Department of Computer Science and Engineering

The Chinese University of Hong Kong

vPresent

Collaborative Presentation System on Mobile Devices

Fall 2012

Final Year Project Report

Project Code:	LYU1203
Students:	Sinn Lok Tsun (1155002358)
	Leung Chak Hang (1155000316)
Supervisor:	Professor Michael R. Lyu

This is a blank page.

Abstract

vPresent is a project promoting a new concept of presentation – Collaborative presentation. Collaborative presentation is the concept of presentation allow multiple presenters and views to make contribution to presentation, and making the presentation more interesting and unique. With raising popularity of mobile operating systems and mobile applications, we implement applications on mobile platform for promoting collaborative presentation. The application demonstrate, but not limiting the concept of collaborative presentation. The concept of collaborative presentation do not limit or require form of presentation, but pointing to a new direction for improving presentation style.

Table of Contents

Chapter 1. Introduction

1.1 Background

Presentation systems was a large market for both education, business etc. In education aspect, teachers, lectures and professors use slides show and presentation software for lecture. Commonly used presentation software include Microsoft PowerPoint and Apple Keynote. Some lecturers also use portable document format, as known as PDF format, for packing multiple images as slides for lecture. Other than lecture, presentation system is also used in conferences. Presenting a thesis, researchers and scholars would also use slides to present their thesis in conference. Presentation is widely used in transferring knowledge on educational purpose.

Regarding business use, presentation is also widely use: people use presentation slides to promote their products; staff make presentation slides to report his / her latest work; company use slides show to presenter company's status to shareholder. Moreover, press conference also use presentation and presentation slides for introducing the new products or statistics of company.

Attending and watching lots of presentation, we found presentation composed of two components: presentation style and presentation slides. In most presentation, we can see the presentation style is unidirectional and presenter dominate the presentation; and the presentation slides are just image with limited animation. We are going to do some marketing research of existing presentation systems, looking at the presentation style and presentation slides format.

1.2 Marketing Research

We are going to discuss the following systems:

- Presentation Software
  - Microsoft PowerPoint
  - Prezi
- Presentation Recording System
  - Opencast Matterhorn
  - Echo 360
  - Camtasia Relay

1.2.1 Presentation Software

Microsoft PowerPoint

It is one of the most famous presentation software in the world. The main presentation structure is sequential and linear, which mainly deliver the presentation contents by slides. Also, PowerPoint supports importing graphics and video into the slides.

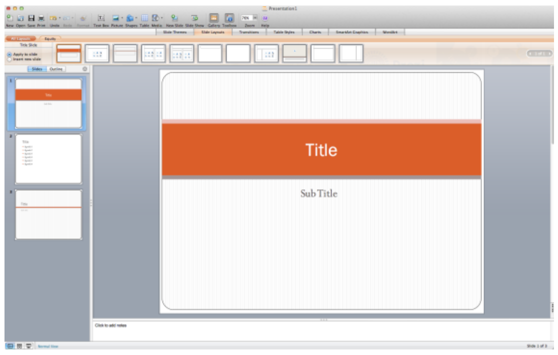


Fig.1 Microsoft PowerPoint 2008 for Mac Screenshot

During presentation, the mouse pointer can be changed into pen pointer, which allow handwrite marking on the slides during presentations. It provides various type of handwrite marking style, including the ball pen pointer with different colors, and also the highlighter style.

Typically, to control the flow of a PowerPoint slides presentation, a list of slides and the animation sequence of the slides have to be obtained. If our application needs to support PowerPoint presentations, the controlling protocol has to obey the flow control rationale of PowerPoint.

Prezi

Prezi was found only a few years ago, but the popularity increased drastically in the past 2-3 years. This brand new presentation software mainly distribute their services through the Internet and edit the presentation content mainly through the web browsers, it introduced another approaches of presentation style.

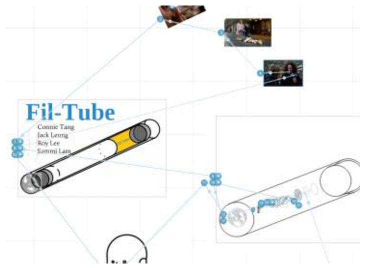


Fig.2 Screenshot of Prezi Editing Screen

One of the main feature of Prezi presentations is that Prezi disposed the idea of slides, but expressing the presentation content using zooming in and out. If our application needs to support Prezi presentations, the controlling protocol will have to be tailor-made for them, for example the zooming ratios and delivery of presentation file contents.

1.2.2 Presentation Recording

Opencast Matterhorn

Opencast Matterhorn is an open source project for presentation recording and video management. The project is initiated by University of California Berkeley, coordinating with several universities and institution.

Fig.3 Logo of Opencast Matterhorn

The system is composed of a server and capture agents as client. Server is used for encoding of presentations, and also a web server for management panel, based on Linux or Mac OS X Server. In contrast, capture agent is an add-on box to the presentation computer using Linux. The presentation computer split the output video signal to two, one to projector and another to the capture agent, feeding into VGA2USB card of capture agent computer.

Presentation record can be done by ad hoc, issuing command to capture agent from server, or according to schedule. After recording of presentation is done, files is sent to server via internet. Receiving the files, server would encode the files and stored on it. After encoding, presentation can be viewed by access a web player hosted on server. The server manage all encoded presentation as video repository.

As an open source system with media and encoding functions, there are many dependency on other multimedia libraries. Fixing outdated dependency is difficult and time consuming during installation, requiring extensive experience on Linux. In addition, the system only provide functions on recording, managing and replaying presentations, without other features enhancing learning or communication with audience.

Regarding to presentation style, it tries to capture the lecture and podium screen in recording, which is good for unidirectional presentation. And discussing about the presentation slides, the capture agent capture the entire screen, no matter what kind of slides using.

Echo 360

Echo 360 is a client-server presentation recording system with rich functionalities. As we cannot obtain a license for the system, we could not have extensive practical experience on it. We here only giving a brief idea on the system.

Fig.4 Logo of Echo 360

Echo 360 aimed at educational purpose, mainly lecture recording. The system can record a lecture by ad hoc, or on scheduled time. Recording could be done using an external box produced by Echo 360 Company, or directly on podium computer. In addition, Echo 360 support video broadcasting, and able to interact with students by asking questions. However, we are not able have experience in broadcasting function.

After recording or broadcasting, the original files upload to server, followed by encoding into package of files. The presentation then can be played with the web player, bundled with bookmarking and note taking functions.

A highlight about Echo 360 system is synchronization of multiple tracks. The synchronization is done by a XML-liked file, recording the offset between echo tracks. During encoding and packing, the offset would be used for synchronization.

We found a lot to inspiration about functionalities from Echo 360, including note taking, bookmarking, broadcasting etc. Moreover, the use of XML give us direction for synchronizing multiple tracks.

In conclude, the presentation style is still presenter-dominated, with limited viewer involvement features like viewer raising questions. Regarding presentation slides, again, recording is capturing the entire screen thus there is not limitation on slides format.

However, Echo 360 is able to capture content of PowerPoint and image-based slides for better integration.

Camtasia Relay

Camtasia Relay is a system focused in recording and encoding of presentations. It provide installed and portable recorder running on presenter computer. Recorded presentation is uploaded to encoding server for encoding, and publish to storage.

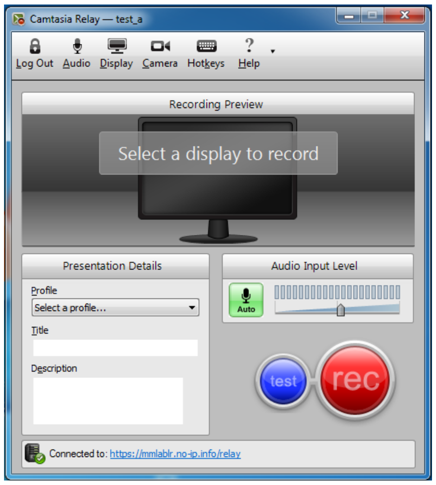


Fig.5 Screenshot of Camtasia Relay Recorder

Focusing on recording features, Camtasia Relay do not have other features for communication or broadcasting with interaction. However, the recording features with complete and clear flow of presentations with comprehensive encoding features is worth for reference. It do not limited or require any presentation slides format. But regarding to presentation style, the camera is still only capturing single presenter and not good for multiple presenter use.

1.2.3 Summary

To summarize, we found most presentation software provide more functionalities for image-based slides, but still able to support other type of presentation slides by capturing entire screen. In addition, we found there is another type of presentation slides named Prezi.

Prezi use zooming animation and not strictly divide slides by different images. Prezi is a new type of presentation showing application.

There is another type of presentation slideshow or content showing application. However, the presentation style is still remaining the same, only single presenter dominate the presentation. Therefore, we are going to focus on presentation style, tackling the problem that single presenter always dominate the presentation.

1.3 Motivation and Objective

In traditional slide-based presentation, usually only one presenter dominate the whole presentation. If there are two or more presenters responsible for the presentation, they have to either use their partner's machine and continue to present with the set of slides; or disconnect their partner's machines from the projector and connect their own device back to the projector. Both methods introduce inconveniences and interruptions towards the presentation flow, especially for the second one, not only because that is time consuming for the projector itself to recalibrate for the newly connected device, but may also induced many unnecessary errors during the exchange of machines, such as the device cannot detect the projector, or the aspect ratio is not correct after the device connected to the projector.

Secondly, we found there are not enough interaction between presenters and viewers. In traditional presentations, presenters dominate the presentation, talking at all the time. We have try other software and systems of presentation and make some review in the previous section. However, features involving viewers are few, limited to allow viewer post questions, or calling a vote among viewers. Therefore we can observe the following: even with different group of viewers, the presentation is similar when the presenter and slides is same. The presentation is just similar to video playback and viewer involvement is few: viewer is not affecting or changing presentation content, and the presentation do not change even the viewer not presence. In addition, listening to presenter's speech and reading presenter's slides, viewers only receive information. Presentation is just a unidirectional information transferal.

Therefore, we are trying to promote a new concept of presentation, namely collaborative presentation. Collaborative presentation aimed at preventing single presenter dominate whole presentation by allowing others join and contribute to presentation, and the ultimate goal is to vague the boundary between presenters and viewers. Promoting the new concept of presentation, we are going to develop an application on iOS, demonstrating characteristics of collaborative presentation.

Chapter 2. Collaborative Presentation

2.1 Concept of Collaborative Presentation

Collaborative presentation is trying to make as more as people presence in presentation to join and contribute the presentation. We first divide people presence in presentation to three types: moderator, presenter and viewer.

