

# Augmented Virtuality with iPad

**Adam Lewis**

Student at Lancaster University  
Lancaster, United Kingdom  
2009-2013  
a.lewis3@lancaster.ac.uk

**Paul Coulton**

School of Computing & Communications  
Infolab21, Lancaster University  
Lancaster, LA1 4WA  
p.coulton@lancaster.ac.uk

## Abstract

Augmented reality is an area of research that is currently enjoying a large amount of interest. With constant developments in hardware, such as the Microsoft Kinect [1], many projects are being developed that augment virtual content onto the physical world. Observing Paul Milgram's mixed reality continuum [2], we directed our research within this paper to explore the inverse of augmented reality, augmented virtuality. Within Milgram's continuum, the levels of mixed reality vary. "Probably the best known of these is Augmented Reality (AR), which refers to all cases in which the display of an otherwise real environment is augmented by means of virtual (computer graphic) objects. The converse case on the virtuality continuum is therefore Augmented Virtuality (AV)." [3]. We hoped to create a system on the iPad (or any of Apple's other iOS touch-screen devices) that would enable us to use physical objects as an extension of the standard input method. Doing so would allow us to augment the virtual world within a device through the use of objects which are more natural for users to utilise. To do this, conductive materials were used to create purpose built objects that would be able to interact with an application through the touch screen of a device.

**Categories and Subject Descriptors** A.0 [General]: Exploring the potential for object to surface recognition systems on iOS touch screen devices.

**General Terms** Design, Experimentation, Theory.

**Keywords** Augmented Virtuality; iPad; iOS; Alternative Input Methods; Human Computer Interaction;

## 1. Introduction

The concept of augmented virtuality is primarily based on taking physical objects from reality and integrating them into a virtual world. These integrated objects are then used to interact with the virtual world in some way to alter its content. An example of this having been done is the Windows on the World project [4]. From the research performed in this paper, a system was created that could augment the virtual world by capturing objects in reality and applying their textures to 3D models. AV technology has also been used in popular culture for some time and examples of this having been done are Sony's Eyetoy [5] and the cancelled

children's show "Nightmare" [6]. Both of these systems controlled game-play elements by incorporating the user from the physical world into a virtual world. The user would then interact with the content of this world via the system to progress in the games. From the example systems, it is clear that the key components of any augmented virtuality system is the ability to integrate real world objects into a virtual world and allowing them to interact with the virtual world they are integrated into. These components of AV systems were major requirements for our research into creating an augmented virtuality system on the iPad. The system design would have to allow for real world objects to become a part of the virtual world and let them interact with the virtual environment of a device.

Interactions with virtual environments are performed on a daily basis world wide by users to complete simple and complex tasks. To perform interactions with a system, we use input devices such as mice and keyboards to alter the state of the virtual environment. We use such devices frequently and rely on them to enable us to interact with computing systems. As users we use these devices based on how suited they are to the required task. The more naturally a device allows us to complete a task, the more useful it is to us. For most tasks performed within the virtual environment, keyboards and mice have proved the most versatile and utile. Some examples of common tasks we perform with these devices, but not limited to, are playing music, the use of productivity tools, web browsing and gaming. The variety of purposes we use computer systems for is limited only by human creativity and as new innovations are implemented on systems, new requirements are often discovered with them.

The solutions designed to meet the requirements of these new systems however can sometimes be difficult to interact with. This is usually due to restrictions in the hardware and software available making tasks more complex for a user to perform. A large amount of research to solve this and increase the quality of user interaction with computing systems has been performed in the area of HCI (Human Computer Interaction) and has had some strong success stories [7]. From the research into HCI, new hardware and software based solutions have been designed to improve the ease of which we utilise the user interfaces available to us.

Our research was motivated by this concept of making user interfaces more natural to users because they are a key component to interacting with any computer system. Oxford dictionary defines "user interface" as "the means by which the user and a computer system interact, in particular the use of input devices and software." [8]. Another definition of this term is "Any designed interaction between man and machine is a user

interface.” [9]. Having a good user interface design is key to making a system useful.

### 1.1 Evolution of Input Devices

There are multiple ways in which we as users interact with computers, each with their own strengths and weaknesses. To improve the quality of our interactions with computers, we occasionally design new devices or redesign existing ones. We do this to create new devices that will be better suited to a required task and a well designed device makes our interactions more natural, intuitive and improves computer accessibility.

For example, graphics tablets are considerably favoured by artists over mice to create digital artwork [10], yet both were designed to function as pointing devices. This is a primary example of how devices can evolve individually to meet different requirements based on their individual strengths and weaknesses. In contrast to this approach, some research has been targeted towards improving current input devices to evolve them and extend their functionality. An example of this occurring recently is research undertaken at Lancaster University to evolve the functionality of the keyboard [11]. In this work, research was pioneered to create a more dynamic keyboard that would become a natural extension to the graphical user interface of a computer.

We decided to take a similar approach in our research by rethinking how we interact with touch screens. The technology behind a touch screen is implementation dependent but commonly conform to the use of fingers or a stylus as the main method of allowing user input. Generally speaking, there are four types of touch screens that are commonly available, each a combination of single/multi-touch and capacitive/resistive screens. Multi-touch screens in comparison to single-touch screens allow for multiple simultaneous touch points and both can be implemented on capacitive or resistive screens.

