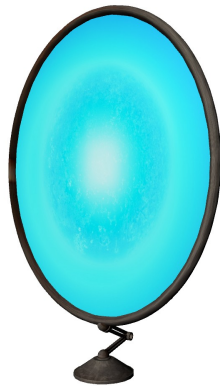


# **MAGIC MIRROR**



**M1560210**

**Gesture Controlled Touchless Interactive Display for Virtual Try-on**

**CS3283 Final Report**

## Report Contents:

- 1 Introduction
  - Abstract
  - Target Users & system installation
  - Requirement & solution
- 2 Related Work (*Links*)
- 3 Design Phase
  - General functionalities
    - Fitting room menu
    - Browser menu
    - Recycle bin menu
    - Photo catalog menu
  - paper prototype (1st phase)
    - Wardrobe menu
    - Fitting room menu
    - Recycle bin menu
  - digital prototype (2nd phase)
    - Use Cases
    - Magic Mirror Architecture
    - Internal Structure
    - Meeting user requirements
- 4 Re-modified Prototype
  - Changes to initial design
    - re-arrangement of icons
    - addition of cursor (tentative)
  - Evaluations
- 5 Discussion (*Future direction*)
- 6 References

Annex B (*API*)

# 1. Introduction

## **Abstract**

The magic mirror was first conceptualized as a virtual wardrobe and mirror that allows users to freely interact with it without any pointing devices, touch screen devices or keyboard. In implementing it, input is made through motion-sensor devices, namely a Microsoft Kinect. An LCD screen of appropriate dimensions is used for output display, and users use recognised hand gestures to interact with the system. The main use of this system is to let users browse for clothes and apparels within the system itself, then subsequently try out those clothes on their “body”, a mirror image projection on the display, once they have selected some. The displayed content consists of icons, information of clothes, and also most importantly, the mirror showing the users’ reflections in said clothes.

## **Target Users & System Installation**

The main target audience for the Magic Mirror would be frequent shoppers, mainly the technologically-adventurous young adolescents and adults ranging between 16 - 39 years of age. Shoppers can both be male or female in this context. This system can thus be installed in shopping malls or at home for users’ convenience. The target group are also assumed to have frequent usage of ICT gadgets such as smartphones, such that they can apply their knowledge in using the proposed system.

## **Requirement & Solution**

Since the target user group is one which is exposed to the latest in technological gadgets, it has been assumed that they are familiar with interacting with other HCI gadgets such as smartphones and tablets. This group of users would expect more than just functionality in terms of user-friendliness and user-experience. With this user requirement in mind, we set out to achieve fluidity in the system’s runtime design, particularly in the GUI and menu navigation. The NUI employed will be intuitive and easy to use for them. The solution, in the form of a “Magic Mirror”, is an alternative to answer the demands of the target group, with the various user-experience elements supporting the basic usability functions.

## 2. Related Work

### **Fitting Reality - <http://fittingreality.com/>**

Fitting Reality is a Russian-based technology company with the main goal of developing a virtual fitting room that will cater to the shopper's needs. Their current model, theVIPodium is based on the Kinect System, featuring an augmented reality built on the Kinect's depth and sensor technologies. A 3D avatar is modeled based on image data from the data, as well as, some user supplied demographic data. All clothing is fitted to the avatar which is synced with the user's movement in real time, virtually "dressing" the user according to items he may select from a catalogue maintained on a cloud server.

The company's model reveals vast potential for growth in this field and our team aims to build on the successes of their current user interface, by further challenging the capacities of the Kinect's gesture systems, for the purpose of extending the usability and enhancement of the user experience.

### 3. Design Prototype Phase

There are 2 design phases for the system that we decide to implement. The first prototype was designed on paper and the second prototype was digitized and developed using Microsoft C# and other relevant Kinect libraries.

#### General Functionalities

Before we get to design the prototypes, some use case functionalities are being considered when users get to use the system (i.e. what the users would do when at each menu) They are as follows:

Note: The starting default menu is always the browser menu when the program is launched.