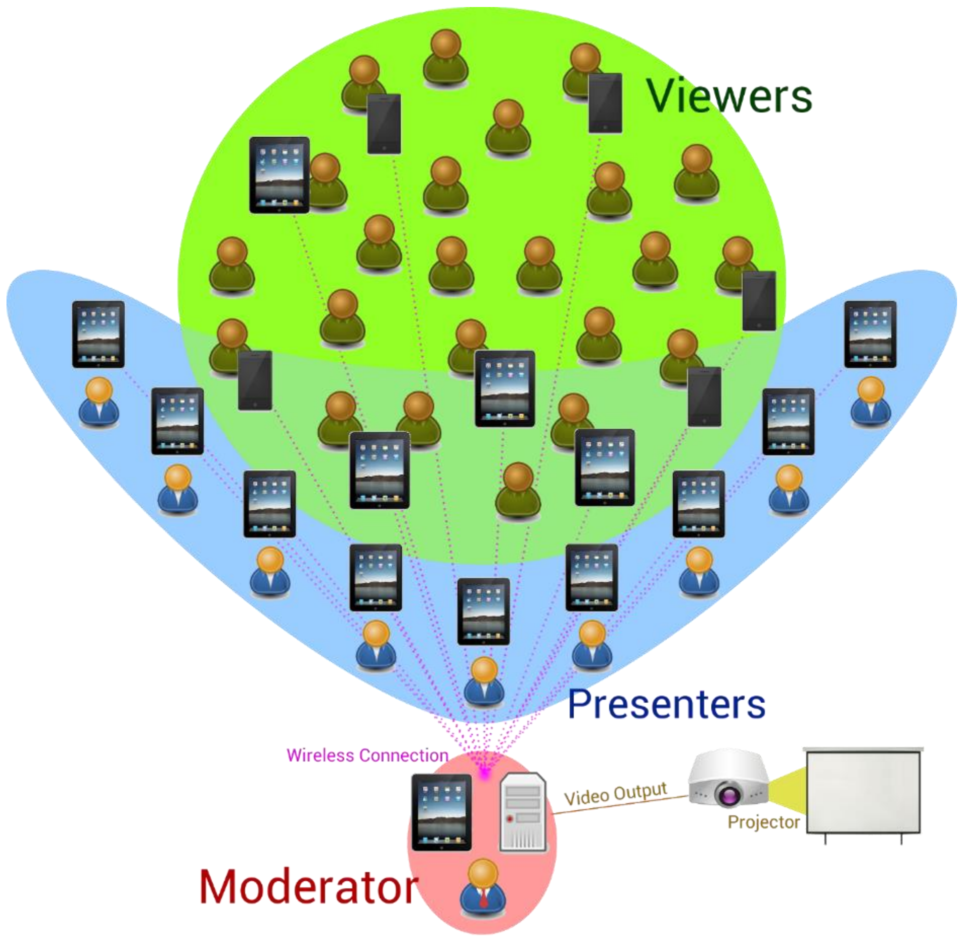


Fig.6 Collaborative Presentation Conceptual Diagram

Moderator should be unique in presentation. The main role of moderator is to monitor the presentation, and coordinating the presentation with presenters and viewers. Device of moderator will also act as server, connecting to external monitor or projector. For some special circumstance, moderator may be able to act as presenter, presenting his / her own set of slides.

Presenter can be a group of people. Those people responsible to present their own content during presentation. Therefore they would bring their own set of slides to presentation, storing in the device. During presentation, each presenter may take turn to present his / her own slides, becoming active presenter. The remaining others may be inactive and can act as a viewer.

Viewer is people coming to listen to the presentation. They would not bring any own material. The viewer would be allowed to make short presentation, but still stick to presenter's materials, and making remark or commentary on presenter's material. By having the control of presentation and giving short presentation, viewer can contribute to presentation and the ultimate goal would be enhancing communication.

In order to make the connection easier, all connection between devices would be using wireless connection. The only cable connection should be connection to external monitor from moderator device. The connection to external display can be done with converters and existing video cables such as VGA, HDMI etc. Converting including 30-pin or lightning to VGA or HDMI for iOS mobile devices; for Android devices, MHL for micro-USB to HDMI, requiring device chip support, and mini-HDMI to HDMI could be used. There are plenty choices in market and most devices would be able to connect to external monitor.

2.2 Key Features

Collaborative presentation contains two features, contribute by other presenters and collaborate with viewers.

2.2.1 Collaborate with Other Presenters

Collaborating with other presenters means during the presentation, there are multiple presenters and forming a group of presenter. As mentioned above, there might be multiple presenters in single presentation, but most cases are limited to four presenter, and they have to make slides together. Collaborative presentation can make the slide do not need to centralize to single machine before presentation starts, and also making seamless handover of presentation control.

Each presenter would bring their own device to the presentation. Their own slides are stored in his/her own device. Before the presentation start, it do not need to send their own slides to server, in most circumstance. When one of presenter start presentation, he / she connect to moderator machine. Connecting to moderator machine, the slides will send to server machine wirelessly. Then the presenter can start his / her presentation as usual.

After current presenter finished his / her part of presentation, another presenter may start the presentation. The flow is similar, presenter would send the slides to moderator, and get control of presentation. The next presenter then can start the presentation. During handover of presentation control, presenters do not need to make any cable connection or data exchange physically, but the presentation can proceed seamlessly with another presenter's device and slides.

2.2.2 Collaborate with Viewers

For collaborating with viewers, we are allowing user to interrupt the presentation after permitting by presenter and moderator.

For sake of discussion, we name the active presenter by presenter A. During presentation of presenter A, a viewer namely B, might have question or comment about current content. In such case, B can send a request to server, asking for control permission. The permission would be grant only when both active presenter A and moderator agree to pass the permission. Getting the permission, viewer B can get control of presentation, add drawing on slides and give commentary on slides. However, the right is strictly monitored and presenter and moderator is able to withdraw the permission from viewer immediate, or with few seconds countdown. After viewer B finish the question or remark, the permission is passed back to presenter A and presenter A can continue the presentation with his / her content.

By adding remark and giving short commentary, viewer is able to contribute to presentation. In optimal case, this could enhance communication between presenter and viewer, and also discussion among viewers.

Note that as inactive presenter is similar to viewer when he / she is not presenting, they also able to contribute to the presentation in this way.

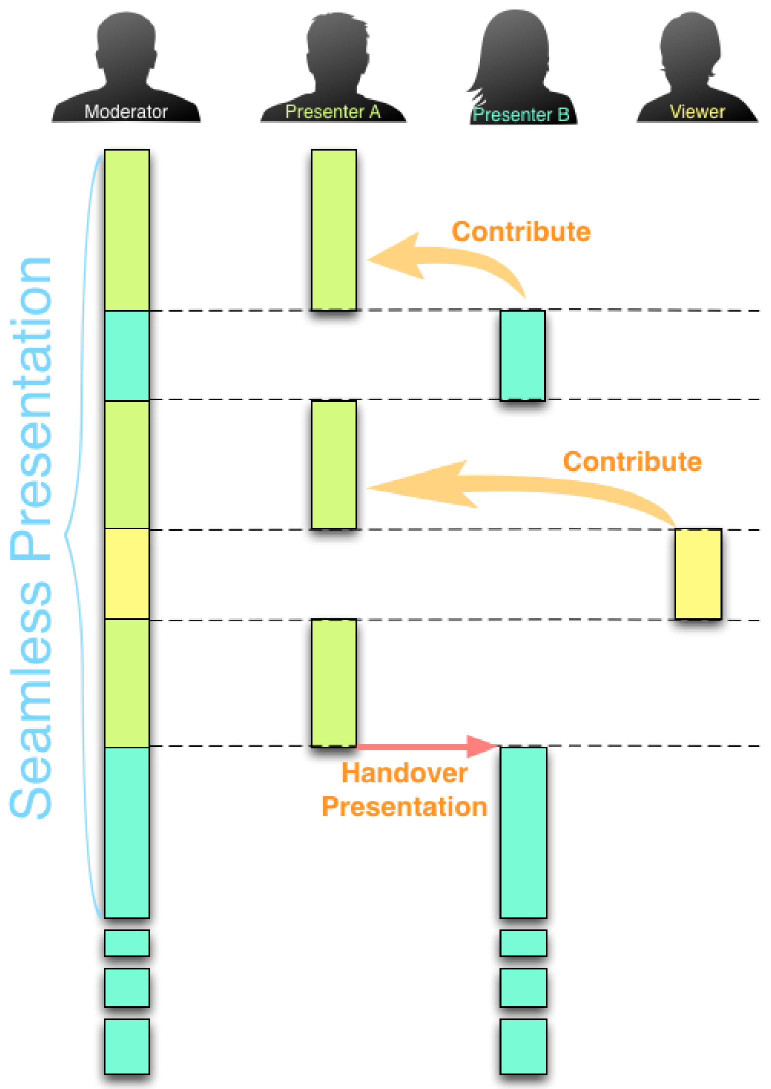


Fig.7 Presentation flow of Collaborative Presntation

2.3 Functionality

Regarding the functionalities, we are going to divide it into two categories. Firstly, functions for controlling the presentation flow and other related system status of collaborative presentation definitely needed to be included. Secondly, we would also include some common presentation software functions.

Seamless handover of presentation control

- Since the moderators is responsible for controlling the whole presentation flow, which have the power to start or cut any presentations delivering by the presenters, this function should be granted to the moderators.

Temporary passing control to others during presentation

- During the presentation delivering by the presenters, other inactive presenters may contribute to the current presentation by sending requests to both the active presenter and the moderator. If both of them accept the request, the inactive presenter would be activated to participate on the presentation originally belongs to another presenter.

Showing slides in External Monitor

- This allow all viewers able to read the slides

In addition to those collaborative presentation-related functionalities, we also include functionalities of basic presentation which would be useful in presentation:

Support of type of Slides

- We are going to support image as slides at first
- With modularization, it is easy to substitute the image with another slide engine by substituting the original image class

Drawing Pad and Canvas

- Allowing presenter to draw control

Recording of Presentation

- As each presentation is unique after collaboration of viewers, recording is important
- We could store metadata instead of capture entire screen
  - e.g. Storing command type and timestamp

Making the entire screen black (or white)

- As whiteboard or blackboard for drawing
- Shielding inappropriate content by presenter or viewer

With those functionalities, we provide all-round features for application.

2.4 Deployment Scenarios

2.4.1 Small Group Meeting

One of the most common scenarios that suitable to deploy collaborative presentation will be a small group meeting that happen every day at offices and campuses. We define "small group" as a group of participants that will not exceed a total number of 10, and among this group of participants, one of them will lead the presentation and this person will be the moderator. Since other participants will have the right to speak, they will all be the presenters.

After the moderator connected his/her own device to the external displaying unit, other presenters can start to connect to a common Wi-Fi access point, and directly connect to the moderator's device thought the IP address and listening port. Once connected, the moderator can see all the connected presenters. Assume all participants can see and hear each other easily, the moderator can simply ask who wish to be the first one to present, and handover the external screen control to the first presenter.

During the handover processes, no external files needed to be exchanged between the presenters' and moderator's device. This saves the overhead when performing handover actions in traditional presentation, either transmitting files between the two devices or reconnecting the external displaying unit. Once the presenters have been granted the control, they can use their own machines and prepared material just have the same experience as their devices have been directly connected to the projectors.

When other inactive presenters want to express some ideas about the content of the current presentation, they can send a request to the moderator and the current presenters, and if both of them allow this inactive presenter to have the control, this inactive presenter will become active and he/she will be allowed to contribute on the presentations with the original presenter. After that, the new presenter can also present their presentation contents and become the main presenter.

For a group size in less than 10 participants, direct connect between moderator and presenters is a feasible solution, as the iPad can theoretically accept much more connections simultaneously, still the battery usage would be the major concern after a prolong time to maintain the states of the connections and needed to process the output to the external displays. So if there are too many presenters, direct connections will not be a favorable solution.

2.4.2 Class and Lecture

In this scenario, usually only the lecturer will be the presenter. Other students will be at the role of viewers. Viewers can have the right of getting the presentation contents, contribute by making request to the presenters and moderator, but they are not allowed to upload their own content and dominant the presentation.

As a result, this scenario will have a small number of presenters, but large amount of viewers. Comparing to the previous deployment scenarios, some modification can be made in order to increase the performance and efficiency of the overall system. A separate server can be setup only for storing the presentation content provided by the presenters, so that to ease the network traffic of the moderator device by reducing the upload content amount.

2.4.3 Conference

Another suitable deployment scenario would be council meetings or conference meetings in very larger scale. For example, the meetings of the general assembly hold at the hall of United Nation in Geneva. There are nearly 200 countries participating the conference, in average every nation will send out 2 delegates to represent their own countries. If we include the delegates from the observer countries and other participants, over 500 participants can be expected in the same assembly hall.