### 1.2 Touch Screen Background

“Resistive touch screens consist of a glass or acrylic panel that is coated with electrically conductive and resistive layers made with indium tin oxide (ITO) ... The thin layers are separated by invisible spacers.” [12]. These layers have electrical current run through them and when pressure is applied by an object, the layers push together and touch. When this occurs, the flow of electrical current changes and the change in current is then processed by a controller. The controller provides the device with a touch point so that the software can respond appropriately to the touch event.

The other technology commonly used by touch devices, capacitive screens, work slightly differently to resistive screens however. Screens such as the iPhone's, use mutual or self capacitance [13] to detect the user's touch by exploiting the electrical properties of the human body. When the body comes into near contact with an electrical field, electrons within it rearrange, altering the original electronic field in the process of doing so. It is this change that is detected by capacitive touch screens as touch events [14] and processed by controllers. Capacitive technology is generally seen as an improvement over resistive screens due to the screens requiring less pressure for input detection and devices not being prone to accidental triggering of touch events by objects.

Touch screen computing devices based on these technologies are becoming ever more popular in the public domain, particularly in the mobile computing environment. Evidence of this can be seen with touch screen smart phones in Q4 2009, taking 55% of the market [15]. Further evidence can also be seen with companies such as Dell incorporating touch-screen technology

into their Inspiron Duo laptops [16], Google into the Android Tablet PC [17] and Apple into the iPad [18]. This trend of touch screens being accepted by the public as the standard method of interaction with mobile devices, has lead us to ask whether we can extend upon how we interact with basic touch screen devices and improve how we utilise them.

## 2. Goals and Design Choices

To improve the capabilities of mobile device touch screens, we chose to research the possibility of object-to-surface recognition within the context of mobile devices. Object-to-surface recognition is a technology that has recently been used by Microsoft in their Surface 2.0 project [19]. Surface 2.0 is a user-touch and object-touch controlled system designed to cater towards the creation of new experiences for users. An example of such an experience is Sonicspree [20], a quiz game that tests players on their musical knowledge by utilising both touch points and dice as inputs to the game. This is done by enabling the system to capture objects at the pixel level through the use of PixelSense [21], a technology which integrates infra red cameras into display pixels, and computer vision techniques.

The largest con of Surface 2.0 however is the 4 inch depth of the hardware required to implement the system [22]. We felt that creating a similar system to Surface 2.0 to function on mobile touch screen devices would be more beneficial to the general public. The implementation of a similar system on a mobile platform, such as tablet PCs, would allow for new innovations that wouldn't be possible using the standard methods of input. Additionally, if our system were successful we would be able to potentially reduce the device size of systems such as Microsoft's Surface 2.0.

Making a system of object-to-surface detection mobile, it would be of a much greater benefit to the general populace as access to this classification of technology is expected to increase rapidly [23]. The trend of mobile touch screen devices being accepted by the public suggests that these devices will be common place in the future and a working object-to-surface system on mobile platforms would be highly accessible to the public. Enabling such a system on mobile platforms would provide benefits to the public relating to how they interact with their devices for a variety of purposes. To create such a system however, we first had to face design challenges that required solving.

### 2.1 Recognition of Objects

The first challenge we faced, and the core component in our system, was to discover a method that would allow object-to-surface recognition on a mobile touch device. To implement this system, we decided to use the touch points themselves to define an object. This is an approach that is only possible on multi-touch devices unless touch area data information is available. Using the positions of touch points on a device, we concluded that we would be able to recognise an object based on the shape the touches made. By exploiting the properties of shapes, such as length, area and angles, we would be able to use this information to determine the identity of an object on the surface of the screen from a set of preprogrammed objects. This approach was decided to be the most suitable implementation on a mobile platform as it would be able to function correctly with any already existing multi-touch capable hardware. There would be no need to use any external hardware (such as cameras) or specialist screens (such as those in Surface 2.0) for object recognition.

## 2.2 Using objects as input devices

The second challenge we faced was to determine how objects placed onto the surface of a device could be used for input. To augment the virtual environment, the augmentation of an object in reality has to be detected and processed by the system. Without a method to detect this, we can not create a system that allows for interaction, which observed from previously discussed systems is a vital component of augmented virtuality. A method that allowed objects to transfer user input to touch screen devices was required. We designed and evaluated three potential ways that this could be achieved with a generic mobile touch-screen device.

The first proposed method was to use the object detection event as input to the system. For example, a user could place objects onto the screen representing groceries they wish to purchase, such as apples and bananas. The application would then automatically add the grocery item to the user's shopping list. This application would simply sit on top of the object detection system and use an object discovered event as the input trigger. This type of application would be simple to implement as once the device is trained to recognise a set of objects, the program can respond appropriately to each object detection event. This method of input, though simple and has feasible uses for a "touch-and-do" application, doesn't have any advantages over similar simple applications where all of the potential objects are represented virtually. By using a menu based system such applications remove the need for physical objects and are generally far more advantageous to users.

The second input method devised allowed for objects to provide information relating to their position and orientation on the screen. The concept behind this method was to arrange the touch points of an object in such a way that they could also define the position and orientation of an object. As a system would already be able to recognise objects based on the shapes created by touch points, it would simply require the use of a-symmetric touch patterns to enable position and orientation calculations. Allowing for objects to be detected and tracked through the use of this method provides us with a system that could make a wider range of interactions possible. With the data provided by the inclusion of position and orientation monitoring, the quality of input could reach a point sufficient enough to equal or surpass traditional touch based inputs.

