Touch Wall

Team 38: David Xiao, Mark Menezes & Prateek Dahal
Client: Microsoft Research
Client Contact: Dean Mohamedally

Microsoft Research touchdevelop

UCL

1. Table Of Contents

2. Abstract

The application is designed to allow a user to interact with Windows. The interactions occur on a flat surface with the screen contents displayed on, either because the surface is a screen, or if the surface is backlit by a projector. The interactions are detected by the Kinect 2 sensor, processed by our application, and then used to move the cursor accordingly.

The final application is to be used by Microsoft Research, where they hope to be able to combine Touch Wall with TouchDevelop to create a learning platform for schoolchildren to aid their learning of programming.

3. Context

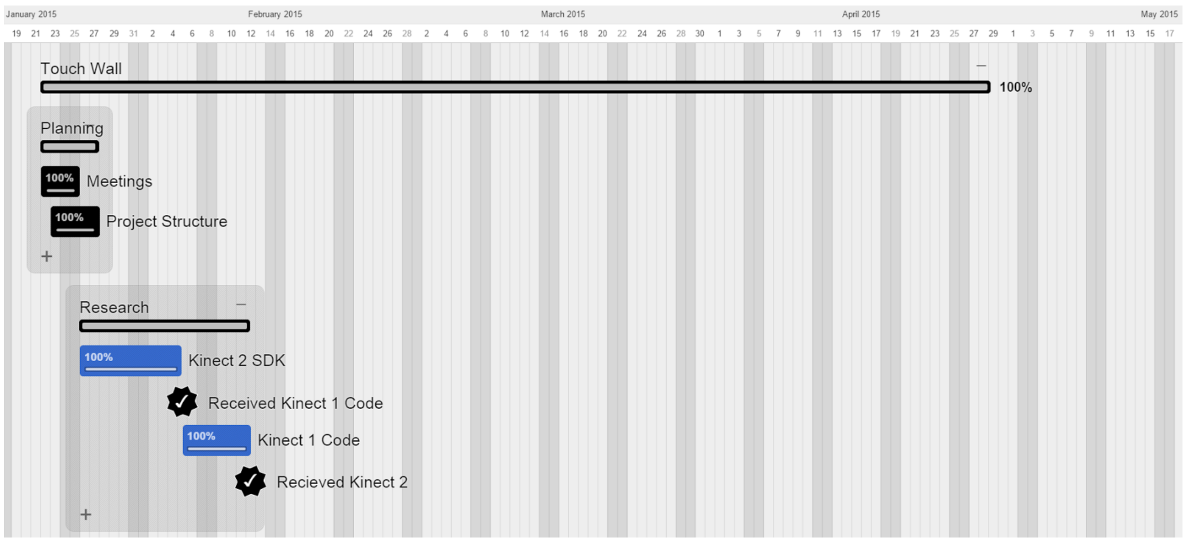
In the UK, schoolchildren from primary school onwards will take part in a new computing curriculum. This will include programming lessons, requiring teaching staff to have the correct training and tools. This new curriculum was introduced in response to the shortage of qualified employees in computing related positions.

Microsoft Research created TouchDevelop in response to the changes in the way many people interact with their devices. Instead of keyboard and mouse, there are many devices that have touch screens, are connected online all the time and have other sensory information such as location. TouchDevelop aims to address these changes by providing a unique programming environment that can be used on many devices would previously not be considered as programming tools.

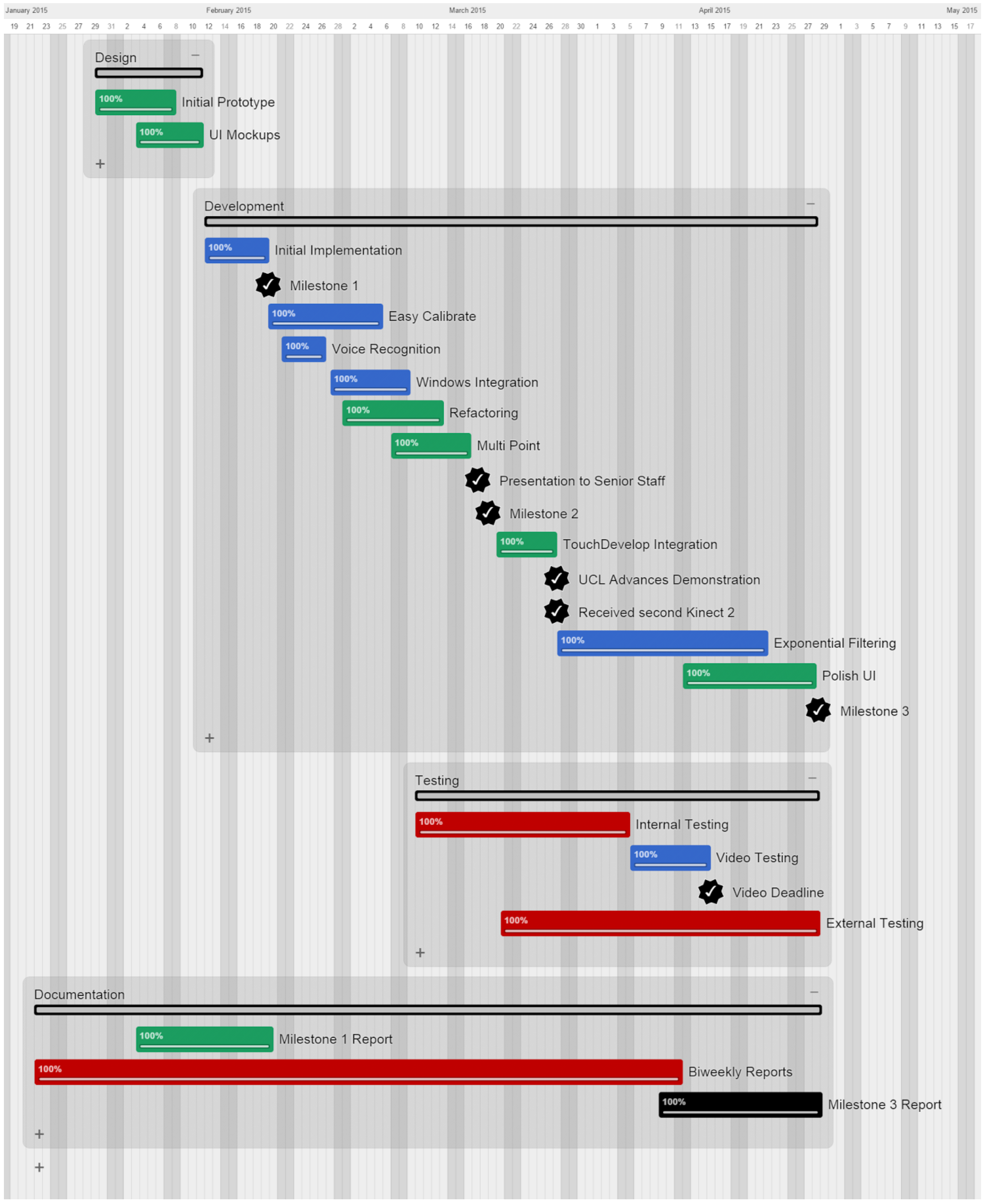
Touch Wall aims to turn any flat surface, such as a wall, into a touch sensitive surface using a Kinect sensor. One of its end goals is to use it with TouchDevelop to make the coding experience more interactive. Implementing the Touch Wall will cater the needs of children who are more comfortable with the kinesthetic learning style.

4. Workplan

Blue: David Green: Mark Red: Prateek Black: Team



Blue: David Green: Mark Red: Prateek Black: Team



5. Team Summary

5.1. Prateek Dahal

- Roles: Team Leader/Project Manager, Lead Interviewer for Requirements, Lead Tester
- Key Skills: I am a certified Apple Support Professional for Mac OSX. My previous experience consists of the C and Haskell languages. Working as a salesman in PC World for 6 months has given me experience with working under pressure in addition to learning Team skills.
- Prior Project Experience: I have co-coordinated a Talent Show project for CODEC-UK, a charity organisation as well as being an active volunteer for the charity since it was founded in 2009. I was part of several projects at Historic Royal Palaces as a member of the Desktop Support team.
- Communication and Teamwork: I gained communication skills during my time at HRP. As a member of a team of six, communication was vital on a day to day basis to have team integrity. Teamwork has been a key part in the four jobs I have had so far hence I have lots of experience in working as a team.
- StrengthsFinder 2.0 strengths: Competition, Learner, Input, Futuristic & Intellection.