That would be definitely impossible for all participants to directly connect to the Secretary-General's device. This deployment scenario stated out the connectivity limitation of a single device. So we need another deployment strategy to overcome this bottleneck.

In this large-scale case, we need another standalone server to serve as a registration server and presentation content delivery server. All participants, no matter in which role, connect to this server with a valid credential, and they will be assigned with different access rights and power. Still, the chairmen have the power to control the whole conference flow by managing and counting the speech time of each delegate, which act as the role of a moderator. Delegates from different countries can send request to the chairmen for requesting a time slot for giving out speeches, and the observers can connect to the network and obtaining the materials from different presenters.

2.5 Implementation Platform

2.5.1 Criteria

Regarding implementation of server, we have to ensure all viewers are able to read the presentation slides during presentation. Ways include broadcasting slides to all viewers and projecting slides onto large screen. Broadcasting slides is a good option but difficult in practical. This assume all viewers are holding a device, able to connect with current presenter or server. The server loading would be heavy during broadcasting. Another option would be using web server and web application for broadcasting but this would be difficult to make active communication. In addition, interactive web application using latest technology may give different experience to user using different browser. Therefore, we device to use project or external monitor for viewers to read slides, and we need server implementation support external monitor.

Regarding implementation for client application, as we want to allow more viewer to join the presentation, the platform should be as common as possible, and preferably one person one device with the platform. If we develop the application based on rare platform, viewer would have to bring a specify device in order to join the presentation, which is inconvenient. In order to lower to entrance to join the presentation, the platform should be common to everyone. Becoming popular in these years, mobile applications based on mobile operating systems would be a good choice.

2.5.2 Mobile Applications and Mobile Operating Systems

There are several mobile operating systems in market, including Apple iOS, Google Android, Microsoft Windows Phone, RIM Blackberry, Samsung Bada, Symbian etc. We have found some statistics about the market share about those mobile devices.

Operating System	**Thousands of Units sales in 2012 Q2**	**Thousands of Units sales in 2011 Q2**
Android	98,529.3	46,775.9
iOS	28,935.0	19,628.8
Symbian	9,071.5	23,853.2
RIM	7,991.2	12,652.3
Bada	4,208.8	2,055.8
Microsoft	4,087.0	1,723.8
Others	863.3	1,050.6
Toal	156,686.1	107,740.4

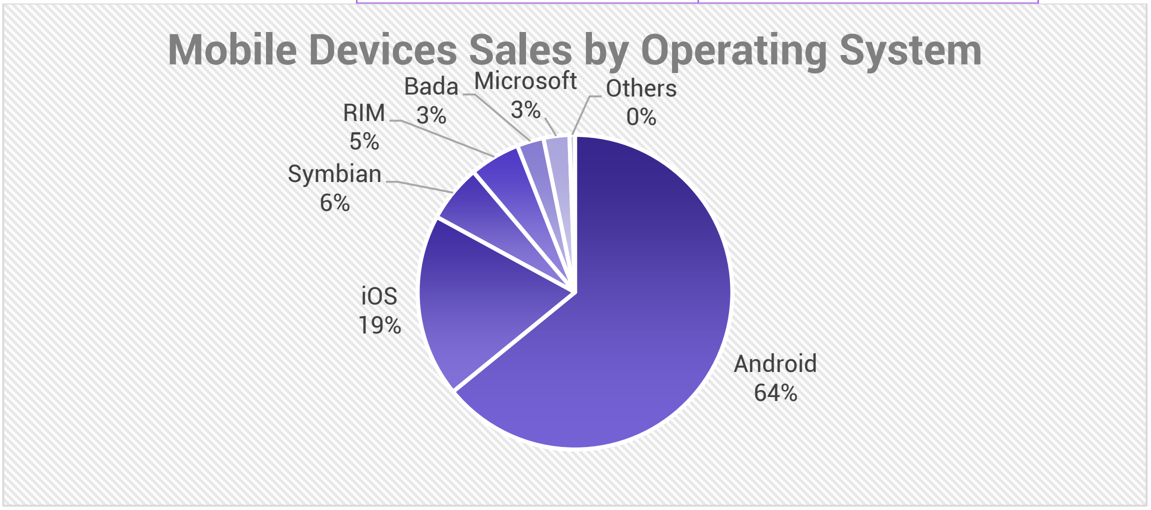


Fig.8 Mobile Devices Sales by Operating System in 2012 Q2

The number is based on statistics by Gartner published on August 2012[1]. From the statistics, we have two observation: firstly, the sales of mobile devices have increased by 50% from 2011 Q2 to 2012 Q2, which proved mobile devices becoming popular in these

[1] *From http://bgr.com/2012/08/14/mobile-phone-q2-2012-market-share-sales/*

days, as stated in previous section; secondly, the top sales by of mobile devices categorized by operating systems are Android and iOS. Therefore, we narrow down the choices to two: Android and iOS.

As discussed before, presentation relies on projector and external display a lot, and we need the server to support external monitors. Despite Android is having large market share in Q2 2012, it do not support multiple display until Android 4.2, announced on late October 2012. Android 4.2, with code name Jelly Bean, is an update to Android 4.1 with same code name announced on June 2012, providing new API level on 17 with some new features. Before release of Android 4.2, connecting to external monitor can only mirror device screen to external monitor. As our project starts on May 2012, and there is no official support of at that moment, we decided not using Android as our development platform. In addition, upgrade of Android major version is slow, as this would involve varieties of devices and manufacturers. It takes long time for Android version to become popular, usually after next major version upgrade. Therefore, it is not worth to develop based on latest version within a short time.

Therefore, we have decided to develop our applications based on iOS with tablet devices based on iOS which is iPad.

2.5.3 Our Decision – iOS

iOS is the mobile operating system developed by Apple from 2007. Latest version of iOS is iOS 6, announced in June 2012 and released in September 2012. iOS only come with iPhone, iPod touch and iPad where interface in iPad is differ from that in iPhone and iPod touch, optimized for large screen tablet experience. As there is increasing trend of using iPad as computer replacement, paperwork and presentation, we focus our development to iPad interface at first.

iOS applications based on Cocoa Touch, a specialized application environment of Cocoa framework. Writing Cocoa applications, or iOS application, Objective-C is most common programming language being used. Objective C is an object oriented programming language, based on and also superset of ANSI C with Smalltalk-liked syntax. Moreover, Cocoa application prefer using MVC design pattern, separating program logic and user interface.

Chapter 3. System Design

3.1 System Structure

3.1.1 System Structure Diagram

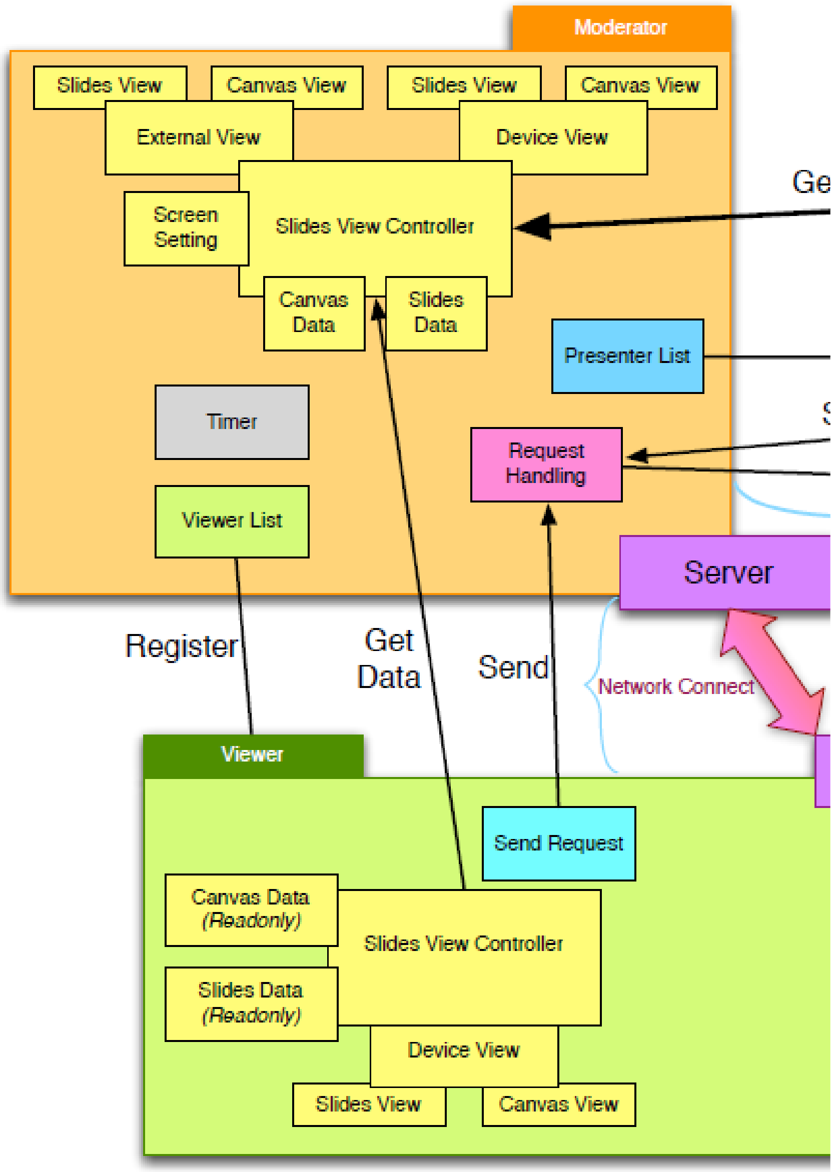
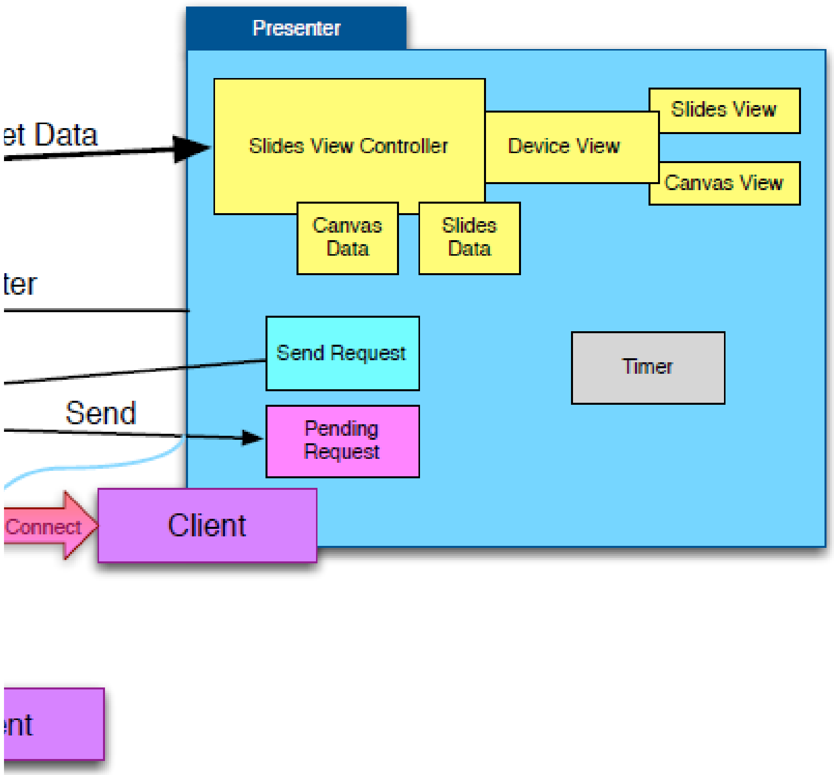