The third concept proposed was based on the concept of allowing objects to connect with a device itself through the use of any connectivity functionality available. The use of such a system could pair wireless objects to devices automatically when they came into contact with each other. Any input data from the object would then be transferred to the device through the wireless communication channels. This as a method of input improves the convenience of device pairing but was decided to be against the ideology of the research being performed. To use this third method creates a hybrid of currently available technology and our first proposed concept of object detection triggers. The hybrid also provides us with no tangible benefits as there are already software based solutions available that enable wireless devices to automatically synchronise and exchange data with each other [24].

## 2.3 Experimental Device Choice

To discover the capabilities of our concept system we required an experimental device to use as a platform for a system prototype. From the analysis of the concept, we were able to conclude on a set of requirements that were crucial to our choice of device. In

order to implement a workable prototype of the system, we required a programmable multi-touch mobile device.

For the implementation our prototype, we chose to use Apple's iPad as our main experimental device. We chose this device for several reasons. One core factor for choosing this device was its support of multi-touch through a capacitive screen. The use of a capacitive screen as opposed to a resistive one provides us with the benefit of heightened sensitivity when detecting touch points. This was deemed a property that would likely improve the reliability of the prototype's object-to-surface detection and improve the quality of any implementation.

Another benefit of the iPad is that all of Apple's mobile touch devices can be developed for in Objective C. Objective C is a simple object orientated language that allows for the creation of prototypes in a minimal amount of time. A secondary benefit of this is that the same code would function across several devices with minimal porting. This was a strongly beneficial factor in our choice as it allowed us simultaneously explore our concept on both the hand held and tablet platforms by porting our code to an iPod Touch.

Apple released its first generation iPod on October 23rd 2001 [25] and since then approximately 225 million units had been sold by 2009 with the device holding a 73% share of the MP3 player market [26]. The iPod was later paired with the second generation of iTunes [27], Apple's own music management software, and both products have continued to gain significant popularity in the market since. As they have both developed over time, new features have been integrated to benefit both iTunes and the iPod. One update that is considered revolutionary to the music industry [28] was the provision of the iTunes store in the iTunes version 4 update. This update provided Internet users with a portal to download music legally [29] and led the way for the distribution of other forms of digital content such as music videos, TV shows and films.

The most recent and influential addition to Apple's iTunes store are applications or "apps" sold through their own App Store, a feature which was launched July 10th 2008 [30]. The inclusion of apps into the iTunes store revolutionised the way users could access programs to run on their own mobile devices. Making apps easy to find and download provided users with what could be described as "on-demand programs" which would always meet an expected level of quality due to Apple's control over what apps get published on the store. Many companies have tried to emulate the same success Apple has had, such as Google's Android Market [31], due to how popular this concept is with consumers. By allowing the iPod touch, iPhone and iPad to run a wide variety of applications, Apple created a new market for custom built applications which can be created by anyone who registers as an Apple developer.

It was clear that using Apple's iPad device as our main prototype system would enable us to test our concept for augmented virtuality through extending the basic touch screen functionality available. The support for multi-touch input and ability to develop for the device would fulfil the basic requirements to explore the concept system. The secondary benefits of being able to test the concept on multiple systems and having access to a wide range of information relating to iOS software development also proved to be critical in influencing the choice of experimental device.

## 3. Research into Device Limiting Factors

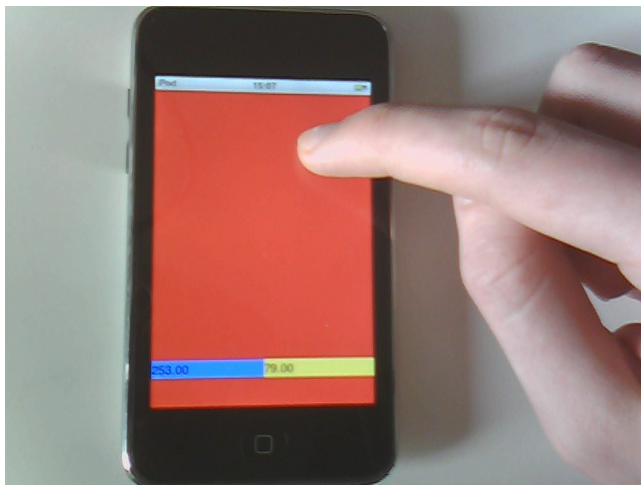
Due to the pioneering aspect of the research, experimentation was required to push the project forwards. Information about the limitations of our devices was needed to discover the limits they

would in turn place on our implementation of a prototype system. Unfortunately, the information we required is not easily or reliably available and the first question that required an answer was “How many touch points can these devices support?”. The limitation enforced by the number of trackable points would effectively define how many objects we could have on the screen at once or how complex our shapes could become to define an object.

**Note:** All of our experiments were performed on a 3<sup>rd</sup> generation iPod Touch and a 1<sup>st</sup> generation iPad. All code and results can be accessed at the URL specified at the end of the paper.

### 3.1 Experiment One – Upper Limit of Distinct Touch Points

To discover how many touch points the iPad and iPod could track simultaneously, a very simple application was written and executed on both devices. This application simply displayed the x and y co-ordinates of a new touch whenever the OS informed us with a new touch event. This application can be downloaded at the website referenced at the end of this paper, under the name of “SingleTouch” and works on both iPod and iPad. To discover the limit of each device, it was only a simple matter of saturating the device with touch points. To do this, one finger at a time was placed then held onto the surface of the screen until the application stopped displaying the co-ordinates of a new touch event.



