makes it unfit for its intended purpose in many situations the user could manipulate the object so that it can function correctly again. However if the content item which was rendered unfit for purpose was non-transformable then it would remain unusable. For example when a content item containing text is occluded to an extent that the text cannot be read, a transformable implementation of the object would allow the user to move the item away from the

occluding object thus making the text readable. But if the item was non-transformable then the user would not be able to move it and the item would remain unreadable. This difference between transformable and non-transformable content items can be used to make criteria more specific. One of the influencing factors on the criteria for managing the impact of solutions is whether the software used contains transformable content items, non-transformable content items or a combination of both.

4.5.2 Evaluation Criteria

The criteria which can be used to judge a solution's ability to resolve initial occlusion, detect changes in the display shape and manage its impact on content items are presented as a list. Each criterion consists of one or more variables which can be measured in some way. For these measurements parameters are defined. The criteria stipulate what values for these measurements are unacceptable. Theses values are derived from the observed measurements at which content items started to become unfit for purpose in the experiment. The criteria are numbered, in no particular order, for later reference.

The first criterion relates to the initial occlusion of content items. If at least half a content item is visible then there should be enough of the item on display for it to be manipulated by the user to a different position. Most multi-touch software places limits on the scaling of transformable content items. This is so that the content items cannot be made so small that the user is unable to manipulate them with two touch inputs. This is because in most systems the manipulations for scaling content items require gestures that use two touch inputs. Therefore when an item is scaled to its smallest possible and half of it is occluded, enough of the item should be visible to allow at least one user touch to manipulate it. The user should be able to move the content item away from the occluding object and perform a gesture to scale the item up when it is fully visible. This criterion derives from the initial issue of occlusion which occurs when using software with a different display shape.

Criterion 1: The maximum potential amount a visual content item can be occluded by the display border.

a) If transformable content items are occluded by the display border by more than 50% then the solution is unsuitable.

b) If non-transformable content items are occluded by the display border by more than 0% then the solution is unsuitable.

Criterion 2 relates to the potential impact of solutions and applies to visual items being occluded by non-transformable content items. If a non-transformable content item is occluded by another non-transformable content item neither of the items can be moved so the occluded item would remain covered. This would mean that any visual information it may be displaying could be unusable. Occlusion of transformable content items by non-transformable content items is acceptable as long as enough of the item being covered is visible to allow for it to be moved by the user. This is similar to criterion 1 but relates to occlusion caused by non-transformable content items rather than occlusion caused by the display borders. However occlusion by transformable content items of any amount is acceptable as the item causing the occlusion can be moved by the user. This criterion derives from Impact A described in Section 4.3.

Criterion 2: The maximum potential amount a visual content item can be occluded by non-transformable content items.

a) If transformable content items are occluded by non-transformable content items by more than 50% then the solution is unsuitable.

b) If non-transformable content items are occluded by other non-transformable content items by more than 0% then the solution is unsuitable.

As tabletop interfaces are horizontal there is no single vantage point that users will be expected to view the display from. For this reason any rotation of the content items is acceptable. To make the most of this display feature content items should ideally be rotated in different directions. For transformable content items this is possible so no action should be taken to rotate them to the same angle if their orientations differ. Non-transformable content items though may be expected to have a relationship between their rotations. For example one item may be expected to have the same

orientation as another item. For this reason all non-transformable content items should undergo the same amount of rotation to preserve the possible relationships between their orientations. This criterion derives from Impact B described in Section 4.3.

Criterion 3: The difference between content items' rotations from their original orientations.

a) If the difference between any two content items' rotations from their original orientations is not 0% then the solution is unsuitable.

There are many possible content item layouts that developers may employ in their software. Due to many content items being transformable in multi-touch software though these layouts are not always required to be preserved. However the layout of some non-transformable content items may be required to be maintained in some form. Therefore a solution should be capable of preserving the layout of non-transformable content items in some form. Transformation and deformation of the layout is acceptable when applied in such a way that makes the relation between the content items' positions clear to the user.

Transformation and deformation of the layout can allow for more of the display to be used. Due to changes in the display shape and size regions of the display area may be unused by the initial layout of content items. By transforming the layout content items can be moved into these areas without losing the formation of the items' positions. For example a grid layout of content items on a new display shape may only use a particular region of the display. The content items could all be moved the same amount so that the grid of items is now in the centre of the display. The distance between the items could then be increased. Now more of the display is used up but a grid layout is still present. The layout of transformable items will not need to be preserved as users can easily disrupt the configuration by moving the items so there is no need for the software to enforce the preservation of their layout. This criterion derives from Impact C described in Section 4.3.

Criterion 4: The integrity and scale of the relationship between content item positions.

a) If the layout of non-transformable content items can be deformed or transformed in any way that does not maintain the relationship between the items' positions in any form then the solution is unsuitable.

b) If the layout of non-transformable content items does not utilise as much space as possible then the solution is unsuitable.

As multi-touch systems can display a wide range of content in their displays, including text, many deformations are not acceptable. This is because even to a small magnitude, deformations may make text unreadable. The only deformation that can be accepted when performed to relatively large magnitudes is scaling which maintains the aspect ratio of the content item. Only allowing scaling ensures that the deformation of a content item will not result in any visual content the item may display becoming unintelligible to the user. For the scaling deformation, more precise criteria can be outlined regarding the magnitudes which are acceptable. For transformable content items any size is acceptable as long as the content item can still be scaled by the user through multi-touch gestures. This means that if the item is of a size that makes any visual information it contains unfit for its intended purpose the user can scale the item to an acceptable size. As this is not possible for non-transformable content items it should be ensured that these items are scaled so that any visual information they contain can be understood by the user at all times. This criterion derives from Impact D described in Section 4.3.

Criterion 5: The possible deformations that can be performed on a content item and their extents.

a) If any deformations to the content items beyond scaling is possible then the solution is unsuitable.

b) If a non-transformable content item can be scaled to an extent that any visual information it contains becomes incommunicable to users then the solution is unsuitable.

c) If a transformable content items can be scaled to an extent that users can no longer manipulate it with two finger gestures then the solution is unsuitable.

For a solution's requirement of detecting a display's shape it is important that accurate information concerning the display's geometry can be attained by the software. Therefore the geometrical information attained is required to be accurate. This allows the software representation of the display shape to correctly align with the real world display borders. With multi-touch interfaces there is an additional requirement of not just detecting the visual output area but also the acceptable input regions of the interface. Some hardware implementations of different display shapes, such as covering regions of the original display, may result in the system being capable of detecting touches outside the display shape. Therefore it is important for software utilising different display shapes not to respond to any touch inputs detected outside the display area. This criterion derives from Requirement A described in Section 4.4.

Criterion 6: Accuracy of geometry information attained by the software.

- **a)** If the software generated border does not align with the hardware border then the solution is not suitable.
- **b)** If the software reacts to touch based interaction outside the display shape then the solution is unsuitable.

With these criteria defined any proposed solutions can be assessed. This assessment will identify whether the solution resolves the initial occlusion in such a way that leaves content items fit for purpose.

4.6 Chapter Summary

In this chapter the impact of solutions attempting to resolve the occlusion caused by the initial placement of content items in multi-touch software used with different shaped displays was discussed. The issues that result in content items becoming unfit for purpose which may arise from the solutions were identified. These issues related to the overlapping, rotation, deformation and layout of content items. Also identified was the requirement for all solutions to be informed of changes to the display shape. From the original issue of occlusion, the issues potentially resulting from a solution's impact

and the requirement for display shape information a set of criteria was developed. With these criteria now defined, potential solutions allowing for multi-touch software intended for horizontal interfaces to utilise different display shapes could be considered for use. It is important to note that though a potential solution may be found to be unsuitable by these criteria it may be suitable to other systems with different criteria. These criteria can not only be used for the evaluation of solutions to the issue of initial occlusion for multi-touch software but can also be used to inform the design of these solutions.

Chapter 5

Solutions

5.1 Chapter Introduction

This Chapter outlines several potential solutions to the issue of occlusion caused by the placement of content items in software used with different display shapes. These solutions must correctly manage their impact to ensure all visual content items affected remain fit for purpose. There is the possibility for several solutions to be used together to ensure that occlusion is successfully resolved and that the solutions' impact is adequately managed. Solutions also require a method of being informed of the current display shape a system is using. For this reason each solution, or combination of solutions, will require a Display Shape Detection Method (DSDM).

From the potential solutions put forward in this Chapter, several will be selected for implementation into Multi-touch Software. These solutions will be eventually assessed using the criteria outlined in Section 4.5.2. The criteria also can be used to compare these potential solutions with each other. This will allow the most suitable potential solutions for the scenario of systems with multi-touch table interfaces to be identified.

In addition to resolving the initial occlusion, potential solutions put forward could find ways of adapting software to make the use of certain features of the display shape. Maximising the use of the display shape is not a requirement of the solutions as it is not always possible. However, when a solution can make use of the features of

a display it becomes the preferred option to alternatives that do not capitalise on the display's features. For example, one solution may rearrange content in such a way that it successfully may result in content items being fit for purpose but as a result of its use the solution will cause all the system's visual content items to occupy a small region of the display. The solution is adequate but could be improved by finding ways it could make full use of the display shape. If an alternative solution which resolves the initial occlusion with the same impact and makes full use of the display is available, it becomes the desirable option. Therefore, in addition to how solutions resolve occlusion, their capitalisation on features of the display must also be considered.

The solutions presented in this section are a selection of potential methods which can be used to resolve initial occlusion. These solutions also provide methods of capitalising on display shapes when possible. The solutions listed in this chapter could also be considered for other systems with different criteria.

This chapter discusses the potential solutions to the occlusion of content items caused by their initial placement. The combination of these solutions is then discussed. DSDMs are then detailed in the following section.

5.2 Resolving Occlusion

In this section potential solutions to the initial occlusion of content items, caused by their original positioning when used with different display shapes, are outlined. Three potential solutions are discussed here. These are the Virtual Rectangle Environment (VRE), Pull Content Item Positions (PCIP) and Warp Output (WO) Solutions as shown in figure 5.1.

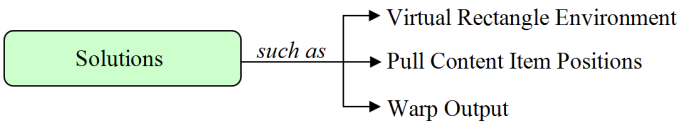


Figure 5.1: Hierarchy of occlusion solutions.

5.2.1 Virtual Rectangle Environment Solution

The VRE solution is built on the concept of taking the original visual output of some software and transforming it to fit within the borders of a display shape. As established in Section 2.4.4 the original visual output of most current software was likely to be rectangular. Therefore this solution is built for adapting the contents of a rectangular software environment to different display shapes. A rectangle of the same dimensions of the original rectangular environment is created. By scaling, rotating and translating, the representation of the software's original rectangular environment can be positioned so that it fits within the display shape being used. The transformations which are applied to the representation of the rectangular environment to make it fit within the border are also applied to visual content items of the software. This ensures that the visual content items have the same positioning and scaling in relation to the representation of the software environment as they previously had relative to the original software environment.

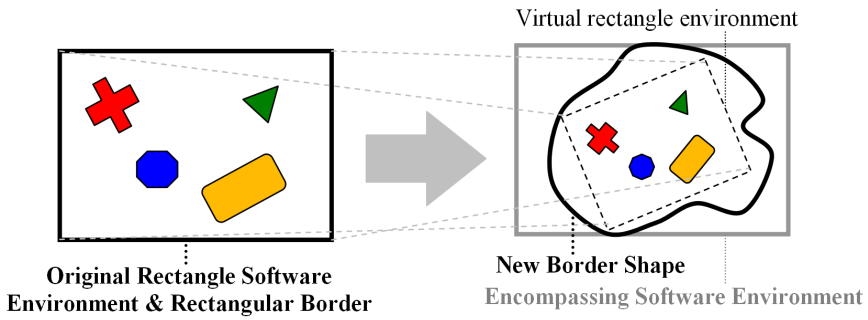


Figure 5.2: The VRE solution.

The transformed representation of the software environment is not the software environment itself, hence it is referred to as a VRE. This ensures that the contents will incur no occlusion from the borders of the new display shape as the content items will be within the VRE which fits within the display. The true software environment still encompasses the display shape so that the entire display area can still be managed by the software system. As Figure 5.2 shows the display shape sits within the true software environment and the original software content is transformed with the VRE

to fit within the new display shape.

This solution manages the initial placement of the visual content items. With the true software environment encompassing the display, users could interact with areas of the display outside of the VRE. This means that if a content item is transformable the user could move content into areas outside of the virtual rectangle. Methods can easily be put in place to ensure users do not move content items beyond the display borders. This means that though content is placed by the system in the VRE which only inhabits a single region of the display to avoid occlusion, content items can be moved to and manipulated in all areas of the display.

This solution requires the VRE to be placed entirely inside the display shape. The VRE cannot overlap the display borders but can touch the edges of the display shape. A larger VRE will utilise more of the display. Therefore finding the largest rectangle that can be placed within the display is desirable. The concept of finding the largest rectangle inside a shape is a known as the 'largest empty rectangle problem' or the 'maximal rectangle problem' and has several existing solutions.

Vandevoorde [100] presents a straight forward method similar to a brute force approach. The method processes the encompassing shape by using the pixel representation of the shape and involves travelling along each pixel row until an edge is found. The pixel where the edge is found is treated as a potential corner of a rectangle. All potential rectangles that could use this pixel as a corner which has one or more non-adjacent sides to the corner which touch, but do not cross the display shape, are tried. The method then continues through the pixels until another edge is found and the process is repeated if it is not an outside edge. The largest potential rectangles for each edge pixel considered are compared so that the largest potential rectangle is found. Due to this method being a form of brute force approach, the running time could be undesirable for many systems.

Another method of finding the maximal rectangle is presented by Alt [101] where a rectangle in the centre of the shape is expanded in a specific pattern. The method uses collisions between the expanding rectangle and edges of the shape to inform how the rectangle could be expanded further. This method is relatively fast compared to other methods with a running time of O log(n) where n is the number of vertices of the

shape. A drawback to this method is that the solution will not work if there are holes in the shape. Though the presence of holes in the display shape is unlikely there may be situations where it may be desirable. Therefore this solution would only be viable for use when it is known that there will be no holes in the display.

The rectangles returned from any of the current solutions to the largest empty rectangle problem will always be parallel to the environment axis. This is problematic as several potential rectangles positioned at an angle not parallel to the axis may be larger than those returned by any of the existing maximal rectangle finding methods. A potential solution to this is to rotate the shape to various orientations and repeat the algorithm. This however results in multiplying the running time by the number of orientations checked. The size of the change in orientation between each pass of the algorithm when reduced will increase running time but will also improve the chances of finding the true maximal rectangle. When a rectangle is found by a pass of any of these algorithms when the shape is rotated it is important to rotate the resulting rectangle so that it matches the shape's usual orientation.

Another problem with the rectangles which result from these methods is that their aspect ratios will vary depending on the shape. This is not desirable for when a rectangle with a specific aspect ratio is desired. However rectangles with the correct aspect ratios can be found in linear time within the resulting rectangles from these methods. As it is unlikely that a rectangle with the correct aspect ratio will occupy the entire rectangle returned from one of these methods further space is left unused. An alternative to using these methods is for a manufacturer of the display to identify the maximal rectangle. This could be done by initially employing a simplified version of the maximal rectangle finding methods to produce an axis-parallel rectangle. Human intuition could then be employed to identify where rotating and translating the rectangle may allow for it to be made larger. The issue then becomes how to inform the system of the decided upon rectangle position and rotation. Additional information from a DSDM can provide this information as discussed in Section 6.3.

The VRE solution ensures no additional occlusion of content items by the display border. This is because content is kept within the virtual rectangle which is inside the display. This provides the means to resolve occlusion, both from the display borders

and from other content items. To make the virtual rectangle fit the display the rectangle and its contents may need to be rotated. As the content is rotated together the solution can be seen to manage the impact of solution rotation.

The virtual environment will need to be scaled down to fit the display. The method of scaling can be implemented in a number of ways. If a system has a minimum acceptable scale value for its contents the scaling of the VRE can be used to check that the content items are not scaled beyond these limits. The VRE solution does not perform any deformations other than scaling and therefore gives the solution scope to resolve the issue of deformation. As the content of the software environment all undergo the same transformations the layout of the content items is preserved. The layout may be rotated, scaled and transformed but it is not deformed in any other manner and therefore would still be clear to the user. Developers can implement limits to how much the virtual environment is transformed which directly affects the possible magnitudes of transformation to the layout. As the solution preserves the layout and allows the developer to dictate the extents to which the layout can be transformed it is clear that this solution has the ability to reserve the relationship between content item positions.

This solution currently requires the software environment to be rectangular. This is acceptable for current software systems as most exclusively use a rectangular environment. However, if non-rectangular displays become more common place in the future the solution will need to be modified. The virtual environment will need to be capable of being any shape to accommodate software originally intended for non-rectangular display shapes. The solution could simply be adapted to do this with a polygon (meaning a shape of any set of geometric values). However the process of fitting the environment into the display shape would become more complex. There are many solutions to the challenge of finding the largest particular shape such as circles and triangles inside a polygon [102, 103]. But methods for finding the largest polygon in another polygon are few in comparison and require much more time for processing.

The only current guaranteed method is an adaptation of Vandevoorde's algorithm [100]. This adapted algorithm entails using only the shape of the former software environment in place of a rectangle. This is equivalent to finding the solution through

brute force and therefore would render the solution non-polynomial which may not be desirable for systems with resource constraints. Again the solution of a position and rotation for the virtual environment defined by a user, developer or manufacturer could be used. The VRE solution does provide the scope to resolve the initial occlusion and correctly manage its impact in such a way to leave content items fit for purpose. However, this solution does not make full use of the display. Content items are placed in a single area of the display meaning that parts of the display could be initially unused. Users could move contents to areas outside the virtual rectangle but if all the content items are non-transformable this renders areas of the display completely unused. Combining this solution with others could improve the usage of the display as discussed in Section 5.3.

5.2.2 Pull Content Item Positions Solution

The PCIP solution uses the differences between the shapes of a software environment and the display it is output through to move content items so that they are placed without being occluded. For each content item their x and y axes parallel to the edges of the software environment are used. In both directions from the centre of the content item each axis intersects the display shape border at least once and the edge of the software environment. Vectors are created along the axes starting at where an axis intersects the edge of the software environment and ending at where it intersects with the display border. For both of the axes of the content item, two of these vectors are created. These initial vectors are shown on the left hand side of Figure 5.3.