Fig.9 Structural Diagram of vPresent System


Presenter
et Data
Slides View Controller
Device View
Slides View
Canvas View
Canvas Data
Slides Data
ter
Send Request
Timer
Send
Pending Request
Connect
Client
nt

3.1.2 UML Diagram for Moderator

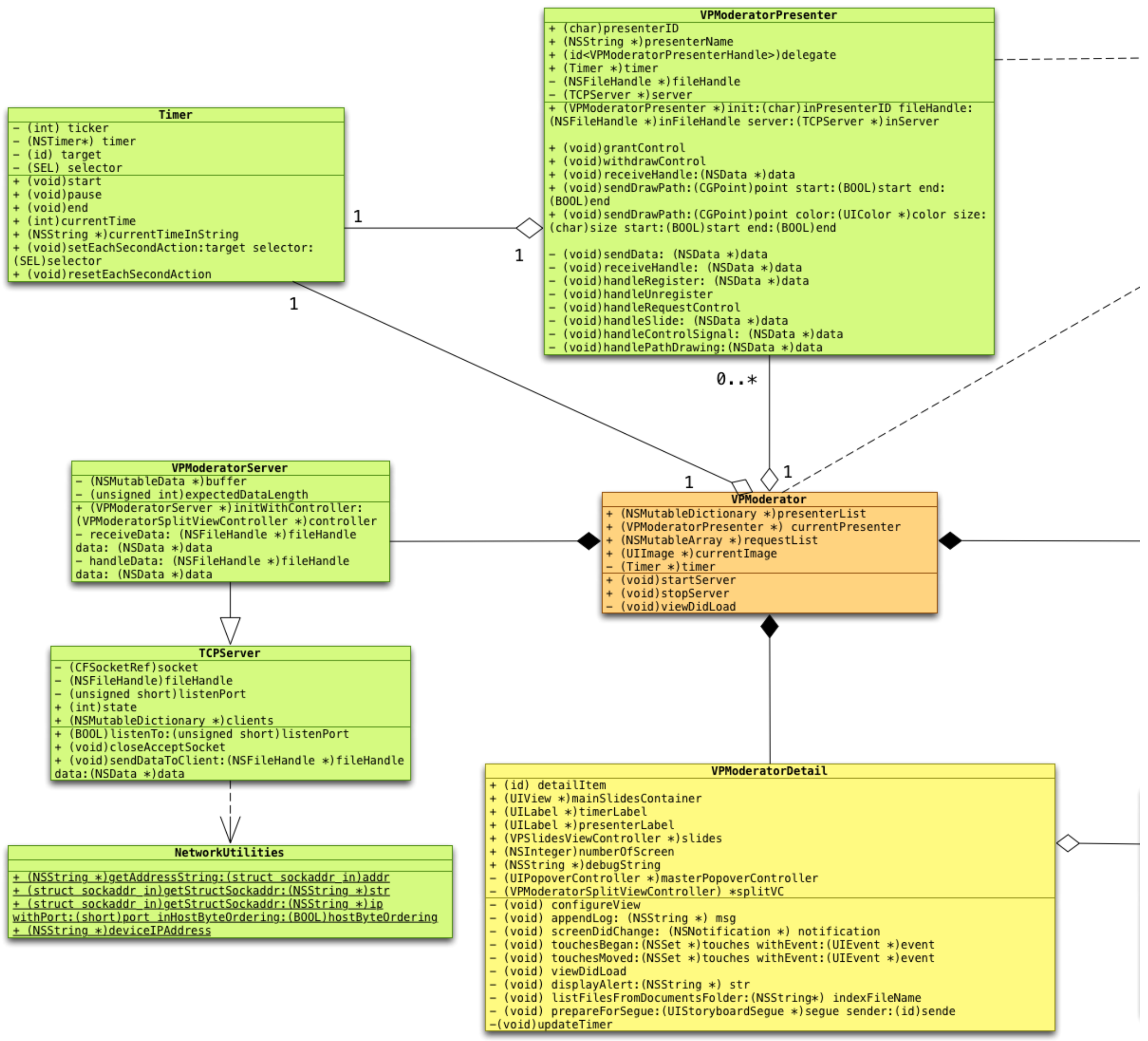
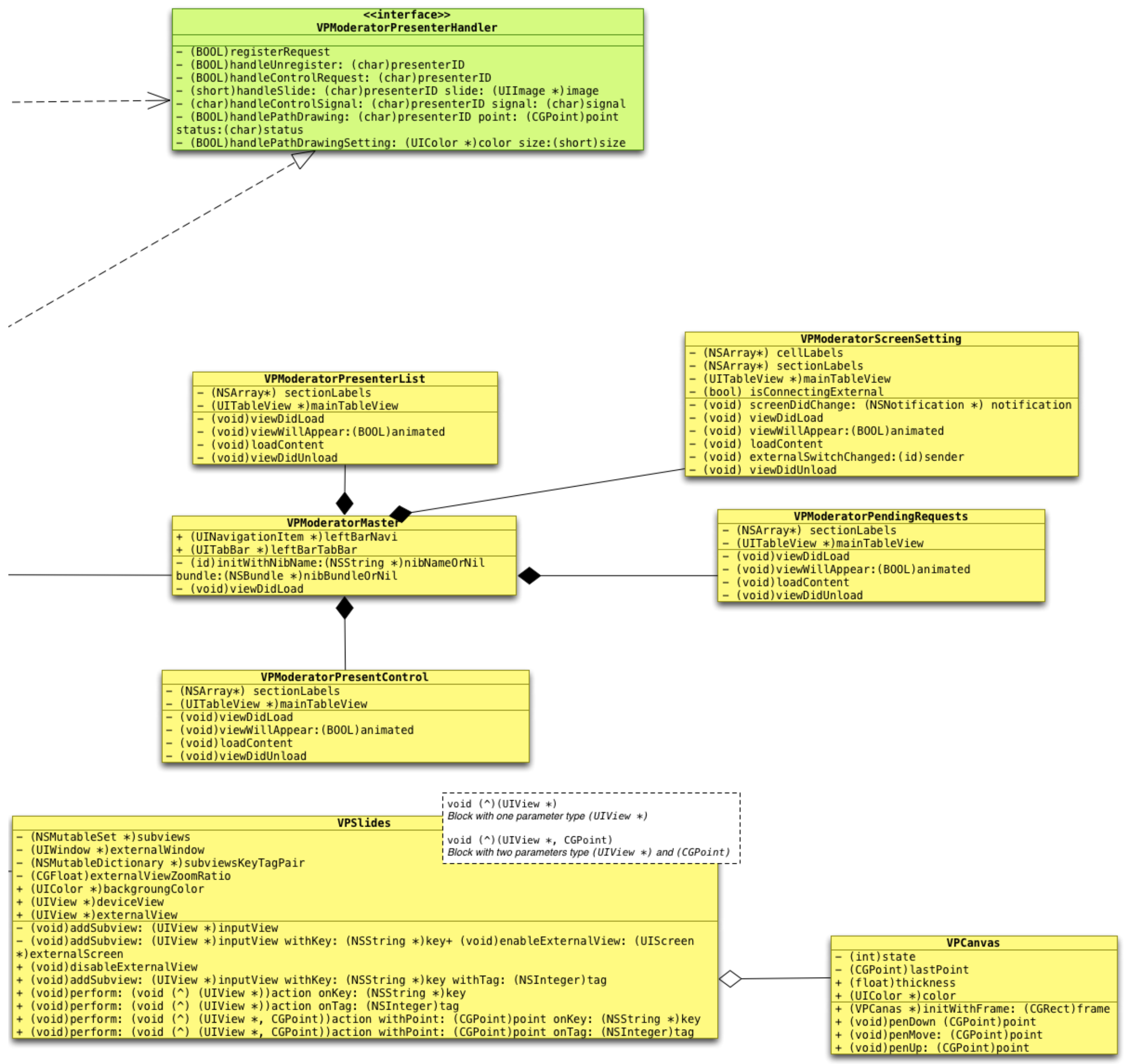

VPModeratorPresenter
+ (char)presenterID
+ (NSString *)presenterName
+ (id<VPModeratorPresenterHandle>)delegate
+ (Timer *)timer
- (NSFileHandle *)fileHandle
- (TCPServer *)server
+ (VPModeratorPresenter *)init:(char)inPresenterID fileHandle:(NSFileHandle *)inFileHandle server:(TCPServer *)inServer
+ (void)grantControl
+ (void)withdrawControl
+ (void)receiveHandle:(NSData *)data
+ (void)sendDrawPath:(CGPoint)point start:(BOOL)start end:(BOOL)end
+ (void)sendDrawPath:(CGPoint)point color:(UIColor *)color size:(char)size start:(BOOL)start end:(BOOL)end
- (void)sendData: (NSData *)data
- (void)receiveHandle: (NSData *)data
- (void)handleRegister: (NSData *)data
- (void)handleUnregister
- (void)handleRequestControl
- (void)handleSlide: (NSData *)data
- (void)handleControlSignal: (NSData *)data
- (void)handlePathDrawing:(NSData *)data
Timer
- (int) ticker
- (NSTimer*) timer
- (id) target
- (SEL) selector
+ (void)start
+ (void)pause
+ (void)end
+ (int)currentTime
+ (NSString *)currentTimeInString
+ (void)setEachSecondAction:target selector:(SEL)selector
+ (void)resetEachSecondAction
1
1
1
0..*
1
1
VPModeratorServer
- (NSMutableData *)buffer
- (unsigned int)expectedDataLength
+ (VPModeratorServer *)initWithController:(VPModeratorSplitViewController *)controller
- receiveData: (NSFileHandle *)fileHandle data: (NSData *)data
- handleData: (NSFileHandle *)fileHandle data: (NSData *)data
VPModerator
+ (NSMutableDictionary *)presenterList
+ (VPModeratorPresenter *) currentPresenter
+ (NSMutableArray *)requestList
+ (UIImage *)currentImage
- (Timer *)timer
+ (void)startServer
+ (void)stopServer
- (void)viewDidLoad
TCPServer
- (CFSocketRef)socket
- (NSFileHandle)fileHandle
- (unsigned short)listenPort
+ (int)state
+ (NSMutableDictionary *)clients
+ (BOOL)listenTo:(unsigned short)listenPort
+ (void)closeAcceptSocket
+ (void)sendDataToClient:(NSFileHandle *)fileHandle data:(NSData *)data
NetworkUtilities
+ (NSString *)getAddressString:(struct sockaddr_in)addr
+ (struct sockaddr_in)getStructSockaddr:(NSString *)str
+ (struct sockaddr_in)getStructSockaddr:(NSString *)ip withPort:(short)port inHostByteOrdering:(BOOL)hostByteOrdering
+ (NSString *)deviceIPAddress
VPModeratorDetail
+ (id) detailItem
+ (UIView *)mainSlidesContainer
+ (UILabel *)timerLabel
+ (UILabel *)presenterLabel
+ (VPSlidesViewController *)slides
+ (NSInteger)numberOfScreen
+ (NSString *)debugString
- (UIPopoverController *)masterPopoverController
- (VPModeratorSplitViewController) *splitVC
- (void) configureView
- (void) appendLog: (NSString *) msg
- (void) screenDidChange: (NSNotification *) notification
- (void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
- (void) touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
- (void) viewDidLoad
- (void) displayAlert:(NSString *) str
- (void) listFilesFromDocumentsFolder:(NSString*) indexFileName
- (void) prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sende
-(void)updateTimer

