Shaocong Mo
PRIME 2012

Multi-touch Interactive Applications

Introduction:

There were two parts for this project. The first part was to create two gallery programs that communicate with each other (one for touch table and one for android device) and provide location-based services for users to navigate San Diego Museum of Art (SDMA) with their preferred artworks in their android devices. The second part was to build an interactive educational puzzle game that provides users detailed information about an artwork whenever they finish the puzzle game for that specific artwork. The project was a collaboration between PRIME from UC San Diego, SDMA, and National Institution of Information and Communication Technology (NICT) from Japan.

The Big Challenge: Time

To be honest, this project was quite frightening for me because the second part of the project was suddenly requested by SDMA after we started working on the first part for quite some time, which means we had to build three programs in total in such a short time frame. Although my major is computer science, we never need to build actual applications for our classes. Then all of a sudden I had to create three applications with Michael Yao, another applicant for PRIME 2012 who is also a computer science major. While fearing that we might fail to complete all three applications, I kept working as hard as possible because I know being worried was not a solution. Challenge accepted!

Project Part I: Touch Table

General Idea:

After some brainstorming with the help from our mentor Dr. Jason Haga, we decided to build a simple grid interface with 25 artworks for the touch table that allows users to drag the artworks they want to know more about to a drop area. After they finish selecting the artworks, they can press the CREATE TOUR button to create their personal mobile gallery tour. As they press the button, the information of the artworks is sent to their android devices and they are set to go.

Multi-touch API– Open Exhibits:

Since we were working with a touch table, we needed ways to detect touch points and gestures created by users. To start off, we searched online for different options that were suitable for our applications. Out of the many options, Tangible User Interface Object (TUIO) protocol seemed to be a good option. Its library was provided for many different languages, such as C , Java, C#, and many more. As we learned more about it, however, we figured it would take us a long time to learn how to set up the whole multi-touch system and create our application on top of it. After some series of studying and testing, we decided to use Open Exhibits multi-touch SDK, which was recommended by Dr. Haga and was used in last year's PRIME project. Open Exhibits SDK was developed using a Flash-based language called ActionScript. Using Open Exhibits gave us a few advantages over TUIO. First, Open Exhibits has pre-made templates that work on Adobe Flash and everything inside the templates is set up and good to go, which had helped us save enormous amounts of time. We just needed to simply apply the multi-touch gestures to the objects we created, and that was it.

Another advantage was that since last year's PRIME students had used this API before, we knew it could work and we knew who to ask for help if we somehow got stuck and ran into a dead end. This was a big advantage because it was important to know if the multi-touch SDK could work when there was little time for us to work on the rest of the functions.

Interface:

After we chose the multi-touch SDK, we started to work on building the interface. Surprisingly, at first I thought someone from SDMA would help us with the designing aspect. It turned out that we needed to sort out everything on our own. Since both Michael and I had little user interface designing experience, it was a tough job to come up with the actual interface. After days of drawing the interface in Adobe Flash and adding actual functions to each individual part, we finally came up with this interface and gave the application a temporary name—Mobile Gallery:

