# The fluid photo browser

**Pelle Krøgholt**
IT University Copenhagen
Rued Langgaards Vej 7,
DK - 2300 Copenhagen S
pelle@itu.dk

**Andreas Bok Andersen**
IT University Copenhagen
Rued Langgaards Vej 7,
DK - 2300 Copenhagen S
aboa@itu.dk

**Stine Bierre**
IT University Copenhagen
Rued Langgaards Vej 7,
DK - 2300 Copenhagen S
sbie@itu.dk

**Abstract**
*The need for a photo-printing expert is not dead with the digitalization of photography. To simplify the process of handing over the digital photos from photographer to print expert a communication between a phone and a tabletop is made. The user interface of the ubiquitous system has been developed with ten rules for good interface implementation in mind. To implement functions of multi touch and sending photos with gestures the MT4J library is added to the tabletop.*

**Author keywords**
Ubiquitous computing, tabletop, Android, MT4J, TCP, multi touch, gesture

**Introduction**
Nowadays it is possible to shoot 400 photos in one day without having to delete any of them. To highlight some photos as being special the photographer therefore need to treat some pictures differently from the others. One way to add value to a photo is to print it. When photos are selected as valuable to the photographer a better print than from your own personal printer is needed. The need for photo shops and distributers is therefore not dead with the digitalization of photo shooting. Furthermore it is preferable to be able to discuss the editing of the photos by pointing and showing features and elements in the photo to the expert. The best way to hand over the photos is by showing up in the photo shop and have a face-to-face communication with the expert. To hand over digital photos from a photographers mobile device to a photo printing experts tabletop does not have to be a complicated process.

**Requirements for the system**
The photographer that walks into the photo store might have taken hundreds of photos of this morning's sunrise, but he only decided to get prints of three of them. When entering the store he activates his FluPho (Fluid Photo) application on his Android phone and goes to the desk that is also a tabletop. While transferring the photos the photographer and the expert discusses the quality and after the transferring they edit them together on the tabletop. When finished the expert sends the photos to the printer (with his own gesture) and back to the photographers phone. What is needed for this process apart from the hardware is the applications on the devices and a network connecting the devices (described in the System design chapter).
To implement the system into daily life activities as trading and sharing it has to be intuitive in terms of assimilating the physical worlds objects and movement patterns. To use a tabletop as desk makes it possible to hand over and point and edit the photos with gestures that would also have been used with physical photos. This tabletop is due to that an augmented reality where the possibility of editing the photos directly on the desk for example with a digital pen is augmenting the desk of the physical worlds functions, such as handing over objects. The fact that the photos will be visible on the tabletop after transfer is contributes a feeling of privacy at the photographer that has just transferred what to him are valuable photos. To see them end at the desk verifies that the pictures have not ended in the big cyberspace. This is only possible due to the collocation of the participants and to the synchronous transfer of the data. The combination of face to face communication and technological transferring will in that way

expand the interactive editing of the photos without obstructing the customer's feeling of his objects being treated as private in a secure manner.

In addition to the augmented reality the need for intuitive handling of the system is needed. The need for making movements in addition computer devices intuitive is an important task for getting people to integrate the device even further in their everyday life.
The photo expert will have to deal with the fluid photo browser system a lot in his everyday. Therefore it can be a good idea to have an easy way for him to transfer the photos back to the customer and to the printer after editing them on the tabletop.
This can be done with a gesture. The system has an implemented possibility of the user to define the gesture to make the transfer from the tabletop to the phone. This makes it possible for the user to implement rhythms and routines in the system and thereby fit it to his specific spatial patterns. This socio-technical solution[3] can make the system more intuitive to the user because it leaves a possibility for the photo expert to design a an ergotic[1] gesture that assimilates the way he would hand over a printed photo on a desk or the way he thinks makes most sense at the moment.

