IT 11 059

Examensarbete 15 hp
Augusti 2011

A proposal for an Android-based tablet client used in one-to-one computing in teaching environments

Alexander Rangevik

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

A proposal for an Android-based tablet client used in one-to-one computing in teaching environments

Alexander Rangevik

The technology of today gives opportunity to completely new ways of developing educational systems for both teaching and learning.

Portable devices to reasonable prices contribute to a large majority being able to acquire laptops, netbooks, smartphones and tablets.

Therefore, students that use computers, for instance taking notes at lectures become more and more common. The educational form should, instead of opposing and discourage the use of computers during lecturing, take the advantage of the rapid development of these systems to ease the way of education for both students and teachers.

This report will examine the possibilities of developing this type of one-toone computing systems, from a tablet’s point of view. Where the teacher can provide, for instance, Power Point slides to all student’s devices and give them the possibility to send feedback.

An already existing system, which is computer-to-computer based, is built upon this idea and developed by a master’s student within the department of Computer Science at National Taiwan Normal University (NTNU), Taiwan.

The task assigner of the existing project now wants to extend this system to work with tablet platforms as well, which will enable students to use both computers and tablets for the same task.

Furthermore, I will with this report look into the Android platform and its possibilities for the required task and come up with an appropriate design solution; what difficulties that can occur with an eventual implementation on a tablet, and also develop a suitable graphical user interface.

The result is a user interface made in Android for sliding through Power Point slides as JPEG-images, a user login procedure and a solution for finding existing Bonjour services run on an actual Android device.

Handledare: Greg C Lee
Ämnesgranskare: Lars Oestreicher
Examinator: Anders Jansson
IT 11 059
Tryckt av: Reprocentralen ITC

Index

Abbreviations

API	Application Programming Interface
GUI	Graphical User Interface
IP	Internet Protocol
JPEG	Joint Photographic Experts Group
mDNS	Multicast Domain Name System
MCV	Model-Controller-View
NTNU	National Taiwan Normal University
RCTP	Real-Time Control Protocol
SDK	Software Development Kit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
QoS	Quality of Service

1 Introduction

1.1 The ongoing project at NTNU

The task assigner at National Taiwan Normal University has an ongoing project involving several students in different graduation stages, from senior to master's students. The project focuses on building a universal solution for one-to-one computing in student-teacher environments. An existing system is built upon a Java framework for compatibility reasons. The system is still on a development level and therefore has not been subject to any extensive user testing, which means that the compatibility status with other platforms is still uncovered grounds – this is where my part comes in.

The server, typically being run by the teacher, has functionality that provides:

- changing of slides
- drawing on the slides with different colors, for instance, to highlight important areas, and
- collecting answers from question slides.

The slides are of Power Point type, but converted into images during runtime.

The server can see all connected devices and their corresponding student IDs, in case it wants to send any individual data. The client, on the other hand, works as a receiver of the content sent by the server. However, the client can send feedback to the server in forms of drawn-upon slides and also answering custom-made question slides.

The custom-made question slides are created on the server side with image processing software. The algorithm used for this task will distinguish question slides from ordinary content. These slides will have 'clickable' buttons that provide question alternatives. The students can thereafter press any of the answers of choice and the feedback will be sent to the teacher with the corresponding student ID.

Both the server and the client are embedded within the same executable and as you start the program the option to choose between 'Server' and 'Client' is shown to the user.

This kind of interaction between teacher and student is ideal, for instance, when the teacher wants to know how the class is progressing in a certain field.

1.2 Task description

Initially, and as stated in the project proposal, the task description was more focused on finding a suitable GUI approach, solve some algorithmic problems regarding the image processing part and setting up the networking between the devices using Bonjour. However as the project progressed a lot of issues regarding the network solution appeared. It appeared that the Android platform, on which the solution was supposed to be built, had some incompatibility problems with the network protocol Bonjour[1] and also problems with other solutions based on Zero-configuration. So after extensive researching of the Android platform's possibilities and limitations, the task has come to change into something more of a User Interface-oriented kind, which also the task assigner approved with. Therefore and foremost, I will look into how the implementation could look like and also propose a design solution, a GUI. Secondly, if there is time, implement further functionality.

1.3 Goal and purpose

The purpose of the study and research will be to:

- evaluate the development of a tablet-based client, with Android as the platform of choice,
- evaluate how different solutions could look like when it comes to the networking part between client and server,
- investigate how an appropriate User Interface would be designed and how it would best suit this kind of system, and finally
- look into possible image processing functionality of the client and how to solve this in a suitable manner.

The purpose has been modified from the initial plans.