5.2. David Xiao

- Roles: Programming and Repository Lead, Technology Platform Research Lead
- Key Skills: I have prior programming and debugging experience includes C, C++, Java, HTML, Haskell, Python and Pascal. I also have had experience with developing applications for Android.
- Prior Project Experience: I have worked in multiple projects either for my own purposes or for a client. These personal projects have provided a learning platform for many programming languages.
- Communication and Teamwork: I have worked in multiple placements ranging from small web development companies to retail stores, each showing how important both communication and teamwork is to the work environment. I have also set up multiple societies and clubs during my time at A-Level.
- StrengthsFinder 2.0 strengths: Input, Learner, Harmony, Achiever, Analytical

5.3. Mark Menezes

- Roles: Lead UI Designer and User Testing, Documentation Lead
- Key Skills: I enjoy problem solving and I have a good work ethic. I have prior programming experience with C, Java, Haskell, HTML, CSS and JavaScript, and experience using Adobe Fireworks, Flash and Dreamweaver.
- Prior Project Experience: I had a work experience placement at Transaction Network Services in Customer Implementations, where I helped with their current projects, making presentations and modelling their progress.
- Communication and Teamwork: I was part of a team that updated my school's website weekly and we eventually worked on building a new website. Additionally, I volunteer each week as a Welcomer at my Church, where I enjoy practising my communication skills by greeting and helping the community.
- StrengthsFinder 2.0 strengths: Achiever, Discipline, Responsibility, Futuristic, Intellection

6. Requirements

6.1. Must have requirements

- Kinect needs to be modified so that it can connect to a Windows computer and communicate via USB.
    - Completed: Kinect successfully sends depth data to computer
- Application needs to be able to read the depth data* from the Kinect, interpret it and be able to find where the user's finger is.
    - Completed: It can detect multiple fingers
- Application needs to convert the coordinates of the camera space to the user space, and convert it into cursor coordinates.
    - Completed: User is able to move the cursor
- Application needs to be able to calibrate to the screen dimensions via user interaction.
    - Completed: "Easy Calibrate" mode where user touches each edge of screen once

6.2. Should have requirements

- Application should be able to detect more than one finger for multi touch with multiple users.
    - Completed: Feedback for multiple fingers touching the screen simultaneously
- Adapt to number of users and remember which cursor belongs to which user
    - Completed: In "multi touch mode", each user is denoted by a colour
- Application needs to be able to use filtering and averages to make the cursor control as accurate as possible.
    - Completed: Gestures are a lot more stable than first version
- We should create interactive Touchdevelop examples to effectively demonstrate our project.
    - Completed: "Highway Racer" and "Balancer" games and tutorials created
- A means of launching TouchDevelop from our TouchWall program for easy access.
    - Completed: A button launches TouchDevelop in the browser

6.3. Could have requirements

- Could add voice control to allow users to speak their commands to the Kinect
    - Completed: User can tell the Kinect to begin or cancel Easy Calibrate mode
- Allow users to control Windows (instead of just Touchdevelop) with the Kinect
    - Completed: Our application manipulates the Windows cursor
- Allow the Kinect to detect how much pressure the user is exerting on the screen
    - Completed: Depth data is used to give values for pressure applied on screen
- Guess the position of the user's hand if another hand is blocking it from Kinect's view.
    - Incomplete: Could not complete in given time frame

6.4. Would have requirements

- Integrate multi touch with Touchdevelop to allow more than one pupil to program simultaneously.
    - Incomplete: TouchDevelop currently does not support multi touch

** Depth Data - The 3D coordinate data created by the application using input from the kinect, and used to interpret where the user's finger is relative to the Kinect sensor (if any)*

7. Work Distribution

7.1. Prateek Dahal

- Produce bi-weekly reports
  - Completed: All bi-weekly reports have been detailed and submitted on time
- Create the logo for our project
  - Completed: A simple yet effective design was created for the logo
- Test the application
  - Completed: Achieved throughout the development of the application
- Find volunteers to test and review our solution and comment on how to improve it
  - Completed: Found multiple people at various stages of development for testing

7.2. David Xiao

- Allow application to detect the user's hand
  - Completed: The Kinect can scan a designated area for a finger
- Add Easy Calibrate mode
  - Completed: With some user interaction, the application can adapt to any screen size
- Speech recognition for starting and cancelling calibration with user's voice
  - Completed: Many voice commands are now available
- Allow users to control Windows using the Kinect
  - Completed: The mouse_event function allows the application to control the cursor
- Add a post filtering algorithm
  - Completed: Added exponential filtering to make interactions smoother
- Recording our team video
  - Completed: Team video was filmed, edited, voiced over and published on Youtube

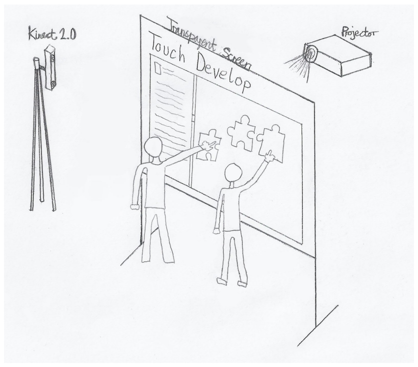
7.3. Mark Menezes

- Refactor code
  - Completed: The application has been fully restructured during its development
- Create multi touch mode
  - Completed: Kinect can detect more than one finger simultaneously
- Allow the application to adapt to multiple users
  - Completed: Application can track multiple points
- Create interactive Touchdevelop tutorials to effectively demonstrate our project
  - Completed: "Highway Racer" and "Balancer" games and tutorials created
- Make the application useable on Touchdevelop
  - Completed: Application contains a button to open TouchDevelop
- Designing and implementing the UI
  - Completed: Created a user interface that is very useful for debugging

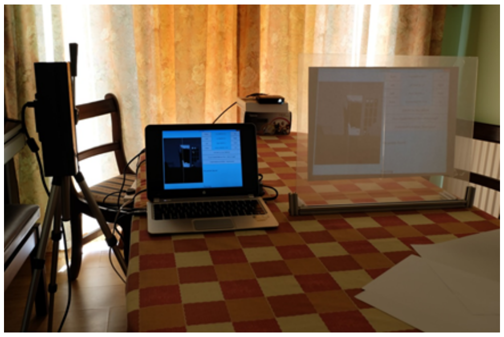
8. Application Design

8.1. Setup

The screen will be a transparent surface, made of PVC with a coating. A projector will be placed behind the screen, and light will be scattered on the screen, allowing the user to see what is on display.

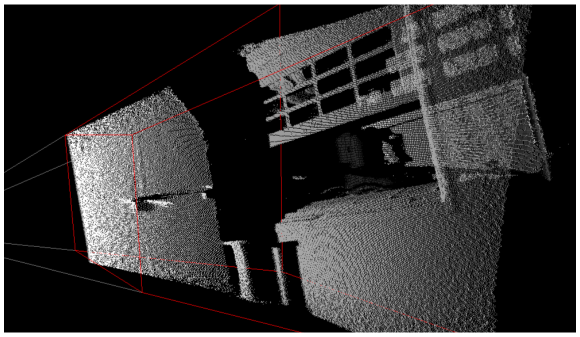


The Kinect will be placed to the side of the screen, parallel to it, screwed onto a tripod. It will allow touch input by measuring the coordinates and depth of the user's hand and translating that into mouse movements and clicks.



We originally wanted to use the Kinect from below, as this would make detecting multiple points side by side easier. This detection would be more useful than detecting multiple points vertically. We eventually scrapped this idea due to the Kinect 2's lack of horizontal vision.

8.2. Explanation



The application uses the Microsoft Kinect Library for the depth data and the Microsoft Speech Recognition Library to allow voice commands to be issued.

Using the Kinect Library, the application is able to read the depth data in our `FrameDataManager` class, which then creates a new Frame when new data has been fed in. This then creates a map in 3D using the `MapDepthFrameToCameraSpace` function. This gives us a 3D space that we can then analyse within the screen dimensions to look for a finger. The depth data is stored as an array of ushorts and the 3D map is stored in an array of `CameraPoints`.

The program then scans the entire array, searching for points that are within bounds that it knows. These bounds ideally match up with the physical screen. As this is rarely the case in real life, we have also implemented a method of calibration with the help of some user interaction.

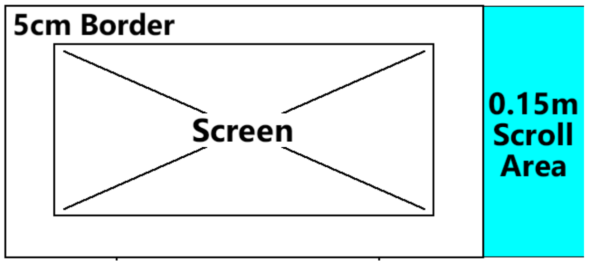
The program then acts differently depending on whether it is in Multitouch mode or not. If it is, then it runs a method called `FindGestures`, which looks for multiple points in the array. It works by going through the array, and for each point, it checks if it is distinct from any previous points. Once complete, it then compares the final selection of points to pick up to four points that are nearest to the screen.

On the other hand, if it isn't in Multitouch mode, then a method called `FindSingleGesture` is called. This function looks for the single point that is nearest to the screen, then it stores all of the points that are around it. The program then goes through each of these points and records points that have neighbouring points. From this last list, the method then picks whichever is closest to the screen.