#### When the user is in the Fitting room menu

- Access Browser/Catalog menu
- Access Photo catalog menu for previously taken pictures
- Access shopping cart wheel for clothes to “wear”
- Access Camera feature to take photos
- Add unwanted clothes to recycle bin
- Access Recycle Bin menu for previously discarded clothes

#### If the user is in the Browser/Catalog menu:

- Access Fitting room menu
- Access Men’s clothes and add them to cart
- Access Women’s clothes and add them to cart
- Access Newly released clothes and add them to cart
- Access Sales clothes and add them to cart
- Access Price of clothes for their information
- Access Photo catalog menu for previously taken pictures
- Browse through the shopping cart wheel
- Add unwanted clothes to recycle bin
- Access Recycle bin menu

#### If the user is in the Recycle Bin menu:

- Remove clothes permanently
- Restore the clothes back to the shopping cart wheel
- Go back to previous menu

If the user is in the Photo catalog menu:

- Browse photos
- Go back to previous menu

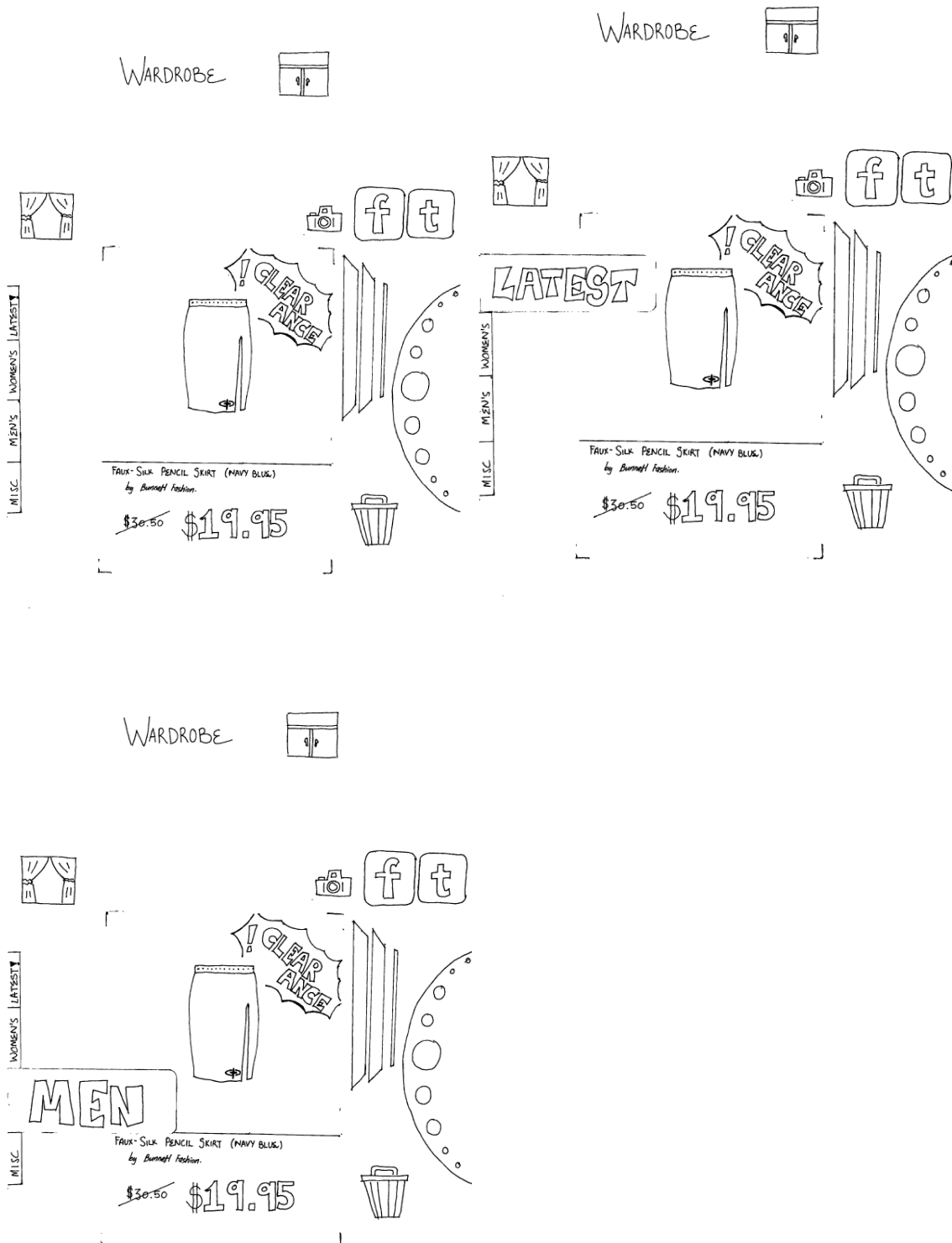
These user functionalities are largely unchanged (except for some minor tweaks when it was ported over from the paper to digital format, see Section 4) when we implemented the prototype. The core and essence of the system remains the same, and we target to achieve a smooth experience for users when they access this system, whether from the comfort of their homes or in shopping malls.

