Exchanging Images on a shared surface

Jette Marie Juul Jerndal
IT-University of Copenhagen
Copenhagen, Denmark
jejj@itu.dk

Jacob Glerup Bachmann Andersen
IT-University of Copenhagen
Copenhagen, Denmark
jacb@itu.dk

Nikolaj Søgaard Mosbæk Nielsen
IT-University of Copenhagen
Copenhagen, Denmark
nnie@itu.dk

ABSTRACT

A pervasive system has been developed, where multiple users can share images from their smartphone via the Microsoft Surface 2.0, by using a drag and drop functionality. For this to work networking is a big priority, since it is the link between platforms.

A smartphone application has also been develop, which will display all images on the device's SD card and which will also handle the TCP connection to and from the Surface, and the ability to send images to other phones.

ACM Classification Keywords

Pervasive Computing, Tangible User Interface, Microsoft Surface 2.0, Smartphone, Networking, Visualizer Tags.

General Terms

Documentation, Theory.

INTRODUCTION

This report is a part of the mandatory assignment #2 in the course Pervasive Computing at the IT-University of Copenhagen (ITU), relating to the subjects of tangible computing and tagging. The technology used is Microsoft Surface 2.0; an interactive platform that allows people to use touch and real world objects as well as sharing digital content. An application was implemented allowing people to exchange images from their smartphones on a shared surface. Though the core of the subject concerns tangible computing a major part of the project is the exchange of data between the phones and the Surface. Although the network component will seem trivial to some, due to the inherent nature of pervasive computing it is a vital part that shouldn't be underestimated.

CONCEPTUAL DESCRIPTION

The concept of exchanging images on a shared surface is given as a Mandatory Assignment #2, in the course "Pervasive Computing-F2013" at the ITU, however in this section the concept will be described, and throughout this report the system will be referred to as The Image Exchange System.

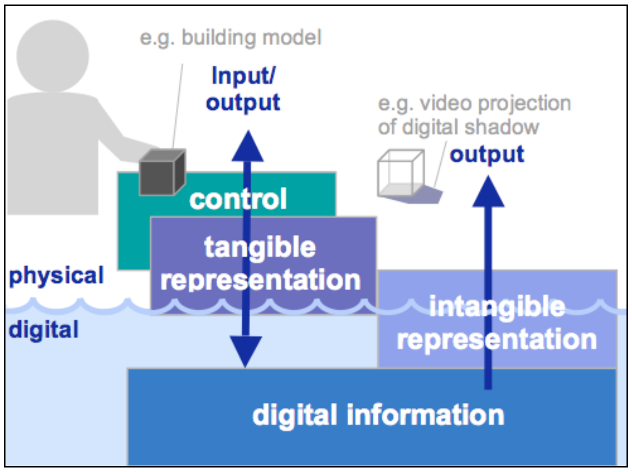

Figure 1: Interaction model for TUI[3]

The Image Exchange System supports multiple users transferring images that are located on their smartphone, to each other simply by laying their phones on the Microsoft Surface 2.0. All the images from their phones will then by visualized on the Surface, with a small icon in the corner showing which phone the image is stored on. The users can then drag and drop images onto their phone, thereby transferring the image to their phone - a new icon will be added to the visualization, showing that both phones now contain this image. If the user pins the phone by clicking the pin-button, a visualization of the phone will be added to the Surface, and the user will be able to remove the phone; thereby still being able to transfer images to that phone. A conceptual example can be seen on figure 2.

An image browser application called Image View, for the smartphone will display all the images found on the phone's SD card, and will automatically update when new images are added.
All these elements are meant as an easy way of sharing pictures between smartphones, using the Microsoft Surface 2.0.

Tangible User Interface

In this section the main concepts concerning the theory behind the behind tangible user interfaces (TUIs) will be described in short terms, as well as how it is related to the Image Exchange System.