Fig.10 UML Class Diagram of Moderator

3.1.2 UML Diagram for Moderator (Continue)


<<interface>>
VPModeratorPresenterHandler
- (BOOL)registerRequest
- (BOOL)handleUnregister: (char)presenterID
- (BOOL)handleControlRequest: (char)presenterID
- (short)handleSlide: (char)presenterID slide: (UIImage *)image
- (char)handleControlSignal: (char)presenterID signal: (char)signal
- (BOOL)handlePathDrawing: (char)presenterID point: (CGPoint)point status:(char)status
- (BOOL)handlePathDrawingSetting: (UIColor *)color size:(short)size
VPModeratorScreenSetting
- (NSArray*) cellLabels
- (NSArray*) sectionLabels
- (UITableView *)mainTableView
- (bool) isConnectingExternal
- (void) screenDidChange: (NSNotification *) notification
- (void) viewDidLoad
- (void) viewWillAppear:(BOOL)animated
- (void) loadContent
- (void) externalSwitchChanged:(id)sender
- (void) viewDidUnload
VPModeratorPresenterList
- (NSArray*) sectionLabels
- (UITableView *)mainTableView
- (void)viewDidLoad
- (void)viewWillAppear:(BOOL)animated
- (void)loadContent
- (void)viewDidUnload
VPModeratorMaster
+ (UINavigationItem *)leftBarNavi
+ (UITabBar *)leftBarTabBar
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
- (void)viewDidLoad
VPModeratorPendingRequests
- (NSArray*) sectionLabels
- (UITableView *)mainTableView
- (void)viewDidLoad
- (void)viewWillAppear:(BOOL)animated
- (void)loadContent
- (void)viewDidUnload
VPModeratorPresentControl
- (NSArray*) sectionLabels
- (UITableView *)mainTableView
- (void)viewDidLoad
- (void)viewWillAppear:(BOOL)animated
- (void)loadContent
- (void)viewDidUnload
void (^)(UIView *)
Block with one parameter type (UIView *)
void (^)(UIView *, CGPoint)
Block with two parameters type (UIView *) and (CGPoint)
VPSlides
- (NSMutableSet *)subviews
- (UIWindow *)externalWindow
- (NSMutableDictionary *)subviewsKeyTagPair
- (CGFloat)externalViewZoomRatio
+ (UIColor *)backgroungColor
+ (UIView *)deviceView
+ (UIView *)externalView
- (void)addSubview: (UIView *)inputView
- (void)addSubview: (UIView *)inputView withKey: (NSString *)key+ (void)enableExternalView: (UIScreen *)externalScreen
+ (void)disableExternalView
+ (void)addSubview: (UIView *)inputView withKey: (NSString *)key withTag: (NSInteger)tag
+ (void)perform: (void (^) (UIView *))action onKey: (NSString *)key
+ (void)perform: (void (^) (UIView *))action onTag: (NSInteger)tag
+ (void)perform: (void (^) (UIView *, CGPoint))action withPoint: (CGPoint)point onKey: (NSString *)key
+ (void)perform: (void (^) (UIView *, CGPoint))action withPoint: (CGPoint)point onTag: (NSInteger)tag
VPCanvas
- (int)state
- (CGPoint)lastPoint
+ (float)thickness
+ (UIColor *)color
+ (VPCanas *)initWithFrame: (CGRect)frame
+ (void)penDown (CGPoint)point
+ (void)penMove: (CGPoint)point
+ (void)penUp: (CGPoint)point

3.1.3 UML Diagram for Presenter

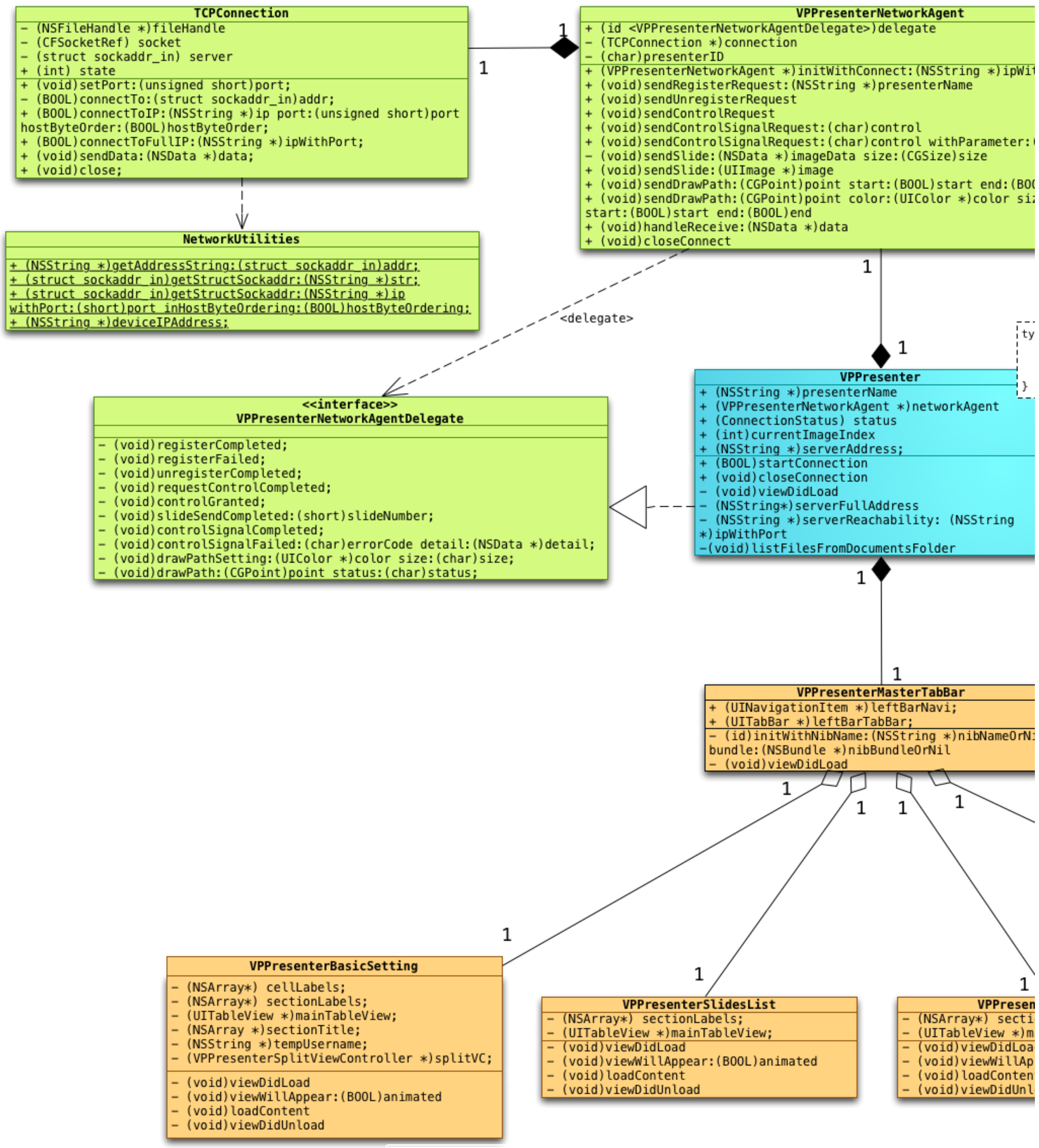
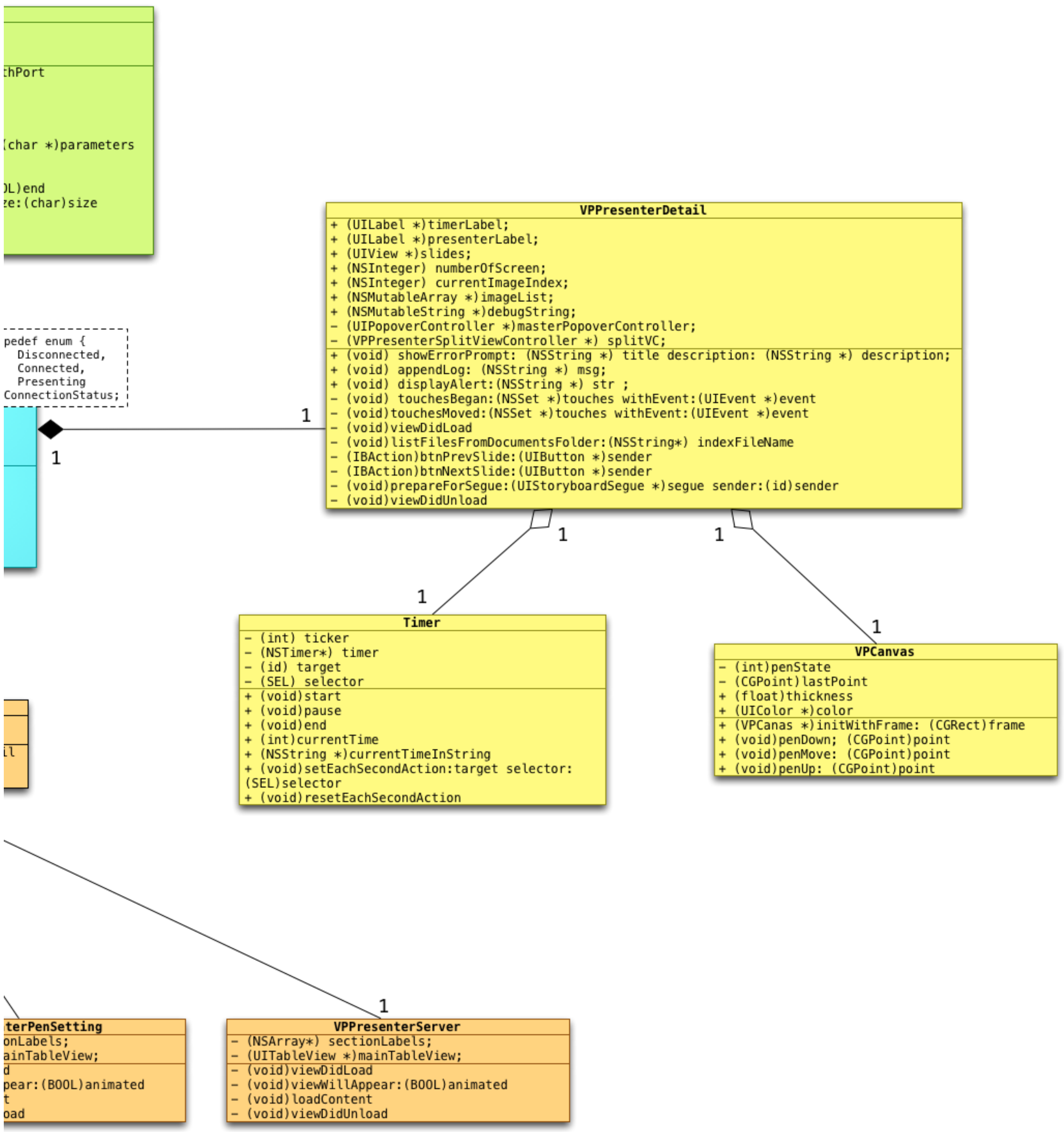