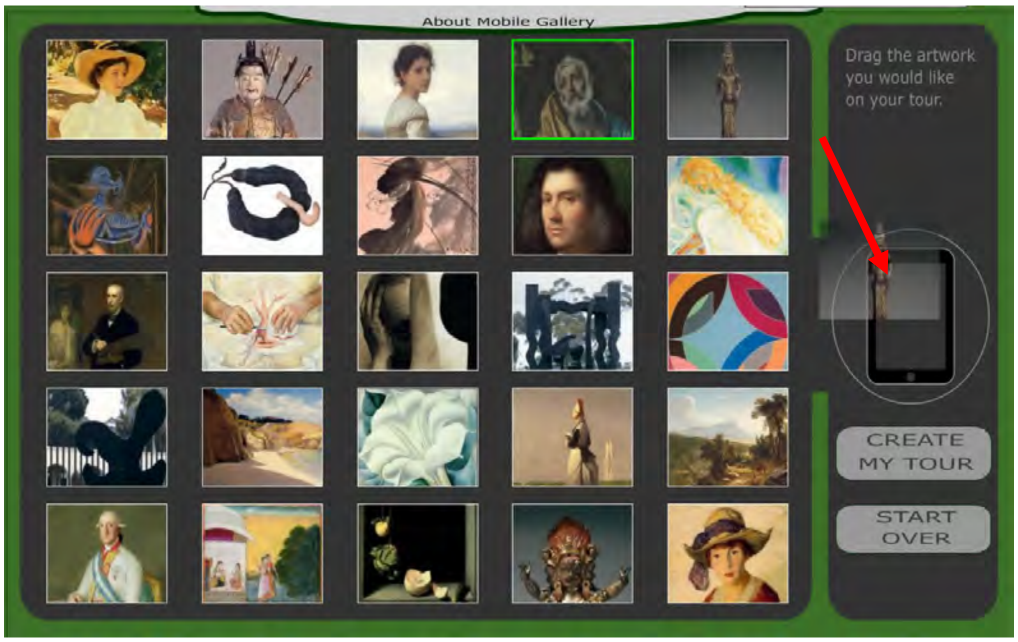


Fig 1. Mobile Gallery Interface

As figure 1 shows, when a user drags an artwork toward the drop area, a ghost image of the artwork appears just like when we drag icons around on our computer. When the artwork's center point touches the drop area, the artwork is highlighted with green borders around it and its opacity is lowered to 0.8 as visual cues that tell the user which artwork has been selected. As simple as this interface may look, we indeed spent some time to draw it even though we already had some discussion about the design with Dr. Haga. Fortunately I am not and will not be in any part of the design industry.

Tweener:

To add some animations to the interface, such as the sliding down of the about panel and the fading in/out effect of some pop out screens, we used an open-source library called Caurina Tweener. Tweener is an ActionScript library that allows smooth single or multiple animations on objects using codes (more flexible) instead of using timeline in Adobe Flash. To show how it works, here's an example. If we changes the x-coordinate of an object from 0 to 20, then the object simply teleports to x=20. When using Tweener, the object's x is changed slowly from 0 to 20 within a certain time frame to create a delusion of moving object.

```
Tweener.add( object, { x:20, time: 1 });
```

This line changes the object's x to 20 within 1 second. If the object originally is at x=0, then after 0.5 second, its x is 10.

Although this Tweener class is not very essential for this application, its smooth animation allows for a better user experience by avoiding awkward, teleporting popup windows.

<u>Project Part I: Android Device</u>

R.java:

Playing with android and building application for it had been my favorite part of this project; however, the excitement halted for a short while as we began our first try of building a simple Hello World program (a program with no function but showing “Hello World!” on the screen) on the android device provided by NICT.

We encountered a problem where eclipse (a super convenient text/program editor) cannot compile even the simple Hello World program. It kept saying that R.java was missing. When we tried to import R.java from Android SDK, eclipse showed another error message saying it cannot find the specific properties needed from R.java. We did some research and found out that we needed to select the “build project” option to have eclipse to generate the R.java file to link all the resources for the application. Without solving this problem, we cannot even start programming. For us, this was THE little step that led to the following great creation.

Android application workflow:

The name for the android application is also called “Mobile Gallery” to correspond to the touch table application. To avoid confusion, I will use “android application” and “touch table application” instead of two “Mobile Gallery”. Here’s the rough workflow of the android application. For the first screen, we created a vertical, single-columned grid that holds the artworks, the authors, and the titles. Below the grid are two buttons—“Update Tour List” and “Start Tour”. When a user presses “Update Tour List”, the android device requests information from the touch table and shows the selected artworks for the user to see. Then, the user can press “Start Tour” to see the artworks (see Fig 2).