### **Paper Prototype**

This is the first original prototype that we designed to cater to the user's need and feel when using the system.

The focus of the paper prototype is on the GUI and aesthetics design methodology, whilst keeping in mind the target group's using habits and demographics.

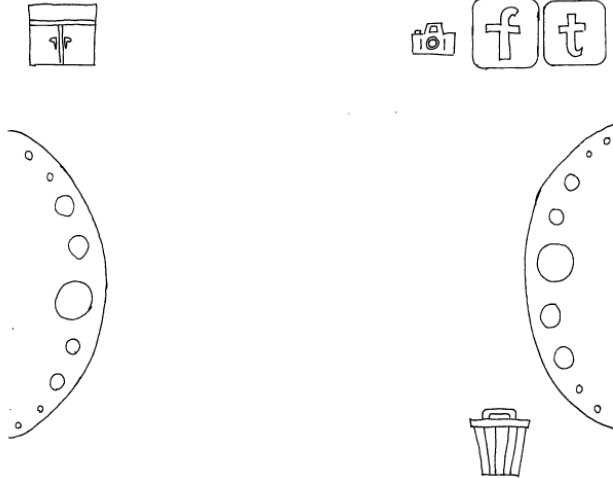
## Wardrobe menu:



Above shows the example when the user hovers his hand over the icon categories at the left hand side.

Fitting room menu:

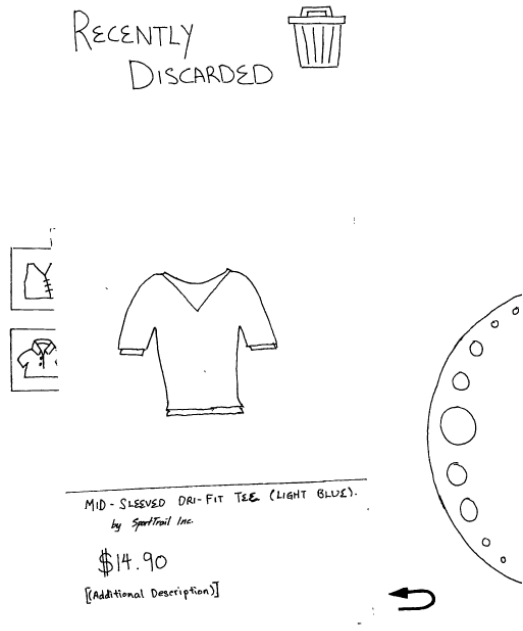
FITTING ROOM. 



The fitting room menu when the user navigates into it. The user can choose to pick which clothing he likes to wear, and take a photo from the camera icon above.



### Recycle Bin menu:



The recycle bin menu with clothes selected. Users can exit this menu and navigate into other menus as well.

Reasoning for the particular GUI layout and placement of icons are at the end of this chapter. Do note that the naming convention and shapes for the menus might have changed when ported over to the digital format for some of them, but functionalities remain mostly the same. For more information regarding the initial paper prototype design, please refer to Annex D.

### **Digital Prototype**

The paper prototype is converted into the digital format on the PC after some research and findings on Kinect technologies. Basic functionalities are transferred from the paper prototype over to the digital format, but more complex functionalities are simplified into basic functions due to the early exploratory phase. Hence, more sophisticated functions such as Facebook and Twitter sharing are not implemented in this phase. They will be integrated, modified and enhanced in the next phase after testing and improving the system. There are also some minor adjustments in the shape of the icons after the design is being ported, since we took aesthetic design aspects into consideration when

the prototype is being ported over. More details of this justification is at the end of this chapter.

### Magic Mirror Use Case

The following use cases will be detailed to showcase how the program works when the user interacts with it.



System: Magic Mirror

Use Case: Choose clothes to pick in browser menu

MSS:

1. User navigates into "Man" menu (hovers his hand across the icon)
2. System displays the Man clothing catalogue for view.
3. User hovers his right hand on the arrows (left or right arrows) to browse through the men clothes catalog
4. System displays next/previous mens' clothing based on user's hand navigation over the arrows
5. User found a clothing he likes and hovers his right hand above the "add to cart" icon
6. System adds the selected clothing into the shopping cart wheel.

Users may repeat steps 3 and 4 until they find a clothing they like. Users may repeat steps 3 to 6 to pick other clothings.