Dynamic exponential filtering is then applied to smoothen the coordinate. We found that when using raw data, the noise would cause a lot of "wobble". Our first filtering method was to use a moving average, but after feedback on how the resulting cursor still wobbled too much, we changed it to an exponential filter. We were also given a suggestion to use a Kalman Filter, but we were unsure how to use it in this circumstance. We later realised that a dynamic exponential filter would improve upon the exponential filter without the complexity of the Kalman filter.

Upon finding the user's finger, we created a function called `ProcessGesture` in our Gesture class that will take coordinates of the finger in the `CameraSpace`, and convert it to coordinates for the cursor. In finding the finger, the 3D map also allows us to find how far away the user's finger is from the screen, so we can use how close the user's finger is to the surface as a way of deciding when to click the mouse and how much pressure the user is exerting on the screen.

Once we have the coordinates for the cursor, we can then do some interesting things with it.

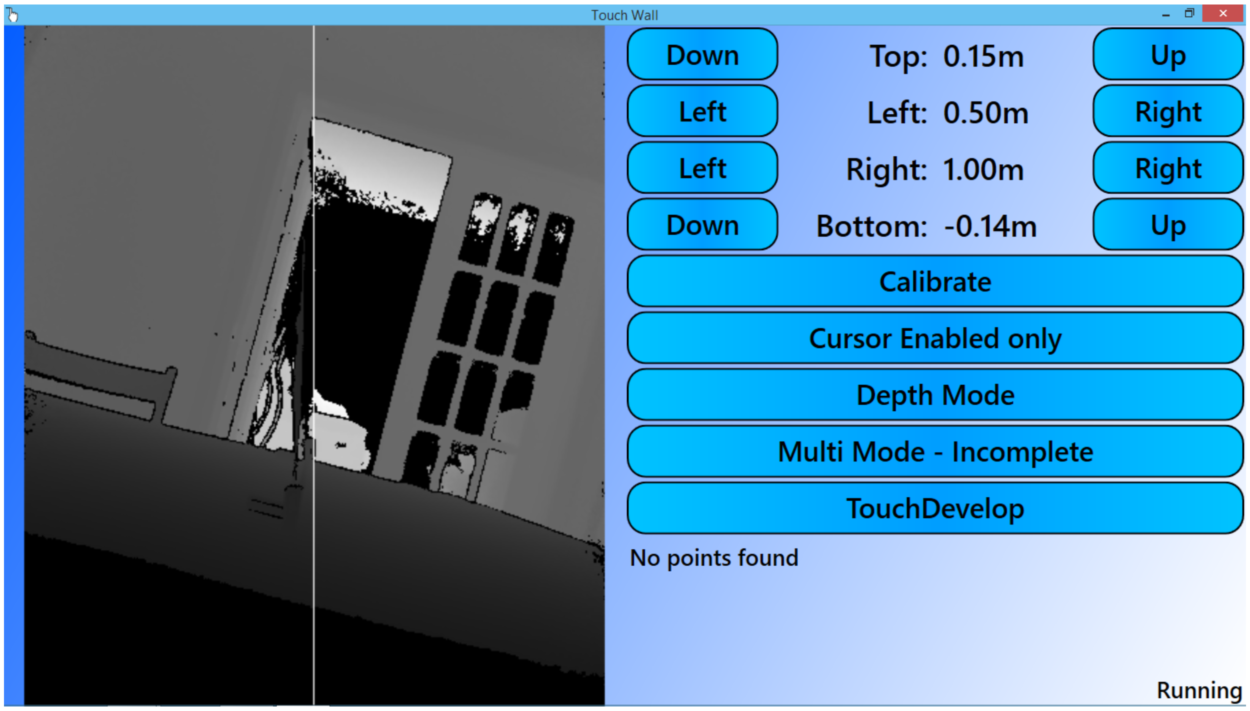


In single touch mode, these coordinates are then plugged into the Windows cursor via our `InteractWithCursor` function. If the coordinates are in the scroll zone, then the mouse_event method is fired with flags to indicate that a mouse scroll is desired. On the other hand if the coordinates are within the screen boundaries, then mouse_event is fired with different flags to move the cursor. If the user has moved their finger close enough to the screen, then a left click down action is fired and when the user's finger moves away again, the left click up action is fired.

In depth mode, a circle is shown on the screen at the coordinates of the users finger, the size of the circle representing how close the finger is to the screen. The circle changes colour and shape depending on whether the user is touching the wall or not. The same process of using the dynamic exponential filter applies here as well.

In multi touch mode, a circle is shown on the screen at the coordinates of where the kinect has found the user's finger(s). Like single touch mode, the multi touch mode also has exponential filtering after the success we had with it in single touch mode. This proved more challenging, as this required the code to know where the previous coordinate was for each of the current point. Unlike single touch mode, multi touch mode does not affect the cursor; it is purely there as a proof of concept.

8.3. Screenshot



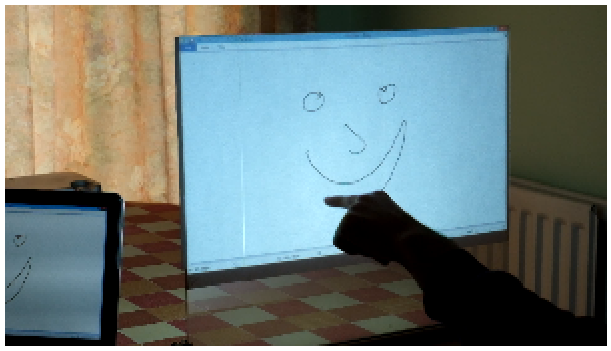
8.4. User Interface

We designed our user interface so that it is intuitive to use on a touch screen, with large, easily-accessible buttons. We decided to stick to a blue colour scheme with bold text that stood out so it is easy to read. On the left of the main window, we have a live image feed from the Kinect that represents the depth data and on the right we have the buttons to control the application.

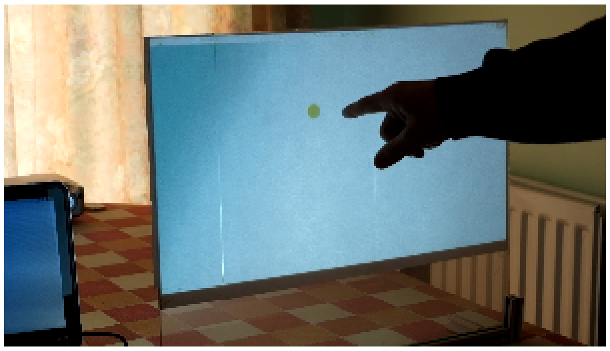
The first four rows of buttons show the values that the calibration has been set to, which include the distances of the top, bottom, left and right edges of the screen relative to the Kinect sensor. There are individual buttons on either side of these values, that can be used to incrementally adjust each one.

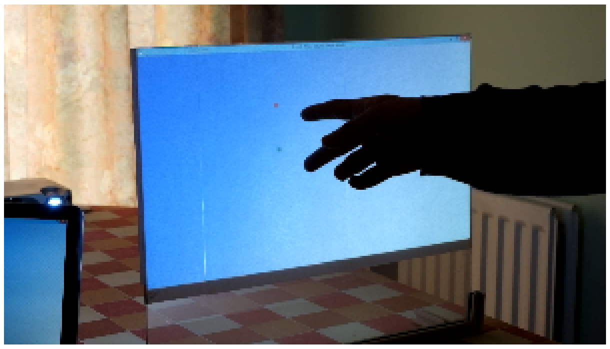
There is also a Calibrate button that takes the user through a sequence of steps, where they have to touch each edge of the screen once, in order to calibrates the values in one go.

We then have a button which controls the status of the cursor, including whether cursor movement, clicking and scrolling are enabled or disabled. There are four statuses that the cursor can be set to: "Cursor Enabled Only" allows movement of the cursor without mouse click, "Cursor Enabled, Click Enabled" allows movement and clicking of the mouse,


Cursor Mode With Click

“Cursor Enabled, Scroll Enabled” allow the user to scroll across the screen using hand gestures, and “Cursor Disabled” disables the cursor entirely.


Depth Mode In Use


Multi Mode In Use

Underneath the cursor status button, there is the Depth Mode button which opens a new window that represents the cursor as an ellipse. This ellipse changes size depending on the distance of the user’s finger from the screen, becoming smaller as the user’s finger moves away from the screen. When the user’s finger is very close, the ellipse changes to a rectangle as well as changing colour. This allows the window to behave as though it is pressure sensitive.

Pressing on the Multi Mode button launches another window containing a blank canvas. Ellipses are drawn at multiple points to represent multiple fingers. Each ellipse is given its own colour, although the functionality to decide which ellipse should have which colour is incomplete, hence the incomplete label.

The last button, the TouchDevelop button, launches TouchDevelop in the default browser for easy access.