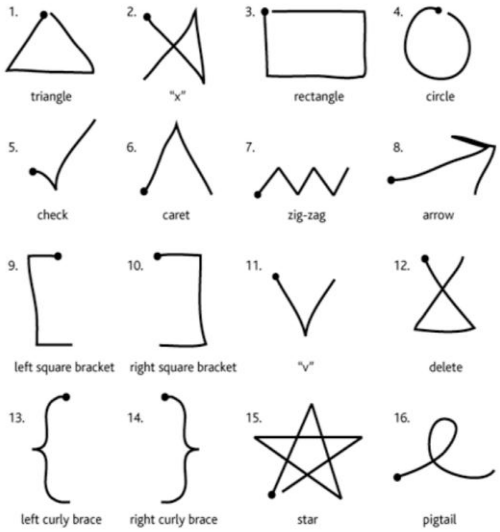


*Figure 1. Examples of strokes for the user to define[3]*

[1] Ergotic means to create and manipulate artifacts through physical action (Cadoz, 1994).

In figure 1 there is a lot of examples of strokes that could be preferable for the user. The *X*, the *delete* and the *pigtail* could for example for some users mean *end the session and send the photos back.* The *check*, the *V* and the *star* could all mean *good work is done – send.* And the *arrow* and the *caret* could mean *send in this direction*. Furthermore a single line could be added that just depicted a finger on a physical printed photo handing it over to the receiver meaning *Here you go!*
Since the photo expert will use this system a lot there has been made a possibility for him to store the gesture he defines so that he can just use that gesture without defining a new one every time he needs to send the photos.

**Ubiquitous User Interface design**
The design of the user interface has been implemented with awareness of the ten rules for Ubiquitous User Interface design (see figure 2)

| Rule | Meaning | Example |
|---|---|---|
| Bliss | Learning to interact with a new UUI should not require people to learn another skill or complex command language. | Good interaction design as discussed in Section 6.2 |
| Distraction | Do not demand constant attention in a UUI. Inattention is the norm not the exception. | Ambient User Interfaces as discussed in Section 6.3.3 |
| Cognitive Flow | Ubicomp systems that are everywhere must allow the user to retain total focus on the task at hand. | Multimodal interfaces as discussed in Section 6.4.3 |
| Manuals | Do not require a user to read a manual to learn how to operate the current UUI. Do leverage prior experience. | Use of affordances (e.g., UI overlay) on real world in Figure 6.5 |
| Transparency | Do not rely on users to hold application state in the mind to operate the UUI. | Tangible User Interfaces as discussed in Section 6.3.1 |
| Modelessness | Avoid "modes" where the system responds differently to the same input stimulus dependent on some hidden state information. | State visible in SharePic as shown in Figure 6.8 (Section 6.3.2.1) |
| Fear of Interaction | Provide easy means to undo actions, otherwise users may become paralyzed with fear when using the UUI. | Use of well-understood design patterns as discussed in Section 6.2.4 |
| Notifications | Feedback to the user can be piggybacked and layered into interactions with their physical environment. | Display of power usage as shown in Figure 6.12 |
| Calming | Interfaces will support situated actions, interfaces will rely on a wide array of human inputs and human senses. | Surface interfaces as shown in Figure 6.6 |
| Defaults | Good interfaces judiciously exploit what the system knows or can deduce. | Applications that reuse user input |

*Figure 2. Ten Rules for UUI design [6]*

Basically the ten rules says that the interfaces have to be intuitive so that the user do not have to use a lot of time to get into the "language" used in the user interfaces. Besides the possibility for making an intuitive gesture for sending the

interface as a whole is implemented in a simple and intuitive style.
The photo gallery on the Android (see figure 3) just have the photos and nothing else that can be distracting. Furthermore it does not require constant concentration since it is operating without any inputs. Thereby it meets the rule of distraction.

*Figure 3 – Screenshot of the Android photo gallery*

On the tabletop there are just three separate scenes. One that shows the photo sent from the Android – the gallery (see figure 4), another in which the user has to define the send gesture – the gesture definition (see figure 5) and the last where the photo has to be sent – the sending (see figure 6).