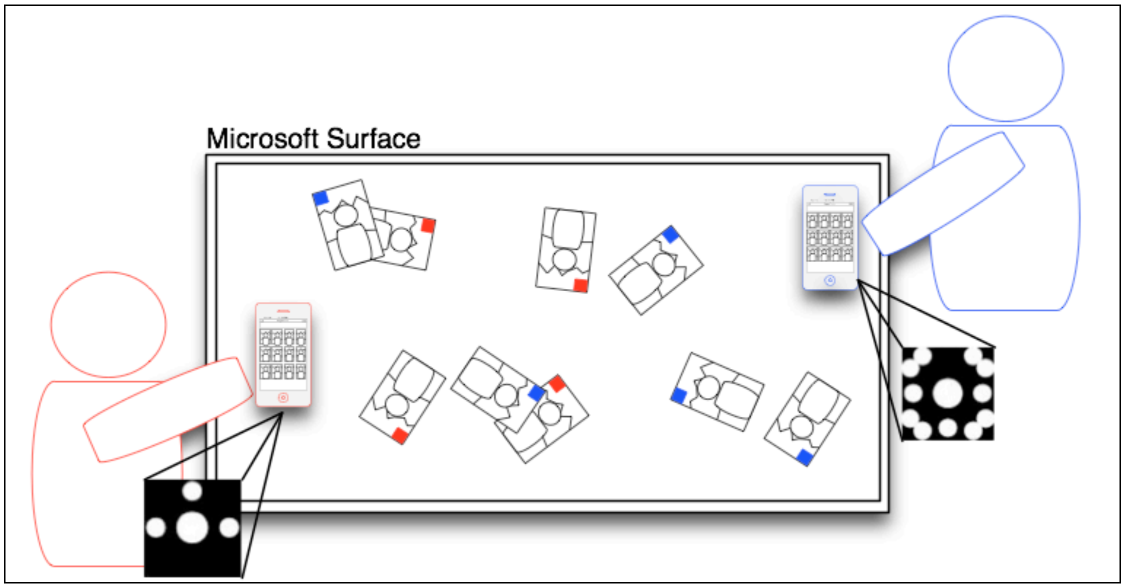

Figure 2: Two users using the Image Exchange System, having two smartphones with Visualizer Tags attached

As David Güiza Caicedo describes on his web site:

"*A tangible user interface is one in which the user interacts with a digital system through the manipulation of physical objects linked to and directly representing a quality of said system*"[1].

A form of tangible user interface is the tabletop. Tabletops have been produced since the mid 90s, and have evolved since then [2]. Some tabletops can support multiple user interaction types: single touch, multi touch and pattern recognition. One of the ways to draw advantage of the pattern recognition is by using tags. A tag is a unique pattern, which can be printed onto an object and put on the table for the tabletop to recognize. An example of how this is used in the Image Exchange System can be seen in figure 2. In this report these tags will be referred to as tags or Visualizer Tags.

In 2006 Hiroshi Ishii created an interaction model for tangible user interfaces (see figure 1).[3] This model can be recognized in the Image Exchange System where the tangible element is the smartphone, with a tag for recognition. The Surface could be seen as the tangible representation, which then digitally processes the information from the tag, giving the user an intangible representation of images on the surface. The fact that the Image Exchange System, when used with the Microsoft Surface 2.0, fits with the interaction model of TUI shows us that it is a good example of a tangible user interface system.

Setup

In order to start the entire image exchange system, there are a few steps that need to be completed in order to get a proper and stable connection between the different components. First the Microsoft Surface should be running at all times. Secondly, the smartphones must have an active Internet connection, and run on the same network as the surface. Every phone/user must be registered at the Surface in order to have a proper Visualizer Tag from which the Surface will recognize the user and his phone; an example of these can be seen in figure 2.

When the above has been done, the user should start the 'image view' application on his or hers smartphone, and place the backside of the phone with the Visualizer Tag facing the surface of the table. The user will now be faced with the images from the phone's external memory and the table is ready to be used as described in the conceptual description above.