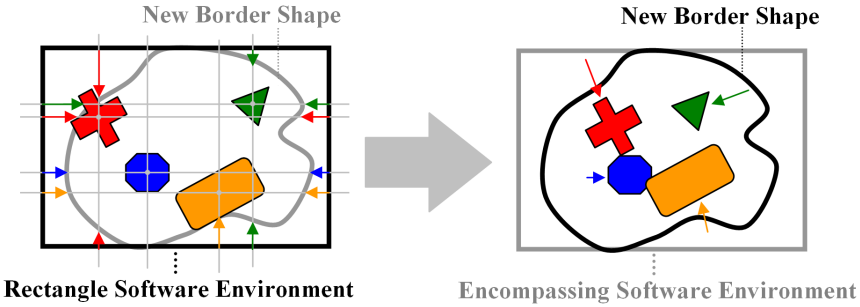


Figure 5.3: The PCIP solution.

The components from two initial vectors along each axis are added together to create a new vector. Addition creates a single vector for each axis of the content item determining the difference between the initial vectors. Component addition then occurs between the resulting axes vectors. The resulting vector can be used to translate the content item to a new position inside the display shape as shown on the right hand side of Figure 5.3. For some display shapes the lines travelling out from the content item may intersect the border edges several times before reaching the software environment edge. Where this is true only the intersection closest to the content item is relevant to the solution. This is because the edge closest to the content item will have the most influence on the item.

The process of calculating the vectors for this method will need to occur separately for each content item. Therefore the amount of work the software does in performing this method is multiplied by the number of content items in the visual software environment. However, since only the addition of the vector components is needed the calculations are simple and require relatively little time and memory. The effect of this solution is that content items are moved away from large gaps between the edges of the original display shape and the new display shape edge. As a result of this method the layout of content items is squashed and deformed to fit the new display shape.

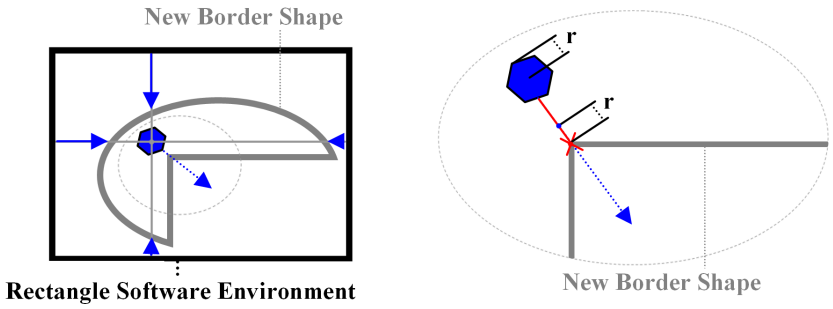


Figure 5.4: A potential problem in the PCIP solution.

This method however does not always guarantee that a content item will be translated correctly into a position in a display with no occlusion. There may be situations where a content item is outside of the display area and the resulting translation vector from the method may not be large enough to move it entirely into

the display. One such situation is shown on the left hand side of Figure 5.4. A method to counter this problem is to detect when the item may be in a position where occlusion occurs.

First the method must check that the proposed content item position is not outside the display. If so there must be a display edge between the content item's proposed position and its current position. The content item should be translated to the position on the display edge between its current and proposed positions. If the item is left here it will incur some occlusion as the point it is translated around will be on the display edge. This may be acceptable for transformable content items which allow an amount of occlusion. However it is not acceptable for non-transformable content items where no occlusion is permitted. in these circumstances the non-transformable content item must be translated again to the nearest position with no occlusion. This can be done by calculating the largest distance present in the content item between the point the item is translated around to any other vertex of the item, referred to as the radius distances. By moving the content item towards its original position by this distance from the display edge the content item will no longer be occluded by this border edge. This is shown in the right hand side of Figure 5.4 where **r** represents the radius distance.

If no occlusion is desirable then checks need to be carried out whenever a content item is placed within the display. The radius can be used to create a bounding circle around the visual content item. If the display borders intersect the bounding circle then the content item can be moved to be a radius distance perpendicular from the edge to remove the occlusion. Occlusion caused both by overlapping non-transformable content items and the display borders is potentially reduced to an acceptable level by this solution when used with this enhancement.

With the visual content items being moved towards the centre of the display there is a possibility that the content items may overlap. Measures can be taken to ensure that non-transformable items are not placed too close together so that they do not overlap and cause occlusion. If occlusion is caused from non-transformable content items overlapping the items can be scaled down to reduce overlapping. When scaling is incorporated as part of this solution the implementation can be designed to adhere to limits on the magnitude of the deformations. As the items are only translated, and

scaled in specific instances, other translations such as rotation are not possible.

Due to the nature of this solution the layout of content items will be deformed. The deformation which can be seen in the change between the previous software environment shape and the current display shape will also apply to the layout. Therefore the layout can be seen to be persevered in that it becomes a deformation of the original layout. The translation of content items is not constrained to one area and as a result all regions of the display could potentially be made use of. This demonstrates that this solution does capitalise on the display shape.

This solution can be implemented in a number of ways. The differences between these implementations may not change how the solution resolves the issues but do affect how the software makes use of the display. The solution as it stands does not take into account the distance of the content item from the edges of the display. This means that a content item will both be influenced equally by both the initial translation vectors on an axis. A vector on a distant side of the display shape will affect the positioning of the content item as much as the vector on the closest edge.

There are several techniques which can be used to ensure that the distance of the content item from an edge is taken into consideration when computing the final translation vector. One such method is to calculate the distances between the content item and the display edges along each axis. The distances between the edges of the border and the software environment are calculated as normal. Then by using the formula stated in Equation 5.1 a scale can be calculated for each initial translation vector. In this equation 5.1 *itemDistance* is the distance between the content item and the display edge. The *edgeDistance* term represents the distance between the edges of the display and software environment.

$$scale = \frac{itemDistance}{itemDistance + edgeDistance} \tag{5.1}$$

The vector components are multiplied by the resulting scale to represent the influence that the distance from the initial edge has on the content item's translation. The addition of the vectors along each axis and of the resulting two vectors is executed as normal. With this technique if an object lies on an edge of the display, the adjacent

vector will have a 100% influence on the vector whereas the vector on the distant side will have a smaller influence. An alternative method of representing how the proximity of display edges to content items influences the resulting translation is to only use the vector of the closest edge. By ignoring the influence of the more distant vector along an axis the translation is more likely to be towards the centre of the display shape. This has its advantages for situations where there may be larger vectors on remote sides of the display shape. In these circumstances developers may wish the remote vectors to have no influence on the translation of the content items.

By ignoring the influence of distant edges the relationship between a content item position and a feature of the display shape can be preserved with minimal alterations. For example an item which is positioned in a corner will retain its positional relationship with this feature of the display. This is because without the influence of distant edges the item will not be pulled away from the corner. This is an example of how this solution can be adapted to preserve the relationship between content items and features of the display shape. This is beneficial for systems which require this in their criteria, though it is not relevant to the multi-touch software this thesis is concerned with. The method of changing the influence of an initial vector based on their proximity to the content items can be incorporated into the technique of ignoring the distant edge. This allows for the influence of the closest edge to be determined by the closest edges' proximity to the content item. This solution provides a method which makes use of the full display area and has been shown to have the potential to leave content items fit for their intended purposes.

5.2.3 Warp Output Solution

The WO solution is designed around the concept of deforming the visual output of a system to fit the shape of the display. This method is only suitable for systems in which the deformation of content materials is acceptable. This solution takes the differences between the shape of the software environment and the new display shape and applies the same changes to the visual output. Content is stretched and squashed in a non-uniform manner to fit to the display shape as shown in Figure 5.5. The

content can be deformed through either software or hardware approaches. A software implementation could be achieved with many methods. One such method involves creating a single image of the software's visual output and performing deformations on the resulting image to match its shape to the display shape. An example of a hardware implementation of this solution is present for projector based outputs. A projector lens can be lens warped to correspond to the display shape to deform its projected image accordingly.

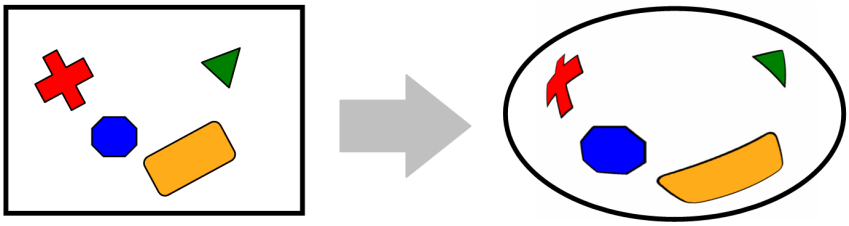


Figure 5.5: The WO solution.

This solution ensures that no content is occluded by the display borders and no additional occlusion will be caused by content items overlapping. This demonstrates that this solution is capable of managing occlusion both from the display border and non-transformable content items. Content may or may not be rotated depending on the solution's implementation. This solution is not suitable for systems where the content items cannot be deformed in anyway. In some systems where all types of deformation are acceptable there may be limits on the extent to which the deformations are performed. For these systems a software implementation of the solution can incorporate methods of managing the extent of the deformations.

As the visual output in its entirety is deformed the layout of the content items is deformed. This means that this solution is adequate for systems where the deformation of the content layout is acceptable. It is worth noting that if a feature of the original software environment is present in the new display shape then items in close proximity may maintain their positional relationship with this feature. This is beneficial for systems whose criteria require this positional relationship to be maintained. For example a content item placed along an edge of the original software environment

will remain near the corresponding edge if it is still present in the new display shape.

The WO solution can resolve all of the issues resulting from resolving the initial occlusion for some systems where the deformation of content items is acceptable. Multi-touch systems often contain visual information such as text which when deformed becomes incommunicable to the user making the item unfit for purpose. This solution is therefore not suitable for use with the majority of multi-touch software. The solution does have benefits despite the limited number of systems it may be suited for such as its capitalisation of the display shape by utilising the entire display area.

5.3 Combining Solutions

Some potential solutions on their own may not resolve the initial occlusion of content items in such a way that leaves content items fit for purpose. Combing these solutions could succeed in doing this. The combination of solutions is not only useful for resolving occlusion and managing their impact, but can also improve the utilisation of features of the display shape. In this section a potentially beneficial combination of solutions of the VRE and PCIP solutions is discussed.

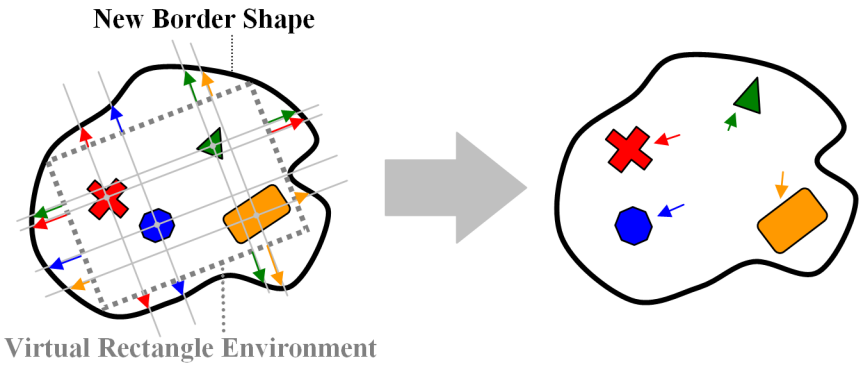


Figure 5.6: A combination of the VRE and PCIP solutions.

The VRE solution resolves the initial occlusion in a manner that leaves content items fit for purpose. However the solution does not make use of all regions of the display. Using a form of the PCIP solution, the content in the virtual rectangle could be translated into the unused areas of the display. In this implementation the initial vectors

of the PCIP solution do not measure the distance between the software environment edge and the display borders. Instead the initial vectors for this implementation will measure from the virtual rectangle edge to the display shape edge. The effect of this solution is that content items are moved towards large gaps between the edges of the new display shape and the VRE. This will have the effect of stretching the content item positions out as shown in Figure 5.6.

Many of the same considerations from the PCIP solution are still applicable. One such consideration is the amount of influence different vectors should have on a translation depending on their distance from the item. This additional translation of content items may deform their layout which will effect how the resulting combined solution manages their impact.

Developers could be permitted to choose which content items can and cannot be moved by the PCIP solution. Those which are not moved by this solution will be preserved in the layout resulting from the VRE solution. This means that developers would have greater control over the solutions' influences over the content items. This combination of solutions potentially resolves the initial occlusion, leaves content items fit for purpose, capitalises on the display shape and allows developers greater control over its impact.

The WO solution is not considered for combination with either of the other proposed solutions. This is due to the fact that it affects the entirety of the display. As a result of this it is likely to influence, and therefore potentially counter-act, the effects of any other solutions.

Whatever the combination of solutions chosen, at least one method for informing the solutions of the display shape used is required. Section 5.4 discusses several potential DSDMs which could be used with either an individual solution or a combination of solutions. Some specific combinations of occlusion solutions and DSDMs can have additional benefits as discussed in Section 6.3.

5.4 Display Shape Detection Methods

To allow for the software to be used with any display shape no assumptions can be made about the shapes a system may potentially use. Therefore, rather than just identifying a display shape, DSDMs are required to produce geometric information from which virtual representations of the display shape can be built. In this section three potential DSDMs are discussed. These are the VS, User Calibration (UC) and Display Border Storage (DBS) DSDMs as shown in figure 5.7.

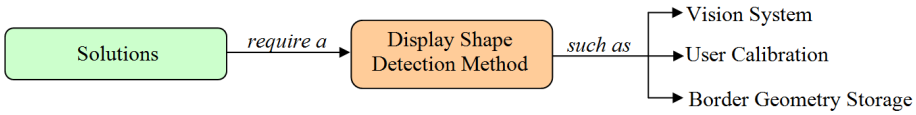


Figure 5.7: Hierarchy of DSDMs.

5.4.1 Vision System DSDM

For systems which utilise a vision based input the display shape could be detected through the system's sight. There are many methods which are used to collect the geometric information of the objects a system can see. However for these methods to be accurate the system will normally require additional information that cannot be provided by a single camera with no additional features. Such additional information includes the systems' camera position relative to the object being viewed or a selection of the viewed object's measurements. It is not always possible to collect this additional information, especially when no measurements of the display being used are known to the system. Features of the system which can be used to attain additional information include specific placement of the camera, additional cameras, projected patterns and the use of visual markers.

Specific placement of the camera and display means that the distance of the camera from the display would be known as well as the orientation of the display in relation to the camera. If this set up is possible then image processing methods, such as Canny's edge detecting method [104], can be used to detect the display shape if its edges are well defined. However this set up is not always possible. Poor lighting could make the

detection of the display edges impossible as could objects obscuring the system's view of the display. Another potential problem with this approach is that the display may be too large to be viewed by a single camera if space requirements do not allow it to be placed far enough away. Therefore for a single camera to detect the display shape the position of the camera relative to the interface must be known. The system must also be able to see the display clearly in its entirety and nothing should obscure the camera's view of the display.

The use of multiple cameras can allow a system to build a three-dimensional view of an object [105, 106] such as a display. For the three-dimensional view to be built the positions of the cameras relative to each other need to be known as well as their orientation. Therefore this method of collecting geometric information on the display is acceptable for systems where the cameras are unlikely to be moved separately. A three-dimensional view allows for the orientations and positions of the object to be known. This allows for the measurements of the display to be calculated without the need for any additional information. The multiple cameras view the display from different perspectives and identify features of the display that two or more cameras can see. These positions are called landmarks and are used to stitch together the view from all the cameras. Knowing the positions of the cameras then allows this view to be translated into a three-dimensional virtual model. Some implementations of this technique will require markers to be added in certain positions on the display to act as landmarks. The use of multiple cameras may not be possible for some systems however due to constraints on resources, software limitations or the placement of the system.

If the system can only use one camera then the use of visual fiducials [20] may provide a method of collecting a display shape's geometric data. By placing several fiducial on the display's borders the additional information on its positioning and orientation relative to the camera can be calculated. With this additional information the system can then identify the edges of the display and correctly interpret their geometric values. However there may not always be adequate positions to place the fiducials on the display border. Also if the lighting is poor the system may still not be able to correctly identify the edges of the display or the fiducials.

Another feature which could be used by a system to identify a display shape is

the use of pattern recognition. Similar to the techniques used in three-dimensional scanning [107] the system can use a pattern shown on a display to identify its geometric values. The pattern used is known to the system; therefore deviations from the pattern can be identified and used to calculate the display shape. One method of employing a pattern would be to project it over the display and its edges. Different types of light can be used for the system as long as the camera is able to see it. Infra-red light (such as that used in DI multi-touch systems) could be used to avoid the possibility of interference from external visible light sources. As the system is set up to provide additional light the likelihood of parts of the display not being visible to the system's camera due to poor lighting is reduced. One potential problem is that the light may not be visible on the surface of the display. For example if the display is glass it may not reflect the type of light used to project the pattern.

However this may not be an issue. This is because the system should be able to see the light reflected from the display's border and calculate the display shape from the geometrics of the void in the centre of the pattern. The use of a patterned light source however may not always be possible. The system may be used in an environment where the lighting may change frequently in such a way that affects the light pattern.

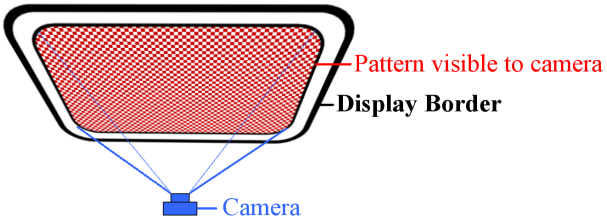


Figure 5.8: The use of a vision system to detect a display shape.

An alternative is the use of a pattern board. The pattern board is a large surface with a pattern displayed on it which is known to the system. The board is placed over the display in such a way that the camera can see the pattern through the display. As parts of the display are occluded by the border the system can only see the pattern through the display area. The system can calculate the geometric information of the patterned area within the borders that the display occupies. The use of a pattern for collecting

geometric information of a display shape is shown in Figure 5.8.

A drawback of using a vision based method of inputting the display shape is that the area between the camera and the display must be clear. This is so that no occlusion of the display border occurs in the system's view. Also without the use of an additional storage method the process of scanning the display would need to be repeated whenever the relevant software is used. With a storage medium the collection of geometric data would only be needed when the display shape changed. This technique could be used to monitor the display shape to ensure that any changes to the display shape during use are noted and adapted to.

5.4.2 User Calibration DSDM

Several systems exist which require users to perform a calibration process before use. Examples of these include vision-based multi-touch systems [23, 52, 51] and interactive white boards which use single touch and stylus inputs [108, 109]. This activity is already required to be performed before use on many multi-touch systems. Therefore modifying this activity so that it can also detect the shape of the display would be beneficial. This would make better use of an already existing system input which is desirable [110]. Methods similar to the existing technique of touching the display at specific points could be developed. If a point is outside of the display a user performs some action to inform the system that the relevant location is outside the display area. Using the knowledge of which locations are inside the display area and which are outside allow for the display shape's geometric information to be calculated. The number of points used in calibration improves the accuracy of this geometric shape. However more points will require more time from the user which is not desirable when the shape of the display needs to be changed quickly with minimal effort. Also this technique is dependent on user accuracy. An inaccurate set up by the user could result in the system calculating an incorrect display shape.