**Figure 1.** Example of the application detecting and displaying a touch event on an iPod Touch

Our touch events were all separated by both time of contact and location to ensure there could be no blending of touch points or notifications by the OS. From the experiment it was discovered that our iPod Touch could detect and track five points simultaneously and that the iPad could handle eleven simultaneous touches. However, we were required to perform further experimentation into how these touches were handled within the device. We needed to expand our knowledge of how the iOS system operated and tracked touch points.

### 3.2 The Life Cycle of a Touch

Touch events on an iPod touch or iPad are detected and handled by iOS. From the hardware's point of view, when a user touches the screen it recognises an elliptical touch point and

calculates a co-ordinate of where it thinks the user was trying to touch [32]. This is done so that it can provide the operating system with useful touch point data that it can translate into actions, events and gestures. Examples of these are double tapping an area, pressing a button or pinching the screen to zoom. Touches like this can be tracked through the use of Objective C classes such as gesture trackers or through actual touch point tracking. The operating system will inform the running application of any changes by calling one of the following methods if the running application supports it:

- touchesBegan
- touchesMoved
- touchesEnded
- touchesCancelled

Each of these methods provide the application with a set of UITouch objects, contained within a NSSet. UITouch objects are grouped into NSSets as occasionally, more than one touch has been detected or has changed state. For example, pinching involves the concurrent movement of two fingers so iOS places two UITouch objects into the NSSet. Each UITouch object in the set represents a touch that has changed in accordance with the method they've been passed to. To clarify, if touchesEnded is called with two UITouch objects in the set, both of those touches have been detected by the OS as having left the screen. This API allows for simplistic coding to handle the different touch states that can arise during the execution of an application.

When a new touch is detected by the device, the operating system allocates memory to this touch as a UITouch object and retains this object in memory. The UITouch is then made a member of a set of NSObjects and passed to the application via the touchesBegan method. The application can then behave as it wishes with the data provided to it. Within our proposed system, this is where touch point tracking and shape detection code would operate in order to determine the object(s) that have been placed onto the screen.

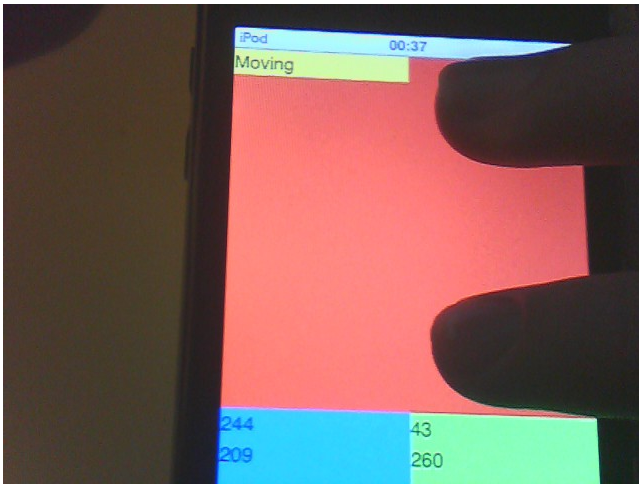
This touch is then tracked by the OS, which in turn informs the running application of any changes until the touch ends or is cancelled. Whenever movement occurs on the surface by touch points, touchesMoved is called by the OS to provide the running application with a set of UITouch objects that have moved. By using the pointer values of each UITouch, it is possible to track each individual touch point and its position on the screen. It is through this method that we can track the position and orientation of our objects during their lifetime on the screen. Additionally, by tracking only the points that are known to be an object from the touchesBegan method, we can also optimise the code to ignore any noise induced by additional touch points moving on screen.

Eventually touches being tracked by the OS are either ended or cancelled. The key difference between these two methods is that a touch ends when a user lifts their finger from the device while touch cancellation is triggered when the OS deems it necessary to force a program to stop tracking touch points. An example of when this could happen is on the iPhone when a phone call is received. The OS forces the program to stop tracking the current touches as it believes them to be irrelevant as the user needs to accept or reject an incoming call.

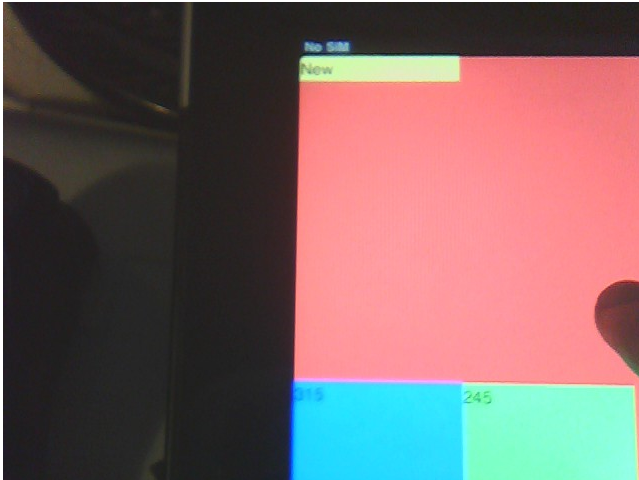
### 3.3 Experiment Two - How a Device Handles Over-Saturation of Touches

With the knowledge of how iOS provides touch events to applications we wished to know how the device would react when

we over saturated the amount of distinct points it could handle. From our first experiment we discovered that a device will stop registering new touch points once saturated but how would it treat those that were already in contact with the device? This would be an important factor in any design decisions made to implement an object-to-surface based system as there is always the potential that a device could be saturated. To discover what would happen we wrote two device tailored apps, one for the iPod and one for the iPad. These applications tracked all of the touch points in real time and displayed their co-ordinates. As well as doing this we also used another label that would tell us the state of our touches being passed to the application, whether they were new, moving, ended or cancelled. This small program would allow us to very simply discover what occurs within the device once it is saturated with touch points. The applications can be seen in use in figures 2 and 3. Both applications are available for download at the website referenced at the end of the paper under the titles MultiTouchiPod and MultiTouchiPad.