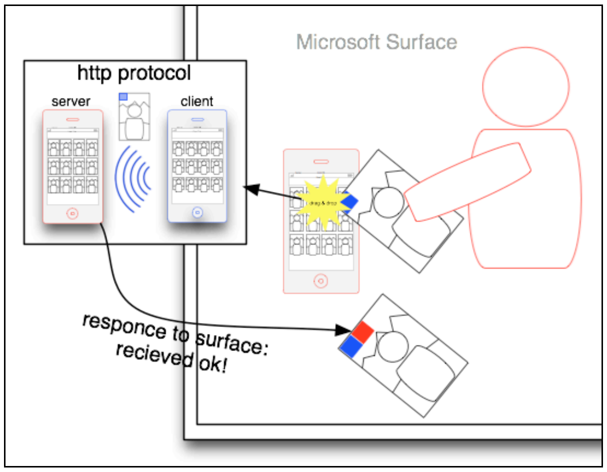

Figure 3: A '*drag and drop*' operation

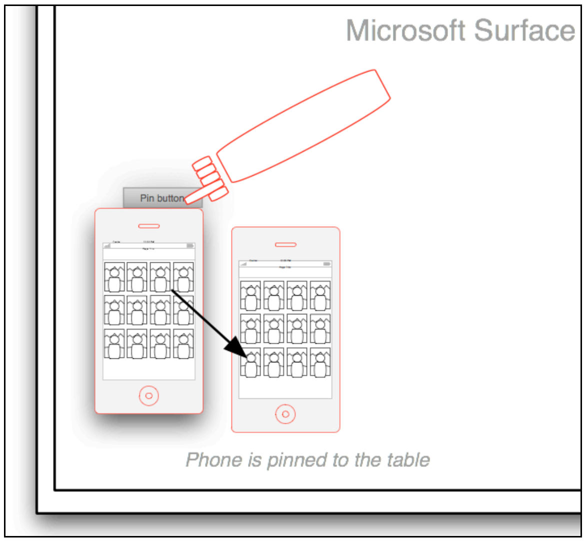

Figure 4: A user pins the phone to the table, and removes the physical phone

One word of caution is; when the table or phones has been disconnected from the mutual subset, the user has to update the IP setting in the Surface system; this also goes if the user has received a new Visualizer Tag.

TECHNICAL DESCRIPTION

This section is divided into three parts; one regarding the Microsoft Surface, another regarding the Smartphone device and the last being a short network description. The complexity of the Image Exchange System is rather large, so this report will not go into microscopic details of the code, however in the end of each part, there will be a "sum up" of the most important features in code or design.

Microsoft Surface

The Microsoft Surface UI consist of two important UI components, one being a Tag visualizer that visually displays a phone image representation based on the tag, and the other a Scatterview that displays a pool of images from phones recognized by the Surface.

The Tag visualizer depends on predefined user information, such as tag definition and a phone image. The Image Exchange System saves this information within a User class object, which of course requires that the user be already registered as mentioned earlier. When the tag is recognized by the surface, it displays an image related to it, and the user can now choose to pin it to the table. The pin operation means that the user can remove the physical phone and its visual representation stays on the surface (see figure 4). The user can now perform drag and drop operations (see figure 3) with images in the Scatterview.

The Scatterview, which can be seen in figure 2, takes a

```
<s:ScatterView.Resources>
    <DataTemplate x:Name="ScatterviewItemDataTemplate"
                  DataType="{x:Type r:Watermark}">
        <Border Child="{Binding}"></Border>
    </DataTemplate>
</s:ScatterView.Resources>
```

Figure 5: Visual elements on the Surface, is defined by a *data Template*

pool of images, which is handled by an observable collection, and then scatters them across the visible area. Whenever an image is received by the surface, it is added to the observable collection, where the ownership and image information is stored as a Watermark class object.

There are two kinds of visual element on the MS Surface in this project, one being a image from a phone, another a tag visualizer image, both of these are define by a *dataTemplate* from within the SurfaceWindow1.XAML file. Each visual element is displayed accordant with its data template, which is defined by the element's type (an example of this can be seen in figure 5.

Smartphone

The smartphone User Interface is rather simple, and requires little knowledge of the communication with the Microsoft Surface. The smartphone UI automatically refreshes the image browser, when a new image is successfully transferred from the Microsoft Surface, an example of the smartphone UI can be seen in Figure 6.

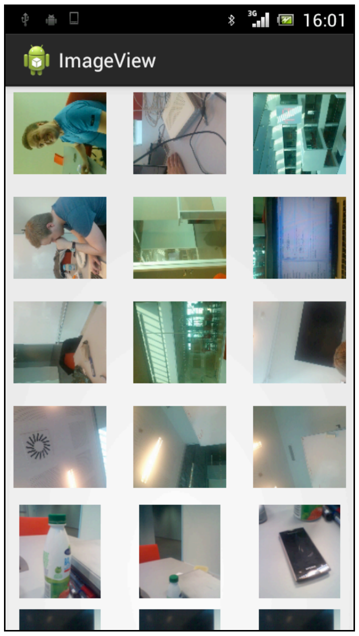

Figure 6: The UI for the Smartphone device

When handling images on phones, there are a number of issues that needs to be taken care of, one of them is the image size, and the other is a side effect of this. In this project the source phone had an average image size of 700 kB to 1.5 MB per image, and assuming that the user has 15 images on the phone, he now has a collection of image with a size of 10.5 - 22.5 MB. Due to this size, the phone's images are copied and resized to a separate folder, from which the images are transferred; this helps to improve the transfer time as well as handling memory issues on the phone.