An alternative method for utilising a user's direct touch input for detecting the display shape is for users to provide an outline of the display. This could be done by a user placing their finger on the edge of a display at one position then dragging their

touch around the whole outline of the display. Again this process is time consuming and is also open to user errors. If the interface is large the user could suffer discomfort being required to drag their finger around the entirety of the display. For some systems users could use objects such as styluses to reduce discomfort when dragging a touch input around the display edges. Having to perform this process whenever the software is used could prove frustrating and time consuming. Storing the data calculated for retrieval in the future use of the system would be beneficial.

5.4.3 Display Border Storage DSDM

The DBS DSDM functions by providing a representation of the geometric information concerning the display shape used. The geometric data can be represented by typical modelling data such as the vertex locations, edges, faces and normals of the shape. For this information to be used by software the geometric data should be stored in a format that can be parsed. There are several feasible formats containing geometric information which are intended for parsing such as the scalar vector graphic and the waveform object formats. The benefit of the storage approach is that the user will not need to perform a task which defines the geometric information each time a different display is used. Once geometric information has been created for a particular display then it can be easily stored and recalled when needed. Of course this is not desirable for systems where the display may change frequently [81, 82, 83] as the information stored may be static and the software will not be able to update it.

How the geometric data is collected can be implemented in a number of ways. The two other DSDMs discussed in this section provide methods for collecting geometric data. The geometric data could also be provided by someone involved in the use or production of the system. Relying on a user to input the geometric data directly however causes the quality of the data to be reliant on the user's accuracy and understanding of the data structure used.

There are also requirements for the user's time and effort to be invested in measuring and entering the geometric data. These user dependencies and requirements make this option undesirable [110]. An alternative is for the developer of software

to create the border geometry though they may not know all the shapes that may be used with the system, especially if any display shape could be employed. However the producers of the displays used with a system would know all the relevant geometric information. The display manufacturer is the best suited of those involved in the creation and use of a system to collect accurate geometric information relating to the display. Manufacturers are likely to have the absolute measurements of the displays on hand from their design work. As many manufacturers develop their designs in modelling software the relevant geometric data could be exported from modelling software to a data structure which could be used by other software. The geometric information could then be included with the display in a number of ways such as part of the display driver or as a download from an online resource. This DSDM is very flexible and thus allows for it to be used in conjunction with a number of other DSDMs.

5.5 Chapter Summary

In this chapter several solutions to the initial occlusion of content items were discussed. These solutions could be used in combination to comply with the criteria to resolve occlusion and leave content items fit for purpose. Also discussed were DSDMs so that these solutions would be informed of a display's shape. With the right solutions and DSDMs implemented, software can become display shape independent. This means that any of the visual components of the software can be designed and placed without any need to make considerations for how certain display shapes may affect them. A combination of solutions which successfully fulfils the criteria will allow multi-touch software to function correctly on tabletop displays of any shape. This is because the solutions will apply correct right transformations and deformations to content items so that they are always fit for purpose A good combination of solutions, in addition to resolving the initial occlusion, is also expected to capitalise on the display shape used and reduce user input. In addition to this a combination of solutions should allow the developer control over the solution's influences and make effective use of the system resources. It is therefore important to select the right combination of solutions and DSDMs for implementation into a system.

Chapter 6

Implementation

6.1 Chapter Introduction

Figure 6.1: The multi-touch tables used by Durham's TEL research group [111] built by Ness Furniture [2].

With the potential solutions outlined in Chapter 5 a set of suitable solutions could be implemented into a piece of software. In this chapter the process of implementing a combination of solutions into software intended for multi-touch tabletop interfaces is discussed. The software chosen for implementation was the SynergyNet [5] MSF which provides functionality for numerous applications intended for classroom use.

An example of the type of interfaces that the SynergyNet MSF is intended for use with is shown in Figure 6.1. With the implemented solutions the software would become compatible with multi-touch tables that use different display shapes. This chapter details the selection of adequate solutions and DSDMs for SynergyNet. Also detailed is the process of their implementation into the MSF.

6.2 SynergyNet

SynergyNet is a higher level MSF built on the SynergySpace MSF discussed in Section 2.3.4. SynergySpace itself is built on JME [55] and a number of other third-party libraries which enable it to perform a wide range of functions. These functions include hand writing recognition, support for receiving inputs from several multi-touch protocols including TUIO and the ability to render various media. The significant feature of SynergySpace is its ability to enable multi-touch events such as user gestures to interact with content items rendered by JME in two or three-dimensional environments. SynergySpace is intended for education based activities but due to the wide range of features it supports, many other types of application could be supported by the MSF.

One of the aspects of SynergySpace that differentiates it from other MSFs is its use of networking. A single instance of SynergySpace is able to discover other instances of the MSF running on the same network. Instances of SynergySpace running on the same network can then communicate with each other. Through the use of third-party libraries such as x-stream [112], Java objects can be sent across the network. This allows for a visual content item to be copied and sent across the network to other instances of SynergySpace which can then recreate and use the item.

The SynergyNet MSF is used to provide a simple platform for developers to produce multi-touch applications for SynergySpace. SynergyNet takes the features of SynergySpace and allows developers to utilise them through simplified API calls. This higher-level MSF also provides a simple menu system for navigating between the supported applications. Applications designed for SynergyNet are by default created using an orthogonal JME environment. In this virtual environment all the visual

contents rendered are two-dimensional and placed the same distance from the scene camera. An ordering system is used to manage how the items overlap in relation to the user's view.

Developers can choose to override the default application settings of SynergyNet if needed. For example a developer could use a three-dimensional environment in this application if they desired. SynergyNet also offers a content system which allows developers to easily create content items in application and set their relevant parameters. The types of content items include videos, images and shapes. The relevant parameters which the developers can set for these objects, relate to their transformations and ability to be manipulated by the user.

Using the established multi-touch gesture of flicking [113] SynergyNet can be configured so that a flicked content item can travel beyond the edges of a display to another iteration of SynergyNet running on another system. Using networking and some prior configuration, instances of SynergyNet can be made aware of where other instances are running in relation to their display's position. When an item collides with the edge of a display it usually bounces away from the edge, otherwise it would be lost. However if the item's parameters dictate whether it is permitted to be transferred to other displays. If it is permitted to be transferred when the item collides with an edge the system checks to see if there are any displays showing a SynergyNet environment along its current direction of movement. If so the item is then transferred via the network to the instance of SynergyNet running on the remote display, if not the item bounces as normal. This is one of the many features that may be affected when adapting this framework to utilise different display shapes and therefore must be taken into consideration during implementation.

6.3 Selection of Solutions

The VRE and PCIP solutions were chosen for implementation. This is because of their ability to revolve occlusion, mange their impact and capitalise on the features of the display shape. This combination of solutions also offers the developers greater control over its impact as discussed in Section 5.3. This is useful for SynergyNet as the

developers of applications for the MSF may want control over how solutions affect the visual contents. The alternative to this combination was the WO solution. However, since this solution entails the deformation of content items, it was discarded as this is not acceptable for MSFs such as SynergyNet.

With the solutions for occlusion chosen at least one DSDM is required. Chosen for this implementation was the DBS DSDM. This was chosen due to its lack of reliance on particular features of the system used such as the alternative DSDMs.

The VS DSDM could be implemented into any lower level MSFs designed to utilise a vision-based multi-touch technology. These MSFs would have access to cameras which are set up to view the display that could be used to detect the display shape. However this solution was discarded due to the fact the SynergySpace, and therefore SynergyNet, could be used with any lower level MSF that produces an output with a compatible protocol. This means that the SynergyNet MSF could be used with multi-touch systems that do not use cameras. Also if the lower level MSF is not open source it cannot be modified to detect display shapes even if it is designed for vision-based multi-touch technologies.

The UC DSDM was discarded due to its requirement for prolonged interaction by the user. The use of the DBS DSDM means that geometric data will already be available to the system. Therefore any additional DSDMs would only be required for identifying which set of geometric data to use. It is not necessary for the user to provide a wealth of information, as would be required by the UC DSDM, as the user can directly identify the adequate set of geometric data. The DBS DSDM could be used in conjunction with other DSDMs if they are implemented as part of future work.

As discussed previously in Section 5.2.1 the VRE solution requires a rectangle within the display border to be specified. One method of specifying the rectangular environment is for the geometry of the rectangle to be provided by the manufacturer. The geometry of the virtual rectangle can be implemented as part of the border storage. Both the border and virtual rectangle geometry need to be input into the system. Also both these sets of geometry need to be provided by someone with knowledge of the display dimensions such as the display's manufacturer. This demonstrates how some combinations of occlusion solutions and DSDMs can have additional benefits. The

combination of the VRE and PCIP solutions with the DBS DSDM was therefore chosen for implementation into SynergyNet.

6.4 Implementation into SynergyNet

In Section 6.3 a combination of solutions and a DSDM was identified as being capable of leaving content items fit for purpose in multi-touch software designed for tabletop interfaces. With the solutions and DSDM to be implemented identified, the process of their implementation is documented in the remainder of this chapter. The implementation of the occlusion solutions is discussed in Section 6.4.1. In addition to this the implementation of DSDMs is discussed in Section 6.4.2.

6.4.1 Resolving Occlusion

To resolve the initial issue of occlusion the combination of the VRE and PCIP solutions was chosen for implementation. This combination of solutions allows for the layout of content items to be preserved in some form when needed. The solutions chosen also reduce the of content items occlusion to acceptable levels and avoids any deformation of the items beyond scaling. The solutions are intended to make content items fit for purpose but also offer additional benefits such as full utilisation of the display area. By stretching content items into areas outside the VRE there are no areas which cannot be used in the initial layout of content items. This combination of solutions also offers the potential for developers producing applications for the MSF to have a large amount of control over the influence of the solutions. The combination of these solutions was implemented into SynergyNet's content management system. The solutions were specifically implemented into methods in the content management system which related to the transformation of content items. This means that any object which is produced and managed through the content item system can be influenced by the implemented solutions.

The solutions implemented required two inputs to function correctly. These inputs were the display shape geometry and the virtual rectangle geometry. Both were

provided through the waveform object discussed in Section 6.4.2. The transformations to content items instigated by the solutions were performed in stages. The first was the translation, scaling and rotation of content items using values calculated by the VRE solution. These values are created by collecting the four rectangle vertices from the waveform object.

A virtual rectangle object is built using these vertices and is scaled to fit appropriately within the display shape border object. The virtual rectangle is compared to an untransformed one thousand and twenty four by seven hundred and sixty seven pixel rectangle which has not been rotated, scaled or moved from its position in the centre of the software environment. This allows the rotation, scale and translation of the virtual rectangle to be found. These transformation values could then be applied to the content items to ensure that they fit within the virtual rectangle with no occlusion or loss of layout. This set up means that whenever a content item is initially placed in an application using SynergyNet's content system additional transformations resulting from the implemented solutions will also be applied. The first of these transformation result from the VRE solution.

After applying transformation deriving from the VRE solution to content items the next stage is to apply transformations derived from the PCIP solution. From the point around which content items are transformed, usually the centre of a content item, rays are fired in both directions along the x and y axis. These rays are objects from the JME software framework which can be used to detect the distance from the ray's origin to any objects they collide with. There is an issue with the use of rays however in the SynergyNet MSF. Because the applications in SynergyNet are by default created in an orthogonal environment any objects in the environment have no presence along the z-axis as they are perfectly flat. In addition to this, objects are placed on different planes along the z-axis to manage how they appear to overlap to the user. This means that when a ray is fired along only the x and y axes it cannot detect a collision with any of the content items in the application. In orthogonal environments JME's collision detection can correctly identify when two content items are overlapping or colliding.

However JME's collision detection cannot provide any additional information, such as the location of the collision, about collisions between objects in an orthogonal

environment. As the location of the collisions is required for calculating distances along the rays a method to resolve this issue was devised This method involved adding depth to SynergyNet orthogonal content items when needed. This method would first calculate the vertices on the outer edges of a content item, referring the edges with only one adjacent face of the content item. Using these vertices JME triangle objects which stretch into the z-axis could be created along each outer edge of a shape. For each triangle the ends of an edge are used as two of the triangle vertices. Then the third triangle vertex is defined halfway between the two other triangle vertices in the x and y axis but with a different z. This allows the triangle placed along the outer edge to have some presence in the z-axis.

By performing this for every outer edge of a content item shape a wall of triangles is created along the outside of the item which stretch into the z-axis. This allows the JME ray objects to detect collisions with content items in the orthogonal environment. This method can require additional processing time and memory for the creation of the triangles which could be avoided with a correctly managed three-dimensional environment. With this method incorporated the rays used as part of the PCIP solution implementation can detect distance along each axis of the content item. This means the distances between item and the relevant positions on the edges of the border object can be found. Using these distances the resulting translation vector for the PCIP solution can be calculated and applied to the content items.

The final stage of transforming the content items when placed by SynergyNet's content management system involves reducing any additional occlusion. This is occlusion that may have occurred during the previous stage of transformation. Though the PCIP solution does make use of more of the display shape area it can result in further occlusion with certain display shapes. The cause and resolution of this problem is discussed in Section 5.2.2. Each content item is checked to make sure they are not overlapping with the border object. This uses JME's collision detection as no additional information about the overlap is required.

If an overlap is present this means the content item will be occluded. The content item is moved along a vector perpendicular to the edge that overlaps it into the display area. The content item is moved along this vector until it no longer overlaps with the

display border. The item however may be moved to a position where it is overlapping with another edge. Therefore the method may need to be repeated until there is no overlap between the content item and the display border object. With all these stages of transformation complete the application's content items will have no occlusion and will be placed in all regions of the display. However the content item layout, preserved by the VRE, will be lost by the later stages of transformation. This may not be desirable in some applications or for particular content items. Therefore as part of the implementation, methods for applications to define which stages of transformation are used in the initial layout of content items are employed.

The use of a variable called 'steadfastness' is employed to define the stages of transformations which are applied to a content item on their initial placement. The variable is defined as part of the content item super class as are the methods to modify it. This variable is an integer used to define the stages of transformation which should be applied to a content item. By default 'steadfastness' is set to three which means that all stages of transformation are applied. Therefore applications which do not wish to preserve the layout of content items but just wish to make full use of the display area will not need to modify the 'steadfastness' variable at all. If the variable is set to one then the content item will only undergo the transformations from the VRE solution. This is useful for content items which need to be kept in a specific layout. If the variable is set to two then the content item will undergo the transformations from both the VRE and PCIP solutions. However, any of the occlusion caused by the second stage of transformations will not be resolved. This may be useful for content items that are intended to extend beyond the display border.

In addition to changing the 'steadfastness' variable for each content item specifically there is also the ability to change the default 'steadfastness'. This is done through the use of a 'default steadfastness' variable defined as part of the SynergyNet application class. By changing this value any content items which do not have their 'steadfastness' directly modified will conform to the application's 'default steadfastness' value. An example of using this variable is for an application where the layout of all the content items must be preserved. Rather than setting each content item's 'steadfastness' variable individually they can all be modified through changing

the application's 'default steadfastness' variable to one. This gives the application developers control over the influence of the solutions. It also allows for the differences between the criteria for transformable and non-transformable content items to be accommodated for.

Applications can also define acceptable amounts of deformation and transformation. The application super-class has a default 'scale maximum' variable. SynergyNet will not allow an application to run on any display shapes which will force the content items and their layout to be scaled beyond this threshold. There are other deformation and transformation threshold variables which can be set to define acceptable amounts of rotation and translation incurred from the implemented solutions. The scale threshold is useful for stopping applications running on displays that may be too small or thin. On these displays the VRE may be scaled to an extent that the items within will be much too small to manipulate or attain visual information from.

When an application's deformation or transformation threshold is exceeded due to a certain display shape's influence on the implemented solutions a message is displayed to the user. This message states that the application they are trying to access is not compatible with the display shape. This is only likely to occur with display shapes with very small maximal rectangles or applications. This is beneficial to application developers who do not want the contents of their applications to be influenced too much by the implemented solutions. Developers may design applications for specific display shapes and therefore may not want the application to be available on any other display shapes, despite the MSF's ability to use any shaped display. Allowing application developers to set these deformation or transformation thresholds for their applications is another method of allowing greater control over how the solutions' influence. With these different stages of transformations implemented and an application's control over their influence, content items should remain fit for purpose when used with different display shapes.

6.4.2 Detecting Display Shape Changes

For the implementation of the DBS DSDM the waveform object file format was chosen. This format was used for representing three-dimensional shapes and uses a simple text structure to store information on the shape it represents. The structure of the waveform object is very simple where each line can be parsed using spaces as the tokens. The first unique string on each line identifies what the rest of the line defines. For example a line starting with a v is a vertex and three numbers will follow defining the x, y and z values respectively. The waveform files define the vertices, edges, faces and normals of the shape which are all used for defining the geometry of the display shape. As the shapes represented in the implementation are all in two-dimensions all the vertices' z values are zero and the vector normals all face the same along the z axis.

To create the waveform objects the modelling software Blender [114] was used. In the modelling software the shapes could be designed from scratch or existing geometry data could be imported from various file formats. This could be useful for display manufacturers who may have rendered the interface during their design in modelling software. The format that designers use for storing their work is likely to be a three-dimensional model file. This file can then be imported into Blender where modifications can be made, if needed. Such modifications could be the flattening of the shape into two-dimensions or removing any elements of the model which are not part of the display shape. The shape can then be exported into the waveform format.

For every display file a rectangle was used to define the outsides of the border model. This rectangle was modelled on a typical one thousand and twenty four by seven hundred and sixty seven pixel rectangle which all the border vertices are expected to be placed in. The size of the rectangle was only selected as it is a common resolution. The scaling of the rectangle to fit a software environment makes its size and aspect ratio unimportant. The display border object is effectively the shape of the display cut out from the rectangle. For the placement of content items a virtual rectangle is needed to be defined. In this implementation the virtual rectangle is to be created in the border files. As part of the creation of these border files the maximal rectangles in the border's silhouette were identified by using representations of the shapes with maximal

rectangle finding software tools [115]. A developer's judgement was then used to find better placements of the rectangles (meaning the edges not parallel to the software environment's axes) that the software may not have found. The rectangles vertices are all that are needed to be defined in this implementation as the relevant information can be calculated in the MSF as discussed in Section 6.4.1.

Lines defining the rectangle vertices begin with a hash symbol which typically in waveform objects denotes a comment. This is so that any parsers not looking for the rectangle definition will ignore the line and will not encounter errors with this custom structure. Each line then uses an identifier, such as 'Rectangle1', to state which vertex it relates to. This is then followed by two numbers which define its x and y values (no z is needed as the rectangle is two-dimensional). Several border files were created with shapes identified as being likely to be used in multi-touch table designs in the near future. These shapes are discussed later in Chapter 7 and included the typical rectangle border. The rectangular border is treated as the default border due to its current wide usage.