[1] For a description of Bonjour, se section 3.2.1

1.4 Method

By a thorough study of the Android platform architecture I will try to find a design solution suited for the purpose. I will do research on if there are any eventual problems with the solution and how, if they exist, should be solved.

Studies will also be made in the field of Human Computer Interaction in order to find a good graphic and interactive solution, for the purpose of providing a user-friendly experience for the users.

The task assigner will provide an Android 3.0-based device for testing purposes and evaluation.

1.5 Delimitation

Implementation issues may occur since the Android platform is still on an immature stage and it still lacks extensive documentation. And also because of the somewhat unexplored area of this thesis's focus, it may be hard to find sources that will support me in the research. This may become an obstacle when it comes to find a proper implementation solution.

Although the Android platform uses Java as programming language, it is hugely different to develop for mobile devices than computers. Component lifecycles, memory management, threads etc. in mobile development are also somewhat unexplored areas for me.

Since my solution is dependent on another student's solution and code, the server, which is developed on a different platform, compatibility issues may occur. The project is also dependent to actually be tested on a real device since the emulator supplied by the Android SDK environment does not allow network interaction.

2 Background

2.1 One-to-one computing

One-to-one computing is defined as "*a student uses at least one computing device for learning*" [8]. In teaching environments this means that every student and teacher possesses a computer, Internet and software that allows every user to be accessible. The term "computer" means PC, laptop, net-book, hand-held devices and tablets [10].

The formal definition stated above is quite old, from a time where students possessing their own computer are considered somewhat unusual. Nowa-days, the case is different since most people already have their own laptops, so supplying them with either software to be used on their own devices or even supplying them with computers in some cases, is a problem of no con-cern.

2.2 Tablets

Since the development of more and more sophisticated smartphones the in-terest for the hand-held computer market has increased. The introduction of, among others, Apple's iPad did not only change the market but also created a whole new way of using the new generation of tablet computers. The so-phisticated touch screens, the high-end hardware and long battery life ena-bles countless of ways and environments for these devices to be used in.

Other major companies as Acer, Asus, HTC and Motorola also accompanied the trend and created their own similar devices. The main differences be-tween these devices however are foremost the operating system, in which there are two major participants competing for the customers, Apple's iOS and Google's Android. There are also some Windows based operating sys-tems on some devices, although these have not gained as much recognition on the market so far.

With the fast growth of these kinds of systems and especially with the ease of use and user friendliness they provide. The fields that these devices could be used in are a very interesting and current topic.

2.3 The Android platform and devices

Android [3] is a software platform initially developed for smartphones, but is now also developed for tablet use. Android's SDK supplies tools and APIs necessary to be able to develop for the Android platform, with Java as primary programming language. Android is based on a modified Linux kernel. The operating system's software consists of Java applications that run on an object oriented application structure. Android was developed by Android Inc., which later was bought by Google the year of 2005.

An "Acer Iconia Tab A500" device is used for testing and is based on the Honeycomb 3.0 operating system for Android. It has a 10.1-inch screen with a resolution of 1280 x 800 WXGA and a 16:10 aspect ratio.

2.4 Human-Computer Interaction

When developing for mobile devices and tablets the importance of HCI becomes more significant. On a computer screen the resolution is high and the user navigates using the cursor, which gives very high accuracy. This enables buttons, text and graphical components to be very small. It also enables a lot of information to be crammed into the workspace without it feeling cluttered. When it comes to mobile and tablet design we have to rethink our layouts a bit more thoroughly. The workspace is an issue and choosing the right content to show in the right moment becomes crucial.

The importance of a seamless mobile system becomes more important than for traditional computer environments. The user should not need to look around in an application for different functions, as we are used to with today's computer software. For a mobile device the intuitivity of a solution is the first most important factor when using an application.

3 Methods and design development

3.1 User Interface

To find a suitable User Interface for the purpose of creating a user-friendly environment, I will look into mobile and portable device guidelines supplied by Android and information from previous courses taken in Human Computer Interaction.

An initial proposal was made during early stages of the research process, this image is actually an image mockup based on an iPad framework, *Figure 1,*
It was only intended to be used as a proposal and a way to build further ideas from. On the top a progress indication and the four buttons represent, from left to right; send slide to server (teacher), previous slide, next slide and draw on slide.