\*(a) at any time the user hovers his hand over other menus (fitting room, recycle bin, photo catalogue)

\*(a) 1. System redirects user to selected menu

System: Magic Mirror

Use case: Choose clothes to wear from cart in fitting room menu

Pre-cond: Clothes are present in shopping cart wheel

MSS:

1. User hovers his right hand on the arrows (up or down arrows) at the shopping cart wheel to browse through previously selected clothes
2. System displays next/previous clothings based on user's hand navigation over the arrows
3. User found a clothing he likes and hovers his hand over to "wear" it
4. (System fits the clothing over to the user's body in the display)
5. user hovers his hand over to camera
5. (a) 1. refer to camera use case.

User may repeat step 1 and 2 until he finds a clothing he wants to "wear"

\*(a) at any time the user hovers his hand over other menus (fitting room, recycle bin, photo catalogue)

\*(a) 1. System redirects user to selected menu



System: Magic Mirror

Use case: Take self shot from camera

MSS:

1. User hovers his right hand on the camera icon
2. System (counts down from 3 seconds), then take a snapshot of the user with/without the clothes on. Picture file is saved in local drive

\*(a) at any time the user hovers his hand over the back button

\*(a) 1. System redirects user to previous menu

System: Magic Mirror

Use case: Delete clothes permanently from the recycle bin

MSS:

1. User hovers his right hand on the arrows (up or down arrows) at the recycle bin wheel to browse through previously deleted clothes
2. System displays next/previous deleted clothings based on user's hand navigation over the arrows
3. User found a clothing he wishes to remove permanently and hovers his hand over to the "delete permanently" bin icon
4. System removes the selected clothing permanently

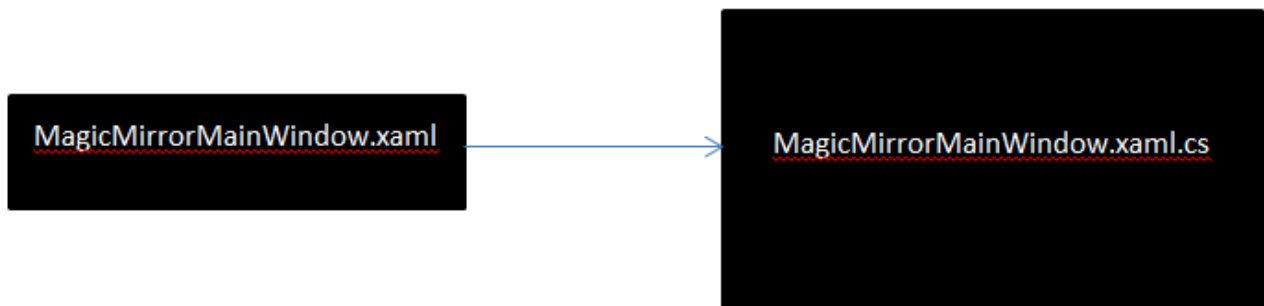
User may repeat step 1 and 2 to navigate clothings until he finds the one he wants to delete.

\*(a) at any time the user hovers his hand over the back button

\*(a) 1. System redirects user to previous menu

### Magic Mirror Architecture

The diagram below showcases the overview of the Magic Mirror Architecture. It consists of 2 components, namely MagicMirrorMainWindow.xaml and MagicMirrorMainWindow.xaml.cs