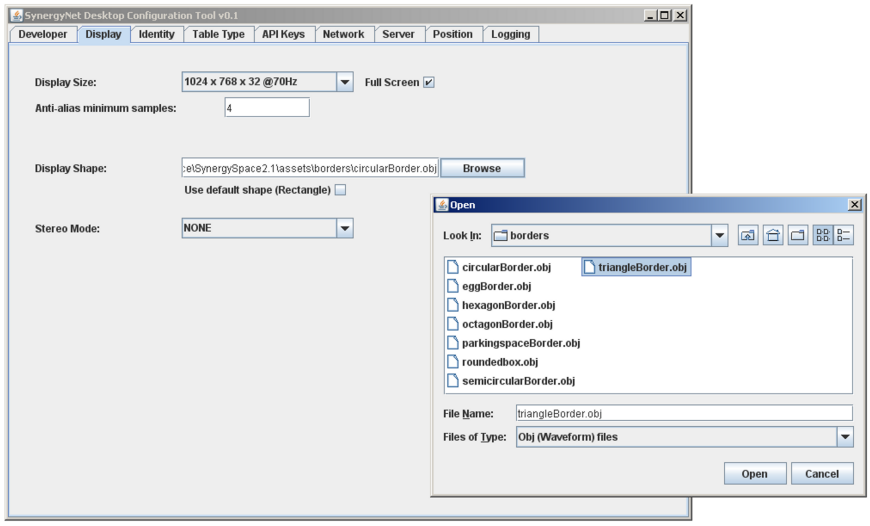


Figure 6.2: SynergyNet's configuration tool.

SynergyNet uses a configuration GUI that allows users to establish and modify the settings of the MSF before its use. This GUI is shown in Figure 6.2. These settings

relate to the management of the system's appearance, inputs, network connections and logging. Added to these, as part of the implementation, was a field allowing for the path to a border file to be specified. The option to browse for the file was also implemented along side this field. The file browser would only show waveform object files with the suffix of .obj to ensure that only files with the valid format could be selected.

Once a file is chosen the preference system stores the current directory so that the user does not need to reselect the same file whenever the software is used with the same display shape. This setup facilitates methods of attaining border files beyond those supplied with the MSF. Border files existing outside the MSF's working directory, such as those downloaded from online repositories, can be accessed through this method of loading the border files. The use of border files allows users to access the geometric data they contain with various pieces of software. This is necessary when users may want to modify border files or create their own. In the configuration system GUI there is also a check box which is used for selecting the default rectangular display shape. The rectangular display shape could be navigated to via the file browser. However, due to its frequent use, because of the current prevalence of rectangular displays, the ability to quickly select it was seen as beneficial.

When SynergyNet runs an application it consults the preference systems to find where to load the border file from. If the path to the file is incorrect (meaning that it does not lead to a valid waveform object file) then SynergyNet will display a message to inform the user of this error. When no valid file is specified SynergyNet will load the default rectangular border file using a hard coded path value. This is possible due to the rectangular border file being stored in SynergyNet's workspace. Therefore its location will always be known to SynergyNet. When a file is loaded its geometric information is used to create a content item. If this geometric is incorrect in some way that stops the solutions from functioning correctly, the user is informed and the rectangular border file is used instead.

As part of the implementation of these solutions a new type of content item was created in SynergyNet. This content item uses JME's function which converts a waveform object file to a mesh of triangles which can be used to render content items in JME applications. As the additions to the border file for establishing the virtual

rectangle are commented out with the hash symbol they do not affect this function. The implementation therefore allowed waveform object files to be used to create shapes in SynergyNet applications. For the border files the shape is read in and placed in the centre of the display. This is because the point at which the waveform object is transformed around is its centre. The shape is then scaled to fit the resolution and aspect ratio of the software environment. This shape is used to represent the border of the display in SynergyNet applications.

The borders are made black so that if a projector is used there will be no bright light output outside the display area where it could be shining into the users' eyes. The waveform object shape is then made to be non-transformable so that it cannot be moved or manipulated by the user. With the border in place the values for the VRE can then be read from the border file using a custom parser. Using the geometry of the virtual rectangle (specifically the locations of its four corners) information for the placement of content items can be calculated, as discussed in Section 6.4.1. With the border file and its accompanying VRE information loaded, their corresponding Java objects can be kept in memory. This allows for the appropriate objects to be recalled whenever an application is started. This means that the consultation of the preference system, the creation of the border object and the parsing of the border file for the virtual rectangle information only need to be performed once. This implementation of the DBS DSDM provides the occlusion solutions with the relevant information needed to manage content items so that they remain fit for purpose.

6.5 Chapter Summary

In this chapter the process of implementing the selected solutions and DSDM into SynergyNet were discussed. The process of implementing the solutions has been relatively issue free with no major problems stopping the integration of the solution. The most major obstacle in this implementation was the orthogonal nature of content items in the SynergyNet applications. This made it difficult to find additional information about collisions when they occurred. As these pieces of additional information were required repeatedly in the implementation a method of overcoming

this obstacle was required. The use of triangles along content item edges which stretched into the z-axis to give the items depth proved to be an adequate method of allowing extra collision information to be collected.

This implementation also highlighted how certain features of the system can influence which solutions can be used. The VS DSDM, though identified as beneficial to the system, could not be implemented. This was because the solution could cause SynergyNet to become dependent on a single lower level MSF. This would undermine SynergyNet's ability to utilise many different input protocols which is not desirable as this is one of the main features of the MSF. The DBS DSDM used proved to be beneficial in aiding the VRE solution. This was due to the border file's provision of the virtual rectangle information, rather than requiring the MSF to calculate it. Due to the software being a MSF that supports a range of applications the solutions were implemented so that application developers could have some control over the solutions' influence. This control is important, particularly in software frameworks.

Chapter 7

Evaluation, Analysis and Discussion

7.1 Chapter Introduction

Evaluation is needed to assess the solutions implemented into SynergyNet discussed in Section 6. The adequacy of the solutions chosen can also be judged as part of the evaluation. The results of evaluating the solutions and their implementation will aid in proving their worth to future developers. If the solutions are shown to be suitable for SynergyNet they can be considered for implementation into other multi-touch systems. The evaluation may identify where the solutions, or their implementations, could undergo improvements and may also highlight any deviations from the solutions' designs in their implementation.

The framework of criteria outlined in Section 4.5.2 were used to evaluate the implemented solutions and DSDM. Using the criteria the solutions' impact on the content items and other features of the MSF could be assessed. In addition to this, observations can be made on the use of the MSF with different display shapes to provide additional data. Prior to the evaluation's execution a structured approach was designed around assessing the implemented solutions using the appropriate criteria.

This ensured that the evaluation of the implemented solutions and other changes to the SynergyNet would be fair and accurate.

7.2 Evaluation Design

The design of the evaluation focused on finding whether or not the criteria, defined as part of the evaluation framework for the solutions, were successfully fulfilled. These criteria were used as a form of check list to identify if the solutions have influenced the visual contents in a satisfactory manner. The check list took the form of a series of questions where each question was based on a specific criterion. The questions were designed to only infer a yes or no answer. This meant that the resulting answers acted as boolean indicators of whether or not the criteria relating to the relevant question was fulfilled or not.

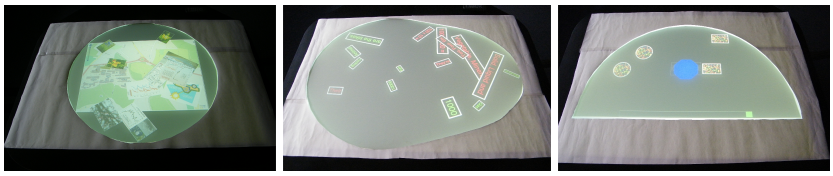


Figure 7.1: Use of different display shapes cut from card to multi-touch tables running SynergyNet applications.

A range of different display shapes were used to ensure that the influence of the implemented solutions worked correctly with more than one display shape. Also a selection of SynergyNet applications was used with each display shape rather than a single application. This allows for the influence of the solutions of different combinations of content items types to be assessed by the criteria derived questions. By using combinations of different display shapes with different SynergyNet applications the influence of various factors which may affect the solutions' influences can be assessed. To apply different display shapes to the visual output of the SynergyNet MSF, multi-touch tables are used. The displays in the surface of these tables are covered in card. The intended display shape is cut out of this card as shown in Figure 7.1 to form the display area.

To represent the criteria of the evaluation framework a series of questions were used. The questions used in the evaluation were grouped based on the evaluation criteria. Each question relates directly to the criterion of the same number given in Section 4.5.2. These questions are stated below.

Question 1: Initial occlusion.

- **a)** Do all transformable content items have less than 50% occlusion from the display border?
- **b)** Do all non-transformable content items have 0% occlusion from the display border?

Question 2: Additional occlusion.

- **a)** Do all transformable content items have less than 50% additional occlusion from other content items?
- **b)** Do all non-transformable content items have 0% additional occlusion from the display border?

Question 3: Rotation.

- **a)** Are all content items rotated the same amount by the solution?

Question 4: Content Layout

- **a)** Is the layout of non-transformable content items preserved in such a way that it is still recognisable?
- **b)** Is the change in scale of the layout of content items less than 50%?

Question 5: Deformation.

- **a)** No content items deformed in anyway other than scaling?
- **b)** No transformable content items scaled to such an extent that they can no longer be manipulated correctly?
- **c)** No non-transformable content items scaled to such an extent that any visual information they contain becomes incommunicable to the user?

Question 6: Display shape detection.

a) Does the software border align with the hardware display border?

b) Does the software ignore user touches outside the display area?

Questions 1a - 5c were answered through observations made on content items when an application was initially started. Question 4b is not directly related to the preservation of the layout of content items but was used to assess whether the combination of solutions capitalises on the display area. For question 6a the answer was determined by changing the display border object colour to white. Normally the display border object colour is set to be black to occlude any light that may be produced outside the display shape. With a white border shape its edges can be checked to make sure they align with the physical display borders. If there are any major deviations the answer to the question is no, otherwise the software display border can be seen to correctly align with the hardware display border. To determine an answer to question 6b several touches were made outside the display area, on the card covering parts of the original rectangular interface. Several gestures and attempts to interact with features of the MSF were made outside the interface with every display shape and application combination to try to illicit a response from the system. If the MSF did not respond in anyway, this indicated that the software was not ignoring user touches outside the display.

For the evaluation, a selection of display shapes would be required. Though the software should in theory be compatible with almost any shape, it was not possible to test an infinite set of potential geometries. Therefore a selection of shapes representative of the likely design of future displays was required. To attain this selection of shapes a discussion took place between those responsible for the design of the evaluation and those involved in the manufacturer of multi-touch tables. The multi-touch table manufacturers involved in this discussion were also producers of typical tables. Therefore these manufacturers were in a useful position for predicting the likely shapes to be used in future multi-touch table designs.

In the discussion with the manufacturers the emulation of typical designs for multi-touch tables was highlighted and noted to be likely to continue. Therefore it is

likely that some future display shapes will be similar to the shapes of current tabletops. Also highlighted was how future table top display shapes may take advantage of ability to collaborate which multi-touch interaction offers. From this discussion eight likely future display shapes were identified and agreed upon for use in the evaluation. These shapes were chosen due to the likelihood of their resemblance to the shapes that may be used with the SynergyNet MSF with the next generation of multi-touch table designs. These shapes are shown in Figure 7.2.

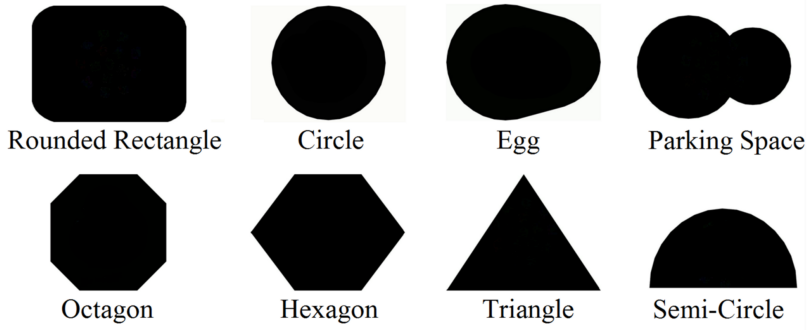


Figure 7.2: The selection of display shapes used in the evaluation.

In addition to the use of the eight shapes shown in Figure 7.2 the rectangular display shape for which SynergyNet and its applications were originally designed, was used. This shape acted as a control for the evaluation, allowing for the use of applications with different display shapes to be compared with the typical use of the system. The layouts of content items when used with different display shapes were compared with the layout of content items from the same applications when used with the rectangular display. Observations on the translations resulting from the implemented solutions could be made from this comparison. Measurements of the occlusion, rotation, translation and scaling which result from the influence of the solutions were used to answer the criteria derived questions. These measurements were made through a combination of SynergyNet's logging functionality and measurements made on the interface with a ruler and protractor.

The non-rectangular shapes used were derived from display shapes deemed likely to be used in future multi-touch systems. Some of these shapes were derived from

common table shapes which could be used with multi-touch interfaces such as the semi-circle design and rounded rectangle. The rounded rectangle shape has an additional reason for its use due to its similarity to the control shape. The similarity of this display shape to the control allowed for observations on whether the magnitude of the influence of the solutions is proportional to the size of the differences between the display shapes The other shapes used were identified as likely to benefit collaboration around a multi-touch interface. Many of these shapes are regular, meaning all their sides and corners are congruent [116]. This allows for users positioned around the display in various places to have an equal share of the interface which can be beneficial to collaboration. These regular shapes include the circle, octagon, hexagon and triangle.

Other shapes were designed to make further use of multi-touch's facilitation of collaboration by providing focal points and areas to share content items in. For example the 'Egg' shape has a focal point at the centre of the large curved edge, towards the left hand side of its centre as shown in Figure 7.2. Items intended to be displayed to multiple users can be placed in this focal point area. In addition to this the 'Egg' shape has a lot of space to the right of the focal point. This area allows users to manipulate content items without interfering with anything intended to be displayed to other users.

The 'Parking Space' shape is similar to the 'Egg' shape. Here the larger circle can be used for one activity such as displaying content items whereas the smaller circle could be used for another activity such as manipulating the items. For these test shapes waveform objects were created to represent them in the software. For these waveform objects the maximal rectangle was found with an alternative representation of the file's geometry that could be used with an online tool [115]. This tool found the maximal rectangle which could then be manually entered into the waveform objects. However, as a result of this tool using Alt's algorithm [101], all the virtual rectangles used were parallel to the software environment's axis.

With each shape the same selection of applications were used to evaluate the effect the combination of the solutions and display shapes had on different collections of visual content items. The applications can be divided into three groups based on the types of content items they contain. These groupings are applications that use only

non-transformable content items, those which use only transformable content items and applications which use a mix of both types of content items. The group of applications that use a mix of content items represents a share of SynergyNet Applications greater than the other two groupings combined. For this reason four applications were used, with one application to represent each of the smaller groups and two to represent the largest group. These groupings are relevant to this evaluation because of the difference in the criteria concerning the two types of content items. Evaluating applications with only a single type of content item aided in evaluating the criteria related to this item type. This is because there would be no possible interference from instances of the other content item types. The two applications which use a mix of content items could then be used to evaluate how solutions manage the two types of content items in the same application. This allowed for all possible content item type combinations to be represented and the influence of the implemented solutions on them to be evaluated.

To represent the grouping of applications which contains only non-transformable content items the SynergyNet application known as Network Presenter was used. This application initially comprises of four buttons in a row which cannot be moved. Pressing the button creates content items in the centre of the display. These additional content items are transformable and can be flicked to other displays over the network. However, since this evaluation is only concerned with the initial state of the application, it is considered by this evaluation as containing only non-transformable content items.

To represent applications containing only transformable content items the XML Puzzle application was used. The application reads in question and answer text from an XML file. It then creates corresponding content items which show the text from these questions and answers. These content items are randomly placed, rotated and scaled on the display and the user must align the appropriate questions and answers. There are settings which affect the set up of this application. For example the questions can be changed to be non-transformable and placed in a column, but in this evaluation the application is set up to be transformable content items only.

For the mix of transformable and non-transformable content items grouping the two applications used were SynergyNet's Sandbox and Simple Map applications. The Sandbox application comprises of several transformable content items which can be

flicked around the display. In addition to these items a bar is placed towards the bottom of the display which is non-transformable. One of the environment's transformable content items is a virtual keyboard which can be used to enter text that appears in the non-transformable bar. The bar also has a button attached which can toggle the keyboard's state between being transformable and non-transformable, though its initial state when the application starts is transformable. Though this application has no true objective its use in evaluation includes typing short messages into the bar and bouncing all the transformable objects off each other.

The Simple Map application uses a map as a background image. For some applications the background image may not need to be influenced by the implemented solutions. However the map is intended to be seen in its entirety and therefore cannot be occluded at all. Therefore the map object is classified as a typical non-transformable content item. On top of the map background several transformable content items are placed which can be moved around the display, though they cannot be flicked. The evaluation was carried out by attaching a display shape to the multi-touch table then using these four applications. To navigate between the applications the SynergyNet menu system was used. Once the applications were tested the shape was changed and the process was then repeated.

The results of this evaluation are represented as the answers to the questions derived from the criteria. With the design of the evaluation outlined, a hypothesis of the results could be made. The hypothesis made before this evaluation took place was that: *changes in the display shape will not affect content items in such a way that leaves them unfit for purpose*. As the questions are worded to give a 'yes' answer when the related criteria have been successfully fulfilled then if the hypothesis held true then the answers to all the criteria derived questions for each combination of a SynergyNet application and a display shape would be 'yes'.

7.3 Results

The results for the evaluation were collated using the following methods:

- *Direct observations from those conducting the evaluations.*
  Notes were made during the use of the applications with different display shapes. These notes include the measurements made of the content items' positions and orientations on the table's interface.

- *Collation of data files created by SynergyNet's logging functionality.*
  These files included positional information related to content items and any possible errors that may have occurred during the MSF's use.

- *Capturing video with screen-capture software.*
  The lab in which the evaluation was carried out contains a number of dedicated machines used for capturing video of the multi-touch systems in use. The videos captured for this evaluation were recordings taken directly from the software environment. The output from the computers used with the multi-touch tables was sent to two destinations. The first being the tabletop interface and the second was sent to the capture system. Therefore videos were created in this evaluation of the output of the SynergyNet MSF. With the display border object in SynergyNet turned white for the evaluation, the display shape can be easily seen in these evaluation videos against the black background of the SynergyNet applications. As the video recording took place on a remote computer it did not use resource local to the instance of SynergyNet being used and therefore did not effect the MSF's responsiveness. Images taken from these videos can be seen in Figures 7.3, 7.4, 7.5 and 7.6.