**Figure 2.** Photo of the application capable of detecting up to five on the iPod Touch



**Figure 3.** Photo of the application capable of detecting up to eleven moving touches on the iPad

The apps behaved exactly as expected, with new touches showing a status of “New” on the label, moving touches as “Moving” and touches being lifted as “Ended”. The interesting

results however were related to how the device behaved once the screen was overloaded with touch points.

When the device is saturated with touches, the operating system passes a set of all of the currently tracked touch points to the `touchesCancelled` method. From this point on the OS no longer informs the application of any changes in state relating to the touches that were already in place on screen. The OS however does appear to continue tracking our touches until they end.

This conclusion is based on how the device behaves when we try to establish a new touch point after the `touchesCancelled` method has been called through saturation. When the `touchesCancelled` method is called by the OS, the touch points that were being tracked become “zombie-touches”. A zombie-touch (a term only used within the context of this paper) refers to a `UITouch` that is still tracked by the OS but the application believes to be dead.

While there are at least the maximum number of supported touches on the screen in the zombie state, the OS will not register any new touches on the screen. A new touch point is unable to replace a zombie-touch and will not be passed to the running application. It appears that the effects of a zombie-touch can only be remedied by the removal of the dead touch point from the screen. For every zombie-touch removed from the screen, memory is freed within the OS to allow for the tracking of a new touch point.

In essence, touch-tracking can be thought of as an array structure where the space for data only becomes available once a touch is lifted from the screen. This experiment has shown that when the OS is saturated with touches, it simply stops informing an application of all of the touches on screen yet still tracks them to ensure that they finish. This is a strange quirk of iOS and may have some unknown logic behind why it is implemented this way. This property of iOS touch devices however is a key guiding factor in limiting how we design objects for our object-to-surface recognition system. Any object designed for use in the object-to-surface recognition system must be considerate in how many touch points it uses. They must be designed to prevent the accidental creation of zombie touch points by allowing for a “touch-buffer” in case of user errors.

### 3.4 Constraints of Object Construction

At this point of our research, we decided to experiment further with respect to the limitations placed on potential input objects based on the limits enforced by our experimental devices. Up until this point we had discovered that objects with up to 5 or 11 contact points could be implemented depending on the device we chose to utilise. In reality, we would implement objects that were a few points below this limitation to build in resilience for accidental touches.

In order for us to construct objects that would be able to replicate fingers on a capacitive device, we needed to make them out of conductive materials. While a user is in contact with such an object, it acts as an extension to the skin and allows the object to interact with capacitive devices. Proof of this can be seen by David Ayman Shamma's implementation of *iSteelPan* [33]. To implement his system, Shamma used conductive fabric and metal rods as an extension of his body's natural capacitive properties. To construct our objects we also used conductive materials that would replicate this effect. The basic design of any touch-extending object is to use conductive cloth to create points of contact with the device and to connect these points by means of a conductive body and adhesive. This allows for the creation of conductive objects that will not damage the screen of the device

and can be detected by a capacitive screen whenever they are placed on the screen whilst being in contact with a user.

In order to be able to construct such objects however, information relating to their constraints needed to be discovered. Experimentation was required to discover the size and proximity restrictions we had to work with. We required knowledge of the smallest touch points that we could implement to be detected by the OS and how close we could put points together in an effort keep our objects to reasonable dimensions.

### 3.5 Experiment Three – Discovering the Minimum Size of Touch Points

To create the touch points of our objects we used pieces of conductive felt. This conductive material would form the base of any conductive objects we created for use with our object-to-surface system. Before using this cloth in our objects however we required an understanding of how well it worked with the iOS devices. To test this, experiments were performed on the iPad and iPod using the very simple application from experiment one.

For this experiment however, pieces of cloth that varied in size were tested as a trigger for input to the device. To do this each piece of cloth was individually placed on the screen of the device and positioned using a non-conductive object such as a ruler or pen. The piece of cloth was then contacted with a hand held paper clip, creating a circuit in which the device would detect a touch if the area was suitable for the device. To perform this experiment, each piece of fabric was moved then touched ten times on both screens. If they provided a change in co-ordinate value without the application of too much pressure, the detection was viewed as a positive result and recorded. The results of our experiments on both the iPod touch and the iPad can be seen in figures 4 and 5 with original data available at the URL specified at the end of the paper. An example of the tools used in our experiments can be seen in figure 6 and an example of a touch being detected through a piece of fabric in figure 7.

The results of our experiments showed us that a 5mm squared touch is the minimum value that can be reliably detected by the OS on both devices and a minimum value of 4mm<sup>2</sup> with very little reliability. In reality though, further unrecorded experimentation with the cloth taught us that using a piece of cloth at 5mm<sup>2</sup> was highly unwise. Due to being on the borderline of detection pieces of this dimension were highly susceptible to degrading and becoming unresponsive through use. We suspect this is due to the cloth becoming dirty over time and 5mm being highly likely as boundary value. Evidence backing up this claim can be seen in the fact that 5mm corresponds to the width of text entry keys on an iPod Touch's keyboard and 4mm<sup>2</sup> being an area that is very unreliably detected.