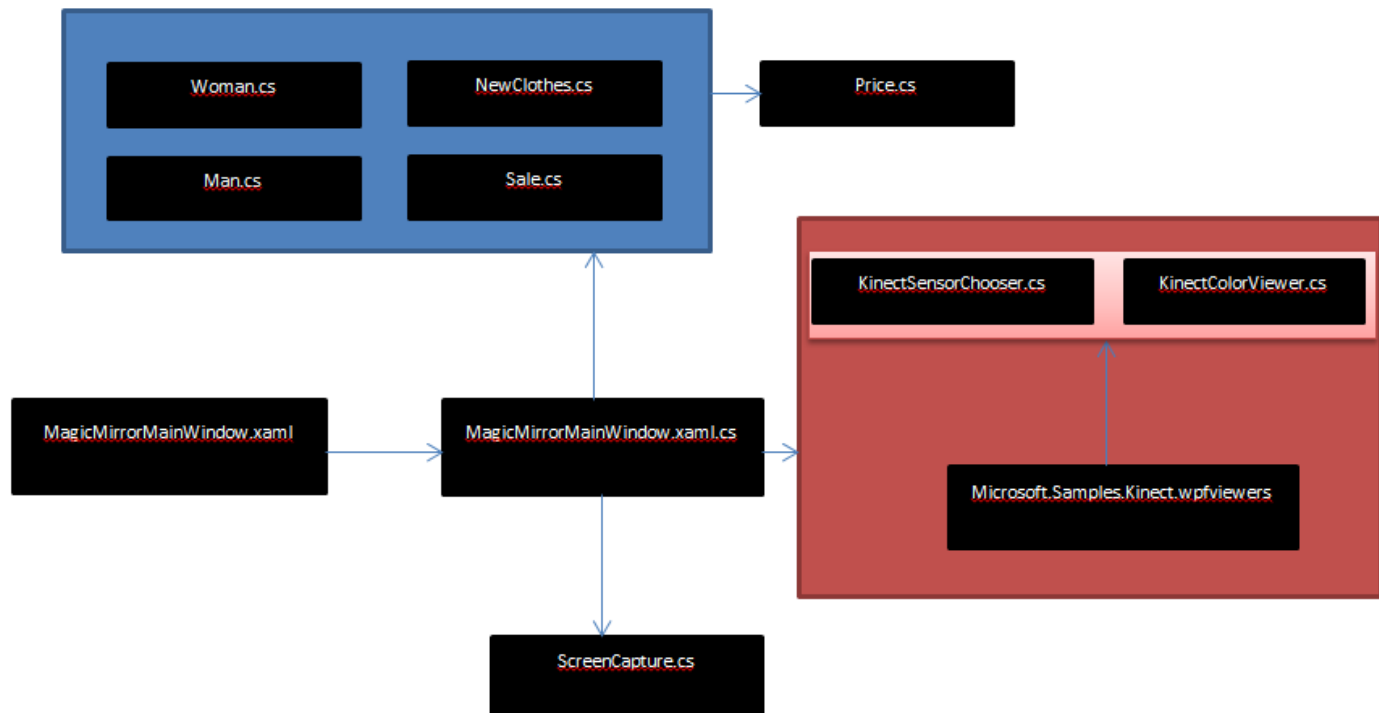
### *MagicMirrorMainWindow.xaml:*

Provides a user friendly UI where the user can view and execute gestures for the system to comprehend. This is the place where all button objects and background are instantiated. The functionality of this class is stated at the first part of this chapter.

*MagicMirrorMainWindow.xaml.cs:*

Reads in commands/gestures from MagicMirrorMainWindow.xaml and execute them, internally, while displaying the output back to MagicMirrorMainWindow.xaml.

Below depicts a semi-complete architecture of the system:



*3rd party libraries(Red box):*

Secondly, the system made use of 3rd party libraries (resources can also be found at Microsoft Website), namely the Microsoft.Samples.Kinect.WpfViewers class. Within this class, this system made good use of the KinectColorViewer class (display the Kinect camera data) and KinectSensorChooser class (activate the Kinect sensor and all the data streams). There are also other relevant .dll files installed in this project, for more information, please refer to Annex B of this document.

The reason for using 3rd party libraries and .dll files is due to their efficient implementation by Microsoft. There is also no need to “reinvent the wheel” when this is available, hence we used their default implementation for the basic functionalities of this project.

It is also possible to implement the sensor objects and display the camera data on the screen, but with a 3rd party library available, smoother and better implementations can be achieved. This thus will result in better feedback by the system to the users.