Below these buttons, the statuses of the Kinect sensor are shown. When the Kinect detects the user’s finger, a status is shown that gives the X, Y and Z values of the point found. If it cannot see the user’s finger, “No points found” is displayed.

The status at the bottom right of the window displays whether or not the application can see the Kinect. If the Kinect is not found, “Kinect not available!” is displayed. If the Kinect is connected, the status says “Running”.

We also implemented a scroll zone to the right side of the screen, in the style of a scroll zone on a touchpad. Its exact location can be seen in the diagram to the right.

As we realised that reopening the application to change a setting would be inconvenient, we opted to include a verbal user interface.

The user is able to start the full calibration process with their voice, as well as cancelling it, should the calibration process go wrong, or if the calibration process was started accidentally.

The cursor mode can also be adjusted, with separate commands to turn the cursor movement on and off, turn clicking on and off, and turn scrolling on and off independently. When turning on scrolling, the click functionality is turned off, as scroll mode and click mode are separate entities.

Depth Mode and Multi Mode can also be launched and closed using voice commands. If one is launched when the other is open, the currently open one will be closed. Moreover, if a request is sent to open a mode that is already open, then nothing happens.

The final voice command is to launch TouchDevelop in the default browser of the computer for mouse free access.

So that the user gets the best experience from the UI and speech recognition, we added a "Help" window, that can be opened by pressing F1. This gives a list of commands for voice control as well as a description of what the user can do with each button.

8.5. Design Patterns

8.5.1. Singleton

The singleton design pattern aims to restrict the number of instances of a class to only one. This can be useful when only one object is desired, or when multiple objects of the same type can lead to disastrous effects.

It has been implemented in several areas in our application, namely the `MultiTouchWindow`, `DepthTouchWindow`, `NullCursor` and `UseCursor` classes. Each of them contain private constructors, preventing external classes from being able to create a new instance.

They also contain a getter method that initializes the object by calling the constructor if the object has not already being initialized, and then returns a reference to the object. A static variable holds a reference to the object, and contains a reference to the singleton object after the getter method has been called once.

We decided to use this design pattern as we didn't want any confliction between any multiple instances, as well as avoiding the need to create a new object on every frame.

8.5.2. Null Object

The null design pattern involves the use of an object defined with methods that have a lack of commands to execute. It is particularly useful in avoiding the need to check for null references as one can instead use an object that has null behaviour.

It was used to make the logic between cursor modes easier to understand. Our application features two cursor classes, `NullCursor` and `UseCursor`. Whilst `UseCursor` is fully functional, `NullCursor` contains the same methods, but with no statements executed.

Thus, when the cursor is disabled, the application assigns a local variable of type `ICursor` to the singleton `NullCursor`, whereas when the cursor is enabled, the same variable holds a reference to `UseCursor`.

We decided to use this design pattern as it allowed for cleaner code, as well as the convenience of no longer needing to use if blocks to check for `null` references.

8.5.3. Template Method

The template method design pattern is used to create a skeleton method that can then be specialized in subclasses that inherit it. This means that we could create a single super method that would contain the similar characteristics of its subclasses, and then specialize.

We have used this technique in the creation of the `MultiTouchWindow` and `DepthTouchWindow`. They share many properties: they require a static variable that holds a singleton reference to itself, they both require a canvas item to be able to draw onto the screen, and they both require a method to be able to draw onto the canvas.

Thus we created a `CanvasWindow` abstract class. This class holds all of the properties that both `MultiTouchWindow` and `DepthTouchWindow` share. Its subclasses meanwhile specialize in deciding exactly where and how large the circles displayed on the screen should be.

We decided to use this design pattern as allowed for cleaner code as it reduced the need for duplicate code.

8.5.4. Factory Method

The factory method design pattern is used to deal with the problem of creating objects, without specifying the desired class that will be created.

We have used this technique in order to allow the code interact with the cursor, and not know whether or not the user has selected the cursor movement to be enabled or not. This was achieved with a Factory method that returns a reference to either `NullCursor` or `UseCursor`, depending on the user's choice.

As these two objects are also singleton classes, the factory method actually doesn't instantiate a new object, only getting a reference to it. Thus the code actually asking for the cursor to be moved has no idea whether or not the cursor has moved.

This design pattern was implemented in order to separate dependencies in the code as well as to satisfy object oriented philosophies.

8.5.5. Facade

The facade design pattern is used to simplify interactions where desired. In our case, we have used them in order to make updating the UI text easier.

In using this technique, we have simplified the code, and made it more readable. We have the `UpdateLabels` method that calls many smaller methods, each independent of each other. Each of these smaller methods is in charge of updating a small subsection of the UI, so making a change to any single part of the UI is simplified.

Another area the facade design pattern has been implemented is in the process of calibrating. The screen class takes care of the entire process of calibrating, and also contains a previous state that can be restored if the user decides to cancel. This could be expanded to use the Memento design pattern, but we were constrained for time.

This design pattern was implemented in order to make the code easier to read, whilst also making it easier to modify in the future.

8.5.6. Memento

The Memento design pattern is used capture an object's state without violating encapsulation whilst allowing the the state to be restored at a later time.

Our application is able to adjust its internal values for the position of the screen using a calibration system that we have implemented. However, it is possible that the user would want to cancel any changes he or she has done. Thus, we have applied the memento design pattern to facilitate that functionality.

The `Screen` object creates a `ScreenMemento` object that contains the current state of the screen size when calibration begins. Thus, the currently saved position of the four edges of the screen relative to the Kinect is stored in the memento object. If the user decides to cancel halfway through, then a `CancelCalibrate` method is called that restores the information contained in the `ScreenMemento` object.

We used the memento design pattern in order to simplify the means to be able to undo changes in the process of calibrating the application.

8.6. Algorithms

8.6.1. Filtering Noise

Throughout the development of our application, we have put a lot of effort to minimise the effect of noise. In doing so, the responsiveness of our application increases. The first step we took to minimise noise was to apply post filtering.

8.6.1.1. Moving Average

This was the first method of filtering applied to make the point detected smoother. The algorithm implemented very fast in a very crude way. Whilst it worked, it wasn't written with efficiency in mind. The implementation starts with an array with a fixed size. Then new values are entered, and the position at which it is entered is iterated. This provides a way of storing a log of fixed length of previous values. These values can then be used to find a moving average.

If the number of previous values used in calculating the mean is sufficiently large, then the effects of noise are greatly reduced. However, the side effects to doing this that make this approach impractical is that the moving average shifts any real time influences by a factor proportional to the size of the log. This gives the cursor the impression that it is being dragged through water.

On the other hand, if the number of previous values is too small, then noise would be a greater influence, making the cursor move in the general area, but with some wobble. Given that the Windows interface consists of many small icons relative to a finger, it becomes very hard for the user to hit a button.

This technique of filtering was the one that was used at Milestone 1.

8.6.1.2. Exponential Filter

This was the second method of filtering that was applied to make the application smoothen the point detected. Implementing this for a single point was very simple, as only a single value needed to be stored between frames. The effect was immediate: the responsiveness of the cursor was greatly improved whilst at the same time reducing the effect of noise. With only the smoothing factor being the only variable to adjust, it didn't take long to settle on a default value.

At the time we integrated the exponential filter into the code, the multi point system had already been implemented, and it had had no smoothing filter on it at all. In order to apply tracking to multiple points, the code needs to know the previous point of a point if there is one. Creating an implementation of this took time as we found debugging frustrating.

This filtering was implemented after Milestone 2.

8.6.1.3. Kalman Filter

This is the filtering algorithm we were going to implement. However, time complications arose meaning that we were unable to implement this. This filter was in fact one of the suggested things to look into when feedback was given at the presentation at UCL Advances. The main reason we did not have the time to implement this filter was due to the trouble we had with getting filtering to work with the exponential filter for multi points. In order to be able to implement the Kalman Filter, the code will need to be aware of the previous points when tracking, the same requirement for the exponential filter.

The Kalman Filter is capable of taking a series of noisy data, and creating a statistically optimal estimate of what the current value really is. It uses past data to create an estimate of where it thinks the new point should be, and compares that to the actual sensor reading, and by means of a weighted average, generates its estimate.

The real strength behind the Kalman Filter is that the weights used in the weighted average react to how well or not well the past estimates have been. It can react to new noise levels, and due to its recursive nature, it is capable of running in real time with information about its current state and the previous state.

8.6.1.4. Dynamic Exponential Filter

After creating the exponential filter, we were very happy with it. After some thinking, it was realised that it could be improved upon in our case. The normal exponential filter uses a constant smoothing factor, however in this version of the algorithm we made uses a variable smoothing factor. We realised that we wanted the best of both worlds; the responsiveness of not having a smoothing factor combined with the smoothness that having a smoothing factor gives.

We noticed that when the user’s finger is moving slowly, responsiveness is not as important as smoothness, whereas when the user’s finger is moving fast, the responsiveness is more important. Thus we made the smoothing factor variable, making it depend upon how far the user’s finger has traveled since the previous frame.