TCPConnection
- (NSFileHandle *)fileHandle
- (CFSocketRef) socket
- (struct sockaddr_in) server
+ (int) state
+ (void)setPort:(unsigned short)port;
- (BOOL)connectTo:(struct sockaddr_in)addr;
+ (BOOL)connectToIP:(NSString *)ip port:(unsigned short)port hostByteOrder:(BOOL)hostByteOrder;
+ (BOOL)connectToFullIP:(NSString *)ipWithPort;
+ (void)sendData:(NSData *)data;
+ (void)close;
NetworkUtilities
+ (NSString *)getAddressString:(struct sockaddr_in)addr;
+ (struct sockaddr_in)getStructSockaddr:(NSString *)str;
+ (struct sockaddr_in)getStructSockaddr:(NSString *)ip withPort:(short)port inHostByteOrdering:(BOOL)hostByteOrdering;
+ (NSString *)deviceIPAddress;
1
1
VPPresenterNetworkAgent
+ (id <VPPresenterNetworkAgentDelegate>)delegate
- (TCPConnection *)connection
- (char)presenterID
+ (VPPresenterNetworkAgent *)initWithConnect:(NSString *)ipWi
+ (void)sendRegisterRequest:(NSString *)presenterName
+ (void)sendUnregisterRequest
+ (void)sendControlRequest
+ (void)sendControlSignalRequest:(char)control
+ (void)sendControlSignalRequest:(char)control withParameter:(
- (void)sendSlide:(NSData *)imageData size:(CGSize)size
+ (void)sendSlide:(UIImage *)image
+ (void)sendDrawPath:(CGPoint)point start:(BOOL)start end:(BO
+ (void)sendDrawPath:(CGPoint)point color:(UIColor *)color si
start:(BOOL)start end:(BOOL)end
+ (void)handleReceive:(NSData *)data
+ (void)closeConnect
<delegate>
1
1
ty
}
<<interface>>
VPPresenterNetworkAgentDelegate
- (void)registerCompleted;
- (void)registerFailed;
- (void)unregisterCompleted;
- (void)requestControlCompleted;
- (void)controlGranted;
- (void)slideSendCompleted:(short)slideNumber;
- (void)controlSignalCompleted;
- (void)controlSignalFailed:(char)errorCode detail:(NSData *)detail;
- (void)drawPathSetting:(UIColor *)color size:(char)size;
- (void)drawPath:(CGPoint)point status:(char)status;
VPPresenter
+ (NSString *)presenterName
+ (VPPresenterNetworkAgent *)networkAgent
+ (ConnectionStatus) status
+ (int)currentImageIndex
+ (NSString *)serverAddress;
+ (BOOL)startConnection
+ (void)closeConnection
- (void)viewDidLoad
- (NSString*)serverFullAddress
- (NSString *)serverReachability: (NSString *)ipWithPort
-(void)listFilesFromDocumentsFolder
1
1
VPPresenterMasterTabBar
+ (UINavigationItem *)leftBarNavi;
+ (UITabBar *)leftBarTabBar;
- (id)initWithNibName:(NSString *)nibNameOrNi
bundle:(NSBundle *)nibBundleOrNil
- (void)viewDidLoad
1
1
1
1
1
1
1
VPPresenterBasicSetting
- (NSArray*) cellLabels;
- (NSArray*) sectionLabels;
- (UITableView *)mainTableView;
- (NSArray *)sectionTitle;
- (NSString *)tempUsername;
- (VPPresenterSplitViewController *)splitVC;
- (void)viewDidLoad
- (void)viewWillAppear:(BOOL)animated
- (void)loadContent
- (void)viewDidUnload
VPPresenterSlidesList
- (NSArray*) sectionLabels;
- (UITableView *)mainTableView;
- (void)viewDidLoad
- (void)viewWillAppear:(BOOL)animated
- (void)loadContent
- (void)viewDidUnload
VPPresen
- (NSArray*) secti
- (UITableView *)m
- (void)viewDidLoa
- (void)viewWillAp
- (void)loadConten
- (void)viewDidUnl

Fig.11 UML Class Diagram of Presenter

3.1.3 UML Diagram for Presenter (Continue)



```
VPPresenterDetail
+ (UILabel *)timerLabel;
+ (UILabel *)presenterLabel;
+ (UIView *)slides;
+ (NSInteger) numberOfScreen;
+ (NSInteger) currentImageIndex;
+ (NSMutableArray *)imageList;
+ (NSMutableString *)debugString;
- (UIPopoverController *)masterPopoverController;
- (VPPresenterSplitViewController *) splitVC;
+ (void) showErrorPrompt: (NSString *) title description: (NSString *) description;
+ (void) appendLog: (NSString *) msg;
+ (void) displayAlert:(NSString *) str ;
- (void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
- (void)viewDidLoad
- (void)listFilesFromDocumentsFolder:(NSString*) indexFileName
- (IBAction)btnPrevSlide:(UIButton *)sender
- (IBAction)btnNextSlide:(UIButton *)sender
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
- (void)viewDidUnload
```

```
Timer
- (int) ticker
- (NSTimer*) timer
- (id) target
- (SEL) selector
+ (void)start
+ (void)pause
+ (void)end
+ (int)currentTime
+ (NSString *)currentTimeInString
+ (void)setEachSecondAction:target selector:
(SEL)selector
+ (void)resetEachSecondAction
```

```
VPCanvas
- (int)penState
- (CGPoint)lastPoint
+ (float)thickness
+ (UIColor *)color
+ (VPCanas *)initWithFrame: (CGRect)frame
+ (void)penDown; (CGPoint)point
+ (void)penMove: (CGPoint)point
+ (void)penUp: (CGPoint)point
```

```
VPPresenterServer
- (NSArray*) sectionLabels;
- (UITableView *)mainTableView;
- (void)viewDidLoad
- (void)viewWillAppear:(BOOL)animated
- (void)loadContent
- (void)viewDidUnload
```

```
thPort
(char *)parameters
OL)end
ze:(char)size
```

```
pedef enum {
   Disconnected,
   Connected,
   Presenting
ConnectionStatus;
```

```
terPenSetting
onLabels;
ainTableView;
d
pear:(BOOL)animated
t
oad
```

3.2 External Screen

The device for server would connect to external screen or projector, and show the slides on external screen. While the projector showing slide, screen on device should also show the slide so the user can focus on the device screen and audiences, without frequently looking at projection. In addition to slide, there should be presentation control and configuration detail shown only in device but not projecting to screen. Therefore, we have to make slide synchronizing in device and screen, and controls only showing in device.

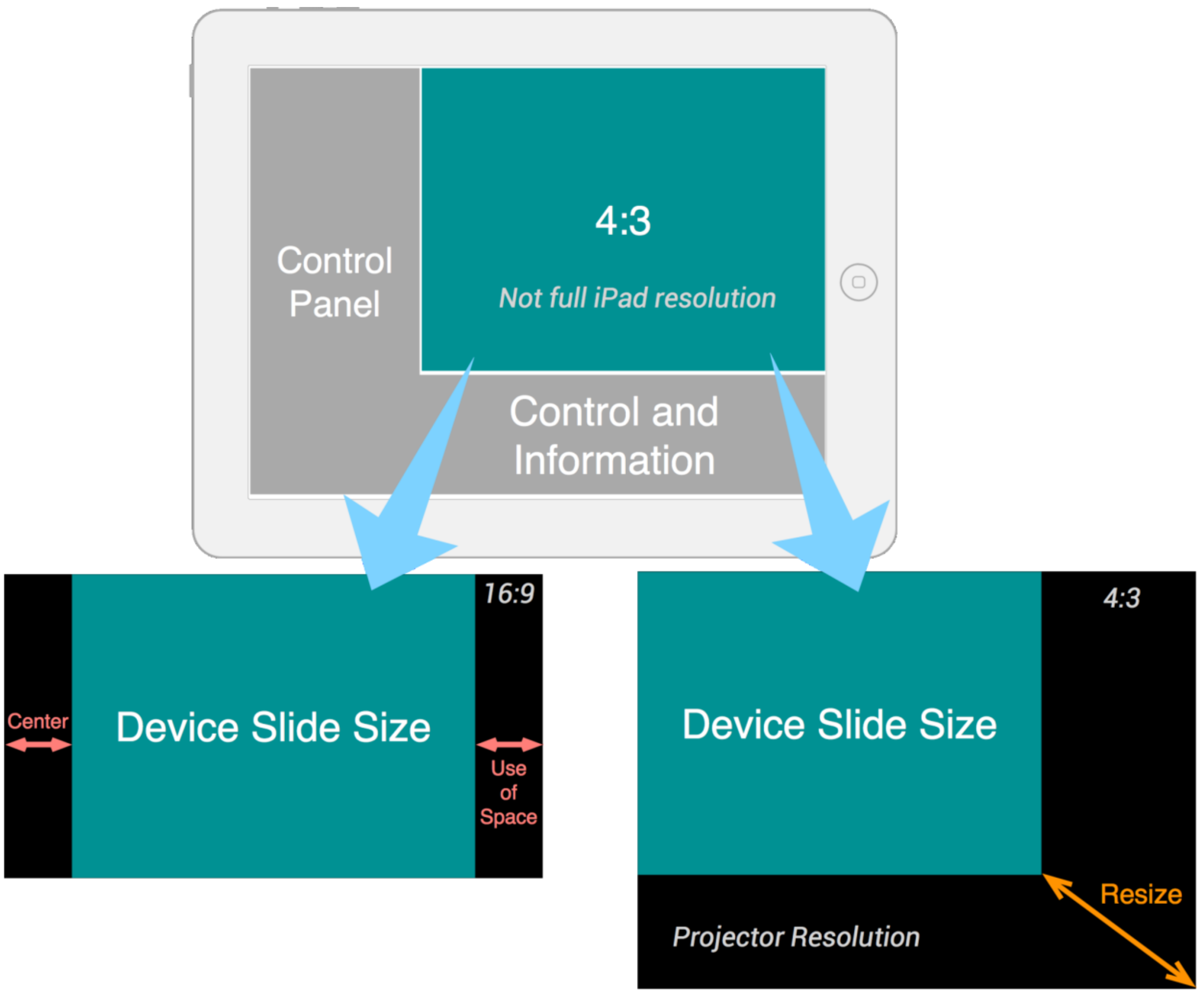


Fig.12 Scaling and Best-fitting for External Screen

Synchronizing device screen and external screen, we can copy the view after each operation, or maintaining two views and perform action on each view when message received. The former method use more computational power for copying but use less space for storing two views, while the latter one use more space for maintaining two views but do not need computational heavy deep copy action. Finally, we decide to use method mixing two: when adding subviews into the view, we copy the view into two copy; then for any further action, it perform on both original view and copied view. This balance computational and space efficiency.

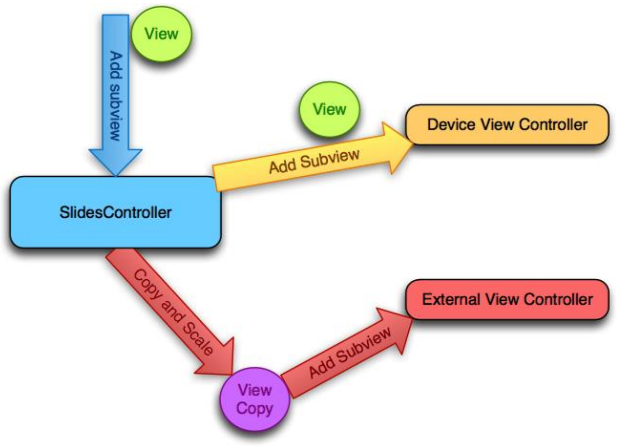


Fig.13 (a) Copying View when adding to external

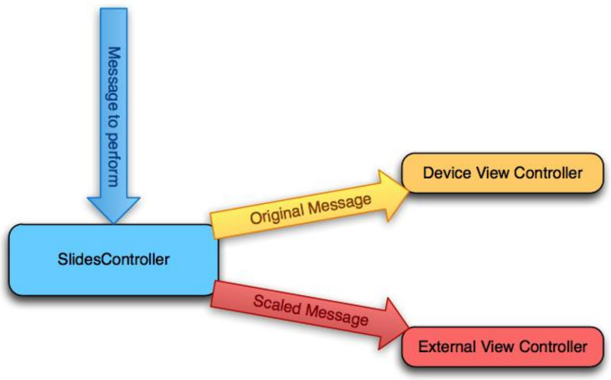


Fig.13 (a) Copying View when adding to external

Another concern is the resolution and resizing. Resolution of iPad, iPad 2 and iPad Mini is 1024×768 while resolution of the New iPad is 2048×1536. However, typical resolution of projector or external screen is 1280×1024 and 1920×1080 (1080p). We separate the problem into two cases: same aspect resolution and different aspect ratio. If the aspect ratio is the same, simply resize the image by multiplying size by a constant ratio. However, if the aspect ratio is different, we need to handle re-calculate the size in order to maintain the aspect ratio. Moreover, center the image and handling of extra space is another concern when making best-fit resizing.