- *Taking photos during the evaluation.*
  In the photos taken both the hardware and software implementations of the borders can be seen. This allowed for their alignment to be checked. These photos provided additional information which could be used to answer question 1b. Three instances of these photos are shown in Figure 7.2.

For each method, the information collected was appropriately labelled to identify which application and display shape they related to. With the notes, recordings and photos of the evaluation each criteria derived question could then be answered and cross

checked. The most relevant observations from these resources were also recorded. To represent the findings from the questions relating to the criteria, a series of tables were used. In total four tables were used, one for each application. Each table uses columns for questions and the rows for the display shapes. Therefore each of the cells containing the results will relate to a specific display shape and question for the application. Each cell may be filled with a 'Y' if the answer to the question was yes or a 'N' if the answer to the question was no.

However for some of the applications some questions were not applicable. Some of the questions only apply to a specific content item type because of the criteria it is derived from. Therefore some questions will not be applicable to an application which does not contain any instances of the relevant content item type. For example question 2a relates to occlusion of transformable content items only. Therefore it is not relevant to SynergyNet's Network Presenter application which only contains non-transformable content items. So where a question does not apply to an application '/' was used in the cells of the application's corresponding table relating to that question.

The appearance of a 'N' response in any of the tables indicated a deviation from the hypothesis. This is because the hypothesis predicted that the answer to the all the relevant criteria derived questions would be positive for all combinations of applications and display shapes. These differences between the expectations and the actual results were investigated to find why they occurred. These deviations could have occurred either in the design of the solutions or in their implementation. When the cause of these deviations was found, it was then be ascertained whether it was a major or minor issue. Minor issues were identified as problems that could be corrected. Major issues were problems which would require the solution design or implementation to be completely overhauled. Sections 7.3.1 to 7.3.4 discuss the results for each individual application. Section 7.3.5 then discusses the results as a whole.

7.3.1 Network Presenter Application

The Network Presenter, despite having few content items, provides a useful representation of application which contains only non-transformable content items.

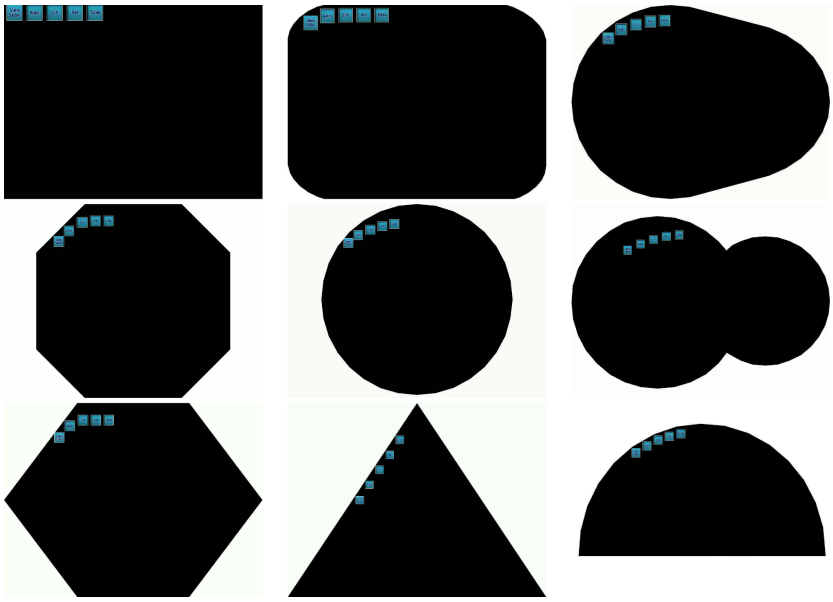


Figure 7.3: The SynergyNet network presenter application in use with a range of display shapes.

The layout of the items in a row provides an excellent platform for observing the influence of the solutions. As can be seen in Figure 7.3 the alignment of the content items is clear with all display shapes. Though not always a straight line it is clear that the items are in a row. A line drawn through the centre of the content items in all the non-rectangular displays can be seen to always be a form of the original line seen in the rectangular control shape. This deformation of the layout of these non-rectangular content items is acceptable due to this preservation of the relationship between the items' positions.

One of the observations about the influence of the solutions on this application is that despite scaling the content items the text they contain is still visible with all the display shapes. The triangle display shape caused the most scaling for this application but the text was still visible. The different display shapes in combination with this application show how the solutions can adequately reposition the content items with indifference to whether the shape has straight edges, gradual curved edges or combinations of both.

The results of the questions shown in Table 7.1 fully conform to the hypothesis. All the answers to the questions, where applicable, are positive, indicating that the criteria have been successfully fulfilled. As there a no transformable content items present in this application questions 1a, 2a, 3a, 5a and 5b are not relevant.

		Questions											
		1		2		3	4		5			6	
		a	b	a	b	a	a	b	a	b	c	a	b
Shapes	Rectangle	/	Y	/	Y	/	Y	Y	/	/	Y	Y	Y
	Rounded Rectangle	/	Y	/	Y	/	Y	Y	/	/	Y	Y	Y
	Circle	/	Y	/	Y	/	Y	Y	/	/	Y	Y	Y
	Egg	/	Y	/	Y	/	Y	Y	/	/	Y	Y	Y
	Parking Space	/	Y	/	Y	/	Y	Y	/	/	Y	Y	Y
	Octagon	/	Y	/	Y	/	Y	Y	/	/	Y	Y	Y
	Hexagon	/	Y	/	Y	/	Y	Y	/	/	Y	Y	Y
	Triangle	/	Y	/	Y	/	Y	Y	/	/	Y	Y	Y
	Semi-Circle	/	Y	/	Y	/	Y	Y	/	/	Y	Y	Y

Table 7.1: Evaluation of the SynergyNet Network Presenter application's compatibility with different display shapes.

The results from this application show that the solutions when used with all the test shapes eliminate the initial occlusion without incurring any further issues. There is occlusion despite the items' close proximity to each other and the display edges. No issues with rotation were incurred either as all the content items remain at the same rotation. This is likely due to the shapes all having axis-parallel virtual rectangles that appear to have the same orientation. There is unlikely to be any differences in the rotation of non-transformable content items with the display shape objects created for this evaluation. Any deviations in non-transformable content item rotations will therefore act as an indication of a problem in the design or implementation of the solutions. There were no issues relating to deformation present because the only deformation that can be applied to content items is scaling. The scaling is acceptable despite appearing to be relatively extreme for some of the display shapes, such as the triangle, as the text on the items was always legible. This shows that the visual information of the content items is still communicable to the user and therefore they are still fit for purpose.

The layout of the visual content items is preserved in such a way that makes the relation between the items' positions clear to the user. The layout is also not scaled beyond 50% so that the content items are not compressed into one portion of the display. The layout of the content items is constrained to a single area of the display with the control shape of the rectangle. The consequent layouts with other display shapes are not compressed into any areas significantly smaller than the area occupied by the objects when used with the rectangular display. The results show that the implemented solutions do not affect the visual content items of this application in such a way that leaves them unfit for purpose.

7.3.2 XML Puzzle Application

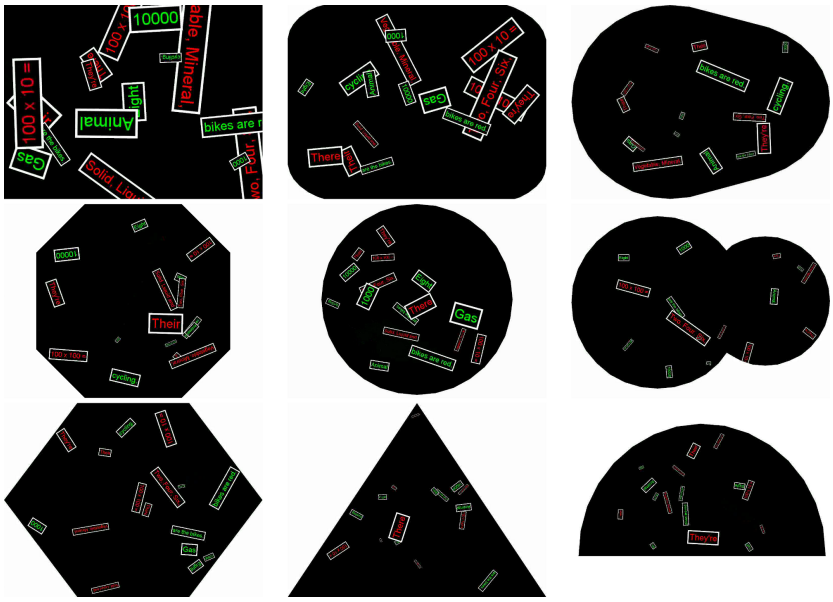


Figure 7.4: The SynergyNet XML puzzle application in use with a range of display shapes.

The XML Puzzle application initially appears to conform fully to the hypothesis that the influence of the solutions will not result in content items becoming unfit for purpose. The initial layout of the content items for each display shape, shown in Figure 7.4, showed that there is an acceptable amount of occlusion for the transformable

content items. The most occlusion that occurred was observed in the control set but it was still within the acceptable amount of occlusion. The content items are placed and rotated randomly so any influence from the solutions is unlikely to be undesirable. However the scaling of content items is also random. This leads to a problem with the additional scaling applied by the implemented solutions. A content item which is scaled to a smaller size by the application is then scaled to an even smaller size by the implemented solutions. This is problematic when text is involved, as can be seen on several of the shapes in Figure 7.4 where some of the content items are scaled far too small to read.

Text being too small to read is not an issue for non-transformable content items due to the fact that the item can be scaled by the user. However there is a problem when these content items are too small to manipulate. When this happens it is not possible to resize a content item so that the text it contains will become readable. This was observed to happen with some combinations of display shapes and this application as shown by the results.

The results of the questions shown in Table 7.2 mostly conform to the hypothesis. However the answers for question 5b show a deviation from the prediction that only positive answers will be returned from the evaluation. All the answers to the other questions, where applicable, are positive indicating that the relevant criteria have been successfully fulfilled. As there are no non-transformable content items in this application, the questions 1b, 2a, 2b, 4a, 4b and 5c are not relevant.

These results show that the majority of the relevant criteria have been fulfilled. However the criteria relating to the deformation of content items was not successfully fulfilled. Specifically the criteria relating to the scaling of transformable content items was not met. There were several instances where a content item was too small to perform two finger gestures. Therefore these content items could not be scaled by the user. This would make these content items unfit for purpose as they cannot be manipulated as intended. The text shown on these content items was too small to read and as the item could not be resized it would remain this way. The results show the content items could be correctly manipulated with the control shape of the rectangle and also with the similar rounded rectangle shape. This indicates that the magnitude of

		Questions											
		1		2		3	4		5			6	
		a	b	a	b	a	a	b	a	b	c	a	b
Shapes	Rectangle	Y	/	/	/	Y	/	/	Y	Y	/	Y	Y
	Rounded Rectangle	Y	/	/	/	Y	/	/	Y	Y	/	Y	Y
	Circle	Y	/	/	/	Y	/	/	Y	Y	/	Y	Y
	Egg	Y	/	/	/	Y	/	/	Y	N	/	Y	Y
	Parking Space	Y	/	/	/	Y	/	/	Y	N	/	Y	Y
	Octagon	Y	/	/	/	Y	/	/	Y	N	/	Y	Y
	Hexagon	Y	/	/	/	Y	/	/	Y	N	/	Y	Y
	Triangle	Y	/	/	/	Y	/	/	Y	N	/	Y	Y
	Semi-Circle	Y	/	/	/	Y	/	/	Y	N	/	Y	Y

Table 7.2: Evaluation of the SynergyNet XML Puzzle application's compatibility with different display shapes.

the influence of the solutions appears to be proportional to the difference between the display shape and the original rectangular shape. The circle shape was also shown not to have this issue, though this may be because of the random scaling of content items.

The question of whether this deviation from the hypothesis originates from a fault in the design of the solutions or their implementations was raised. The VRE solution is the only part of the combination of solutions implemented which would inflict a scaling transformation on the content items. Therefore it is likely that the problem of the extreme scaling will be likely to originate from this solution. The VRE solution design states that there must be a lower scale limit defined which would stop this problem occurring. As this is addressed in the design the problem therefore originated in the implementation. No method of adhering to a lower scale limit is implemented. If implemented this problem would be corrected as it would not allow content items to be scaled below a certain value.

A method of adhering to a scale limit was identified. This method could be simply implemented by making use of SynergyNet's already existing content item scale limits which are used for managing deformations performed by the user. The only potential issue that may arise from this change is that a content item may not be scaled enough to fit within the virtual rectangle. However the third stage of the transformation of content items described in Section 6.4.1 will ensure this will not happen. This may

have the additional benefit of make more use of the display area as content items will not be constrained to the VRE. A single change to the MSF would only need to be implemented in the methods used in the VRE solution. No modification of the applications in SynergyNet would be required for the implementation of these changes. The implementation of this change is discussed in Section 7.4. Therefore the deviation from hypothesis was identified as an easily corrected oversight in the implementation of the solutions. This is only a minor deviation and the implemented solutions should not be considered ineffective because of it.

The other results from this application show that occlusion was resolved correctly when the solutions were used with this application. As all the content items are transformable, their occlusion is not problematic as long as they can still be manipulated. As none of the content items on their initial placement were occluded more than 50% they could still be manipulated in such a way that they could be moved away from the cause of occlusion. Therefore the text they contained could be read after some user manipulation. No issues with rotation were incurred either, as all the content items underwent different, random rotations and remained at different rotations after the influence of the implemented solutions. As the content items were positioned randomly, there was no layout to preserve, hence why none of the questions related to criterion 4 are relevant to this application. The results show that the implemented solutions do not affect the visual content items in anyway, except scaling, which leaves them unfit for purpose. The problem with implementation that causes content items to become unfit for purpose in this application due to scaling can be easily corrected. This allows for no undesirable consequences of the solutions to be inflicted on the content items in future use of the application.

7.3.3 Sandbox Application

The Sandbox Application in SynergyNet contains a combination of transformable and non-transformable content items. Therefore it can be used to evaluate how the implemented solutions influence both content item types when they are used together. The random placement of most of the content items in this application means that their

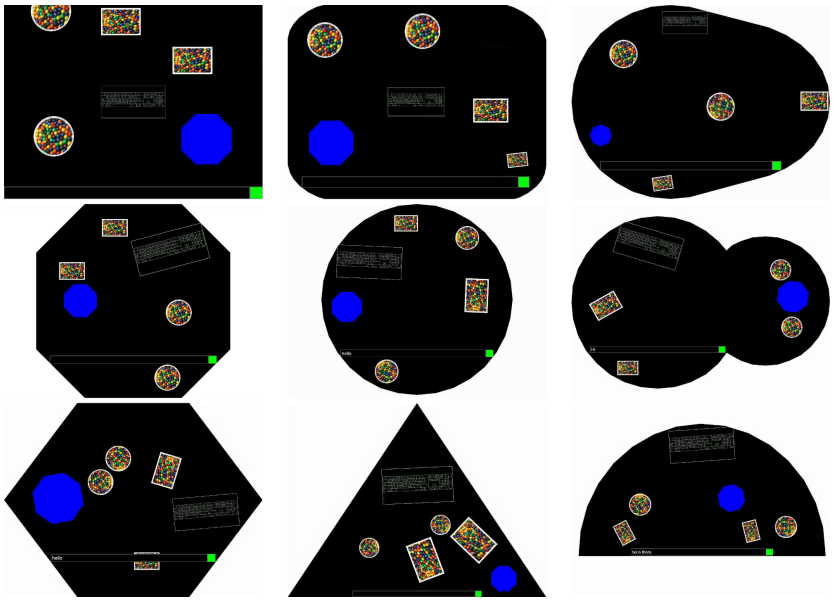


Figure 7.5: The SynergyNet sandbox application in use with a range of display shapes.

positioning is not relevant. However the non-transformable text field is always placed with the non-transformable button for freezing the keyboard object placed directly next to it. Since these two objects are placed side by side, any incorrect management of the content items could easily lead to them overlapping. However as can be seen in Figure 7.5 these two items never overlap and are always correctly aligned. In addition to this, occlusion is never inflicted on these two non-transformable content items which is a desirable result of the implemented solutions. As the non-transformable content item of the bar is relatively large in comparison to other content items, its correct placement demonstrates how effective the implemented solutions are.

As part of the evaluation the keyboard was used to type a short message onto the non-transformable text field object. The message was always legible showing that the content item had not been scaled too small. The keyboard item proved to be usable with each display shape showing that the solutions had no negative influence on it and its use. The results of the questions shown in Table 7.3 fully conform to the hypothesis. The answers to all of the questions were positive indicating that the criteria had all been successfully fulfilled. Since a combination of content item types are used all the

criteria, and therefore their corresponding questions, can be applied to this application.

		Questions											
		1		2		3	4		5			6	
		a	b	a	b	a	a	b	a	b	c	a	b
Shapes	Rectangle	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Rounded Rectangle	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Circle	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Egg	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Parking Space	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Octagon	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Hexagon	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Triangle	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Semi-Circle	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Table 7.3: Evaluation of the SynergyNet Sandbox application's compatibility with different display shapes.

The results show that for all combinations of this application and the display shapes, the implemented solutions eliminate the initial occlusion of all of the non-transformable content items. In addition to this, none of the non-transformable content items incur over 50% occlusion so they can still be moved about the display by user interaction. No issues with rotation were incurred in any instances of the use of this application. This is because none of the content items were rotated in their initial placement and were not rotated in any further when used with any of the display shapes. There are no issues with deformation due to scaling being the only deformation that can be applied to content items. The scaling is acceptable as the transformable content items were always capable of being manipulated correctly. Another demonstration of the acceptability of the scaling was that the text displayed by the non-transformable content items was always legible.

There is no specific layout of the content items beyond the placement of the non-transformable bar and button objects. Since these two objects are always positioned relative to each other the layout can be said to be preserved correctly by the solutions. It is apparent that the solutions influence the content items in an acceptable manner while both content item types are present in the same application. The criteria were fulfilled successfully for all instances of the Sandbox Application's use with the

display shapes. This demonstrates that the implemented solutions do not affect any of the transformable or non-transformable visual content items in such a way that leaves them unfit for purpose.

7.3.4 Simple Map Application

Figure 7.6: The SynergyNet simple map application in use with a range of display shapes.

The majority of the Simple Map application's contents were transformable content items. The map object however was a non-transformable content item in this application which acts as a backdrop. Its full visibility was not required for the use of the application as the map was just a back ground to the application's transformable content items. However the decision was made to attempt to reduce the occlusion of the map. This was so that if a user of the application had a need of the map, rather than just using it as background, the map image could be seen in its entirety. Due to the map object's large size, its occlusion by other content items is irrelevant as the display real estate used by all the transformable content items on their initial placement is less than half of the amount of the display real estate that the map image occupies.