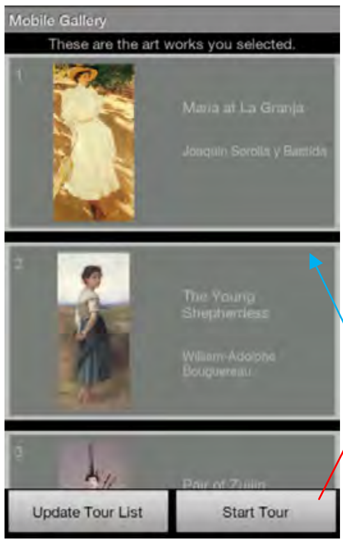

Mobile Gallery
These are the art works you selected.
Maria at La Granja
Joaquin Sorolla y Bastida
The Young Shepherdess
William-Adolphe Bouguereau
Update Tour List
Start Tour

Fig 2. First Screen

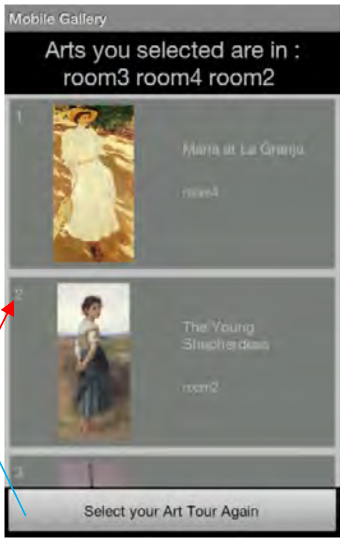

Mobile Gallery
Arts you selected are in : room3 room4 room2
Maria at La Granja
room4
The Young Shepherdess
room2
Select your Art Tour Again

Fig 3. Navigation Screen

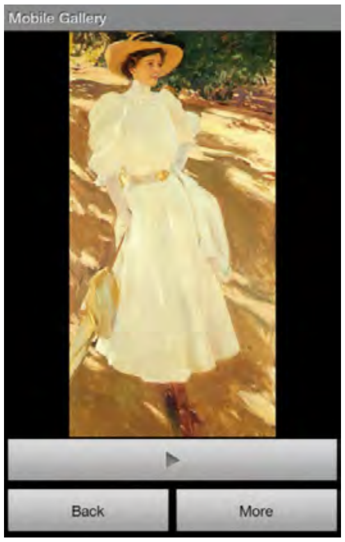


Fig 4. Information Screen

The user can also go back to the first screen by pressing “Select Your Artwork Again” (see Fig 3). As the user gets closer and closer to a certain artwork, if that artwork is selected by the user, the android device allows the user to see the information and video about that specific artwork (see Fig 4).

PlaceStickers:

With the help from Shimojo sensei, we were able to utilize a short-ranged signaling device called PlaceSticker. Each artwork is attached to a PlaceSticker as a means to locate and detect the nearest artwork during the tour session of the android application. PlaceSticker is developed by a company called Information Services International-Dentsu (iSiD).

Indoor Navigation System:

During one of the weekly meetings, Dr. Haga and Shimojo sensei suggested us to build an indoor navigation system using the PlaceStickers to inform users how to get to each artwork. This was an excellent idea and we immediately started to search online and learned about how normal indoor GPS systems work. Sadly we figured it was rather complicated to use short-ranged device like PlaceStickers to achieve our goal. Instead, we marked each artwork with a location so that users can know where the artworks are as their tours start. During the project demonstration day, Dr. Haga, Shimojo sensei, Vivian Haga from SDMA, and other people were satisfied to see how the navigation screen (see Fig 3) turned out.

Communication and Flexibility:

One of the hard parts of this project was that SDMA wanted it such that they can change the information and artworks even if they know nothing about programming. In order to achieve this function, the first thing we could think of was building a database and a server that would allow the android application to access and retrieve information through Wi-Fi; however, setting up a database and a server is not easy at all. We would need to learn a lot about networking before we can even start building a functional database. The time required for that would not be enough for us to complete the application on time. Furthermore, although using a database is efficient, it still requires the data manager to have some knowledge of networking.

Pressure kept haunting us for a few weeks and we still could not come up with any solution. Although Kadobayashi-san and Morimoto-san from Osaka Electro-Communication University were introduced by Shimojo sensei to help us with the application, there was still no progress. The language Morimoto-san used to write the demo application and server was in processing. It was not very smooth and efficient at rendering animations and high-resolution artworks on the touch table. Another problem was that it would surely take some time for us to understand how Morimoto-san's PX server protocol works.