*Sub-menu classes(light blue box):*

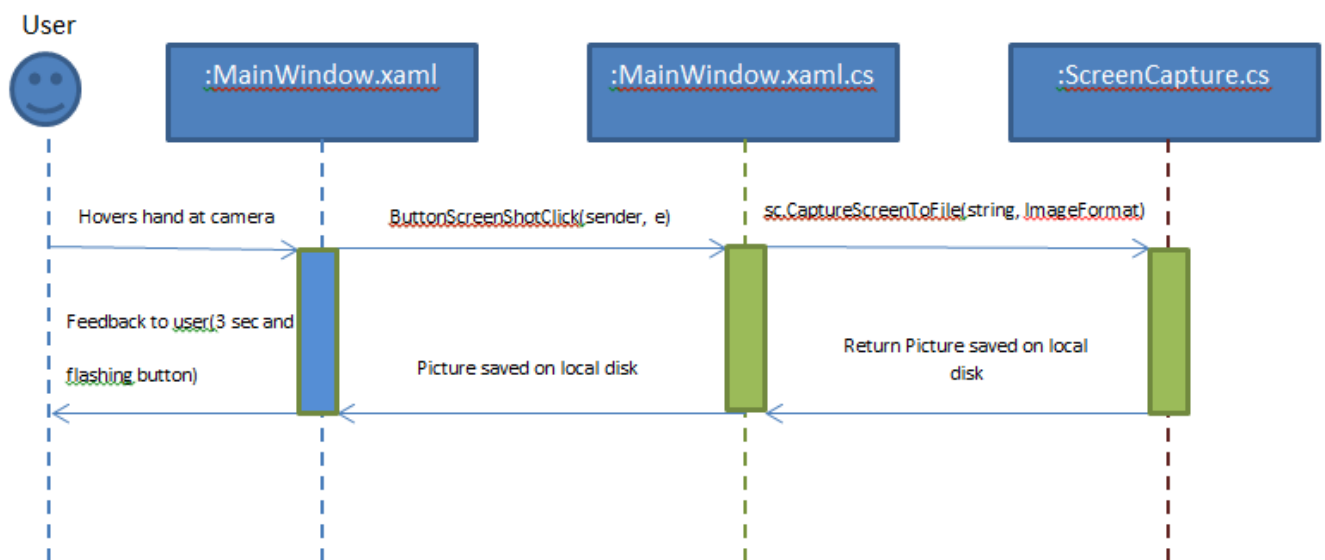
The 4 classes contained inside this box(Man, Woman, NewClothes, Sale) are responsible for handling the respective clothes ID when they are selected, how they are selected, and how the clothes are stored, etc.

*ScreenCapture.cs:*

This is a self written class that allows the system to take screenshots of itself when users activate it, and subsequently save it on a local disk. This is used for the camera function that is outside the GUI display, and will interact with MagicMirrorMainWindow.xaml.cs class to save the picture on a local disk.

### Internal Structure

Below details a sequence diagram example when a user uses the camera function of the Magic Mirror.



This example shows how the system reads the user's hand gesture from MainWindow.xaml, calling the eventhandler function ButtonScreenShotClick and interpreting it, and subsequently calling another function sc.CaptureScreenToFile (sc is a ScreenCapture object created earlier). It then returns a picture file that is saved inside the local disk, and the feedback is propagated back to the main GUI where the user can know from the blinking camera button.

### Meeting User Requirements

With all the design aspects of the system (be it GUI design and internal software design), the system has met user requirements to some extent based on their demographics and user habits as explained earlier in Chapter 1.

## **GUI design:**

Design principles are also taken into consideration when designing the GUI. We focus on mainly two core principles: Aesthetical and Functional

### **Aesthetical**

- Harmony
  - Same shaped buttons on left and right panels for a harmony combination effect to appear visually satisfying
  - Dominant colors used are similar to allow a color harmony effect
- Gradation
  - Menu frames are designed with directional colour gradients for flow and continuity

### **Functional**

- Layout and Symmetry
  - Layout of buttons and panels are ordered and neat
  - All buttons are kept symmetrical
  - Buttons grouped into specific areas helps with ordering
- Placement
  - Buttons with related functions have placements adjacent to one another to optimize usage
  - Options for a related purpose are placed together to optimize use
- Deleting
  - Trash bin icon and menu allows users to delete their clothes selection from the shopping cart
  - Users may also undo their deletion or permanently delete from the trash bin menu

## **Programme/Software design:**

### **External libraries**

- Usage of relevant libraries.
  - Need not reinvent the wheel.
  - Results in better implementation.

### **Feedback**

- Icons are programmed to glow to give feedback to users
  - users will know that they have interacted with it (when their hands hover over the icons).

### **Fast execution time**

- Feedback are given back fast to users.



- Less time waiting will lead to less frustration when using the system.

### **Systematic classification of classes(OOP)**

- Although general users may not know the internal structure of the programme, this feature is important if another team is to continue our project, as understanding the whole system is much easier.