As long as all the content items were scaled by the same amount (which is ensured by the solution design) then it would be impossible for the transformable content items to occlude more than 50% of the map object. The map object is rectangular and is transformed using the virtual rectangle values and is set to not be influenced by any of the further stages of transformation. This means that the map objects acts as a visible representation of the virtual rectangle as can be seen in Figure 7.6.

Despite the correct placement of the map there was a recurring problem with the solutions' influence on it. The map object contained text which was initially quite small. When the item was scaled down to fit within the VRE the text after a small amount of scaling became illegible. The text is barely readable in its initial form and is still legible after the minor scaling involved when the application was used with the rounded rectangle shape. However the scaling needed for any shapes with smaller VREs resulted in the text becoming impossible to read. As the map object is a non-transformable content item it could not be manipulated by the user in such a way as to make the text legible. Therefore the visible information to be presented by the map object's text becomes incommunicable to the user. This is shown by the results of the evaluation.

Most of the question answers shown in Table 7.4 conform to the hypothesis. However the answers for question 5c indicate a deviation from the prediction that only positive answers will be returned from the evaluation. All the answers to the other questions are positive, indicating that the relevant criteria have been successfully fulfilled. Since a combination of content item types are used all the criteria, and therefore their corresponding questions, can be applied to this application.

The results relating to the Simple Map application show that the majority of the criteria have been successfully fulfilled. However the specific criterion relating to the scaling of non-transformable content items has not been fulfilled. There were several instances of the use of the application with different display shapes where the non-transformable content item was too small to read text from. The original text on the map image was already too small for clear use, this is obviously a problem with the application but the influence of the solutions exacerbated it.

The implementation of VRE solution was again identified as the cause of this

		Questions											
		1		2		3	4		5			6	
		a	b	a	b	a	a	b	a	b	c	a	b
Shapes	Rectangle	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Rounded Rectangle	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Circle	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
	Egg	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
	Parking Space	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
	Octagon	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
	Hexagon	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
	Triangle	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
	Semi-Circle	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y

Table 7.4: Evaluation of the SynergyNet Simple Map application's compatibility with different display shapes.

problem. This was because this is the only solution implemented which performs scaling and the design of it states that limits should be placed on scaling. A potential method of resolving this problem was identified. This was the suggested change for making the implemented solutions adhere to existing content item scale limits detailed in Section 7.3.2. The implementation of this change is discussed in Section 7.4. Also improving the map image would be beneficial, though this is a suggested improvement to the application rather than the implemented solutions. Therefore the deviation from the hypothesis can be seen to be due to an easily correct oversight in the implementation of the solutions. This is only a minor deviation and the implemented solutions should not be considered ineffective because of it.

The results relating to this application show that the solutions resolve the initial occlusion of content items adequately when used with any of the display shapes. For the transformable content items their occlusion is not problematic as long as they can still be manipulated. As none of the transformable content items on their initial placement were occluded more than 50% they could still be manipulated in such a way that they could be moved. Therefore users could move any content item away from the cause of occlusion and as a result any visual information they contained could be viewed in its entirety. The non-transformable content item incurred no occlusion on its initial placement, therefore the criteria relating to occlusion was

successfully fulfilled. In addition to this no issues with rotation were incurred. This was because all the transformable content items underwent random rotations and remained at these different orientations when influenced by the implemented solutions. The VRE did not differ in rotation between any of the other display shapes. Therefore the non-transformable map item was not rotated by the solutions due to its alignment with the virtual rectangle. Therefore the questions relating to content item rotation received positive responses.

The transformable content items were never scaled to a point where they could not be correctly manipulated. Therefore their scaling was not the cause of any issues unlike the scaling of the non-transformable content item. As the transformable content items were positioned randomly there was no layout to preserve between them. The non-transformable content item was not positioned in relation to anything. Therefore the questions relating to criterion 4 all received positive responses. With most of the criteria fulfilled successfully for the Simple Map application, the implemented solutions can be seen to not affect the content items in anyway which leaves them unfit for purpose. The only exception to this statement is the solutions scaling. However the problems with implementation that do cause content items to become unfit for purpose due to scaling in this application can be easily corrected. This will allow the solutions to have no undesirable influence on the content items in the application's the future use.

7.3.5 Summary of Evaluation Results

The majority of the results adhere to the hypothesis that the implemented solutions will not cause content items to become unfit for purpose. For all combinations of applications and display shapes the software borders were observed to align correctly with the physical display borders. In no instance of use during the evaluation did the MSF respond to user touches outside the display area. This indicates that the implemented DBS DSDM was capable of building accurate software representations of the display border.

None of the content items in the evaluation were inflicted with unacceptable amounts of occlusion. The orientation of the content items was never altered by the

solutions. This meant that no content items were rotated to unacceptable orientations. This is in part due to the edges of the virtual rectangles environment defined by the border waveform files being parallel to the axes of the software environment. The lack of rotation-based issues indicates that the solutions manage orientation correctly as otherwise problems would have arisen, even with axis-parallel virtual rectangles. The layout of non-transformable content items was highlighted by the results as always being preserved in a manner that made the relation of the items' positions recognisable. The layout of content items was also noted to always be transformed and deformed to make use of the display area. For the applications where non-transformable content items were randomly placed, the layout of the items covered the majority of the display leaving no unused areas. This demonstrates that the implemented solutions are able to correctly manage the transformation and deformation of content item layouts and capitalise on the display shape.

The implemented solutions do not perform any type of deformation beyond scaling. Therefore no unwanted consequences occurred due to the solutions performing undesirable deformations. The influence of the solutions has been seen to fulfil almost all the evaluation criteria. The one exception to this observation is the influence of the solutions on the scales of the content items. In both the Simple Map and the XML puzzle applications content items were scaled down to an extent that the text they carried became illegible. This would not have been an issue for the XML puzzle if the items could have be resized by the user. However the content items were scaled too small to be manipulated by the two fingered gesture required for scaling so they could not be resized. This is a major issue as it results in the users being unable to interact with the application as intended. The content items affected became unfit for purpose as the user could not read the text they contained.

These problems in scaling were identified as having the same cause; a deviation in implementation from the design of the VRE solution. The solution design calls for limits to be placed on the scaling it performs which were not implemented. By making use of SynergyNet's existing scale limits for content items these problems could be easily corrected as discussed in Section 7.4. The scaling problems were not present in all applications where content items displaying text were included. This demonstrated

that the deviation from the solution design will only affect content items which display text that is already small or poorly rendered. This means that the appearance of this issue is not evidence for the failure of the design of the solutions; it is evidence of a fault in implementation which could be easily corrected. The evaluation hypothesis stated that: *changes in the display shape will not affect content items in such a way that leaves them unfit for purpose*. The implemented solutions can be seen to almost entirely adhere to this hypothesis. This evaluation showed that with a minor alteration, the implemented changes in SynergyNet allow the MSF to adapt to different display shapes in such a way that ensures content items remain fit for purpose.

7.4 Chapter Summary

Based on the results of the evaluation the VRE solution was modified. This change altered the implementation to adhere to the existing scale limits of content items. This meant that the implemented solutions would not scale transformable content items to become too small to manipulate. This also meant that any non-transformable content item may not be scaled to an extent that the visual information it displays becomes incommunicable to the users. A brief secondary evaluation was conducted after the implementation of the change using the same structure as before. For this secondary evaluation only the two applications whose results initially deviated from the hypothesis were used: the XML Puzzle and Simple Map applications. The results of this evaluation showed that the content items were no longer scaled to values which left them unfit for purpose. When used with all the display shapes the question and answer items in the XML Puzzle Application could all be correctly resized and rotated with two finger gestures.

For the Simple Map application an improved map image was used with much clearer text. For all the display shapes used, the text on the map remained visible when scaled. The change to the map image would have likely corrected the text rendering problem on its own without the changes to the VRE solution being implemented. This demonstrates how developers must be aware of existing problems with their software which could be exacerbated when adapting software to use different display shapes.

The results of the secondary evaluation fully conformed to the original hypothesis and showed that the adaptation of SynergyNet had been successful. The MSF could now dynamically adapt its contents so that its applications could be used with any display shape without causing any of the content items to become unfit for purpose. The evaluation demonstrated that not only was the implementation of the solutions ultimately successful but so were the solutions' original designs.

Chapter 8

Conclusion and Future Work

8.1 Chapter Introduction

The research documented by this thesis was intended to investigate how multi-touch software could support the use of different display shapes. This work has thoroughly discussed and elaborated on the issue of occlusion caused by the placement of visual content items when software is used with different display shapes. This issue, and the result of attempts to resolve it, were shown to cause software to become unusable by making content items become unfit for purpose. As part of this research several solutions were devised which were intended to resolve occlusion and appropriately manage their impact in multi-touch software. From the evaluation of the selected solutions and their eventual implementations, the most suitable solutions for multi-touch software in similar scenarios could be decided on. The implication of this is that any future multi-touch software developers who wish to develop similar systems can implement the suggested solutions. The implementation of these solutions would allow their software to dynamically adapt its contents to different display shapes. All the solutions and DSDMs developed, not just those chosen for implementation, could be used by future developers for different systems. In addition to these solutions the research demonstrated how the solutions selected for implementation could be chosen through the use of an evaluation framework. This technique could be developed further

and used by future developers to generate suitable evaluation frameworks for judging the resolution of the initial occlusion in other systems.

8.2 Research Summary

The research was begun with a literature survey. The initial intention of this work was to produce a beneficial development for use in multi-touch systems. Therefore this survey was designed to provide an overview of the current state of multi-touch to identify any gaps in research relating to it. The literature survey was performed following an adapted version of an established protocol [6, 7]. The survey identified several trends relating to the current state of research concerning multi-touch. One such trend was how each piece of research was focused on one of the three elements a multi-touch system comprises of; the technology, the MSF or its application. The MSF was the element of multi-touch systems with the least amount of work relating to it. From the results of the survey it was apparent that there is a lack of cohesion between research projects concerning multi-touch. This lack of cohesion has resulted in several pieces of work overlapping and gaps in research.

One such gap identified was the effect of the shape of the interface on the use of multi-touch systems. Collaboration around a display may be influenced by many factors. Therefore researchers may soon need to investigate the influence of a multi-touch interface's shape on its use. However for this to be possible, software capable of supporting different display shapes must be used. The literature survey however found no multi-touch software that has this feature. Therefore a secondary literature survey was conducted to investigate research relating to the use of multi-touch software with different display shapes.

The second literature survey was initially intended to find more details on research relating to multi-touch systems which utilise non-rectangular displays. However only one resource was returned by this initial search. This resource related to PyMT's 'Puddle of Life ' [50] application. The scope of the survey was therefore expanded to include software that used any kind of interaction technique, not just multi-touch software. More results were returned, however compared to the original literature

survey concerning multi-touch the resources discovered were relatively few. This survey highlighted how there is growing technological support for different display shapes which indicates a growing desire for systems which utilise them. However the software currently used to support these new display shapes are developed for specific shapes. The software designed for a specific non-rectangular output is only intended for use with that display shape. This is similar to how the majority of current software is intended for use with rectangular display shapes.

There are several examples of software systems returned by this literature survey which can be used with a small set of different display shapes. The layouts of the visual content items of these systems are specified for each of these possible shapes. This method of statically setting the items' layout will require more work on the behalf of the developers when more shapes are intended to be used by the system. The concept of allowing software to automatically adjust the layout of its visual contents was derived from this observation. An example of a system which could dynamically adapt its contents found by the literature survey was discussed by Dietz et al. [98]. In the system discussed the output from several projectors is shown on different shaped objects. This system however used relatively few and basic visual content items which could be placed anywhere within the projection without the loss of system functionality. From this literature survey the following observation of a gap in research was made: there are no instances of display shape independent multi-touch software. The remainder of the research that this thesis detailed therefore investigated the development of methods that allow multi-touch systems to adapt to different display shapes. This can be seen as a beneficial multi-touch development and therefore complies with the initial intention of this research.

The research focused on identifying the issues of utilising different display shapes with multi-touch software. Occlusion was identified as an issue which caused content items to become unfit for purpose. To do this a prototype was produced where an existing MSF designed for rectangular visual outputs was used with a non-rectangular display. The research then focused on resolving occlusion caused by the initial placement of content items when used with a different display shape. From the prototype observations on the impact of attempts to resolve this occlusion were made.

Likely issues which resulted from the attempts to resolve occlusion that solutions would be required to manage were identified. These issues were identified as further occlusion caused by content items overlapping, content item orientation, loss of the content items' layout and content item deformation.

Using these observed issues, an evaluation framework for judging solutions to the initial occlusion was produced. This framework consisted of a criteria derived from the issue of the initial occlusion and the potential issues resulting from attempts to resolve it. Criteria were also derived from a solution's need to be informed of the display shape.

Several potential solutions to occlusion were then created and discussed. One such example of these solutions was the VRE solution where the original layout of the content items is treated as a single object. This object, a rectangle due to the likely original output of the system, is then placed within the display shape. The content items are scaled, rotated and translated by the same values as the virtual environment that encloses them. Another solution discussed was the PCIP solution which deformed the layout of content items to fit the display shape. Also discussed was the WO solution which involved treating the entire visual output as a single object and deforming it to fit the display shape. The solutions were designed to be potentially used in combination with each other to ensure that all content items remain fit for purpose and that the shape of the display was capitalised on.

Several DSDMs were proposed which could be used to inform the solutions of the display shape. DBS was a DSDM discussed in which a display shape's geometric information was stored in a structure which could be accessed by the appropriate software. The VS DSDM, which uses a system's visual input to view the display and calculate its geometry, was also detailed. This DSDM was observed to be capable of being implemented in a number of different ways. These different implementations involved the use of image processing techniques, fiducial markers [20] or methods similar to those used in three-dimensional scanning [107]. Also discussed was the UC DSDM which would use the input from a user to set the display geometry. This solution was identified as being dependent on a user's accuracy and is better suited to direct touch interfaces.

A selection of solutions and DSDMs was identified as being suited to multi-touch

systems intended for use on tabletop interfaces. These solutions were implemented into the SynergyNet MSF. The VRE and PCIP solutions were then chosen for implementation with the DBS DSDM. The implementation of these solutions was relatively straightforward.

An evaluation of the implemented solutions and DSDM was conducted. The hypothesis was that the solutions would adhere to the criteria of the evaluation framework. Adherence to the criteria would indicate that the solutions would allow the software to be used with the different display shapes without any content items being made unfit for purpose. The majority of the results conformed to this hypothesis. However a small number showed that in some instances content items were being scaled too small for them to be manipulated or read from, therefore becoming unfit for purpose. This was identified as a deviation from the solution design in implementation due to the scale limits of content items not being adhered to. This was corrected and further observations on the use of the system indicated that the issue was resolved. Therefore it was shown that the solutions selected and implemented allowed the multi-touch software to become display shape independent. Figure 8.1 provides a full overview of how the issue of occlusion, solutions, DSDMs and the evaluation framework discussed in this work all interact.

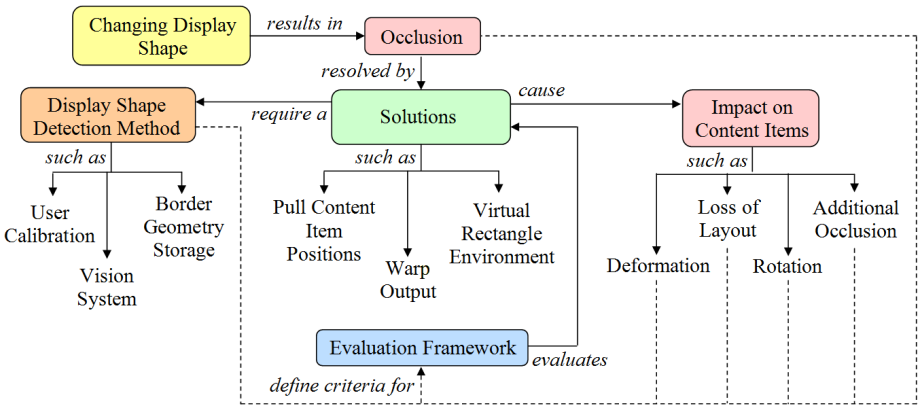


Figure 8.1: Hierarchy of solutions, DSDMs and evaluation framework.

8.3 Future Work

This research was successful in creating multi-touch software which is display shape independent. With new technologies making the use of different display shapes feasible the need for software to use these different display shapes will increase. Using the solutions outlined in this work could enable software to dynamically adapt its contents to different display shapes. Future work may involve the implementation of some of the solutions and DSDMs outlined but not implemented in this work. However the solutions outlined in this work may not be suitable for all software systems which could utilise different display shapes. Therefore future work may involve the creation of new solutions intended to fulfil the criteria outlined in this research. This would allow these new solutions to be compared with the existing solutions.

Different evaluations frameworks with new criteria could be created for identifying which solutions are suitable for software that greatly differs from the multi-touch systems discussed in this work. These new criteria could be derived from the same issue of occlusion and the potential impact of attempts to resolve it. The production and use of these scenario-specific criteria would benefit from being the focus of future research. Both the solutions detailed in this thesis and those potentially produced in future work do not need to be constrained to use with multi-touch software. The use of evaluation frameworks to select adequate solutions could allow for any system to make use of different display shapes with the right criteria. In the same way that many systems have become able of correctly displaying their output on different display resolutions, systems could become capable of producing an appropriate output for different display shapes.

Solutions for resolving initial occlusion need not be used exclusively for fitting content items into a display shape. They could be applied to virtual containers holding content items, for example the windows used in most modern visual operating systems. This would allow software to use windows of different shapes without its visual contents becoming unfit for purpose. Another virtual container that could benefit from the use of these solutions is the TableTrays multi-touch tool [75] discussed in Section 2.3.5.

Future work could also involve focusing on resolving other issues that occur when using different display shapes with software. As discussed in Chapter 3, occlusion caused by the placement of content items is the most prevalent issue in software when using different display shapes. However, as identified in Section 3.2.3, incorrect virtual display borders was another issue encountered when different display shapes were used. The number of systems in which this issue occurs is smaller than those in which the issue of occlusion is present. Despite this, it is still important for solutions to the different issues to be developed to allow these systems to become display shape independent. Resolving the issue of an inaccurate virtual display border could be resolved by using some of the work detailed in this thesis. The geometric data collected by a DSDM could be used to build a more accurate virtual border. Then when an item interacts with the border, for example bouncing off the border when flicked, the correct data can be used to calculate its correct reaction.

All the work discussed in this research is related to the use of two-dimensional displays. However the solutions produced as part of this work could be adapted to work with three-dimensional visual outputs. This would be beneficial for systems that utilise three-dimensional outputs such as those used for virtual and augmented reality interactions. Multi-touch systems which allow for some three-dimensional input, such as those which utilise deformable interfaces [28], could benefit from being able to adapt their visual contents to different three-dimensional shapes. As mentioned in Section 6.2, the SynergyNet MSF can produce three-dimensional applications. The waveform objects used to represent the display borders in SynergyNet are capable of producing three-dimensional shapes. In addition to this the implemented methods for managing the bounce behaviour of flicked content items will function correctly in three-dimensions as well. Therefore the SynergyNet MSF can correctly manage content items in different shaped, three-dimensional environments. This is an interesting potential avenue of future research in finding how these environments can affect the use of the system.