Nonetheless, several days later, we were fortunate enough to find a way that allowed the communication between android device and a java server. We managed to have the server to read an xml file locally with all the information of the artworks and send the information to the android application. This way, people from SDMA can change the information as they see fit by just editing the xml file on the touch table. With that said, once the android application is finished, they do not need to update the application through android market in order to change the information of the artworks. This is also faster for users to download the application since now the application does not contain any large size images and videos.

The hidden problem—Flash platform:

Not until the last few weeks did we finally realize that there was a hidden challenge behind the use of Adobe Flash. As we came to the end of finishing all three applications, we found out that Adobe Flash was web-platform only, meaning that it cannot manipulate local files on its own. Since part of Part I was about the communication between android device and touch table, we needed our touch table application capable of accessing the java server locally to transfer data to android device.

To solve this problem, we did some heavy research online and found out that Adobe Flash can actually connect to the socket port of the local host. Therefore, we created another java server that connects to the local host port of the touch table. This way, this local server can communicate with Adobe Flash and pass the information to the server that connects to android device. Although this method sounds a bit complicated and inefficient, the whole process actually runs seamlessly. SDMA only needs to take care of the xml file.

YouTube Player:

The last screen of the android application contains a YouTube video that contains the information about the corresponding artwork. The problem was that Google has not provided any API for YouTube player yet, although Google made an announcement about it in July. In order to play YouTube video inside the android app, there were only a few options. One of the options was to use the YouTube app on android device to play the video; however, this would mean that the users must have the YouTube app on their devices, and they have to leave our application in order to watch the video. Luckily, there was an open-source library which was originally created by KeyesLabs and updated recently by Rajesh R. This library can create a YouTube player within our android application and was very easy to use. We just needed to call out the YouTube player with the correct website address.

Project Part II: Puzzle Game

Main Screen Interface:

For the puzzle game interface, we used a Flash-based 3D engine called Papervision3D to render a grid of artworks. This grid was designed to be very interactive and attractive to users. Fig 5 shows how it actually looks like. When user drags the grid, the artworks on the grid change. Fig 6 is the result when user drags the grid to another location. Although this interface is very simple, it makes people want to touch it and drag the grid to see what would happen.

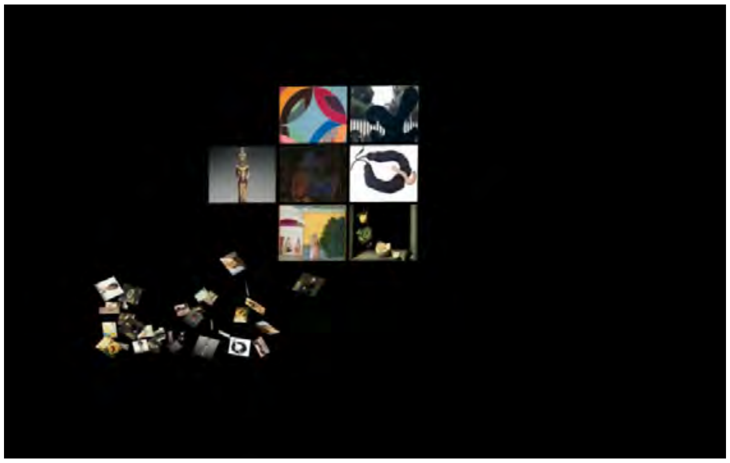


Fig 5. Interactive grid interface (1)