The effect was immediate. This filter satisfied our desire for a responsive yet smooth point detection. This filter was implemented two days before the Milestone 3 deadline (27.04.2015)

8.6.2. Depth Analysis

We also spent a bit of time looking into how to analyse the depth data from the Kinect by using it in such a way to reduce noise.

Note: these algorithms are ones we came up by ourselves. Thus, the names of these algorithms are not “official” algorithms, and are custom designed for our application although may share traits with known algorithms.

8.6.2.1. Simple Minimizer

The first algorithm we implemented was to simply scan the data to find the point that is closest to the screen. It stores a temporary variable that holds a 3D coordinate that it currently thinks is the point closest to the screen, and checks it against all points in the depth data. For each coordinate in the depth data, if it is within the screen bounds, then that point is checked to see if it is closer, the temporary variable is updated to the closer point.

Whilst this algorithm worked, it was very noisy, and provided an unsatisfactory experience.

8.6.2.2. Cluster Finder

This algorithm is the first major attempt to analyse the depth data in a meaningful way. It works by first discarding all points that are not within screen bounds. Then for each of these points, it counts the number of points near it. Once complete, it finds the point that has the maximum number of points near it. If multiple points have the same maximum number of points, then the points are averaged.

This algorithm worked when it came to finding the correct point to use. Unfortunately, it was also very inefficient computationally, reducing the number of times the cursor is updated down to less than ten a second. Thus, whilst the point found was far more stable, the responsiveness of the cursor was worse than with the Simple Minimizer.

8.6.2.3. Mini Cluster Finder

The implementation of Cluster Finder showed that a better use of the depth data can indeed provide an improvement in the point picked for cursor control. With that in mind, we set about improving Cluster Finder. This algorithm is the end result.

The first thing Mini Cluster Finder does is to find the single point that is nearest to the screen using a similar process to Simple Minimizer. A new list of points around this point is created. The number of points to consider is now significantly reduced. Cluster Finder is then applied to these points.

The end result is an algorithm that gains the benefits of Cluster Finder without the computational cost.

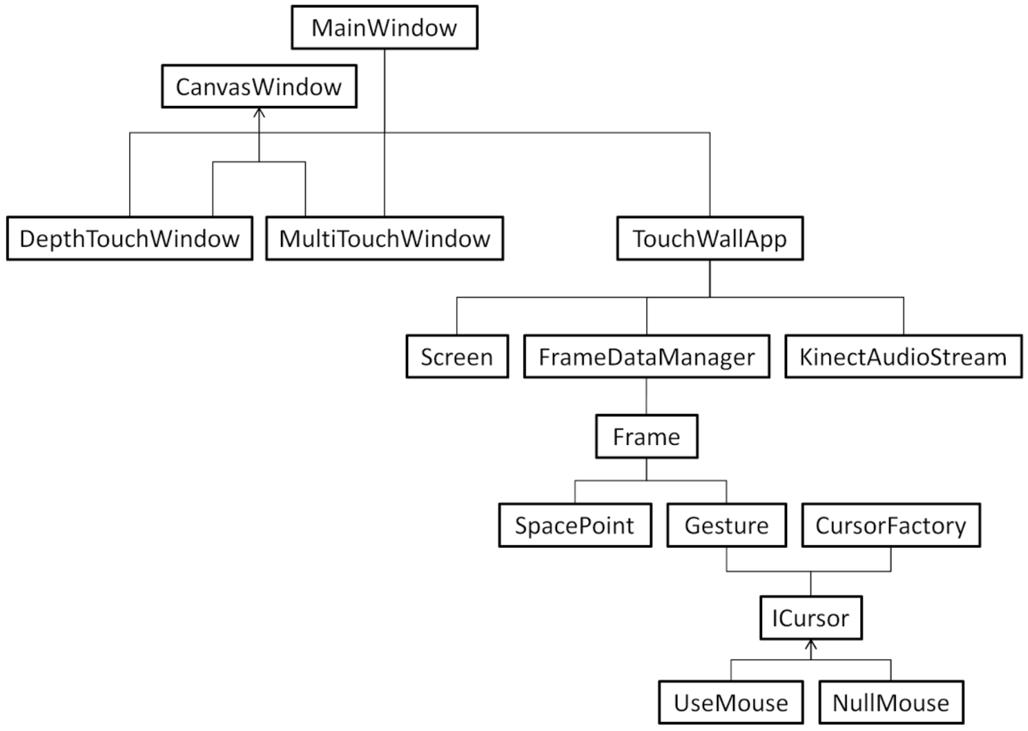
Sadly, expanding it to be able to find multiple points would be computationally expensive.

8.6.2.4. Simple Minimizer Extended

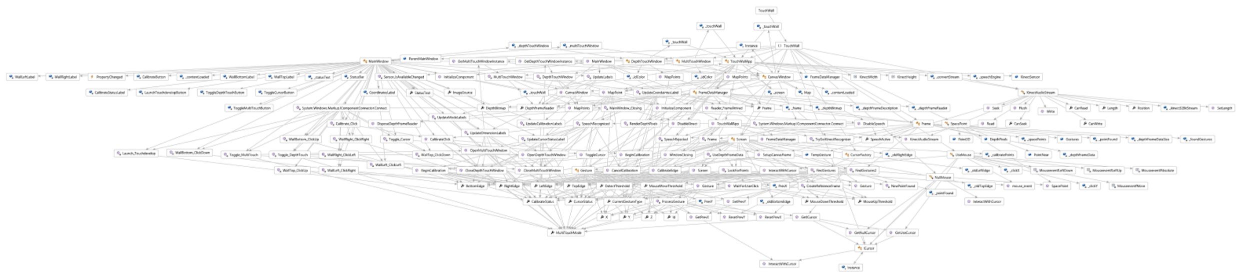
This algorithm is designed for multi point mode. Its method is similar to the Simple Minimizer algorithm described above, but it has been extended to search for multiple points. To find the first point, it uses the Simple Minimizer algorithm. The point found is then stored and the Simple Minimizer algorithm continues with the remaining data, the only difference being that it also checks to make sure that the new point is also not near any stored points. New points found then get stored. Once the maximum number of points found, in our application this number is four, the points are then sent to be displayed on screen.

8.7. Code Hierarchy

Here is a greatly simplified version of the Code Hierarchy:



Here is the complete version:



PNG version of this image is available in the folder this pdf is contained in.
If not, it can be requested via davidxiao93@gmail.com

9. Usage

9.1. Installation Requirements

This application has the same hardware requirements for the computer running the application as the Microsoft Kinect 2 SDK.

These are, at the time of writing:

- Windows 8 or 8.1 (but not any beta version of 10)
- GPU (discrete or integrated) compatible with DX11
- 4GB of RAM or more
- CPU with at least 4 threads
- Compatible USB 3.0 Controller.*

Other required hardware requirements:

- Kinect 2 with adapter for use with a USB 3.0 port and an external power supply
- Tripod to hold the Kinect 2 vertically

Other optional hardware:

- Projector
- Surface for the projector to backlight

Software requirements:

- Kinect SDK 2.0
    - http://www.microsoft.com/en-us/download/details.aspx?id=44561
- Visual Studio 2013 (tested with Ultimate, unknown if Express is sufficient)
    - https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx
- Microsoft Speech Platform - Software Development Kit (SDK) (Version 11)
    - http://www.microsoft.com/en-gb/download/details.aspx?id=27226
    - Required for compilation
- Kinect for Windows SDK 2.0 Language Packs - enUS
    - http://www.microsoft.com/en-us/download/details.aspx?id=43662
    - Only enUS has been confirmed to work
    - enGB does NOT work for some unknown reason at the time of writing
    - Required for Voice Recognition at runtime
- Microsoft Speech Platform - Runtime (Version 11)
    - http://www.microsoft.com/en-us/download/details.aspx?id=27225
    - Required for Voice Recognition at runtime
- .NET 4.5 Platform installed
    - https://www.microsoft.com/en-gb/download/details.aspx?id=30653

** The Kinect 2 requires a compatible USB 3.0 Controller. The amount of bandwidth the Kinect requires is higher than some Controllers are capable of. To make sure that the Kinect can stream all the data, use the Kinect 2 Configuration Verifier that is included with the Kinect 2 SDK. If the final section "Verify Kinect Depth and Color Streams" gives a green tick, then the USB 3 Controller is capable of sending the required data for this application.*

9.2. Compilation Instructions

Simply open the .sln file using Visual Studio 2013. If everything has been setup correctly, you can start the application and begin using it.

If you get compilation errors, make sure that everything in the Installation Requirements section is fulfilled.

If you get a “Kinect not available!” when the application is running then use the Kinect Configuration Verifier to make sure that the USB 3.0 Controller you are using is compatible.