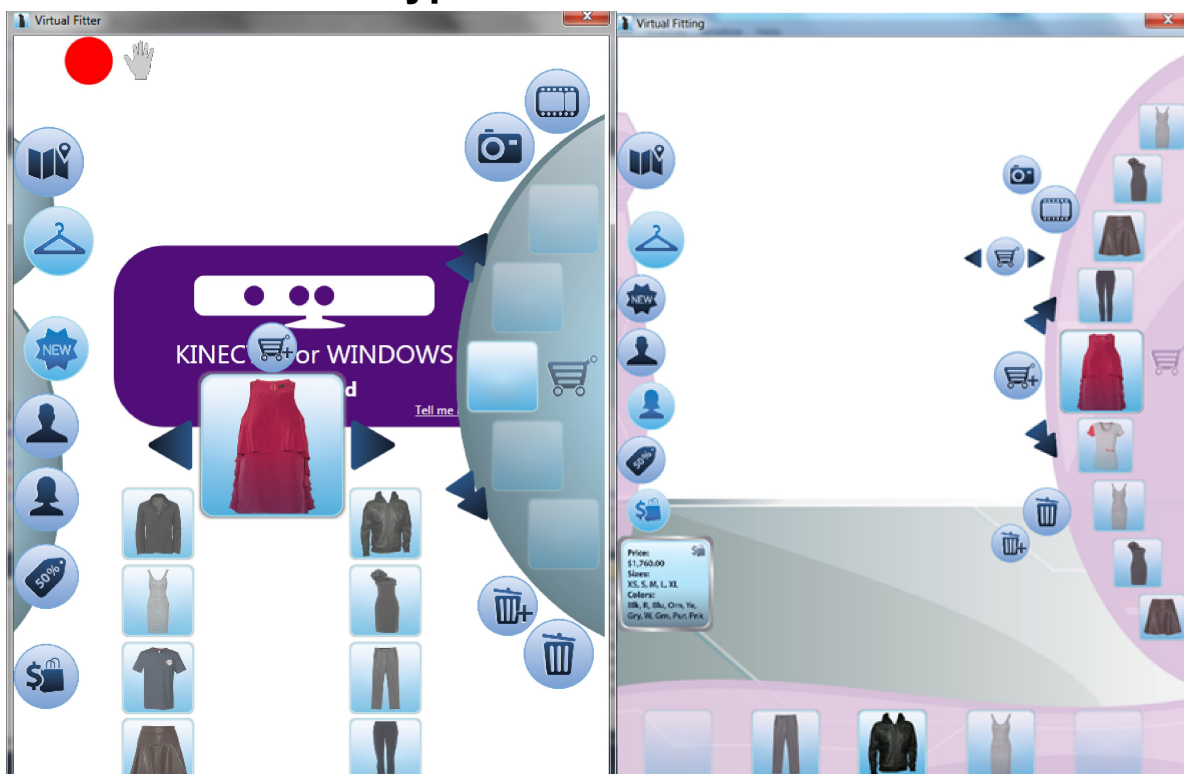
### **Usage of Software design architecture**

- Use cases, overall project architecture, sequence diagrams considered when system is designed
- More smoother flow of software design from top-down view
- Easier understanding of the project by another developer team

Although the system appears functionally stable, some minor changes have been carried out and changes are still ongoing. Further details are in the next few chapters.

In general, design and software principles applied in this system has been effective in showing usability and functionality aspects to some extent, although the system is still a work in progress and has got room for growth and improvement.

## **4. Re-modified Prototype**



*Above pictures showing the change in UI. (Left: old prototype. Right: new prototype)*

### **Changes to initial design**

Minor tweaks to the previous design prototype depicted in the earlier chapter were made upon re-implementation. The goals of the system, in terms of its general functionalities, however, remain intact. The changes and their respective justifications are listed as follows:

#### Rearrangement of icons

Most notably, the browser menu has been edited. Namely, the thumbnails of clothes that were in the middle were shifted to the right wheel menu, replacing the shopping basket, while the shopping basket remains visible as an icon near its original position. There are also some minor rearrangements of the other icons.

- Unnecessary articles on the screen were removed in an attempt to reduce the clutter, which has inevitably formed, in the middle of the screen, due to the inclusion of multiple functional features.
- This hopefully reduces the visual distraction imposed on the user, hence improving user-experience.

#### Addition of cursor

A scaled cursor was introduced to enable users to reach for icons at the far ends of the screen, beyond their natural stretch, more easily.

- The reason the icons were out of reach in the first place was due to the sheer quantity of icons needed, as well as the necessity for the icons to be of at least of a certain spacing from one another to interact optimally with an average palm size.
- Hence it follows that this *may* be just an interim function, to be phased out if future implementation makes use of a larger screen, allowing icons to be more centralised within a radius centering on the user.

### **Evaluations**

The goals of our “Magic Mirror” solution is to provide an interesting, gesture-operated wardrobe application that will enhance users’ shopping experience. This application should let users browse for clothes and apparels within the system and try out those clothes on their “body” reflection. The displayed content will consist of icons, information of clothes, and most importantly, the mirror showing the users’ reflections in said clothes.