The ability for a MSF to be display shape independent opens a new area of research. The influence of different display shapes on the use of the system can now be investigated. This is especially relevant to direct touch interaction techniques where

the display shape will not only affect the system's output, but also it's input. Finding how different shapes, or features of shapes, influence a user's input or perception of the system's output would benefit the design of future software. Certain shapes may prove beneficial for specific scenarios or instances of system use whereas others could prove to be a disadvantage. With software able to support the different display shapes research into finding which shapes have what influence is possible. In addition to this the design of some future software may benefit from investigations into how different display shapes affect collaboration around multi-touch displays. These new areas of research are now unrestricted because software no longer needs to be constrained to a single display shape.

8.4 Conclusion

This research can be deemed successful in answering the research questions put forward in Section 1.2. The following list restates the original questions along side their answers.

1. **What issues occur when multi-touch software is used with different display shapes?**

   The issue of occlusion potentially occurs in any multi-touch software when it is used with different display shapes. The borders of a new display shape will cover areas of a software environment. Visual content items in this covered areas will be occluded and may be made unfit for purpose. This question was answered from observations made on the use of a prototype system. In Chapter 3 this issue is discussed at length.

2. **What evaluation framework can be used for judging the potential methods of allowing multi-touch software to use different display shapes?**

   A list of criteria relating to resolving the initial issue of occlusion, the correct management of the impact of potential solutions and the requirements of solutions was shown to be adequate in judging the potential methods of allowing multi-touch software to use different display shapes. These criteria

specifically related to content items' occlusion caused by the display border, occlusion caused by overlapping with other content items, rotation, layout and deformation. The criteria also related to the accuracy of a solution's software representation of the display shape. In Chapter 4 these criteria are outlined in more detail

3. **What methods can be used to allow multi-touch software to use different display shapes?**
   Several solutions to the initial occlusion of content items were proposed in this work. These were the VRE, PCIP and WO solutions. In addition to this a number of DSDMs, which the solutions require, were detailed. These were the VS, UC and DBS DSDMs. Ultimately the combination of the VRE and PCIP solutions using the DBS DSDM were successfully implemented to allow a MSF to use different display shapes. These potential solutions and DSDMs are detailed in Chapter 5.

4. **Can multi-touch software be adapted or created to be display shape independent?**
   Yes, as shown by this research's successful modification of a MSF to use different display shapes. The combination of the VRE and PCIP solutions using the DBS DSDM were implemented into the SynergyNet MSF [5]. The implementation of these solutions is discussed in Chapter 6. An evaluation of these implemented solutions took place which showed the solutions to be ultimately adequate. In Chapter 7 this evaluation is discussed.

The evaluation ultimately showed that the implemented solutions made SynergyNet capable of utilising the nine display shapes proposed. These shapes were chosen to be representative of likely future display shapes in multi-touch tables. This means that the SynergyNet MSF will be an excellent component for future multi-touch systems which are intended to use tabletop interfaces with different shapes. The results of the evaluation showed the solutions implemented had been successful in allowing SynergyNet to adapt to different display shapes without any loss of functionality.

All visual content items in the system remained fit for purpose after adaptation and all features of the MSF could be employed as expected. The implementation of the solutions was observed to be straight forward. The changes made to the MSF outside the implementation of these solutions were minimal. However some of these changes were made larger due to the restrictions of SynergyNet's structure. The orthogonal nature of visual content items caused issues relating to the collection of collision data which was necessary for some of the implemented changes. However the restrictions were overcome resulting in a MSF that works as expected with different display shapes.

The only problem highlighted by the evaluation results was the scaling of content items to extremes that left them too small to manipulate or read from. The cause of this problem was identified as a minor deviation in implementation from the solution designs. This highlights how developers must put proper thought and planning into the employment of any of the solutions produced by this work. The identification or creation of adequate solutions for a particular system is part of the process of allowing software to adapt to different display shapes which developers must consider carefully.

The growing momentum of developers incorporating Multi-touch interaction into their work provides an opportunity for the adoption of the use of different display shapes. As the development of multi-touch technology requires the rethinking of many existing design paradigms, the inclusion of new guidelines on system design can be considered. The inclusion of methods that allow a piece of software to become display shape independent could be part of these new guidelines. Multi-touch software can benefit from deviating from the typical designs of usual GUI features as shown by the works of Bailly et al. [68] and Flöring & Hesselmann [70] in producing new menu paradigms. Another example of this change from typical software GUI design is how multi-touch software intended for horizontal interfaces loses its dependence on assuming a specific orientation. The opportunity is now available for multi-touch software to lose its dependence on assuming a specific display shape. There is a growing need for different display as shapes as shown through the development of non-rectangular and deformable displays discussed in Section 2.4. Therefore it is important for software to become display shape independent.

At the beginning of this research the question was posed: 'What must be considered when designing multi-touch software for use on different shaped displays?' The work documented in this thesis has shown that the issue of occlusion which arises from using different display shapes must be considered. Also the impact of attempts to resolve this occlusion on visual content items must also be considered. The issues of visual content items' occlusion, orientation, the preservation of their layout and deformation must always be considered by a developer when designing solutions to occlusion. The selection and implementation of an adequate combination of occlusion solutions and DSDMs is required to allow multi-touch software to use different display shapes.

The solutions produced as part of this work can be used for the resolution of occlusion but developers must put serious consideration into the implementation of these solutions. The implementation of a solution can determine how much control the software and other developers can have over its influence. Further issues may arise from an incorrect implementation which deviates from a solutions design. This was demonstrated by the scaling problem discovered in the evaluation of the solutions implemented into the SynergyNet MSF detailed in Chapter 7. The deliverables from this work provide evidence that the software used in multi-touch systems can be adapted to utilise different display shapes. With the appropriate solutions implemented correctly a piece of software can place its content items appropriately within any display shape. With the proper placement content items remain fit for purpose and allow software to function correctly with different display shapes.

A well thought out selection of solutions and a correct implementation can also allow software to capitalise on the features of a display. This work ultimately demonstrates that it is possible to produce software that is display shape independent.

Appendix A

Generic Literature Survey Protocol

Systematic Review Protocol

Review Title: …

Reviewer Names: …

Contact Information: …

Date: …

Abstract

Background: …

Research Questions: The protocol identifies three research questions:

1. …

Method: The protocol describes the manner in which a systematic search of the literature is to be performed to identify candidate primary studies that can be used in synthesis to answer the research questions posed. The studies are selected using defined inclusions and exclusion criteria. Data from these selected sources will then be analysed by the reviewer looking for information relating to the research questions. Any relevant information or correlations between the selected sources will be noted and analysed.

Resources to Search:

….

Data Synthesis: The data will be tabulated and any collected information which is relevant to this survey shall be analysed.

Change Record

Date	Version	Change

Protocol for a Systematic Literature Review of <Survey Topic>

1. Background

….

2. Research Questions

The following research questions have been proposed:

Question 1: …

The following details of the population, intervention, outcomes, and experimental designs of interest to the review will form the basis for the construction of suitable search terms later in the protocol (Section 3.2).

Population: ...

Intervention: ...

Outcomes of relevance: …

Experimental design: …

3. Search Strategy

3.1. Strategy used to identify search terms for automated searches

The strategy used to construct search terms is as follows:

derive major terms from the questions by identifying the population, intervention and outcome;

identify alternative spellings and synonyms for major terms;

use the Boolean OR to incorporate alternative spellings and synonyms;

use the Boolean <and> to link the major terms from population, intervention and outcome.

Results for a)

Term A, *Term B*,….

Results for b)

1st Variation of Term A, 2nd Variation of Term A,…

1st Variation of Term B, 2nd Variation of Term B,…

…

Search Terms

Results for c) and d)

Preliminary Search Terms

("*Term A*" OR "*1st Variation of Term A*" OR "*2nd Variation of Term A*" OR …) AND ("*Term B*" OR "*1st Variation of Term B*" OR "*2nd Variation of Term B*" OR …) AND…

Generalised (post-preliminary search) Search Terms

("*Term A*" <or> "*1st Variation of Term A*" <or> "*2nd Variation of Term A*" <or> …) <and> ("*Term B*" <or> "*1st Variation of Term B*" <or> "*2nd Variation of Term B*" <or> …) <and>… <and> (year >= *earliest year* <and> year <= *latest year*)

Resources to be searched:

…

3.2. Search constraints and validation

…

3.3. Additional search criteria

The search strategy will be based primarily on a search of electronic databases. However, primary sources will all be checked for other relevant references.

3.4. Search documentation

The search will be documented in the format shown in Table 1a and …, which illustrates the search process documentation for the …, … and … online digital libraries. As each of the search engines has a different interface, a preliminary search indicated that the search terms presented in section 3.1 would have to be modified to adapt to the requirements of each search engine. The adapted search terms for the

review, based upon the experience of the preliminary search, are presented in Table 1a and ….

Table 1a - Search process documentation for …

Data Source	Documentation
	Name of Source: … **Search strategy:** … **Search characteristics for each source:** ( ) allows for nested Boolean searches ( ) allows only for simple Boolean searches ( ) indexes full-text ( ) indexes abstract ( ) indexes title ( ) indexes literature written in the following languages: English. **Date and time of searches:** dd/mm/yyyy hh:mm (Preliminary search) dd/mm/yyyy hh:mm **Years covered by search for each database:** … to ….

3.5. Search result management

Primary source references that are potentially relevant will be stored in a Reference Manager database.

4. Study Selection Criteria and Procedures

4.1. Inclusion Criteria

The studies that are of interest to this review should be concerned with …. Therefore, the inclusion criteria are as follows:

- …

4.2. Exclusion Criteria

The following studies will be excluded from this review:

- …

4.3. Selecting Primary Sources

Initial selection of primary sources will be based on a review of title, keywords, and abstract. It is intended to exclude only those primary sources that appear completely irrelevant.

Researcher responsible: ….

Full copies of all primary sources not excluded in the initial selection process will be obtained. These will be reviewed against the inclusion/exclusion criteria.

Researcher responsible: ….

If there is any uncertainty regarding the inclusion or exclusion of a particular primary study then the source will be sent to another reviewer with knowledge relating to the subject of this report.

Researcher responsible: …

The record for each primary source in the Reference Manager Database will be updated to specify whether or not the primary source has been included in the review, and the reason for its inclusion/exclusion.

Researchers responsible: ….

5. Study Quality Assessment

The study quality checklist comprises the following questions:

- Is it clear how the supporting evidence was collected and verified?

- Is the argument for or against a proposed development strongly justified?
- Are the results or other data used in the source available and complete?
- Is the physical design of the multi-touch interface used or that the source relates to made clear?
- Do the findings of the source relate to the design of the physical and software aspects of multi-touch interfaces?

Each question will be answered "Yes/No/Partially", and for each question that is answered partially there will be an attempt to summarise the missing information. An attempt will be made to contact the authors of the papers to seek the missing information.

The study quality checklist will be used as follows:

- If it is not clear how the supporting evidence was collected or verified the source will be identified as *questionable.*
- If the argument for or against a proposed development is not strongly supported then a dialogue between the researchers will take place on whether to include the source in the database or to reference it in the final literature survey.
- If a source's results or data are unavailable the source will be identified as *questionable.* If a source's results or data are incomplete the source will be identified as *incomplete.*
- If a source does not make clear what the physical design of the multi-touch interface used is then a dialogue will take place between the researchers on whether to include or disregarded the source. If the source is included it may be decided that due to this lack of information the source should be identified as *questionable* or *incomplete*.
- If the findings are not relevant to the development of multi-touch tables or software then the source shall be disregarded.

Studies identified as questionable will be used for sensitivity analysis, although initially they will be excluded from the aggregation process. The results of the questionable studies will be investigated to see if they have a significant impact on the interpretation of the systematic review. For example, if including the questionable studies would change the overall conclusions, the results of the systematic review would be considered inconclusive.

Studies identified as incomplete will be included in the aggregation process as far as possible. The results of the incomplete studies will be investigated to see if they have a significant impact on the interpretation of the systematic review. Any systematic difference between the results of complete and incomplete studies will cast doubts on the validity of the systematic review.

Descriptions of all of the questionable and incomplete studies will be appended onto Table 2.

6. Data Extraction Strategy

6.1. Primary Study Data

The aim of the study is to address the three questions defined in Section 2. In order to address question 1 the following information from each primary study will be recorded:

- …

In order to address question 2 the following information from each primary study will be recorded:

- …

The form in Table 2 will be used to record the data extracted from the selected primary studies. The form should be completed electronically.

Table 2 - Data to be extracted

Reference Number	**Value**
1. Reviewer Name	Name of the reviewer conducting the data extraction
2. Title	Title of primary source material.

3. Reference	The Referencing Application record identifier that contains the primary source reference data. (I.e. a Reference Manager ID or reference to locate the paper within other bibliographic software)
4. Database	The name of the database where the primary study was found. (e.g. ACM Portal, IEEE Xplore, etc)
5. …	….

6.2. Data Extraction Process

Initially, each primary study will be read by one reviewer. For each paper, the reviewer will be responsible for extracting the data and checking that the data has been correctly extracted. If the reviewer has any concerns about the data extracted one or more secondary reviewers depending on availability will be contacted for discussion on the concerns and paper(s) the concerns arise from.
Primary reviewer: …
Secondary reviewers: …

6.3. Data Storage

The verified extracted data for each paper will be held…

7. Data Synthesis

The data extraction forms will be examined for any missing data. If any data from a paper or referred to in a paper is missing and cannot be obtained from other sources (for example, the authors of the study) then the word *incomplete* will be used. If the paper makes any claims or puts forward any theories which are not reliably backed up the study will be considered *questionable*.

The papers will be summarised with a bullet pointed list of points of interest relating to claims, observations and data which may be of some interest for further research. The points will be ordered in the order they appear in the paper and annotated with which

section and page of the paper they appear on. Those points considered to have a strong correlation to the subject of this review will be highlighted in an italic font. This list of bullet points will be appended to the form in Table 2 in section 6.1.

8. Validation of the review protocol

A preliminary draft protocol will be sent for comments to …

After updating the protocol to deal with initial comments, a selection of existing papers will be reviewed by the main researcher …. in order to pilot the review and to validate the extraction process and the data extraction form (and also to provide a learning opportunity for the reviewer).

The protocol will be revised as a result of the pilot activities by completing some of the data forms and data summary tables. This will constitute the final draft of the protocol.

The final draft will be reviewed by Dr. …. After any corrections, the protocol will be signed-off as complete by … and the systematic review will move formally into the execution phase

9. Potential Conflict of Interest

….

10. Review timetable.

Task	**Date**

Divergences

Any divergences from the protocol, that have occurred while performing this study, have been recorded:

- <Date>: <Occurrence>

Bibliography

[1] One North East. Welcome to One North East, *Regional Development Agency (RDA) for North East England*. http://www.onenortheast.co.uk, 2010. [Accessed 12 August 2010].

[2] Ness Furniture. Welcome to Ness Furniture, *Ness Furniture*. http://www.nessfurniture.co.uk, 2010. [Accessed 12 August 2010].

[3] One North East. North East Studentships, *Regional Development Agency (RDA) for North East England*. http://www.northeaststudentships.com, 2010. [Accessed 12 August 2010].

[4] R. J. K. Jacob, A. Girouard, L. M. Hirshfield, M. S. Horn, O. Shaer, E. T. Solovey, and J. Zigelbaum. Reality-based interaction: a framework for post-wimp interfaces. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 201–210, New York, NY, USA, 2008. ACM.

[5] A. Hatch et al. SynergySpace, *Google Code*. http://code.google.com/p/synergyspace, 2009. [Accessed 12 August 2010].

[6] M. Turner and S. Charters. Protocol for a Systematic Literature Review of the Technology Acceptance Model and its Predictive Capabilities, *Evidence-Based Software Engineering*. http://www.dur.ac.uk/ebse/resources/studies/protocol/TAM-protocol-v3.5.pdf, 2007. [Accessed 12 August 2010].

[7] B. Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33, 2004.

[8] B. Buxton. Multi-Touch Systems that I Have Known and Loved, *Microsoft Research*. http://www.billbuxton.com/multitouchOverview.html, 2009. [Accessed 03 November 2009].

[9] S. D. Scott and S. Carpendale. Guest Editors' Introduction: Interacting with Digital Tabletops. *Computer Graphics and Applications, IEEE*, 26(5):24–27, 2006.

[10] J. Schöning, P. Brandl, F. Daiber, F. Echtler, O.r Hilliges, J. Hook, M. Löchtefeld, N. Motamedi, L. Muller, P. Olivier, T. Roth, and U. Zadow. Multi-Touch Surfaces: A Technical Guide. Technical report, University of Münste, October 2008.

[11] J. Rekimoto. Organic Interaction Technologies: From Stone to Skin. *Commun. ACM*, 51(6):38–44, 2008.

[12] P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM.

[13] X. Wang. Metaphor of Interaction Design in Multimedia Information Terminals Design. In *Computer-Aided Industrial Design and Conceptual Design, 2008. CAID/CD 2008. 9th International Conference on*, pages 268–270, 2008.

[14] T. Selker. Touching the Future. *Commun. ACM*, 51(12):14–16, 2008.

[15] N. Matsushita and J. Rekimoto. HoloWall: Designing a Finger, Hand, Body, and Object Sensitive Wall. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 209–210, Banff, Alberta, Canada, 1997. ACM.

[16] J. Rekimoto. SmartSkin: an Infrastructure for Freehand Manipulation on Interactive Surfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, pages 113–120, Minneapolis, Minnesota, USA, 2002. ACM.

[17] S. Izadi, S. Hodges, A. Butler, A. Rrustemi, and B. Buxton. ThinSight: Integrated Optical Multi-touch Sensing Through Thin Form-factor Displays. In *Proceedings of the 2007 workshop on Emerging displays technologies: images and beyond: the future of displays and interacton*, page 6, San Diego, California, 2007. ACM.

[18] I. Rosenberg and K. Perlin. The UnMousePad: an Interpolating Multi-touch Force-sensing Input Pad. In *ACM SIGGRAPH 2009 papers*, pages 1–9, New Orleans, Louisiana, 2009. ACM.

[19] R. Hofer, D. Naeff, and A. Kunz. FLATIR: FTIR Multi-touch Detection on a Discrete Distributed Sensor Array. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 317–322, Cambridge, United Kingdom, 2009. ACM.

[20] C. B. Bose and I. Amir. Design of fiducials for accurate registration using machine vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(12):1196–1200, 1990.