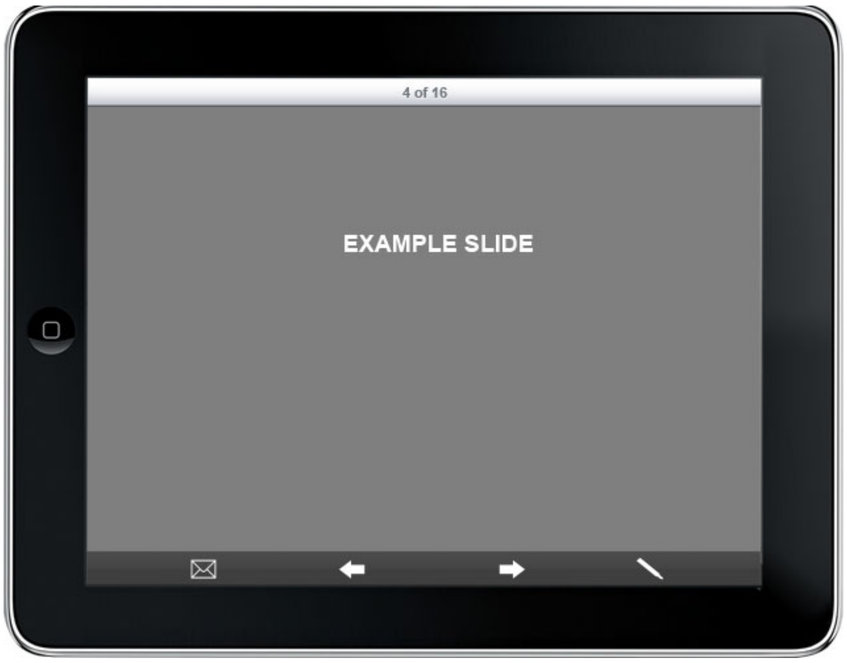


Figure 1. A initial proposal of the layout.

3.2 Server-Client network model

The obvious model for this type of application is of course a client-server network model. Since the existing server is built upon a solution based on the Bonjour protocol and thus on UDP, the idea is that the client will be implemented to seamlessly connect with this system. When the user starts the application a discovery service starts up in the background looking for the server and connects to it. What I will do is to explore how this could be solved using Android in conjunction with the existing server.

3.1.1 Bonjour

The Bonjour networking protocol [11] is an implementation of Zero configuration, a protocol for discovery of network services. Bonjour can discover devices such as printers, computers and other devices' services that are available on a local network.

Bonjour uses something called multicast Domain Name System, mDNS [12]. This means that each computer on the local network stores a list of DNS resource records and then joins the mDNS multicast group. When a client wants to know the IP address of other devices, given its name, the mDNS client sends a retrieval request to an already known multicast address, the device with the corresponding record will reply with its IP address and port number. This is then used to create a socket connection, using either TCP or UDP.

3.2 Image Processing

The client as we know is provided with the broadcasted slides from the server. Some of these slides contain information that the users can interact with, for instance multiple-choice questions. First of all what needs to be done is to convert the Power Point-formatted slides to a suitable image file type, in our case JPEG is fine.

To be able to distinguish what is a question slide and what is just an information slide we use something called horizontal projection [6] within image processing.

To complete this functionality within the client's implementation we would probably need library functions for the Power Point to JPEG conversion. Then it is necessary to apply an algorithmic solution for the horizontal projection to determine where to create clickable links, possibly using histograms to distinguish text from background.

4 Results

4.1 Implementation Android

The Android framework is based on one or more application components; *activities, services, content providers* and *broadcast receivers*. In this application we will focus on *activities* and *services*.

Content providers manage the data stored within the system, such as an SQLite database, content on the Internet or another local storage location. At this stage of the implementation *Content providers* is not necessary. Secondly, *broadcast receivers* are components that respond to announcements made by the system, such as; *a picture has been captured* or *the battery is low*. These broadcasts are also of little concern for us at this stage.

Back to the *activities*, an activity represents a single screen with a user interface. I implemented three activities; one main screen, which handles the login process which requests a user ID, the discovery service for connection to the server and the actual slider part with the pictures. The user can in theory jump between these activities by pressing the 'back' button (application icon) supplied by the OS and then start certain activities again, *figure 2.*

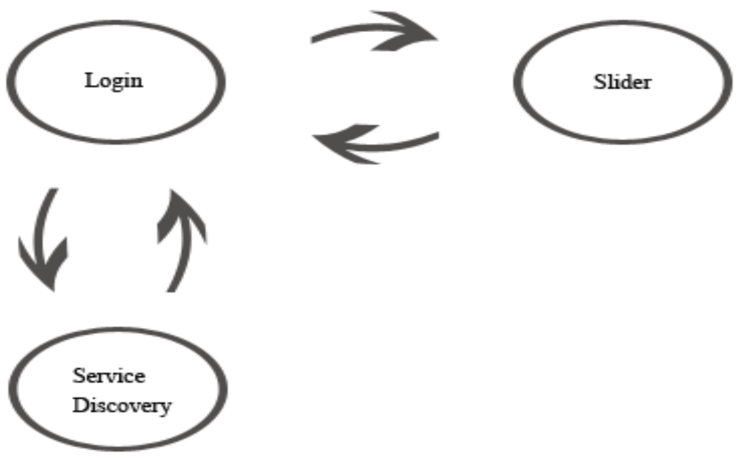


Figure 2. The three main activities.