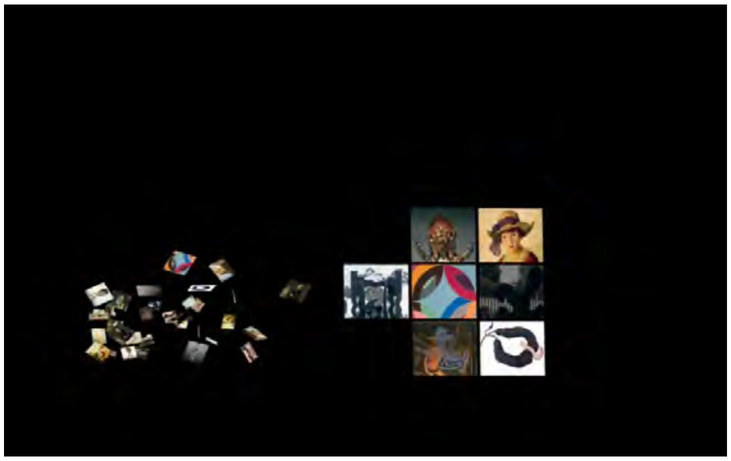
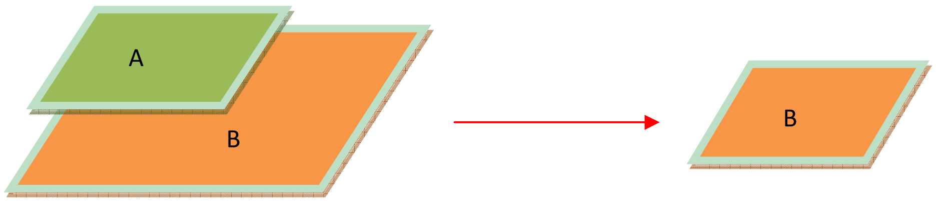


Fig 6. Interactive grid interface (2)

Puzzle:

After the interface for the puzzle game was set, creating the actual puzzle came next. Creating a puzzle might sound easy at first; however, it is a rather complicated process. Although there are lots of puzzle games out there, most of them are not flexible enough for us to automatically cut all the artworks into puzzle pieces. Also, there isn't any ActionScript 3 open-source code out there for us to learn about making puzzle. The ones we could find were in ActionScript 2, which is substantially different than ActionScript 3.

After hours of reading through ActionScript 3's documentation, we figured we might be able to use masking to create a puzzle. Here's the theory. To create a puzzle, we used two classes from ActionScript 3: MovieClip and Mask.



Two layers are combined into one MovieClip and only the area of layer B that is covered by layer A is shown. In our case, layer A is the shape of a puzzle piece that masks over layer B and layer B is an artwork. Using this method, we can create puzzle pieces for all the artworks without cutting them individually. The application is also more flexible that SDMA can change the artworks as they wish.

Conclusion:

As fast as time can fly, we have finally finished all three applications on time. Although some sleeps were lost on the way, the project was a success! This project was by far the largest scale I have ever done, but also the most intriguing challenge. The process of completing the project has taught me important skills on working with real world project.

First, when working in real world setting, one has to be adaptive and knows the meaning of teamwork. Since in reality, there are many factors that can affect the result of the project, such as changes in requirements or target populations. Working in a team can speed up the process of problem solving. Being adaptive has also helped us change our plans accordingly and prioritize to fulfill SDMA's requirements of the applications.

I also learned that the core of computer science is about problem solving. It's not about how many programming languages one knows. Throughout this project, we have used ActionScript 3, Open Exhibits, Java, Android API, and Google API. Although it sounds like a lot, the fundamentals of these languages and API's are pretty much the same. They just have different syntax, grammars, and structures. Indeed it took us some time to learn about them, but in the end they were all the same. They were tools for us to solve problems using our computer science knowledge and techniques.

Bibliography

Android-YouTube-Player. Google Code. August 2012.
< http://code.google.com/p/android-youtube-player/>

Open Exhibits. Ideum. June 2012. < http://openexhibits.org/>

Papervision3D. Open Source Flash. Oct. 2008. June 2012. <http://osflash.org/papervision3d>

Tweener. Google Code. June 2012. <http://code.google.com/p/tweener/>

色素増感型太陽電池(DSC). iSiD. June 2012. <http://www.isid.co.jp/news/2011/1004.html>

Acknowledgements

I would like to thank the following people for giving me this opportunity to work on this multi-touch project and making this PRIME experience a great success:

UCSD:

- Dr. Gabriele Wienhausen
- Dr. Peter Arzberger
- Teri Simas
- Jim Galvin
- Tricia Taylor
- Dr. Jason Haga

NICT:

- Dr. Shinji Shimojo
- Takata Tomoaki
- Masaki Chikama
- Yoshinori Kobayashi

SDMA:

- Vivian Haga

National Science Foundation:

- IOSE90710726