[21] A. D. Wilson. TouchLight: an Imaging Touch Screen and Display for Gesture-based Interaction. In *Proceedings of the 6th international conference on Multimodal interfaces*, pages 69–76, State College, PA, USA, 2004. ACM.

[22] D. Jackson, T. Bartindale, and P. Olivier. FiberBoard: compact multi-touch display using channeled light. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, page 1. ACM, 2009.

[23] Evoluce. Multitouch MIM, *Evoluce Multitouch Software*. `http://www.evoluce.com/index.php?id=13`, 2009. [Accessed 12 August 2010].

[24] S. Izadi, A. Butler, S. Hodges, D. West, M. Hall, B. Buxton, and M. Molloy. Experiences with Building a Thin Form-factor Touch and Tangible Tabletop. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 181–184, 2008.

[25] A. Mazalek, M. Reynolds, and G. Davenport. TViews: An Extensible Architecture for Multiuser Digital Media Tables. *Computer Graphics and Applications, IEEE*, 26(5):47–55, 2006.

[26] A. Mazalek, M. Reynolds, and G. Davenport. The TViews Table in the Home. In *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on*, pages 52–59, 2007.

[27] L. Vlaming, J. Smit, and T. Isenberg. Presenting Using Two-handed Interaction in Open Space. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 29–32, 2008.

[28] Y. Watanabe, A. Cassinelli, T. Komuro, and M. Ishikawa. The Deformable Workspace: A Membrane between Real and Virtual Space. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 145–152, 2008.

[29] O. Hilliges, D. Kim, and S. Izadi. Creating Malleable Interactive Surfaces using Liquid Displacement Sensing. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 157–160, 2008.

[30] M. Weiss, J. Wagner, Yv. Jansen, R. Jennings, R. Khoshabeh, J. D. Hollan, and J. Borchers. SLAP Widgets: Bridging the Gap Between Virtual and Physical Controls on Tabletops. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 481–490, Boston, MA, USA, 2009. ACM.

[31] D. A. Bowman. *3D user interfaces: theory and practice*, chapter 10. Addison-Wesley, 2005.

[32] N. Marquardt, M. Nacenta, J. Young, S. Carpendale, S. Greenberg, and E. Sharlin. The Haptic Tabletop Puck: Tactile Feedback for Interactive Tabletops. In *ITS '09: Proceedings of ACM International Conference on Interactive Tabletops and Surfaces*, Banff, Alberta, Canada, 2009.

[33] M. Hancock, O. Hilliges, C. Collins, D. Baur, and S. Carpendale. Exploring Tangible and Direct Touch Interfaces for Manipulating 2D and 3D Information on a Digital Table. In *ITS '09: Proceedings of ACM International Conference on Interactive Tabletops and Surfaces*, Banff, Alberta, Canada, 2009.

[34] F. Echtler and G. Klinker. A Multitouch Software Architecture. In *Proceedings of the 5th Nordic conference on Human-Computer Interaction: building bridges*, pages 463–466, Lund, Sweden, 2008. ACM.

[35] J. Rivière, C. Kervégant, E. Orvain, and N. Dittlo. CubTile: a Multi-touch Cubic Interface. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 69–72, Bordeaux, France, 2008. ACM.

[36] A. Butler, S. Izadi, and S. Hodges. SideSight: Multi-"touch" Interaction Around Small Devices. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 201–204, Monterey, CA, USA, 2008. ACM.

[37] J. K. Parker, R. L. Mandryk, and K. M. Inkpen. Integrating Point and Touch for Interaction with Digital Tabletop Displays. *Computer Graphics and Applications, IEEE*, 26(5):28–35, 2006.

[38] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. TUIO: A protocol for table-top tangible user interfaces. In *Proc. of the The 6th Intl Workshop on Gesture in Human-Computer Interaction and Simulation*. Citeseer, 2005.

[39] P. Varcholik, J. J. Laviola, and D. Nicholson. TACTUS: A Hardware and Software Testbed for Research in Multi-Touch Interaction. In J. A. Jacko, editor, *HCI (2)*, volume 5611 of *Lecture Notes in Computer Science*, pages 523–532. Springer, 2009.

[40] F. Echtler. TISCH - Tangible Interactive Surfaces for Collaboration between Humans, *Sourceforge*. `http://tisch.sourceforge.net`, 2009. [Accessed 18 November 2009].

[41] F. Echtler, M. Huber, and G. Klinker. Shadow Tracking on Multi-touch Tables. In *Proceedings of the working conference on Advanced visual interfaces*, pages 388–391, Napoli, Italy, 2008. ACM.

[42] P. Hutterer and B. H. Thomas. Enabling Co-located Ad-hoc Collaboration on Shared Displays. In *Proceedings of the ninth conference on Australasian user interface - Volume 76*, pages 43–50, Wollongong, Australia, 2008. Australian Computer Society, Inc.

[43] D. Wallin et al. Touchlib - Home, *Natural User Interface Group*. `http://www.nuigroup.com/touchlib`, 2009. [Accessed 19 November 2009].

[44] T. Cuypers, J. Schneider, J. Taelman, K. Luyten, and P. Bekaert. Eunomia: Toward a Framework for Multi-touch Information Displays in Public Spaces. In *Proceedings of the 22nd British CHI Group Annual Conference on HCI 2008: People and Computers XXII: Culture, Creativity, Interaction - Volume 2*, pages 31–34, Liverpool, United Kingdom, 2008. British Computer Society.

[45] G. Kaindl. Touché Multi-touch Framework, *gkaindl.com*. `http://gkaindl.com/software/touche`, 2008. [Accessed 03 November 2009].

[46] J. Borchers, S. Hafenger, and M. Weiss. The Media Computing Group : Multi-Touch Framework, *Aachen University*. `http://hci.rwth-aachen.de/multitouch`, 2008. [Accessed 03 November 2009].

[47] Natural User Interface Technologies. Snowflake Suite, *NUITEQ*. `http://natural-ui.com/products/snowflakesuite.php`, 2009. [Accessed 03 November 2009].

[48] J. Spadaccini. GestureWorks - Flash Multitouch Made Easy, *Ideum*. `http://gestureworks.com`, 2009. [Accessed 03 November 2009].

[49] P. Varcholik. Bespoke Software Multi-Touch, *Paul Varcholiks Software Development Blog*. `http://www.bespokesoftware.org/wordpress/?page_id=41`, 2008. [Accessed 05 November 2009].

[50] T. Hansen et al. PyMT: Python Multitouch, *TX Zone*. [Online]. Available from: `http://pymt.txzone.net`, 2009. [Accessed 03 November 2009].

[51] M. Kaltenbrunner and R. Bencina. reacTIVision: a Computer-vision Framework for Table-based Tangible Interaction. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 69–74, Baton Rouge, Louisiana, 2007. ACM.

[52] A. Ramos et al. Community Core Vision, *Natural User Interface Group*. `http://ccv.nuigroup.com`, 2009. [Accessed 24 November 2009].

[53] T. B. Downing. *Java RMI: remote method invocation*. IDG Books Worldwide, Inc. Foster City, CA, USA, 1998.

[54] S. Higgins, J. Elliot, . L. Burd, A. Hatch, P. Kyaw, and E. Mercier. Multi-touch Desks for Learning: Developing Pedagogy Through Technology. In *European Learning and Instruction Conference*, 2009.

[55] M. Powell et al. Java Monkey Engine User Guide, *Java Monkey Engine*. `http://www.jmonkeyengine.com`, 2007. [Accessed 12 August 2010].

[56] E. L. Burd et al. Multi-touch in Education: SynergyNet, *Technology Enhanced Learning Group - Durham University*. `http://tel.dur.ac.uk/synergynet`, 2009. [Accessed 12 August 2010].

[57] F. Verier, G. Besacier, C Forlines, C. Shen, and D. Wigdor. DiamondSpin Tabletop Toolkit Project, *Mitsubishi Electric Research Laboratories*. `http://diamondspin.free.fr`, 2006. [Accessed 08 December 2009].

[58] A. Piper and J. D. Hollan. Tabletop Displays for Small Group Study: Affordances of Paper and Digital Materials. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1227–1236, Boston, MA, USA, 2009. ACM.

[59] J. Rick, A. Harris, P. Marshall, R. Fleck, N. Yuill, and Y. Rogers. Children Designing Together on a Multi-touch Tabletop: an Analysis of Spatial Orientation and User Interactions. In *Proceedings of the 8th International Conference on Interaction Design and Children*, pages 106–114, Como, Italy, 2009. ACM.

[60] O. Hilliges, D. Baur, and A. Butz. Photohelix: Browsing, Sorting and Sharing Digital Photo Collections. In *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on*, pages 87–94, 2007.

[61] D. Vanacken, A. Demeure, K. Luyten, and K. Coninx. Ghosts in the Interface: Meta-user Interface Visualizations as Guides for Multi-touch Interaction. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 81–84, 2008.

[62] C. Shen, K. Ryall, C. Forlines, A. Esenther, F.D. Vernier, K. Everitt, M. Wu, D. Wigdor, M.R. Morris, M. Hancock, and E. Tse. Informing the Design of Direct-Touch Tabletops. *Computer Graphics and Applications, IEEE*, 26(5):36–46, 2006.

[63] T. Moscovich and J. F. Hughes. Indirect Mappings of Multi-touch Input Using One and Two Hands. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1275–1284, Florence, Italy, 2008. ACM.

[64] K. Kin, M. Agrawala, and T. DeRose. Determining the Benefits of Direct-touch, Bimanual, and Multifinger Input on a Multitouch Workstation. In *Proceedings of Graphics Interface 2009*, pages 119–124, Kelowna, British Columbia, Canada, 2009. Canadian Information Processing Society.

[65] J. Schöning, F. Daiber, A. Krüger, and M. Rohs. Using Hands and Feet to Navigate and Manipulate Spatial Data. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4663–4668, Boston, MA, USA, 2009. ACM.

[66] S. Taylor, S. Izadi, D. Kirk, R. Harper, and A. Garcia-Mendoza. Turning the Tables: an Interactive Surface for VJing. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1251–1254, Boston, MA, USA, 2009. ACM.

[67] R. Dixon and T. Sherwood. Whiteboards that Compute: A Workload Analysis. In *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on*, pages 69–78, 2008.

[68] G. Bailly, A. Demeure, E. Lecolinet, and L. Nigay. MultiTouch Menu (MTM). In *Proceedings of the 20th International Conference of the Association Francophone d'Interaction Homme-Machine*, pages 165–168, Metz, France, 2008. ACM.

[69] H. Benko, T. S Saponas, D. Morris, and D. Tan. Enhancing Input On and Above the Interactive Surface with Muscle Sensing. In *ITS '09: Proceedings of ACM International Conference on Interactive Tabletops and Surfaces*, Banff, Alberta, Canada, 2009.

[70] T. Hesselmann, S. Flöring, and M. Schmitt. Stacked half-pie menus: navigating nested menus on interactive tabletops. In *ITS '09: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 173–180, New York, NY, USA, 2009. ACM.

[71] J. Rick and Y. Rogers. From DigiQuilt to DigiTile: Adapting Educational Technology to a Multi-touch Table. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 73–80, 2008.

[72] Y. Ghanam, X. Wang, and F. Maurer. Utilizing Digital Tabletops in Collocated Agile Planning Meetings. In *Agile, 2008. AGILE '08. Conference*, pages 51–62, 2008.

[73] D. Wigdor, G. Penn, K. Ryall, A. Esenther, and C. Shen. Living with a Tabletop: Analysis and Observations of Long Term Office Use of a Multi-Touch Table. In *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'07)*, pages 60–67, Newport, RI, USA, 2007.

[74] C. Lo and R. Khoshabeh. TUIO Mouse Release, *NUI Group*. `http://nuigroup.com/forums/viewthread/3447`, 2008. [Accessed 23 November 2009].

[75] D. Pinelle, T. Stach, and C. Gutwin. TableTrays: Temporary, Reconfigurable Work Surfaces for Tabletop Groupware. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 41–48, 2008.

[76] Toshiba Mobile Display. New Circular LTPS TFT LCD, *Toshiba Matsushita*. `http://www3.toshiba.co.jp/tm_dsp/press/2007/07-10-17.html`, 2007. [Accessed 12 August 2010].

[77] E. Murtazin and O. Kononosov. First look at Motorola Aura, *Mobile-Review*. `http://www.mobile-review.com/review/motorola-motoaura-en.shtml`, 2008. [Accessed 12 August 2010].

[78] S. C Deane and S. J Battersby. Non Rectangular Display Device, June 2003. US Patent App. 10/517,286.

[79] S. J Battersby. Non-Rectangular Display Device, June 2005. US Patent App. 11/570,241.

[80] A. G Knapp and S. J Battersby. Non-Rectangular Display Device, May 2005. US Patent App. 11/569,447.

[81] M. Coelho, I. Poupyrev, S. Sadi, R.l Vertegaal, J. Berzowska, L. Buechley, P. Maes, and N. Oxman. Programming Reality: from Transitive Materials to Organic User Interfaces. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4759–4762, Boston, MA, USA, 2009. ACM.

[82] D. Holman and R. Vertegaal. Organic User Interfaces: Designing Computers in any Way, Shape, or Form. *Communications of the ACM*, 51(6):48–55, 2008.

[83] J. C. Lee, S. E. Hudson, and E. Tse. Foldable Interactive Displays. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 287–290, Monterey, CA, USA, 2008. ACM.

[84] I. Poupyrev, H. Newton-Dunn, and O. Bau. D20: Interaction With Multifaceted Display Devices. In *CHI '06 extended abstracts on Human factors in computing systems*, pages 1241–1246, Montral, Qubec, Canada, 2006. ACM.

[85] H. Benko, A. D. Wilson, and R. Balakrishnan. Sphere: multi-touch interactions on a spherical display. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 77–86, New York, NY, USA, 2008. ACM.

[86] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, September 1991.

[87] A. Greenfield. *Everyware: The Dawning Age of Ubiquitous Computing*. Peachpit Press, Berkeley, CA, USA, 2006.

[88] M. Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36:75–84, July 1993.

[89] S. Holloway and C. Julien. The case for end-user programming of ubiquitous computing environments. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, FoSER '10, pages 167–172, New York, NY, USA, 2010. ACM.

[90] A. van Dam. User interfaces: disappearing, dissolving, and evolving. *Commun. ACM*, 44:50–52, March 2001.

[91] J. Steimle, M. Khalilbeigi, M. Mühlhäuser, and J. D. Hollan. Physical and digital media usage patterns on interactive tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 167–176, New York, NY, USA, 2010. ACM.

[92] G. Abowd. Software engineering and programming language considerations for ubiquitous computing. *ACM Comput. Surv.*, 28, December 1996.

[93] K. Hinckley, J. Pierce, M. Sinclair E., and E. Horvitz. Sensing techniques for mobile interaction. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, UIST '00, pages 91–100, New York, NY, USA, 2000. ACM.

[94] D. Kammer, J. Wojdziak, M. Keck, R. Groh, and S. Taranko. Towards a formalization of multi-touch gestures. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 49–58, New York, NY, USA, 2010. ACM.

[95] K. Akeley and T. Jermoluk. High-performance polygon rendering. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 239–246, New York, NY, USA, 1988. ACM.

[96] K. Packard. X Nonrectangular Window, *MIT X Consortium*. `http://www.xfree86.org/current/shapelib.pdf`, 1989. [Accessed 12 August 2010].

[97] E. Bahceci et al. Compiz Window Manager, *Compiz*. `http://www.compiz.org`, 2010. [Accessed 18 February 2010].

[98] P. Dietz, R. Raskar, S. Booth, J. Baar, K. Wittenburg, and B. Knep. Multi-projectors and Implicit Interaction in Persuasive Public Displays. In *Proceedings of the working conference on Advanced visual interfaces*, pages 209–217, Gallipoli, Italy, 2004. ACM.

[99] J. W. Shirley. An early experimental determination of snell's law. *American Journal of Physics*, 19(9):507–508, 1951.

[100] D. Vandevoorde. The Maximal Rectangle Problem, *Dr. Dobb's word of software development*. `http://www.drdobbs.com/184410529`, 1998. [Accessed 12 August 2010].

[101] H. Alt, D. Hsu, and J. Snoeyink. Computing the largest inscribed isothetic rectangle. In *Proceedings of 1995 Canadian Conference on Computational Geometry (CCCG)*, pages 67–72. Citeseer, 1995.

[102] G. T. Toussaint. Computing largest empty circles with location constraints. *International Journal of Parallel Programming*, 12(5):347–358, 1983.

[103] B. Chazelle, R. Drysdale, and D. Lee. Computing the largest empty rectangle. *STACS 84*, pages 43–54, 1984.

[104] J. F. Canny. Finding Edges and Lines in Images. *Massachusetts Inst. of Tech. Report*, 1983.

[105] S. Preibisch, S. Saalfeld, and P. Tomancak. Globally optimal stitching of tiled 3d microscopic image acquisitions. *Bioinformatics*, 25(11):1463–1465, 2009.

[106] O. Bossert. A robust method for alignment of histological images. *Comput. Methods Prog. Biomed.*, 78(1):35–38, 2005.

[107] C. Rocchini, P. Cignoni, C. Montani, P. Pingi, and R. Scopigno. A low cost 3D scanner based on structured light. In *Computer Graphics Forum*, volume 20, pages 299–308, 2001.

[108] SMART. Smart Boards, *SMART Technologies ULC*. `http://www.smarttech.com/us/Solutions/Education+Solutions/Products+for+education/Interactive+whiteboards+and+displays/SMART+Board+interactive+whiteboards`, 2008. [Accessed 12 August 2010].

[109] H. Chen, R. Sukthankar, G. Wallace, and K.s Li. Scalable alignment of large-format multi-projector displays using camera homography trees. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 339–346, Washington, DC, USA, 2002. IEEE Computer Society.

[110] M. Karam and M. C. Schraefel. Investigating user tolerance for errors in vision-enabled gesture-based interactions. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 225–232, New York, NY, USA, 2006. ACM.

[111] E. L. Burd et al. Technology Enhanced Learning Research Group, *Durham University*. `http://tel.dur.ac.uk`, 2009. [Accessed 12 August 2010].

[112] J. Walnes et al. About X-Stream, *X-Stream*. `http://xstream.codehaus.org/`, 2008. [Accessed 12 August 2010].

[113] H. M. K. Koskinen, J. O. Laarni, and P. M. Honkamaa. Hands-on the process control: users preferences and associations on hand movements. In *CHI'08 extended abstracts on Human factors in computing systems*, pages 3063–3068. ACM, 2008.

[114] B. Crowder. Blender. *Linux J.*, page 7, 1999.

[115] D. Sud. Computing the largest orthogonal rectangle in a convex polygon, *McGill University*. `http://cgm.cs.mcgill.ca/~athens/cs507/Projects/2003/DanielSud/`, 2003. [Accessed 12 August 2010].

[116] H. S. M. Coxeter. Affinely regular polygons. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 34, pages 38–58. Springer, 1969.