If you get a “Voice Commands Unavailable” when the application is running, then you are missing the runtime elements of the Microsoft Speech Platform. Refer to the Installation Requirements

9.3. Voice Commands

There are a number of voice commands available to the user. Here is the list of these commands.

Kinect Open TouchDevelop	Opens TouchDevelop in the default browser
Kinect Calibrate Enable	Begins the calibration process
Kinect Calibrate Disable	Cancels the calibration process
Kinect Cursor Enable	Allows the application to use the Kinect to control the position of the cursor
Kinect Cursor Disable	Prevents the application to use the Kinect to control the position of the cursor, clicking or scrolling
Kinect Click Enable	Allows the application to use the Kinect to control the position of the cursor and to control left clicks
Kinect Click Disable	Prevents the application to use the Kinect to control the clicking
Kinect Scroll Enable	Allows the application to use the Kinect to control the position of the cursor and to control wheel scrolling
Kinect Scroll Disable	Prevents the application to use the Kinect to control the wheel scrolling
Kinect Depth Enable	Launches the Depth Mode. If Multi Mode is open, it will be closed
Kinect Depth Disable	Closes the Depth Mode if it is open
Kinect Multi Enable	Launches the Multi Mode. If Depth Mode is open, it will be closed
Kinect Depth Multi	Closes the Multi Mode if it is open

10. Testing

10.1. Development Testing

We dedicated a lot of time to testing. Our application's main criteria was to allow for responsive user interaction. Most of the testing was making sure that we were always heading towards that goal.

We implemented a lot of debug information into early versions of the UI, giving us instantaneous feedback on whether the code was acting as it had been programmed to. We found this more useful in certain circumstances than using the IDE's built in debugger as it meant quicker testing.

The setup for testing was different to the one suggested by the client. We realised quickly that a panel backlit by a projector is equivalent to a flat panel screen. There was also the hassle to set up a projector, including plugging it in as well as darkening the room to the point where the screen would become visible.

Using various flat panel screens gave us the opportunity to test on different sized screens, both physically and in resolution. The panel that we had been given with the projector was limited in size and the projector was limited in resolution. Combined, these factors did not allow the Kinect 2 to show its potential.

Whenever we felt we had reached another milestone, ready for external testing, Prateek would give his verdict on the responsiveness on the application. Upon his satisfaction, he would ask people to test the application, gathering valuable feedback.

10.2. External Testing

Prateek has conducted tests throughout our development, inviting third party people to try out our application. When a person has finished testing, Prateek provides a questionnaire consisting of a score out of ten for Ease Of Use, Intuitiveness, and Responsiveness, as well as any further comments they wish to make.

10.2.1. After Milestone 1

Person 1

- Ease Of Use: 6/10
    - I found it hard to grasp as to what distance I should be away from the screen. Although moving the cursor felt less of a task after a while of getting used to it.
- Intuitiveness: 8/10
    - Brilliant until I wanted to use MS Paint. Moving the cursor was self-explanatory but I had to ask how to click and drag.
- Responsiveness: 5/10
    - Very pesky and impractical due as you had to move your finger very slowly or be patient as the cursor tried to catch up with you. It worked, but just not smoothly.
- Other comments

    - I think you should have a easy way of calibrating the system for different screens.

Person 2

- Ease Of Use: 7/10
    - Easy enough to move the cursor around the screen but rather difficult when attempting to left click, where it was unpredictable where the cursor would end up.
- Intuitiveness: 6/10
    - I can see how some people would have a problem with it.
- Responsiveness: 4/10
    - It was just too laggy for me.
- Other comments
    - The lag needs to be improved.

10.2.2. After Milestone 2

Person 3

- Ease Of Use: 9/10
    - Easy to use, but its lack of precision made it more frustrating that I would have liked to have seen.
- Intuitiveness: 8/10
    - I was very successful in using the click-and-drag feature. The calibration screen was self-explanatory.
- Responsiveness: 7/10
    - It responded to my finger better but still unreliable at times.

Person 4

- Ease Of Use: 8/10
    - Using the calibration screen was very simple to setup.
- Intuitiveness: 9/10
    - Seems easy enough to get an understanding of how to use this.
- Responsiveness: 7/10
    - The lag is quite noticeable.
- Other comments
    - Pointing at the screen for a certain amount of time caused my arm to ache. This isn't a flaw with the system itself but maybe something for you to think about.

10.2.3. Just Before Milestone 3

Person 5

- Ease Of Use: 8/10
    - Easy to setup the sensor to the screen dimensions.
- Intuitiveness: 9/10
    - I got the hang of using this very quickly.
- Responsiveness: 9/10
    - Very responsive correlation between my finger and the cursor moving.
- Other comments

    - Nice work.

Person 6

- Ease Of Use: 9/10
    - Everything seems to work with high efficiency – from the cursor speed to accuracy.
- Intuitiveness: 8/10
    - I’m not good with new technology, but this was very easy.
- Responsiveness: 8/10
    - Feels as though it really was a touchscreen.

11. Evaluation and Final Thoughts

On the whole, our team worked extremely well. We communicated ideas frequently through the use frequent of WhatsApp, and were able to resolve or react to problems very quickly. We are pleased with the progress we have made, and thoroughly enjoyed the project.

We all started with no knowledge of C#, and immediately thrown into the deep end with some source code for the Kinect 1. After familiarising ourselves, we moved onto a Kinect 2. Its development progressed rapidly, incorporating the basic functionality of the Kinect 1 version, and then adding a lot of new ideas.

The vast majority of the development was spent in ways of making the application more precise at tracking a finger. This meant we had plenty of time to research interesting techniques into how we could achieve this, from moving averages to Kalman Filters.

We really wish that we had gotten the Kinect 2 sensors sooner. We received the first one about two weeks after the official start of development, and we obtained the second one four weeks before the final deadline. In having limited access to the Kinect 2 sensors, we have not been able to distribute the workload completely.

We also wish we could find a way of being able to use multitouch beyond simply displaying on screen. We have dedicated quite a bit of time looking for documenation on this matter, and the only good resource we have found is http://social.technet.microsoft.com/wiki/contents/articles/6460.simulating-touch-input-in-windows-8-using-touch-injection-api.aspx which happens to use C++, not C#.

We would like to give special thanks to:

- Dr. Dean Mohamedally: Senior Teaching Fellow
    - For providing us the opportunity to create this application, as well as showing constant interest and giving great feedback
- Dr Yun Fu: TA for COMP103P
    - For providing us the opportunity to demonstrate to senior staff, as well as providing the second Kinect 2 sensor
- Dave Twisleton: UCL Technician
    - For providing the Kinect 1 sensor at the initial stages of the project, as well as the first Kinect 2 sensor
- Tycho Bickerstaff: First Year CS Student
    - For teaching us how to use the Git repository features inside Visual Studio, as well as helping us to understand how the Kinect 1 code worked.

12. References

Kinect 1 Source Code

- Provided by our client
- This code was designed to use a Kinect 1 sensor to convert a user's finger into mouse movements with the Kinect on the side of the screen. This code provided a lot of inspiration for how our application would work

MSDN Kinect 2 SDK Documentation

- https://msdn.microsoft.com/en-us/library/microsoft.kinect.aspx
- There was a lot of useful information contained here, clearly showing the various functions are accessible for the Kinect to use. The function we found most useful for our purpose was MapDepthFrameToCameraSpace as well as the looking up the various data structures

Kinect 2 Example Code

- https://msdn.microsoft.com/en-us/library/hh855380
- Depth Basics example code that taught us a lot about how to use the Kinect SDK. Until we did a lot of refactoring and rewriting, this example code provided much of the original foundation for our application.
- https://msdn.microsoft.com/en-us/library/hh855387.aspx
- Speech recognition example code that showed us how to implement it into code. We were very surprised at how accurate it is in practice.

mouse_event

- http://www.codeproject.com/Tips/371718/The-Kinect-Mouse-Controller-Csharp
- This example provided an explanation of how the mouse_event function worked to move the cursor. Sadly it was written with the Kinect 1 in a different position in mind.
- https://msdn.microsoft.com/en-us/library/windows/desktop/ms646260%28v=vs.85%29.aspx
- Official tutorial explaining how to use mouse_event
- This combined with the Kinect 1 Source Code and the CodeProjects Kinect Mouse Controller provided a lot of information giving a clearer picture on how it is used.

Kinect Depth Smoothing

- http://www.codeproject.com/Articles/317974/KinectDepthSmoothing
- A tutorial that demonstrated various techniques to make the depth data smoother, providing more accurate data

Exponential Smoothing

- https://en.wikipedia.org/wiki/Exponential_smoothing
- Resource used to study exponential filtering to see how it worked, and to come up with a simple implementation

Kalman Filter

- https://en.wikipedia.org/wiki/Kalman_filter
- http://greg.czerniak.info/guides/kalman1/
- Resources used to study the Kalman Filter. These two links represent where we looked for an understanding of it, but we were unable to use it due to time constraints

Canvas

- https://msdn.microsoft.com/en-us/library/ie/windows.ui.xaml.controls.canvas
- This allowed us to understand how to set the coordinates of Ellipses within the canvas to the mouse coordinates of each Gesture in multi touch mode

13. Appendice

13.1. Feedback from clients

Dean Mohamedally, our contact for our client, has often expressed positive feedback throughout the application's development. Sadly, we have no written record of the conversations we have had with him, so here is a summary of the major comments he has made:

1. Great work guys!
2. Do you think you can make it smoother?
3. Can you add in a calibration mode?
4. How large of a screen can this work on?
5. Make the UI look nicer
6. Does multitouch work?