The connection between activities is something called an *Intent object*; it binds individual components together and creates a bridge between them. It provides a means for communication between activities, by passing along messages for instance. In *example 1* you can see the code that creates a button that in turn creates the Intent object and bounds it with Slider.class and starts its activity.

```
/* Button that starts the slider and shows the images */
Button startSlider = (Button) findViewById(R.id.Button02);
startSlider.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
        Intent myIntent = new Intent(view.getContext(),
        Slider.class);
        startActivityForResult(myIntent, 0);
    }
});
```
Example 1. Code to create a button that creates the Intent object and bounds it with Slider.class and starts its activity.

The discovery service is actually at this stage a foreground activity and not a service. The reason for this is that it is easier to work with activities in the foreground and make sure they work properly before we transfer them to work as a background service. However, the Android implementation guidelines discourage the use of network activities in the foreground, as they tend to consume too many resources. Although as a proof of concept, we accept this at the time being.

4.1.1 Differences from the Java point of view

What differentiates typical Java programming that I am used to, from the one used when programming for Android is that of the different application components, mentioned before. When the *Main.class* is started for instance, we need to extend the Activity object to create something visual, a first screen. Then what we need to do here is to override the *onCreate* method with says what we shall do with our new activity object. In this case we bound the activity with a layout file called *main.xml* that structures visual object such as buttons and text fields for instance, see *example 2*.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ...
}
```
Example 2. Code that sets the visual properties, run when Main.class is started.

Below is a graphical representation of the activity lifecycle, see *figure 3*. These methods are all part of the Android API and are crucial in the management of activities. This framework is built upon Java as mentioned but tightly connected to hardware functions within the Android devices, this is one of the core differences when programming for Android compared with regular Java programming.

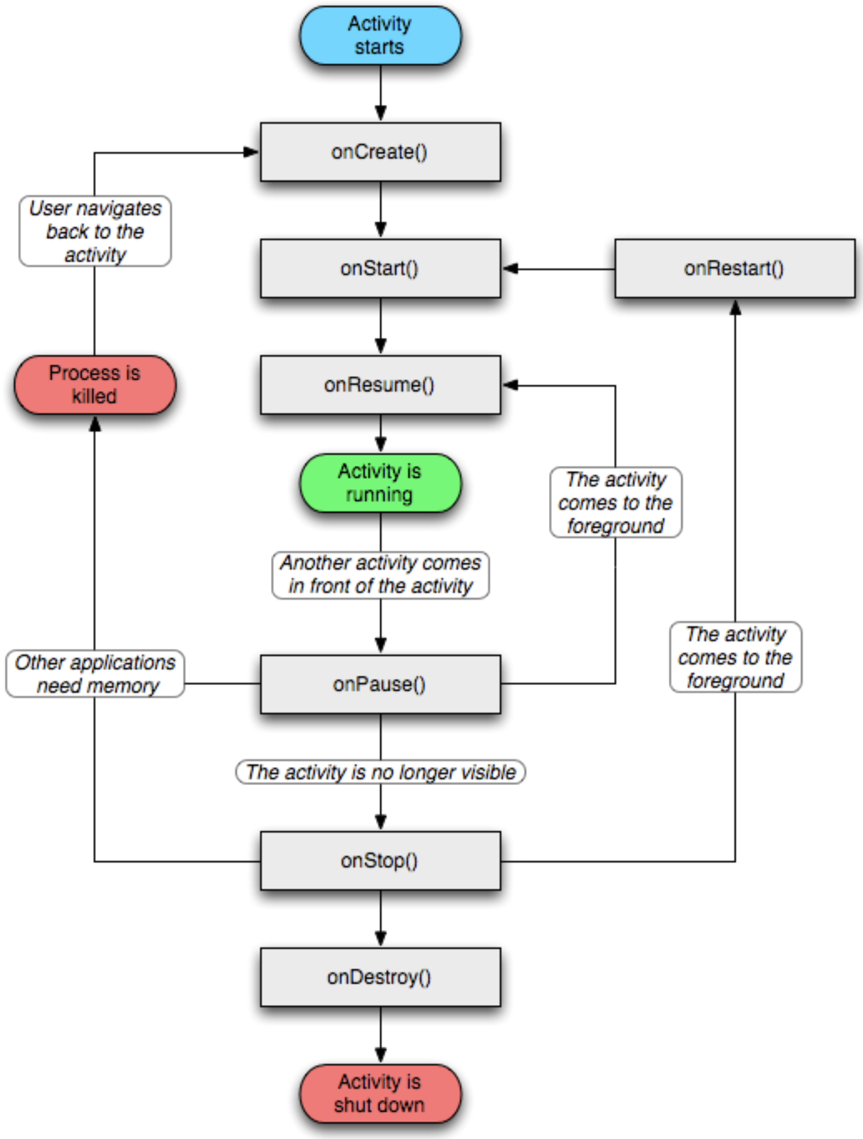

Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Figure 3. The activity lifecycle. (*developer.android.com*).

4.1.2 Network solution

The initial idea was for the client to have a built in service discovery algorithm for picking up the server's broadcasted IP. Connect to the server, let its user send their student ID and begin receiving content from the server, in form of previous mentioned Power Point slides. As it turned out the Android platform does not really support the Bonjour library, which the server uses, this because of the simple fact that there is no implementation made for Android by Apple [1].

What some developers has managed to do is to use a software library called jmDNS [7], which is a Java based library built for Zero configuration communication. The jmDNS works as a broadcast search for devices, in similarity with the Bonjour library.

Initially I only had access to a HTC device running Android to test the already existing solution that was built in Java by the other student, however it seemed that these devices had some bugs [4] when it comes to DNS-broadcasting [5].

Another issue is that, even though jmDNS and Bonjour are built on the same idea of Zero configuration, the API is totally different and would require the re-writing the methods of the jmDNS library to function properly with the Bonjour library methods.