Due to this degradation in reliability we decided that 6mm<sup>2</sup> pieces would be the most practical size for our prototype system. This would be the minimum size we would be able to utilise for our object touch points while maintaining a reliable system. As an additional note, the experiment also provided us with information regarding large pieces of fabric. Larger cloth sizes such as 10mm<sup>2</sup> can provide varying touch points based on where contact is made on the cloth. Due to this the cloth touch point dimensions should not be too large otherwise readings could become unreliable.

| Size of fabric (mm <sup>2</sup> ) | % Successful detected touches |
|-----------------------------------|-------------------------------|
| 10                                | 100.00%                       |
| 6                                 | 100.00%                       |
| 5                                 | 100.00%                       |
| 4                                 | 0.00%                         |
| 2                                 | 0.00%                         |
| 1                                 | 0.00%                         |

Figure 4. Detection results on the iPod Touch

| Size of fabric (mm <sup>2</sup> ) | % Successful detected touches |
|-----------------------------------|-------------------------------|
| 10                                | 100.00%                       |
| 6                                 | 100.00%                       |
| 5                                 | 100.00%                       |
| 4                                 | 10.00%                        |
| 2                                 | 0.00%                         |
| 1                                 | 0.00%                         |

Figure 5. Detection results on the iPad

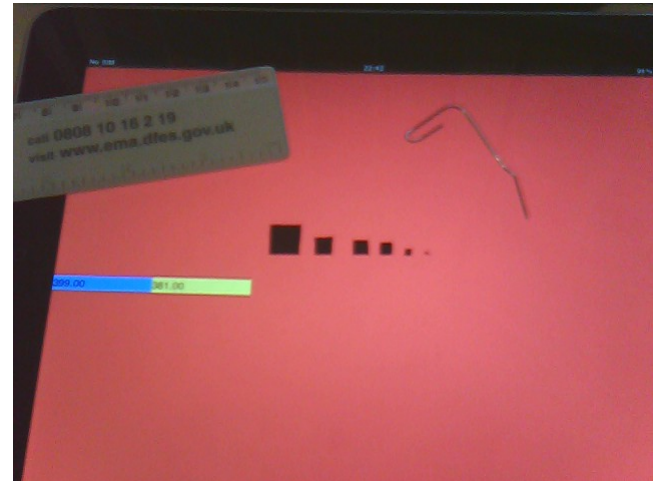


Figure 6. Photo of the tools used to test conductive cloth of varying dimensions on the iPad

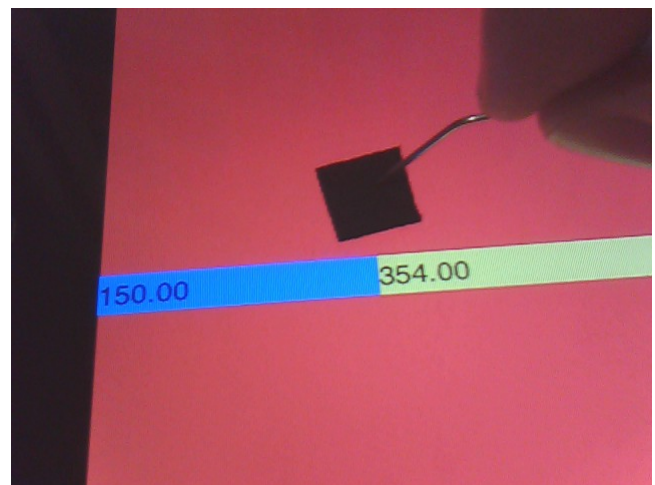


Figure 7. Photo of a touch being detected through the use of a paper-clip and cloth as a proxy on the iPad

### 3.6 Experiment 4 – Discovering Proximity Restrictions

The second restriction on achieving the most minimalistic dimensions for objects in our system was based on how close we could get touch points to each other. Writing an application to calculate the distance between two points is a simple task through the use of the Pythagorean distance formula and this application would be used to discover the minimum proximity value we could obtain when using 6mm<sup>2</sup> pieces of fabric. The application “distance” can be downloaded at the URL referenced at the end of the paper.



**Figure 8.** Photo of an application using the distance between two touch points to determine the iPod's background colour

Through the use of the above application and 6mm squared pieces of fabric we were able to discover a limit of approximately 70 pixels before touch points merged together to form one individual touch point. This value is believable as it is also around the point at which it becomes difficult to pinch your fingers together without leaving contact with the screen. This suggests that the touch point detection drivers are designed to exploit this physical limitation when trying to detect touch points on screen.

Measurements of the iPod's screen revealed to us that it is 320 pixels/5cm wide. Assuming that the driver registers the touch in the centre of our 6mm squared fabric pieces, we can calculate that there is approximately 11mm between the touch points. Taking into consideration the size of our cloth, there must be at least a 5mm gap between the edges of the fabric in order to operate on the borderline of accepted proximity for the iPod and iPad.