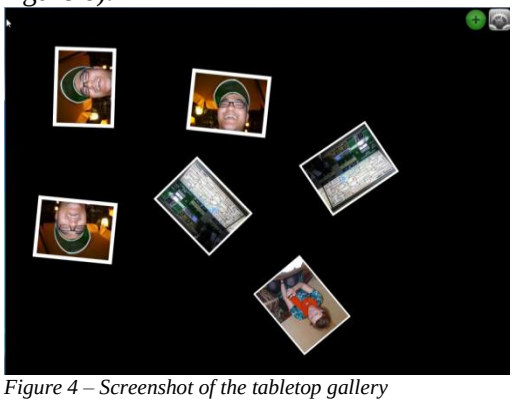
*Figure 4 – Screenshot of the tabletop gallery*

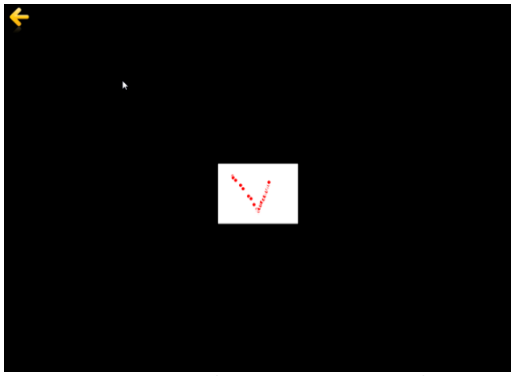
*Figure 5 – Screenshot of the tabletop gesture definition*

The gallery on the tabletop has two features/objects. It has the photo and a settings button that leads the user to the gesture definition scene. The scene is simple and easy to understand and it thereby fulfils the *Bliss rule*. Since the gesture definition, the gallery and the sending are not in the same scene the system allows cognitive flow throughout the process of viewing and sending and the user is not confused about when to do what.
In the gesture definition scene the user is asked to draw a gesture and a distinct square in the middle of the screen shows where the gesture should be drawn. When the gesture is recognized by the system it notifies the user. The notifications make a system user manual redundant and fulfil thereby the rule of *notification* and the rule of *manual*.
When ready to send the photo the photo in the gallery is double tapped and the send

scene opens. The notification asks if the photo is ready to be sent. If it is the defined send gesture is made and recognized the user is notified that it is sent. Furthermore it is possible to go back to the gallery if the photo is not ready to be sent. The possibility of regretting fulfils the *Fear of interaction* rule. The presence of a separate send scene makes it impossible to make a gesture in a scene that has two different outcomes and the system fulfils the rule of *modelessness*. If the sending of the photo was made in the gallery the user would risk to send the photo if he used the defined gesture to just move the photo around on the screen.

**System design**

For the system to work properly the devices needs to be connected in some way. To make the process as easy as possible the connection is made wireless. The connection is established on WiFi with a TCP (Transmission Control Protocol) that is an implementation of IP (Internet Protocol). The TCP makes a free stream between the client and the server and it furthermore requests a handshake between client and server. That means that communication goes two ways, the Android device sends a photo and the server confirms via a handshake that it has received the whole photo. After that the server is ready to send the photo to the tabletop that again with a handshake has to confirm the server that it received the whole package and thereby not half a photo[1]. To make this kind of contact all the photographer needs to do is to connect to the local WiFi when entering the store.

All the Android gallery application should be able to do is to scroll through the photos and send them to the tabletop. The application does not include a lot of multi touch or general gestures and the application is therefore implemented with java. The processing of the images is made at the tabletop.
The tabletop has several functions including multi touch and user defined gestures. To implement those features the external library, MT4J, is added to java.
The transformation of java image to MTImage is all done in the tabletop.

MT4J has a presentation layer for implementing the user interfaces for the code (see figure 7)

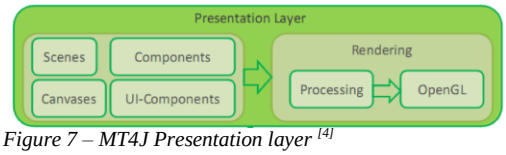


*Figure 7 – MT4J Presentation layer* [4]

*Scenes*

The scenes in MT4J make it possible to implement different interfaces for different functions. The gallery is one scene and the gesture defining is in another scene.