In a latter stage of the work process I acquired an Android 3.0 tablet from the task assigner to test the jmDNS library for service discovery. It turned out that the Android device actually could find other services of Zero configuration kind in the vicinity when I did a live test. However, although the one-to-one computing server was running next to the device, the tablet client was unable to successfully pick up its broadcast.

With some consultation with the task assigner and the other student who developed the server we agreed upon leaving the networking part aside, and continue as if it had worked. One solution to this problem could to be implementing an alternative way of connecting to the server, using a direct socket connection with the server's IP given beforehand. In other words this would mean to hard-code the connection on the client to the server, bypassing the broadcasting part. Doing this would require the server code to be rewritten and tested in close conjunction with my system. However this was something that was beyond my control and influence and will probably be considered in a future version of the server software.

4.2 Human Computer Interaction

4.2.1 User Interface

Smartphone applications are generally designed to show as little information as possible to make them easy to navigate and understand. However when it comes to tablets we have to re-think this general ideology a little bit. All of a sudden we are supplied with high resolution and large screens, but still with smartphone features. We are in this situation stuck somewhere in a boundary country between a smartphone and a computer.

According to Android UI Design Patterns [9] the general idea of “less is more” is emphasized. For instance, it is encouraged that not more that four to six possible choices, or buttons, are presented on each screen to avoid cluttering. However since they refer to smartphones and not tablets using Android, this is not really our concern. However we should still keep this in mind since the user still uses its fingers to navigate and there shall be enough space around clickable items to make the user feel comfortable and not afraid to miss-click. In our case the application does not really have a lot of functionality so staying within the guidelines is done with ease.

After the connection is established to the server an input text box appears centered on the screen, asking for the student ID-number, the virtual keyboard is set to appear. The user input its ID number and presses confirm which is a button directly under the input field, see *Figure 4.*

The slides sent from the server appear on the screen, and the device is now connected to the server and the teacher can broadcast all slides to the students.

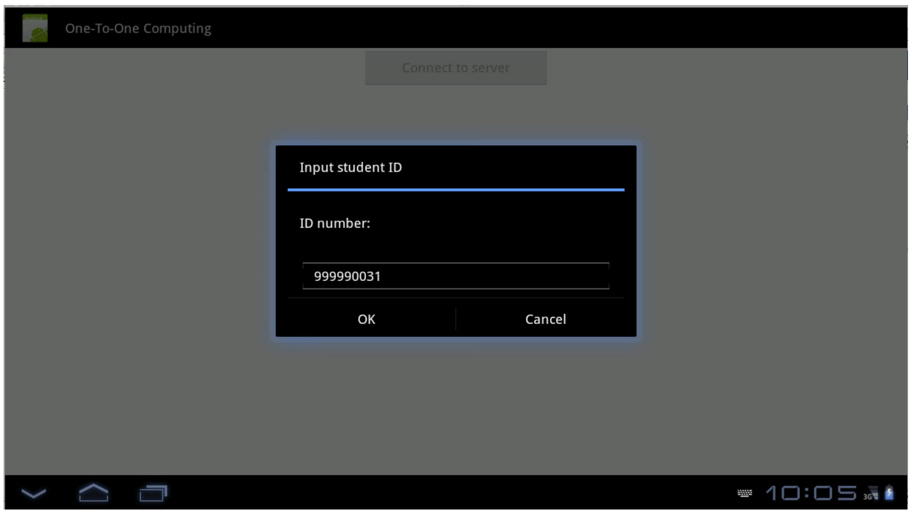
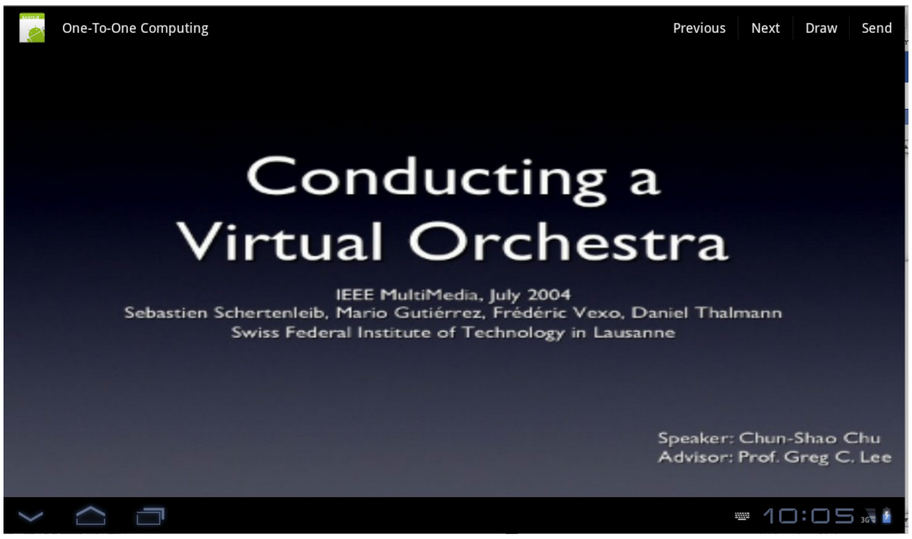


Figure 4. _The Login screen.

What is shown on the screen is now the current slide and the Android action bar, located at the top, with a few buttons. If the user presses anywhere on a slide image a toast message appear, displaying the number of the current slide out of the total number, for instance ‘3 of 14’. A *toast message* is a quick little message that pops up on the screen and stays there for a brief period of time, then fading away. This is a very useful way of informing the user of the current progress through the slides requiring no additional interaction from the user.

On top of the screen is where the action bar is located. To the left of it is the application name with a corresponding application icon. Pressing this icon will bring the user ‘up’ one level or back to the Android OS menu if the user is on the root level of the application, otherwise just back to the previous activity. On the rightmost side are four buttons, *Previous*, *Next*, *Draw* and *Send*, see *Figure 5*. The *Previous* and *Next* are self-explanatory, however the *Draw* and *Send* buttons, need some explanations.