Networking

In Appendix 1 of the life cycle of the Image Exchange System, it is shown that all the network communication is handled in the JSON format language. By using JSON for messaging, each message can easily be used to perform remote procedure calls (RPC) over the network using the HTTP network protocol. Each message contains information regarding which method/function to invoke, but also the data to be handled. Though JSON is a bit more difficult to implement (platform differences), it was judged to the most suitable solution to this project, due to the facts above.

DISCUSSION

Throughout this project many system design choices have been made. In the following there will be reflected upon what could have been different in terms of the technical choices as well as the conceptual, and what would be changed if more time was given.

Technical

Addressing the choices for the networking component, the first choice was to use Bluetooth connections to transfer data between the phones and the phones and the Surface. However after experimenting with this, the idea was abandoned due to the fact the phones needed to pair up first in order for it to work. It was decided to use remote procedure calls over HTTP instead as well as JSON for messaging, as the JSON format is language independent and has implementations for both Java and C#. However as it turned out it was not entirely without problems to implement due to issues concerning differences of the platforms and had to be investigated.

An important issue is robustness of the system however it is impossible to make a system which is 100% robust, this is especially the case when it comes to network communication, which can be picked out as one of the most volatile part of the system [4]. Therefore had there been more time there should have been a greater focus on exception handling for instance the securing of network connections.

Turning to the system design for the Surface it would have been preferable to have isolated the network component from the user interaction by accessing it through an interface, hence making it easier to change this component with another if needed.

As for the user interaction with the Surface, the design was to a large extend hampered by lack of knowledge of the Windows Presentation Foundation (WPF). The idea of having the phones and the images in the same *Observable Collection* as *FrameworkElements* and distinguishing them by type in the code behind worked fine, once the *datatemplate* with correct binding was applied.

An attempt was made to implement the drag and drop functionality with the Surface Drag and Drop Framework, however as this failed more simple drag and drop method was created using a classic method for finding intersected objects.

Conceptual

The application requires the users to register themselves before using it; in a “true” pervasive system no action is need by the user to instigate the interaction [5]. However this brings to mind privacy issues, the user might not be aware of the consequences placing his phone on the Surface i.e. everybody else at the table would be able see the images on his phone and furthermore being able to transfer them to their phones. Again this is a classic concern within pervasive computing.[6]

REFERENCES

[1] What are tangible user interface? http://www.bluehaired.com/2010/05/what-are-tangible-user-interfaces-2/

[2] Christian Müller-Tomfeld, Morten Fjeld: Tabletops: Interactive Horizontal Displays for Ubiquitous Computing, 78

[3] Hiroshi Ishii: Tangible User Interfaces, MIT Media Laboratory, 2006

[4] Kindberg & Fox: System Software for Ubiquitous Computing, IEEE Pervasive Computing, Vol. 1, No. 1. January 2002, p. 77

[5] Mark Weiser:The Computer for the Twenty-First Century. Scientific American, 1991,265 (3), 94-104

[6] Pankaj Bhaskar, Sheikh I Ahmad: Privacy in Pervasive Computing and Open Issues, 2007