Based on informal testing amongst ourselves and feedback from our supervisor, our current prototype does address the above issues effectively in terms of GUI design and system architecture. More work, it is also indicated in the earlier chapter how the prototype meets certain user requirements. However, more needs to be done in order to

make it more interesting and elegant. At its current stage, we believe it is still too crude to meet the market demands for style. A more refined final prototype, thus, is necessary to generate hype *and* enhance the real-life shopping experience.

## 5. Discussion

After some informal testing done on our prototype, and iterative discussions amongst ourselves, as well as with our supervisor, we have come to some conclusions regarding our system. While usability problems were explored and largely resolved for the program (i.e. functionalities of buttons), user-experience remains an issue for the Magic Mirror. With the next semester in mind, more features could be added to the software, so as to enhance the user experience. These include:

- Smooth transitions between “browser”, “fitting room” and “recently discarded” screens
- SNS integration
- Transiting to a higher (1280 x 960) resolution, which will then eliminate the necessity of the cursor.
- More fluid animations for transitions in the GUI (like clothes being browsed, etc)
- Integrating more Kinect hand movement gestures, including pushing and swiping actions
- Refactoring and Optimization

## Annex B (*Important API*)

### **dll installed in project:**

Microsoft.Kinect.dll  
Microsoft.Samples.Kinect.SwipeGestureRecogniser.dll  
Microsoft.Samples.Kinect.WpfViewers.dll  
Coding4Fun.Kinect.Wpf.dll

### **Libraries used in project:**

Microsoft.Samples.Kinect.WpfViewers  
· KinectColorViewer  
· KinectSensorChooser  
· KinectSkeletonViewer

### **Important functions in project(KinectProject.cs file):**

Operation: void kinectSensorChooser1\_KinectSensorChanged(object sender, DependencyPropertyChangedEventArgs e)

Type: void

Description: Function where it enables all Kinect stream(colour, depth, skeleton)

Parameters: sender, event handler for dependency

Pre-conditions: main function called

Post-conditions: Kinect sensor object initialised, all streams enabled, smoothing enabled, allFramesReady eventhandler enabled. Kinect sensor object starts after streams enabled.

Operation: void sensor\_AllFramesReady(object sender, AllFramesReadyEventArgs e)

Type: void

Description: Storing image data from frame into array, getting first skeleton data,

Parameters: sender, event handler for all the frames that are ready

pre-conditions: kinectSensorChooser1\_KinectSensorChanged function executed

post-conditions: image data will be retrieved and stored, get first skeleton data to be tracked, get first camera point, scaling of tracked body part

Operation: Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)

Type: Skeleton

Description: returns the first skeleton data that is tracked.

Parameters: event handler for all the frames that are ready

pre-conditions: all frames of the Kinect must be ready(sensor\_AllFramesReady already called)

post-conditions: returns the first skeleton data that is tracked.

Operation: void GetCameraPoint(Skeleton first, AllFramesReadyEventArgs e)

Type: void

Description: Function where the tracked skeleton data gets mapped from skeleton stream to depth stream, then to colour stream. Also handles interaction with UI buttons

Parameters: skeleton object, event handler for all the frames that are ready

pre-conditions: all frames ready and the first skeleton is tracked

post-conditions: tracked skeleton data gets mapped successfully to colour stream via depth stream.

Checkbutton functions called as well for interactivity with icons.

Operation: private void ScalePosition(FrameworkElement element, Joint joint)

Type: void

Description: scales the tracked joint location to be displayed in the UI based on specified parameters inside the function

Parameters: framework element(UI component), tracked skeleton joint

pre-conditions: all frames running

post-conditions: tracked skeleton joint scaled based on scaled parameters

Operation: private static void CheckButton(HoverButton button, System.Windows.Controls.Image thumbStick)

Type: void

Description: handles the interactivity of the buttons in the UI

Parameters: button, thumbstick(this is the tracked body part)

pre-conditions: all frames running, tracked body part is in midpoint container of the button

post-conditions: hovering action of button executed(by 3rd party library, Coding4Fun.Kinect.Wpf)

Note: This function is also overloaded with other parameters.

Operation: private void StopKinect(KinectSensor sensor)

Type: void

Description: Stops the Kinect sensor

Parameters: Kinect sensor object

pre-conditions: Kinect sensor is running

post-conditions: Kinect sensor stopped.