When the user presses the *Draw* button a toast message appears for a brief moment indicating that the user is in *Draw mode* and they will be able to draw on the screen with their fingers, and when they are done they simply press the *Draw* button again. This could, for instance, be used to mark words or point out things of interest to the teacher.



Conducting a Virtual Orchestra

IEEE MultiMedia, July 2004
Sebastien Schertenleib, Mario Gutiérrez, Frédéric Vexo, Daniel Thalmann
Swiss Federal Institute of Technology in Lausanne

Speaker: Chun-Shao Chu
Advisor: Prof. Greg C. Lee

Figure 5. The Slide Screen.

When the user presses the *Send* button, the actual screen the user sees is sent to the teacher and a status toast message displays that the action is processing, '*Sending slide...*'. The feedback is given since it is important to show the user what is going on when an action is commenced.

To slide from one image to the next or previous image the user just swipes the finger in the opposite direction of the new image, e.g. swipe finger to the left for next image.

The slides are shown as fixed size images, in order to fit the resolution of the actual Android device. Therefore they will look good on any 10-inch screen with similar resolution. Although for the final application to be considered ready for launch, it needs to be scalable to handle different resolutions, screen sizes, new models etc. since we don't want to restrict the system to just tablets of a fixed screen size.

Android has thought about this and implemented good solutions to handle these kinds of matters. The use of XML-sheets for the layout description makes it easier to organize your code and follow the MCV-standard if used correctly, in other words avoiding putting layout features in the model rather than in the view. This makes it fairly easy to implement different styling schemes and layouts for various kinds of DPIs, as Android call it.

As most tablets have gyros they can sense their position, if they are held in landscape view or portrait view.

Since the application cannot really use this functionality to its advantage in its current stage, the orientation detection is turned off. Another reason is that the most common way to look at a Power Point slide is in the landscape orientation, so in order not to confuse the user we keep to this well accustomed tradition.

Each application also comes with something called an *AndroidManifest.xml*-file, the purpose of which is to setup a set of rules and permissions for the application, for instance, which platform-version the application is designed for. Additionally, in our implementation, the manifest file is used to enable picking up Wi-Fi-multicast signals, which is turned off by default to save battery life.

4.3 Image Processing

4.3.1 Horizontal projection

Horizontal projection is done by first detecting the orientation of a binary image; the detection is based on the vertical and horizontal variances on an image. In our case we know that the orientation is horizontal from the beginning, which eliminates the need of orientation detection.

Then by using an algorithm [13] that determines what is foreground and what is text; we are able to produce a projection histogram based on the horizontal and vertical pixels. Then by applying a threshold to the histogram we can determine where the actual text is located on the Y-axis. Finally, by measuring the height of the image we can predict where these clickable areas are located and create links based on their location.

In *example 3* we can see how an algorithm for scanning through an image and extracting the pixel values and putting them in a vertical and horizontal arrays. These arrays can later be used to create the histograms and determine where the text is located on the image in order to create the links.