Appendix 1

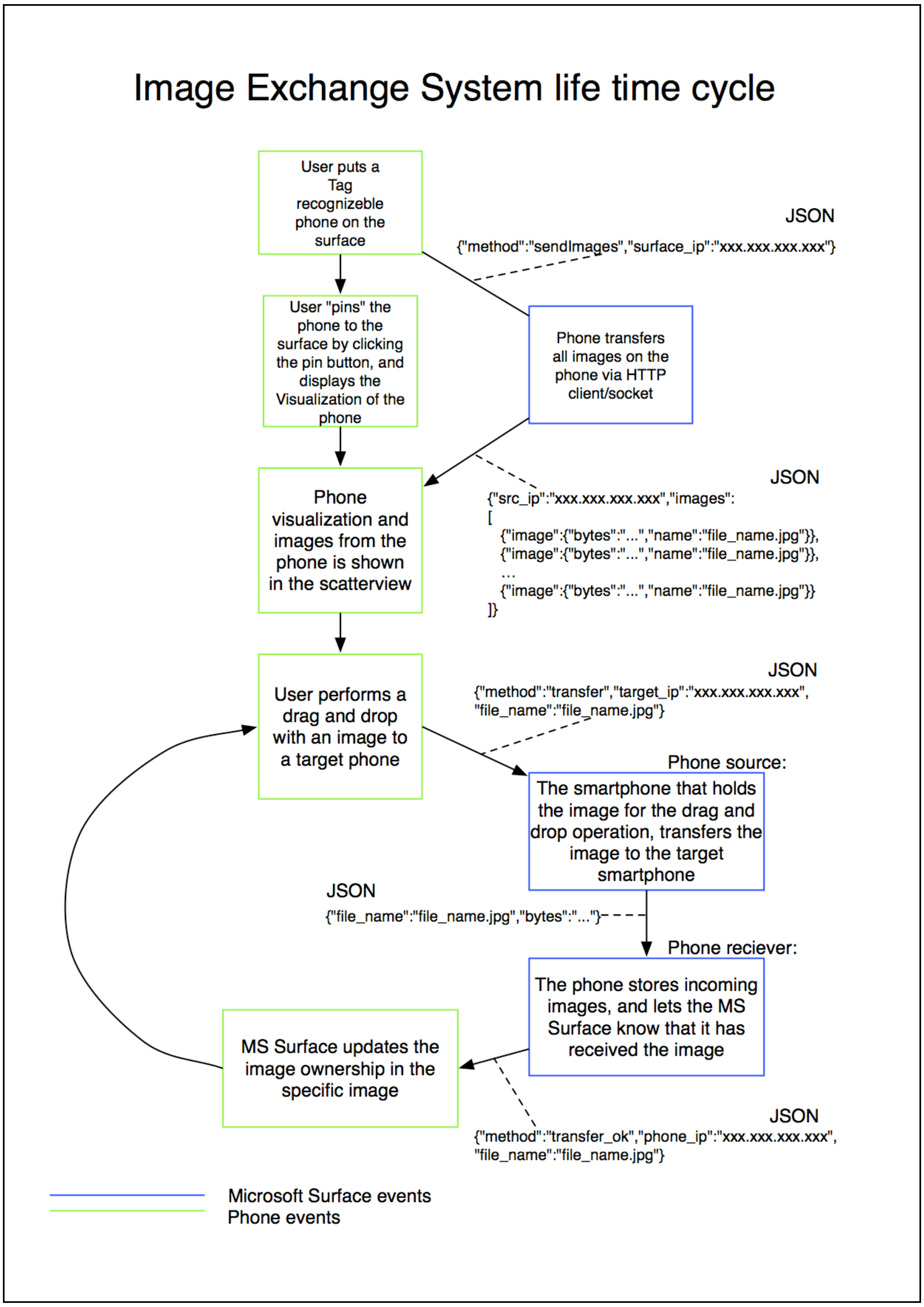
Image Exchange System life time cycle
User puts a Tag recognizeble phone on the surface
JSON
{"method":"sendImages","surface_ip":"xxx.xxx.xxx.xxx"}
User "pins" the phone to the surface by clicking the pin button, and displays the Visualization of the phone
Phone transfers all images on the phone via HTTP client/socket
Phone visualization and images from the phone is shown in the scatterview
JSON
{"src_ip":"xxx.xxx.xxx.xxx","images":
[
{"image":{"bytes":"...","name":"file_name.jpg"}},
{"image":{"bytes":"...","name":"file_name.jpg"}},
...
{"image":{"bytes":"...","name":"file_name.jpg"}}
]}
User performs a drag and drop with an image to a target phone
JSON
{"method":"transfer","target_ip":"xxx.xxx.xxx.xxx", "file_name":"file_name.jpg"}
Phone source:
The smartphone that holds the image for the drag and drop operation, transfers the image to the target smartphone
JSON
{"file_name":"file_name.jpg","bytes":"..."}
Phone reciever:
The phone stores incoming images, and lets the MS Surface know that it has received the image
MS Surface updates the image ownership in the specific image
JSON
{"method":"transfer_ok","phone_ip":"xxx.xxx.xxx.xxx", "file_name":"file_name.jpg"}
Microsoft Surface events
Phone events