3.3 Drawing Pad / Canvas

Writing and drawing on slide is a famous feature of a presentation software. During the design, we have to think about how we serialize the shape, and represent it with minimal size, serving for data transmission through network and saving and storage. We focus the shape to regular shapes and arbitrary path, together with text. Those features in drawing pad cover most needs for users.

Representation of regular shape is simple. Using rectangle as example, we can simple represent it using a point, upper right corner, and a size, height and width of rectangle, with stoke color and fill color. Circle could be represented as center and radius; ellipse could be represented as center, major axis and minor axis. Those regular shape could be represented using geometric properties.

Shape	Storing Data	Special Cases
Line	Start Point, End Point	
Rectangle	Upper right coordinate, Height, Width	Square
Ellipse	Center, Semi-major axis, Semi-minor axis	Circle
Star	Center, two radii: Outer radius and Inner radius	N/A
Regular Hexagon	Center, Radius	

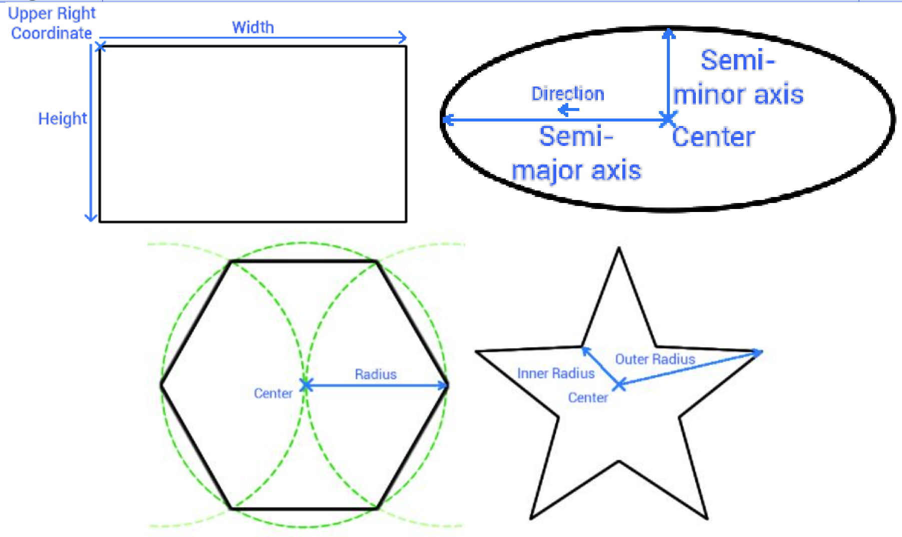


Fig.14 Representation of some regular shapes

Regarding arbitrary path, we can consider it as collection of point. As the points in path should not duplicate, or we can represent it once even the path is overlapping with its own, and also the sequence of point is not a matter when we are storing the whole path, we can use set for storing the path. However, when we consider synchronizing between devices during presentation, we could not use the set serialization. Serializing as set, we have to get the whole path and the set would represent whole path without drawing sequence. The whole path will appear on external screen without drawing animation if we try to use this method, which is not intuitive and difficult for presenter to keep referencing while drawing. Therefore, we decide to serialize it with single point each during presentation, being discussed in the next section.

In addition, we can also allow user to change the fill color, stoke color, thickness of stoke and transparency of shape and path. Those can be serialize as several variables and bytes. Moreover, we can apply transformation to those shapes in order to satisfy users' needs. Transformation including translation, scaling, rotation, shearing etc. Those transformation should also be able to serialize for storage and transmission. We can store serialize those transformation with following value:

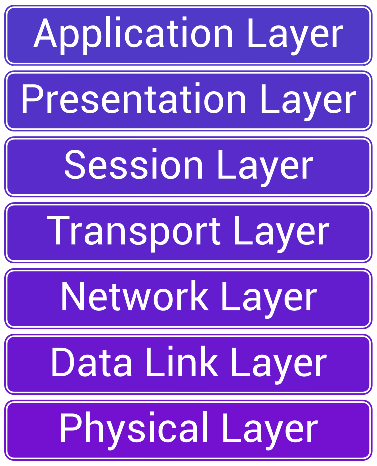
Transformation	Value	Assumption
Translation	x-Coordinate, y-Coordinate	
Scaling	x-scale, y-scale	Centered by reference point
Rotation	Angle	Centered by reference point, anticlockwise
Shearing	x-Scale, y-Scale	

Using the serialization, we can further allow user to undo and redo operation. Serializing it as storage form, we simply push the shape or operation into the stack. By the first-in-first-out property of stack, we simply pop a shape out from stack can make undo. By storing the popped shape into another stack, we can also implement redo, as undo of an undo operation. However, a technical issue is how we could erase a rendered content on canvas.

An additional features is showing pointer on screen. When user touch a point in slide, the screen can show the point touching by a circle. Tracking the point of touching, the pointer move so presenter can emphasize the point or area talking about. However, this might involve erasing a point drawn on screen, and redraw another point. The erasing step might be difficult and need further investigation of APIs together with rendering behavior.

3.4 Inter-Device: Connection and Message Protocol

3.4.1 Overview



As a client-server system, we have to consider about connection between clients and server. Regarding networking and Open Systems Interconnection (OSI) model, there are 7 abstract layers in communications system or network.

Despite it is an abstract concept, it is still worth to make decision about message transmission based on few layers which is more in practical, including data link layer, transport layer and application layer.

Regarding the data link layer, there are much implementation including Wi-Fi and Bluetooth, which are both native supported by iOS devices. We have considered both data link interfaces to implement the inter-device communication, and we decided to use Wi-Fi as our data link layer interface. The data transfer rate of Wi-Fi is generally faster than Bluetooth, as most of the mobile devices are only installed with Bluetooth 2.1+EDR adapters. This Bluetooth standard can give out 3 Mbps of maximum data transfer rate. Comparing to the Wi-Fi standard (IEEE 802.11a/b/g), which have 54 Mbps of maximum data transfer rate, Wi-Fi should have a better performance. In addition, Wi-Fi also supports longer effective distance.

There are two main implementation of transport layer, user datagram protocol (UDP) and transmission control protocol (TCP). In short, UDP is a best effort protocol, lightweight but no guarantee on correctness of data, neither order of data. In contrast, TCP guarantee correctness, order of data, with congestion control and flow control for large amount of data. However, the overhead of TCP to transmit small amount of data is large.

In current implementation, we use TCP instead of UDP because TCP could ensure that received data is correct. This is important when user send slides from own device to server. In addition, this can also avoid a possibility of error and bug during development process. In the applications, there are some message which is tiny and also, directly user driven. Aiming at performance, we may switch to UDP for those message transferal in future.

Application layer protocol is important for message being understand by both devices. As it is self-defined, the information would be more and we will discuss it in the next session.

3.4.2 Message Protocol

Sending message on network is a costly operation, therefore we have to minimize the size of message by considering the protocol in byte.

There are several type of message being sent. During the design of protocol, we should make the type be easily differentiated and distributed to other object for processing. Therefore, we first list out how many type of message we are going to pass between clients and server:

- Register
- Unregister
- Sending slides (as image)
- Control Signal
- Drawing

Message Header and General Specification

We decided to make an application layer header for easy distinguishing command and presenter. The following is the application layer header format