```
/* Horizontal projection */
public void horProjFunc(Bitmap image){
    int W = image.getWidth();
    int H = image.getHeight();
    int[] horProj = new int[W];
    int[] verProj = new int[H];
    for (int v=0; v<W; v++){
        for (int u=0; u<H; u++){
            int p = image.getPixel(u,v);
              horProj[v] += p;
              verProj[u] += p;
        }
    }
}
```

Example 3. A code sample that calculates the sum of the vertical and horizontal pixels.

There is a library called Apache POI [2] for Java, which has the functionality to convert a Power Point slide into a JPEG image. Since we need to have an actual image to work with for the horizontal projection, this is a crucial part. However, after some research, it turned out that the POI library is not likely to work with Android without major re-coding, which has become another cul-de-sac for the application development.

Although, the horizontal projection method is implemented into the slide activity but it's functionality is not really fully developed.

4.4 Result summary

What I have produced in the end will be described as follows. In addition to the read up on the Android platform, it's programming conventions, the GUI conventions and guidelines, network solutions; their compatibility in in conjunction with Android.

I have created three classes; *Main.java, Slider.java and DiscoverService.java* and several xml-layout files in order to create a solid foundation for a computer-to-computer tablet client. What we have is a suitable graphical user interface, following HCI standards for tablets, with all the elements that were requested by the task assigner, proposed solutions for the network connectivity and the image processing functionality.

In more detail we have *Main.java* that starts the other two activities through intents. *Main.java*'s layout is written in the *main.xml* file which styles the buttons used in the activity.
Similar solutions exist in the *slider.xml* and the *menu.xml* files connected to *Slider.java*. The difference in implementation here is that *Slider.java* uses both *slider.xml* and *menu.xml* for styling purposes since the menu at the button is considered an individual object.

There is also a package including library methods from the API, for drawing on canvas with fingers, within the application. These methods although were never implemented due to lack of time. Thus, neither the *Draw*-button nor the *Send Slide*-button functionality is due to the network problems we had when it came to the establishment of the server-client connection described above.

5 Discussion

Android as a development platform has its potential, it gives the developer a lot of freedom, compared to the fruity counterpart. This, however, also has its drawbacks, since it makes it easier to make mistakes and furthermore the developing environment is not as seamless as it could be. Most errors show up at runtime rather than at compilation, which makes debugging tedious.

What made me kind of surprised is that, even though I have some experience programming in Java, the whole SDK of Android is overwhelming and learning what is necessary in this short amount of time to be able to finish the implementation was simply not feasible. Android has, in my opinion, 'teething problems' and does not feel as mature as it should be at this stage, especially when it comes to documentation, examples and general support by its community, which tend to be more focused on professionals than on beginners.

In the initial phase of the project the decision to develop on Android or iOS was discussed. What made Android the candidate of choice was that it would probably be more compatible with the original server, which was written in Java. However since Android's Java and the typical Java did not seem to share the specialized libraries that was needed for this specific project, Bonjour and Apache POI for instance, it did not really help to code in the same programming language. As a matter of fact, developing for iOS could actually had made the whole process smoother, with the support for the Bonjour protocol in mind.

Since Apple has not released a Bonjour library suited for Android at this point, developers need to look in other directions, such as the jmDNS library, which instead lacks the solid foundation of users, implementations and documentation. It is basically a hack, cheating the system is various ways to mimic the functionality of Bonjour. That makes the use of jmDNS less of a long-term solution, but merely useful as a proof of concept.

The server, which was developed by the other student, had not really been tested with other devices than computers at this stage, which made it a bit difficult to solve some of the occurring issues.

For this reason it is important to see the current Android implementation more as a proof of concept, than a final working solution.