13.2. Meeting minutes

Meeting 1
Date: 22/01/2015
Time: 1300
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

We met as a team for the first time to introduce ourselves to each other. We discussed our background regarding programming and previous projects. In doing so, it became a lot clearer to assign the roles appropriately. At this stage, we didn't know what project we would be given therefore we couldn't go into much detail about any particular ideas however, we did think ahead about arranging meetings with our potential client. We also agreed to assign Prateek as being the team leader.

Next meeting's actions: N/A

Meeting 2
Date: 29/01/2015
Time: 1100
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

We received details of the task we had been assigned during the Object-Oriented lab session. Our task consisted of converting a non touch sensitive surface into one that is by using a Kinect V2 Sensor mounted to the side of the surface. We had the opportunity to direct any queries at Dean in order to get as much information as possible to aid us in getting started straight away. After our meeting with Dean, we met up to talk through possible role changes taking the new information into consideration. No alterations were made as we were all content with the initial decision. We also discussed opportunities during reading week for possible meetings.

Next meeting's actions: Research Kinect SDK and set up a meeting with Dean

<u>Meeting 3</u>
Date: 05/02/2015
Time: 1100
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: Dr Dean Mohamedally

We had a group meeting with Dean, where he explained that the project we had been given is an extension of a project given to previous students. We were given the Kinect V1 Sensor along with the code created by the previous students. We then spent time analysing the code to get an understanding of how it works; we then proceeded to edit some of it to our own liking by renaming some functions and variables. The difference in programming knowledge between team members became apparent so in order to overcome this, the more skilled member(s) of the group assisted by answering any questions.

Next meeting's actions: Be familiar with the code by going through it individually

<u>Meeting 4</u>
Date: 09/02/2015
Time: 1300
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

During this meeting, the main topic of discussion was how to improve the code. Each member gave their input on what could be done to improve the program in any way. The problem we had at this stage was the end product needed to work on a Kinect V2 Sensor but we only had access to Kinect V1 Sensor therefore there was a lack of assurance in the program working once we got the new hardware. However, in discussing the code, we had a solid foundation of ideas upon which we could build for the Kinect V2 Sensor.

Next meeting's actions: Try to improve the code individually to share with the group during the next meeting

<u>Meeting 5</u>
Date: 12/02/2015
Time: 1300
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: Dr Yun Fu

Ideas of refining the code were shared to the team which was followed by feedback. We were able to get an approximate four-fold increase in precision which worried us as we we not able to determine the reason the original coders had for not doing this. After some communication with Dean via Yun Fu, we were able to obtain a Kinect V2 Sensor, but its conversion to USB was not complete. Due to several commitments the following day, we agreed to spend time on the project during the week commencing 16/02. One of the reasons for the team meet was due to the fact we only have access to one Kinect therefore we decided it's best to work on it together.

Next meeting’s actions: Work on the app design individually and collate them during the next team meeting. David is also to fix the Kinect V2 Sensor so that it would work in Windows.

Meeting 6
Date: 15/02/2015
Time: 1000
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

After much discussion over the choice of location for this meeting in our WhatsApp group, we met up at Mark’s home. As David had successfully brought the Kinect V2 Sensor to life, we connected the Kinect to a tripod that David brought with him and we set it up next to a TV. We felt that a TV was a close approximation to a backlit projector, and would be sufficient for preliminary testing. We also successfully ran a few of the sample code from the SDK Browser. Satisfied that the Kinect was working, we then discussed what needed to be done for the next deadline (20/02/2015), how to allocate the workload and when our next meeting would be.

Next meeting’s actions: Mark is to create paper sketches that showcase how the Kinect would enable a flat surface to be touch sensitive, David creates the digital versions of these sketches and Prateek writes up the design report.

Meeting 7
Date: 18/02/2015
Time: 1100
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

After meeting up at David’s home, we sat down together to write the design report in Google Docs which would allow all of us to edit a single document at the same time. Following the guidelines, we split up the workload accordingly. The first section required a lot of writing, so Prateek took that part. The Requirements and the Technical description were written by David, whilst the title page and the design sketches and current status sections were fulfilled by Mark. At the same time, we discussed ideas of how we could make the finger detection more accurate by using better data interpretation.

Next meeting’s actions: Prateek is to submit the design report and David and Mark are to continue researching ways to make the finger detection better.

Meeting 8
Date: 24/02/2015
Time: 1230
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

We discussed ideas on implementing an efficient calibration function within the UI. It was agreed that this would be the best thing to do to save time when using the Kinect on different screens, with varying resolutions. The basic idea is to have buttons with the ability to change values determining the working size of the screen. In regards to ways in improving the finger detection, additional research will have to be done.

Next meeting's actions: All team members to build on the idea of calibration and improving finger detection

Meeting 9
Date: 26/02/2015
Time: 1300
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None
We shared notes on our individual research and discussed further ideas on calibration. David was able to alter the code to improve the finger detection accuracy. The interface was now able to work with a lot more precision than with the code we had been given. We were also able to change some of the code even more to our preference allowing us to be more comfortable with its layout.
Next meeting's actions: Make enhancements to the calibration/interface before next meeting

Meeting 10
Date: 03/03/2015
Time: 1000
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: Dr Dean Mohamedally

As we were able to make the enhancements to the calibration as planned in the previous meeting, we were able to discuss any further potential the touch sensory system may have. We decided implementing a multi touch functionality would be useful to provide interaction for more than one person. We had a meeting with our client, who also gave us plenty of ideas for further development. For testing purposes, we were given a projector and a screen panel to project onto.

Next meeting's actions: We are to conduct research on multi touch functionality

Meeting 11
Date: 06/03/2015
Time: 1400
Attendees: David Xiao, Mark Menezes
Other Attendees: None

During this meeting, we realised that the code that we had so far developed was very "smelly". The quality of the code was not up to standard that we would have liked. We decided to postpone the implementation of multi touch, and we agreed to work together to refactor the code over the weekend. This made further development on the code easier due to increased readability and greater modularity.

Next meeting's actions: Be more familiar with the new code individually and try to introduce further changes, if any

Meeting 12
Date: 09/03/2015
Time: 1300
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

In this meeting, we talked through our work on the code over the weekend along with the routine sharing of notes. Further minor changes were made. After much discussion we agreed that implementing a multi touch system required further research and we decided to tackle the user interaction of the calibration interface by using speech recognition in the mean time. We also discussed ideas on how to tackle the one of our client's idea of having one Kinect sensor to take input from multiple panels.

Next meeting's actions: Research detecting multiple objects in the Kinect's field of view

Meeting 13
Date: 11/03/2015
Time: 1100
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

The Kinect's speech recognition was a big talking point in this meeting. Having made excellent progress so far, the team was very eager at this stage to implement speech recognition into the code. We decided to use speech recognition to provide a convenient way of bringing up the calibration system without having to physically interact with the system.

Next meeting's actions: Come up with further ideas of possible uses of the speech recognition

Meeting 14
Date: 13/03/2015
Time: 1000
Attendees: David Xiao, Mark Menezes
Other Attendees: None

As none of us had come up with any useful ideas for speech recognition that would be beneficial to our project, we decided that the time was right to have an attempt at multi touch. As we knew Windows 8.x had no built in API to allow for multiple cursors, we decided that for the time being, we would create a windowed application for Windows that would use the Kinect to draw circles where it detects an object. This application would thus be able to showcase the use of Kinect detecting multiple fingers and converting them into useful coordinates.

Next meeting's actions: Begin writing code to make a window upon which we can demonstrate the Kinect detecting multiple objects

Meeting 15
Date: 17/03/2015

Time: 1000
Attendees: David Xiao, Mark Menezes
Other Attendees: None

After a busy weekend for us, we were successful in creating a simple implementation of multi touch. This version is only capable of detecting where the fingers are within a given area of 3d space. We were able to do this with about the same accuracy as with a single finger, thus limiting the accuracy of the finger's coordinates on the screen to the resolution of the Kinect itself. The downside of this current implementation is that it only detects the fingers presence, it does not implement any kind of tracking mechanism which would be the next logical step.