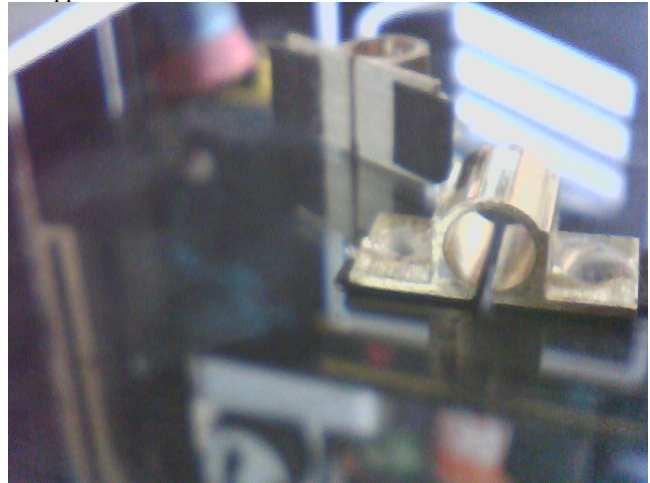
### 4. Applications of Object-to-Surface Recognition

With the knowledge of how the devices handle touch points and the limitations enforced on us by them we could now design systems that would be able to make use of object-to-surface recognition. Some concepts that were thought of to demonstrate the technology are as follows:

- Real sliders used to control an equalizer on the iPad
- Conductive knobs for use as dials in image manipulation on the iPad or iPod, e.g. HSV manipulation
- A conductive knob to be used as a safe combination entry dial on iPod touch
- Pokémon figures from the Pokémon franchise that when in contact with the screen displayed their facts

- An implementation of the retro game “Asteroids” which utilised a real ship

We however chose to implement a remake of the classic video game “Pong” to display our system in its prototype form. To do this, real life paddles were made from parts of a metallic bolt lock, conductive adhesive and conductive felt. These paddles would form the basis of our interaction with the game with each paddle creating two touch points on the screen that could be tracked by the application.



**Figure 9.** Bolt lock paddles made to play the game



**Figure 10.** Pong+, a game that uses paddle objects in reality to control game play based on paddle position and orientation

The use of paddles in this manner adds a new game play element to the classic game. Both paddles can be at any angle or position on the screen, allowing us to create a much more realistic game. To define the location of paddles, very simplistic shape recognition code was used to discover a paddle's location and orientation.

The paddles were implemented by searching the set of tracked touches to find each touches' nearest neighbour. A sort routine is

then executed to find the first pair of touches that are closest to each other and forms a paddle from them. This method is repeated to find the second pair of closest touches to form the second paddle. Using the co-ordinates of the touches, collision detection and deflection angle calculations are possible, allowing for a playable augmented virtuality version of pong.

These paddle updates are applied by the touchesBegan, touchesMoved and touchesEnded methods to track the user's behaviour. When the puck attempts to cross one of these paddles, it is deflected and sped up allowing for competitive two player game play. The code for this game can be found and downloaded at the website referred to at the end of this paper along with a video of demo game play. Experience with the prototype shows that augmented virtuality through our proposed system is viable on mobile devices and could in fact be implemented on larger touch screen devices. With some better coding designed to account for the "fat finger" contact points, games such as Pong+ could become a reality within the touch screen computing domain.

## 5. Conclusions

Touch screens are becoming ever more integrated into the modern world. With mobile computing devices opting to incorporate touch screens, they are rapidly becoming a ubiquitous feature. Within this paper we have discussed a potential extension of how we as users interact with such devices through the expansion of the touch screen interface.

We have presented the foundations for a system that can be implemented on iOS touch devices to allow for object-to-surface recognition and presented some example cases where such a system could be beneficial/entertaining for users. We have also discovered some of the limiting factors that are enforced on such a system on the mobile platform.

Several factors are vital to implementing an augmented virtuality system based on an object-to-surface detection system on any device. The factors are as follows, size of touch points, proximity of touch points and the limitations from touch point saturation. With these factors held in consideration however it is possible to design systems that will allow for reasonably reliable and natural interactions on touch screen devices through the use of objects.

Future advancements for this area of research could be to advance knowledge of shape detection from touch points. The inclusion of more efficient and effective shape recognition code would have many benefits. It could lead the way for more complex systems that support multiple potential objects from a large dataset, or even simultaneous multiple object interaction. It could also allow for objects themselves that can relay input through the touch points. For example, a physical keyboard attached to an iPad which inputs through touch patterns.

The one area this approach to object recognition would be truly advantageous however is within the interactive desktop/tabletop industry. Projects such as Surface 2.0 attempt to create new interactive table tops but may be held back by their bulky (and likely highly costly) bespoke hardware. Implementing a system similar to the one discussed within this paper on a larger capacitive touch screen may prove successful as it could potentially drive down the costs and size of interactive tabletops.

## Acknowledgments

Thank you to Charlotte Jones for creating the Pong+ artwork and to Reuban Edwards for being so patient in handling our requests relating to the Apple Developer Centre.