At the point I received a real device to test with, I was already far in the process of the project learning about the Android environment. Testing with the device and the server unfortunately gave negative results; it was able to discover other services but not the actual server, as it was supposed to. This led to focus and continuation of other components of the project instead and putting the network part aside.

However when this decision was made there was not enough time to fully implement the initially intended functionality, so I chose to focus on the User Interface part for the rest of the project while still implementing a foundation for future development for the other intended parts, such as the send slide and draw on slide functionality. This is one of the down parts with working on something that requires external solutions to work in conjunction.

The implementation that I have come up with is more of a solid proposal for continuation of development. The system can detect other IP broadcasting services, what needs to be added is the connectivity part once we got a server IP address, in terms of network activity.

The graphical user interface is well suited to guide the user through the application in a user-friendly matter, as sough in this kind of application. The user can slide through the slides by swiping his or her fingers, as expected with touch-screen tablet. A quick touch on the current slide brings up a toast message, which shows the actual slide out of the total. The top menu has the application name and the four buttons *previous*, *next*, *draw* and *send*. None of these are implemented but are connected to four corresponding methods. Classes for finger drawing on an image is included in the package and ready to be implemented. Sending a drawn-upon image or answering a question slide should not be hard to implement either, once a socket connection is setup with the server.

For this kind of system to be developed on a professional level a large number of improvements could be made. First of all, the network solution using Bonjour is a good idea, although maybe with TCP/IP as transferring protocol instead. TCP/IP has the advantage of the QoS aspect, packet acknowledgement, retransmissions etc. that UDP lacks. One server with, say hundred students to supply slides to and receiving answers from, is not sustainable with the current solution.

If the server still sticks with UDP for media transfer maybe UDP/RTCP is a solution for the control messages and delivery acknowledgement. However this solution may be too excessive in this kind of environment.

One-to-one computing in learning environments is a really interesting and promising field and doing research on this area has been rewarding to say the least. This since it involves so many aspects, Human Computer Interaction, networking problematic and hand-held devices that need to work in synchronization. What adds up when developing these kinds of systems are security matters on the networking side, UDP is for instance not a good solution when the number of clients start to increase, as mentioned before.

The topic feels very right in time since the technology finally have caught up with the theories.

References

1. Android's official webpage
   http://www.android.com/
   2011-05-18

2. Apache POI
   http://poi.apache.org/
   2011-06-06

3. Apple Mailing lists (no bonjour support for android).
   http://lists.apple.com/archives/bonjour-dev/2009/Jun/msg00024.html
   2009-06-10

4. Google Code Android (Issue with HTC)
   http://code.google.com/p/tunesremote-plus/issues/detail?id=1
   2010-09-26

5. Google Code Android (Issue with mDNS)
   http://code.google.com/p/android/issues/detail?id=2917
   2009-06-08

6. Horizontal projection
   http://archive.nlm.nih.gov/pubs/le/spie93/spie93.php
   Date unknown

7. jmDNS
   http://jmdns.sourceforge.net/
   2011-01-20

8. One-to-one computing
   http://www.educationworld.com/a_tech/tech/tech197.shtml
   2004-09-01

9. Richard Fulcher, Chris Nesladek, Jim Palmer, Christian Robertson
   "Android UI Design Patterns"
   http://www.android-app-developer.co.uk/android-app-development-docs/android-android-ui-design-patterns.pdf
   2010-05-19

10. S. Digangi, Z. Kilic, C. H. Yu, A. Jannasch-Pennell, L. Long, C. Kim, V. Stay, S. Kang, "One to One Computing in Higher Education: A Survey of Technology Practices and Needs." *Association for the Advancement of Computing In Education Journal* (AACE Journal (2007) vol. 15 (4) pp. 367-387

11. Wikipedia about Bonjour
http://en.wikipedia.org/wiki/Bonjour_(software)
2011-04-17

12. Wikipedia about Zero configuration
http://en.wikipedia.org/wiki/Zero_configuration_networking
2011-05-14

13. Wilhelm Burger, Mark Burge, "Digital image processing: an algorithmic introduction using Java" pp. 234.
2007-11-28