Next meeting's actions: Come up with ideas on how to track the finger

Meeting 16
Date: 18/03/2015
Time: 1200
Attendees: David Xiao
Other Attendees: Dr Dean Mohamedally

Dean asked if one of our members, David, could present our progress to a group of senior Microsoft employees and a group of fourth year students. David accepted, and was able to provide a demonstration. David feed back positive feedback from the progress that we had made, highlighting Dean's (our client's) enthusiasm for our progress. The main source of criticism came from the unresponsiveness of the detection algorithm under certain situations.

Next meeting's actions: Design a data model that would be suitable to allow for finger tracking

Meeting 17
Date: 19/03/2015
Time: 1100
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

At this meeting, we were due to demonstrate our application to our TA and to UCL Advances. Unfortunately, David forgot to bring the Kinect, the heart of the application. As such we were unable to make our demonstration. Thankfully, our client Dean made a recording during the previous meeting, which we directed our TA to look at. With Dean's confidence, we were given full marks. We were also asked questions by a member of UCL Advances on user interaction, namely in reducing the latency between the user moving his/her finger and the cursor moving on screen, explaining that since the popularity in touchscreens have boomed, the majority of users expect fluid and responsive interfaces.

Next meeting's actions: Come up with ways to punish David for not bringing the Kinect.

Meeting 19
Date: 25/03/2015
Time: 1300

Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

During this meeting David demonstrated his improved implementation of multi touch, based off the work that was mainly achieved by Mark. The improved version was a lot more responsive, and no longer gave false readings. Whilst this represented progress, we had still not come up with a suitable model for storing Points that would be necessary to allow for tracking of individual points on screen. Punishment was also dealt to David for not bringing in the Kinect by making him buy some beers for the other members.

Next meeting's actions: Think of some more punishments for David, come up with further areas of interest we could expand the project, and continue thinking of ways to track individual fingers.

Meeting 19
Date: 26/03/2015
Time: 1100
Attendees: David Xiao, Mark Menezes
Other Attendees: Dr Dean Mohamedally

After another meeting with Dean, we have been asked to create a video that demonstrates what the program does with a Kinect. This video is to be presented at a Microsoft Paris presentation. After this meeting, we discussed how we are going to create the video, as well as what content we would like to put into it. These discussions took some time, as we knew that this video will be presented at a professional event

Next meeting's actions: start drawing up ideas for a video script.

Meeting 20
Date: 31/03/2015
Time: 1000
Attendees: David Xiao
Other Attendees: Dr Yun Fu

As David had forgotten to bring the Kinect 2 Sensor to the demo on Thursday, he was asked to go to UCL Advances, and demonstrate there. Dr Yun Fu took David to the meeting and the demonstration was met with positive reviews. However, as they focused on the business side, they expressed doubts about the advantage of using a Kinect 2 Sensor over alternative methods. After much discussion, they toyed with ideas on potential expansions for what the Kinect 2 Sensor could do. After the meeting, Dr Yun Fu kindly lent his own Kinect 2 Sensor, meaning we had two Kinect 2 Sensors to develop with, allowing faster development.

Next meeting's actions: Improve the accuracy of the Finger detection, as that was the main piece of negative feedback from UCL Advances

Meeting 21
Date: 01/04/2015
Time: 1100
Attendees: Prateek Dahal, David Xiao, Mark Menezes

Other Attendees: None

In order to improve the finger detection, we decided to employ two different techniques in order to improve the finger detection. Our first method of attack was to use simple techniques to ensure that the point that is detected is connected to something. This will help to remove extra noise. The other technique we are using is to use post filtering to make the point less jolty. During this meeting we successfully implemented exponential filtering, as opposed to the moving average filter that we had been previously using. However, by the end of the meeting, the implementation was not finished.

Next meeting's actions: Finish implementing the exponential filter for single touch

Meeting 22

Date: 03/04/2015

Time: 1000

Attendees: Prateek Dahal, David Xiao, Mark Menezes

Other Attendees: None

By this meeting, David had successfully finished implementing the exponential filter, and had done preliminary research into a Kalman Filter as recommended by a member of UCL Advances from Meeting 20. The next task was to implement this smoothing algorithm to multiple points. This requires each point storing their previous point in order to allow exponential smoothing to occur. In order to do this with multitouch, the program will need to be able to track the movements of points, detecting which point from the previous frame joins with the points in the current frame. Initial modifications were made to form the foundation for this.

Next meeting's actions: Think about how to track the points between frames.

Meeting 23

Date: 06/04/2015

Time: 1300

Attendees: David Xiao, Mark Menezes

Other Attendees: None

During this meeting, initial ideas of tracking points were discussed. More specifically, ideas on how to determine if a point matches with a previous point or not. We had a long discussion on how to determine the state of each current point, as there are several possibilities, such as if two points in the current frame were the exact same distance from a previous point, what should be done.

Next meeting's actions: Think about and attempt to implement a tracking system

Meeting 24

Date: 09/04/2015

Time: 1400

Attendees: David Xiao, Mark Menezes

Other Attendees: Dr Dean Mohamedally

We demonstrated our progress since the last demonstration to our client, Dean. He was pleased to see that we now support the ability to see how close the user's finger is to the surface. The lack of comments that the pointer was jolty was a positive indication that the new filter was doing its job. Our client also provided some potential ideas into which our project could expand into, whilst feasible within the time provided. After demonstrating to the client, we sat down and started to work on our Milestone 3 report.

Next meeting's actions: Make progress on the Milestone 3 report.

Meeting 25
Date: 10/04/2015
Time: 1100
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

Whilst David continued to make progress in making the tracking better, Mark and Prateek decided to resume discussions on the video. An initial script was created, and there were further discussions into filming locations and time. When we turned to discuss who would do the editing, Mark volunteered to do the video editing with help from Prateek. As the video is to be submitted before the reports (Group & Individual), we decided to prioritise that whilst also working on parts of the reports.

Next meeting's actions: Decide on location/time via WhatsApp and record the video followed by editing it in time to be uploaded before 15/04/2015

Meeting 26
Date: 13/04/2015
Time: 1300
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

We felt that we hadn't done enough for the Milestone 3 report, so we all sat down and worked on it. The use of Google Docs meant that we could all edit the same file at the same time. By the end of the meeting, the Abstract, Context, Workplan, Team Summary, and Requirements sections were finalized.

Next meeting's actions: Continue working on the Milestone 3 report

Meeting 27
Date: 15/04/2015
Time: 1100
Attendees: David Xiao, Mark Menezes
Other Attendees: None

On the 14th, we were reminded that we needed to create a video demonstrating our application. Thus, in this meeting, we started early to film the application in action, as well as to edit it and create a voice over. The submission was made moments before the deadline. During filming, we realised that the single touch detection was not as complete as we had thought.

Next meeting's actions: Make the single touch detection more reliable

<u>Meeting 28</u>
Date: 17/04/2015
Time: 1100
Attendees: David Xiao, Mark Menezes
Other Attendees: None

During this meeting, David was keen on making the single touch detection method better. The method in question has already seen multiple complete rewrites, but he insisted on creating another. He delegated roles to the others, and by the end of the meeting, we had created a new version that was far more accurate than the previous version. More discussions about the Milestone 3 report took place.

Next meeting's actions: Make progress on the Milestone 3 report, and test the new method

<u>Meeting 29</u>
Date: 21/04/2015
Time: 1600
Attendees: David Xiao
Other Attendees: Dr Yun Fu

On the 20th of April, Dr Yun Fu sent out an email requesting David attend a meeting to demonstrate the progress we have made in the application, as well as making sure that the application worked on this own laptop for when he needs to demonstrate our application at a Paris meeting. Setting up the application was a good opportunity to see what is required in order to allow the application to compile and run. The main feedback came from the basicness of the UI as well as a desire to have the depth sensing abilities separate from multitouch mode. He also asked that we change the video we made, citing the long pauses as the main problem

Next meeting's actions: Make the UI look better and include depth mode

<u>Meeting 30</u>
Date: 23/04/2015
Time: 1400
Attendees: David Xiao, Mark Menezes
Other Attendees: None

Upon Dr Yun Fu's request, we set about separating depth mode and multi touch mode. The process was simple to do, and thus we were able to spend a bit of time working on Dr Yun Fu's other request of making the UI more pretty. This took longer than we expected as we all had different opinions on what the UI should look like. However, we are all happy with the end result.

Next meeting's actions: Prepare filming for the second video

<u>Meeting 31</u>
Date: 25/04/2015
Time: 1100

Attendees: David Xiao, Mark Menezes
Other Attendees: None

Due to the changed UI, we needed to refilm the application for the demonstration video. After resetting up the Kinect and projector, we showed the newest version of the application. Moreover, as the time constraint was not as much of a factor as before, we were able to film more of the functionality than the previous time. David also then edited the video as well as adding a voice over.

Next meeting's actions: Finalize submissions

<u>Meeting 32</u>
Date: 27/04/2015
Time: 1300
Attendees: Prateek Dahal, David Xiao, Mark Menezes
Other Attendees: None

With the Milestone 3 report and the application complete, we sat down to press the submit button together.

Next meeting's actions: NA