## References

All hyper links were last known to be active 25/03/2011

- [1] Example Kinect projects used in Augmented Reality - <http://thomaskcarpenter.com/2010/12/21/10-awesome-ways-to-use-kinect-for-augmented-reality/>
- [2] Paul Milgram's mixed reality continuum (Introduction)- [http://etclab.mie.utoronto.ca/people/paul\\_dir/IEICE94/ieice.html](http://etclab.mie.utoronto.ca/people/paul_dir/IEICE94/ieice.html)
- [3] Paul Milgram's mixed reality continuum quote (Summary)- [http://etclab.mie.utoronto.ca/people/paul\\_dir/IEICE94/ieice.html](http://etclab.mie.utoronto.ca/people/paul_dir/IEICE94/ieice.html)
- [4] Windows on the World paper, using cameras to model texture onto a virtual environment - <http://www.sics.se/~kalle/published/wow.pdf>
- [5] Evidence reference to Sony's Eyetoy product - <http://blog.us.playstation.com/2010/11/03/eyetoy-innovation-and-beyond/>
- [6] Evidence reference to the show Nightmare - <http://www.imdb.com/title/tt0193677/>
- [7] HCI success stories, slide four - <http://www.slidefinder.net/H/HCI/6756351>
- [8] Oxford Dictionary definition of User Interface - <http://oxforddictionaries.com/definition/user+interface>
- [9] Secondary definition of User Interface, paragraph titled "It can change world history" - <http://www.answers.com/topic/user-interface>
- [10] Evidence to back up claim of graphics tablets preference - <http://www.peachpit.com/articles/article.aspx?p=383855>
- [11] Reference to paper written based on touch display keyboards - [http://www.florianblock.net/downloads/block\\_touch\\_display\\_keyboards.pdf](http://www.florianblock.net/downloads/block_touch_display_keyboards.pdf)
- [12] Quote about the construction of resistive touch screens, first page second paragraph - <http://focus.ti.com/lit/an/slyt209a/slyt209a.pdf>
- [13] Reference to how iPhone screens operate - <http://electronics.howstuffworks.com/iphone2.htm>
- [14] Electrical engineer's explanation of capacitive touch screens - <http://answers.yahoo.com/question/index?qid=20101016144614AAilCSf>
- [15] Evidence supporting the claim that touch screen phones are becoming ever more popular - <http://www.canalys.com/pr/2010/r2010021.html>
- [16] Evidence backing up claim of Dell using touch screens in laptops - <http://www1.euro.dell.com/uk/en/home/Dell-Laptops/inspiron-duo/pd.aspx?refid=inspiron-duo>
- [17] Android Tablet PC Product Reference - <http://www.android-tablet.org/android/eken-m003-a-new-8-inch-android-tablet-pc/>
- [18] Official iPad website - <http://www.apple.com/ipad/>
- [19] Microsoft Surface 2.0 project page link - <http://www.microsoft.com/surface/whatissurface.aspx>
- [20] Sonicspree product link - <http://www.ergonomidesign.com/CASES/Interaction-Design/Sonicspree-Massive-Multi-touch-Gaming-Experience.aspx>
- [21] Reference to PixelSense technology used in Surface 2.0 - <http://www.betanews.com/article/Microsoft-introduces-Pixel-Sense-in-new-slimmer-Surface-20/1294271005>

[22] Evidence of Surface 2.0's 4 inch depth

<http://www.microsoft.com/surface/whatsnew.aspx>

[23] Evidence supporting claim of tablet pc sales expected to increase rapidly -

[http://www.seek4media.com/tech/10238-tablet\\_computer\\_sales\\_set\\_to\\_increase\\_significantly\\_during\\_2011.html](http://www.seek4media.com/tech/10238-tablet_computer_sales_set_to_increase_significantly_during_2011.html)

[24] Bump application as an example of automatic pairing-

<http://www.zath.co.uk/bump-iphone-app-review/>

[25] Evidence for the release date of Apple's iPod device -

[http://www.everymac.com/systems/apple/consumer\\_electronics/stats/ipod.html](http://www.everymac.com/systems/apple/consumer_electronics/stats/ipod.html)

[26] Evidence for the share that iPod commands in the MP3 market -

[http://www.afterdawn.com/news/article.cfm/2009/09/09/ipod\\_market\\_share\\_at\\_73\\_8\\_percent\\_225\\_million\\_ipods\\_sold\\_more\\_games\\_for\\_touch\\_than\\_psp\\_nds\\_apple](http://www.afterdawn.com/news/article.cfm/2009/09/09/ipod_market_share_at_73_8_percent_225_million_ipods_sold_more_games_for_touch_than_psp_nds_apple)

[27] Evidence for the release date of Apple's iTunes software, paragraph titled "iTunes 1 (Jan. 9, 2001-Oct. 23, 2001)" -

[http://www.maclife.com/article/feature/the\\_complete\\_itunes\\_history](http://www.maclife.com/article/feature/the_complete_itunes_history)

[28] Evidence for iTunes store as revolutionising -

<http://www.csmonitor.com/USA/Society/2010/0226/Three-ways-iTunes-and-its-10-billion-in-sales-changed-music-industry>

[29] Evidence for the release date of the iTunes store, paragraph titled "iTunes 4 (April 28, 2003-Sept. 7, 2005)" -

[http://www.maclife.com/article/feature/the\\_complete\\_itunes\\_history?page=0.1](http://www.maclife.com/article/feature/the_complete_itunes_history?page=0.1)

[30] Evidence for the release date of Apple's app store update -

[http://www.readwriteweb.com/archives/apples\\_iphone\\_app\\_store\\_launch.php](http://www.readwriteweb.com/archives/apples_iphone_app_store_launch.php)

[31] Link to Android Market to show how other companies are also trying to use this business model -

<https://market.android.com/>

[32] Expert discussing how iOS detects touches, reference to elliptical shapes at 2:25 at a Stanford lecture -

<http://www.youtube.com/watch?v=iZUtAU6pBf0&feature=BF&list=SPCCE29F69A864766F&index=14>

[33] Example of conductive materials being used to create objects that can interact with an iPad instead of fingers -

<http://www.fashioningtech.com/profiles/blogs/conductive-fabric-drum-sticks>

All documents, programs and results can be found at <http://www.lancs.ac.uk/ug/lewis3>