*Components*

The base component class contains basic shapes and others can be added overriding or adding code to the pre-existing components. In the FluPho application these components are photos, gestures and buttons.

*Canvas*

The canvas is the root component of every scene. It listens on the inputs from the hardware and distributes the input to the destination scene. The canvas is a link between the input processing layer and the presentation layer.

*Rendering*

The renderer uses the processing toolkit (open source java) for rendering MT4J components. It is an actor and is a factor in for example the drawing of the gestures.

For rendering a new gesture a Unistroke processor object is instantiated in a new scene on a new canvas. The Unistroke class makes it possible to make a single touch gesture.
On the processor object the Unistroke template that represents the new gesture is made. It is done by adding the Unistroke event to the Unistroke gesture.

The MT4J library makes it to easily implement single- and multi touch functions into the tabletop.

**System architecture**

We have extended several classes in the *MT4J core framework*. In the *UnistrokeProcessor* a overloaded method

has been added which takes as arguments a predefined *UnistrokeGesture*, the gesture expressed as a list of *Vector3D* points and the direction of the gesture. This method in turn calls *addTemplate* on the Recognizer object. This ensures that the registrered *UnistrokeProcessor* on the canvas also listens for newly added custom gestures. To persist the custom gestures all classes related to the *UnistrokeTemplates* class have been made *Serializable*. On application startup these gestures are loaded and automatically added to the registrered *UnistrokeProcessor*. Settings for the application are saved in an xml file. The customgesture is automatically mapped to the send gesture. A more dynamic mapping could be implemented which allows the user to assign specific commands to a gesture.

**Reflections**

In the implementation of the FluPho for transferring, viewing and sending photos from an Android based mobile device to a tab or tabletop the usability is an important factor. The main idea of keeping Ubiquitous User Interface simple and dividing the different functions on the tab into several scenes makes the FluPho very usable. The gesture function on the phone (scroll) reminds the photographer that he is still working on a digital device. When the photos are sent to the tabletop the photos become more like physical printed photos because of the gestures that can be executed here. The connection established between the phone and tabletop is implemented as a TCP. The drawback of that is that the photographer has to manually select the tabletop as receiver to the photos.

An alternative could be to establish a Bluetooth connection between the two devices. That would be less demanding. The drawback of a Bluetooth connection is that there still are a lot of Android phones running on version 2.2 and in that version the Bluetooth is determined after 120 seconds, which would be rather disturbing in the transferring of the photos.

Another alternative solution would be to implement the Near Field Communication (NFC) that can establish a connection between an Android powered device and a NFC-tag. That would of course require that the tabletop had such a tag. The connection is established when the Android device and the tag are within a range of 4 cm. [5] That kind of connection would make the activation and the selection of a receiver unnecessary and would furthermore feel more secure because the physical distance to the receiver is very short.

The system would not only be useful in a photo shop but would also be good to have for sharing e.g. family photos. The scenario could be four family members seated around the coffee table sharing their vacation photos on the family tab. The FluPho can be used at several occasions and in several forums and is therefore a useful ubiquitous system.

**Literature**


[1] Cadoz, C., Les realites virtuelles. Paris, Dominos, Flammarion, 1994

[2] Coulouris, G. et al., Distributed Systems – Concepts and Design, Pearson, Essex, 2005

[3] Lottridge, D., Mackay, W. (2009) Generative Walkthroughs: To Support Creative Redesign, Proceedings of The 7th Creativity and Cognition Conference (CC09).

[4] MT4J: Wiki, Architecture, 1999. Retrieved October 27, 2011, from MT4J, http://www.mt4j.org/mediawiki/index.php/Architecture

[5] Near Field Communication, 2011. Retreived October 27, 2011, from Android Developers, http://developer.android.com/guide/topics/nfc/index.html

[6] Quigley , A., From GUI to UUI: Interfaces for Ubiquitous Computing, in Krumm, J., Ubiquitous Computing Fundamentals, Taylor and Francis Group, LLCBoca Raton, 2010, p. 237-285