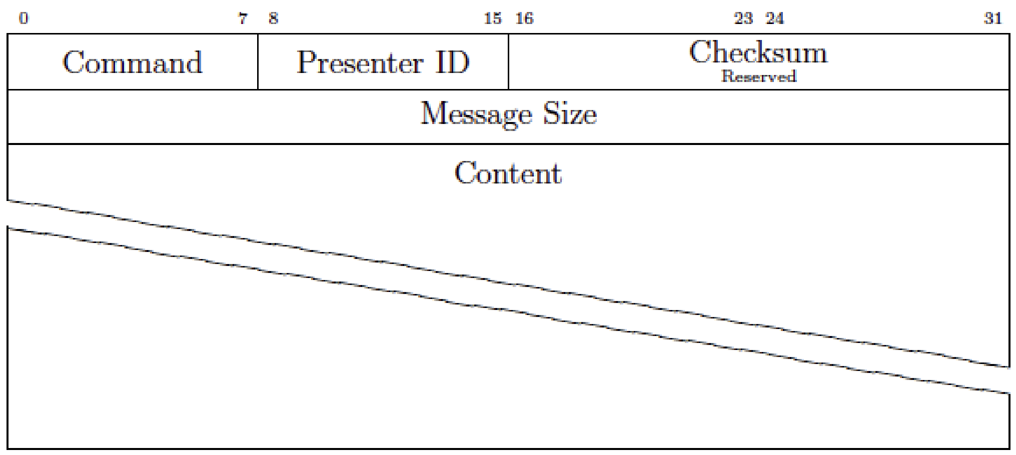


Fig.15 Common Message Structure with Header

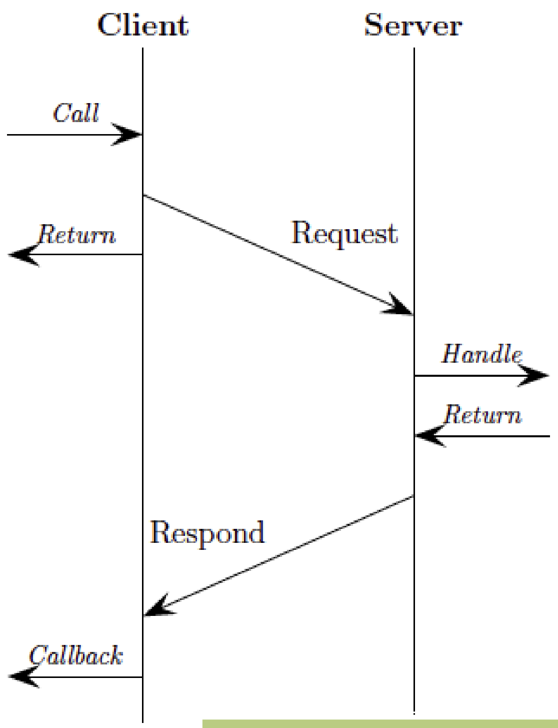


Fig.16 Common Message Sending Chart

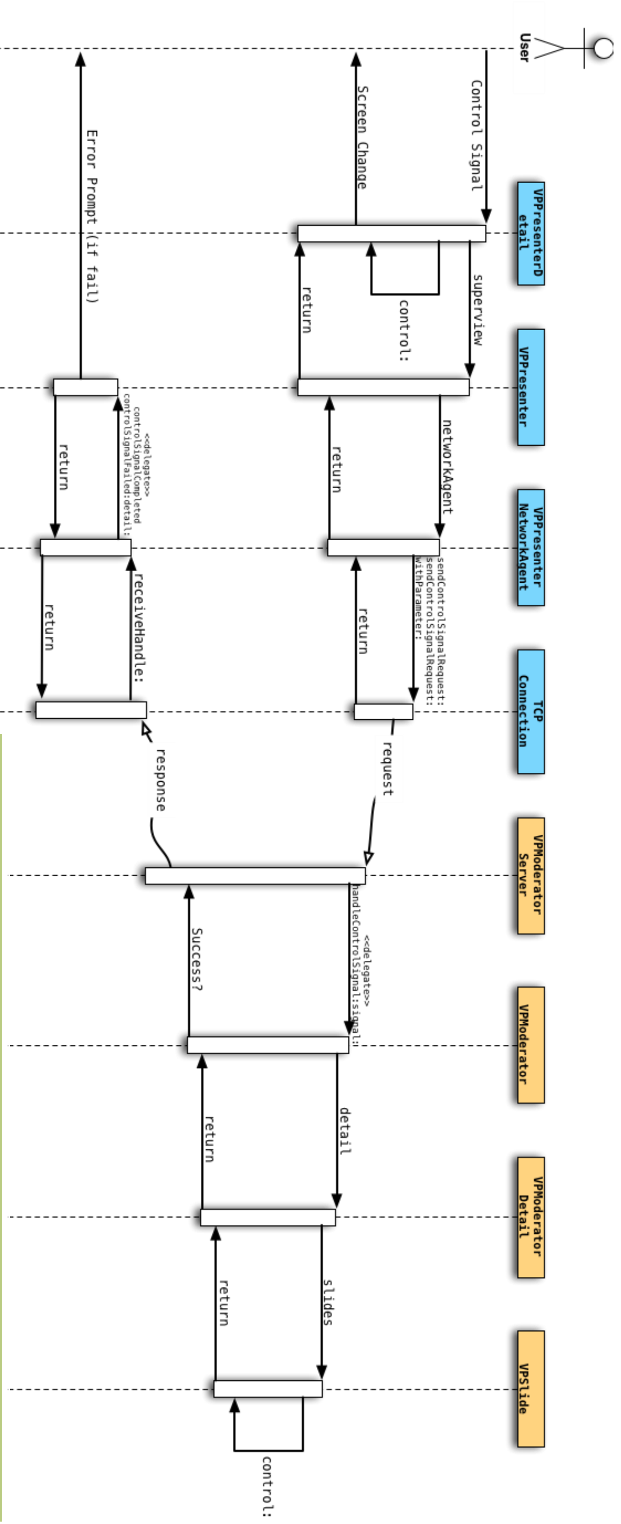


Fig.17 UML Sequence Diagram for Handling Control Signal

The header is total of 8-byte long, including command, presenter ID, checksum and message size.

Command is used to distinguish which type of message is. Adding this field ease the difficulty in partitioning message. Each type of message is associated with a single-byte command value, as shown in the following table:

Type	Action	Source	Command *(in hexadecimal)*
Register	Request	Client	`0x01`
	Success Respond	Source	`0x02`
	Failure Respond	Server	`0x03`
Unregister	Request	Client	`0x04`
	Respond	Server	`0x05`
Control Permission	Request	Client	`0x06`
	Respond to Request	Server	`0x07`
	Grant Permission	Server	`0x08`
	Withdraw Permission	Server	`0x09`
Slides Exchange	Slide data	Client	`0x0A`
	Acknowledge	Server	`0x0B`
Control Signal	Request	Client	`0x0C`
	Success Respond	Server	`0x0D`
	Failure Respond	Server	`0x0E`
Drawing Path	Request / Respond	Client / Server	`0x10`

Regarding the presenter ID, it is a single byte unique value for each presenter. The range of presenter ID is from 1 to 254 inclusively, and 0 is reserved for server, 255 is reserved for unregistered presenter. In usual, presenter ID in a message would not be 0 as the value in message means the client communicate with server. The only case for presenter ID equals 0 is moderator is broadcasting message to all registered presenter.

Checksum is a two-byte length value for ensuring no error during transmission. As we are using TCP in current version, and TCP guarantee correctness of received data, the checksum field is not used. The reserved checksum is going to use if we are optimizing by using UDP for small packet. Before checksum calculation, the field in message need to be zeroed.

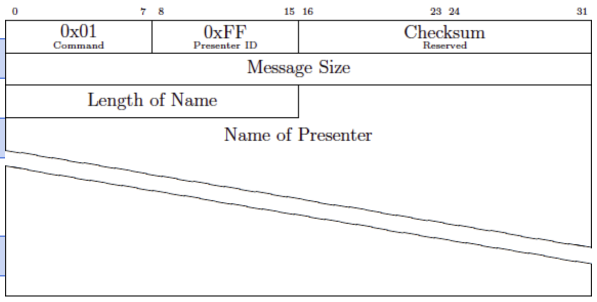
Message length is a four-byte integer for the length of whole message, including the 8-byte header. It is used for buffering and ensure arrival of whole message before processing.

Following is specification of content in each type of message.

Command	`(Command value)`	
Direction	(Source) → (Destination)	
Argument Count	(Argument count)	
Argument (number)	(Length of argument)	(Content of argument)
…	…	…

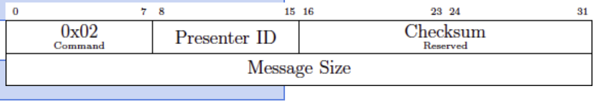
Register Request

Command	`0x01`	
Direction	Client → Server	
Argument Count	2	
Argument 1	4 Bytes	Number for Name
Argument 2	*Vary*	Name



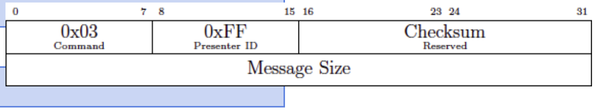
Register Success Respond

Command	`0x02`
Direction	Server → Client
Argument Count	0



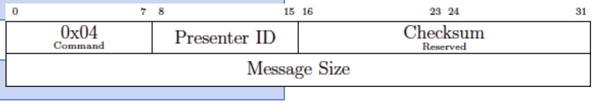
Register Failure Respond

Command	`0x03`
Direction	Server → Client
Argument Count	0



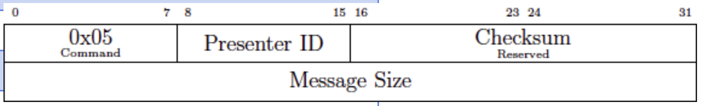
Unregister Request

Command	`0x04`
Direction	Client → Server
Argument Count	0



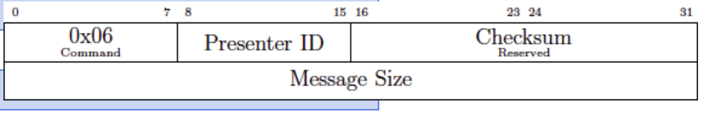
Unregister Respond

Command	0x05
Direction	Server → Client
Destination	Client
Argument Count	0



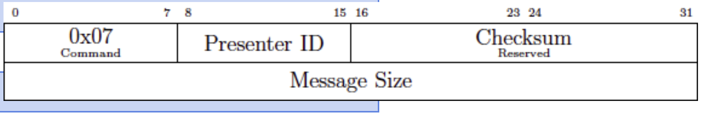
Control Permission Request

Command	0x06
Direction	Client → Server
Argument Count	0



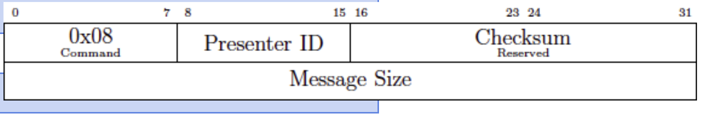
Control Permission Request Respond

Command	0x07
Direction	Server → Client
Argument Count	0



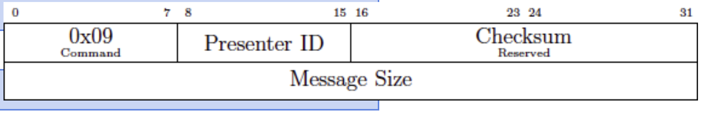
Grant Control Permission

Command	0x08
Direction	Server → Client
Argument Count	0



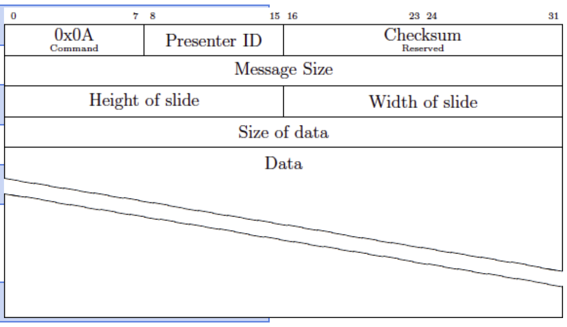
Withdraw Control Permission

Command	0x09
Direction	Server → Client
Argument Count	0



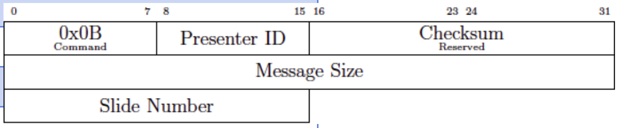
Slide Data

Command	0x0A	
Direction	Client → Server	
Argument Count	4	
Argument 1	2 Bytes	Height of Slide
Argument 2	2 Bytes	Width of Slide
Argument 3	4 Bytes	Number of byte of Slide Data
Argument 4	*Vary*	Slide data



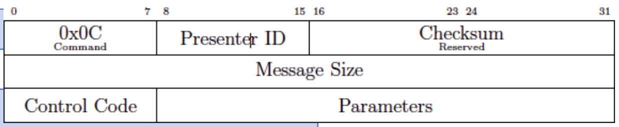
Slide Data Acknowledgement

Command	0x0B	
Direction	Server → Client	
Argument Count	1	
Argument 1	2 Bytes	Slide number



Control Signal Request

Command	0x0C	
Direction	Client → Server	
Argument Count	2	
Argument 1	1 Byte	Control Signal Type
Argument 2	3 Bytes	Control Signal Parameters

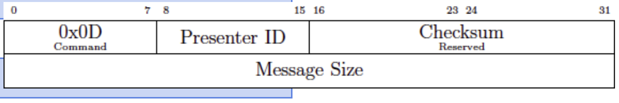


We further define control signal type by the following:

Type	Description	Parameters
0x01	Next Slide	N/A
0x02	Previous Slide	N/A
0x03	Jump to a slide	First byte: Slide number
0xF0	Black the screen	First byte if non-zero → White screen
0xFD	Handover control to viewer	
0xFE	Withdraw control from viewer	First byte: Countdown value
0xFF	Return control	N/A

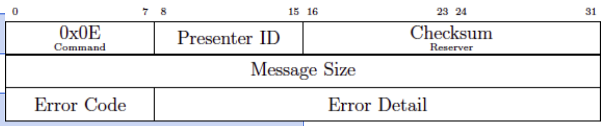
Control Signal Success Respond

Command	0x0D
Direction	Server → Client
Argument Count	0



Control Signal Failure Respond

Command	0x0E	
Direction	Client → Server	
Argument Count	2	
Argument 1	1 Byte	Error Code (Reserved)
Argument 2	3 Bytes	Error Detail (Reserved)



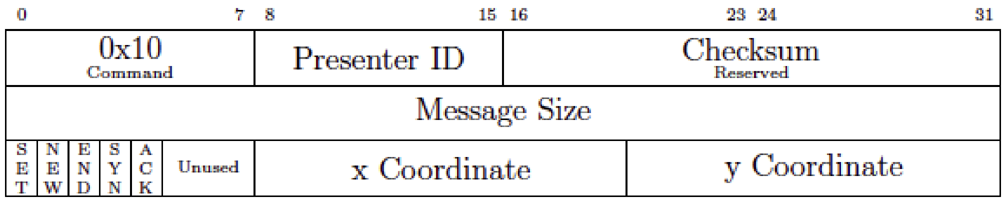
Drawing Path

The message for path drawing is a bit complicated. First, there is a single-byte bitmap for flag of message, defined as following:

0	SET	If message contain content used for setting path property, i.e. argument 4 – 8
1	NEW	If this point represent start point of new path
2	END	If this point represent end point of current path
3	SYN	If message sender ask for synchronization of this point
4	ACK	If message sender acknowledge of previously synchronization request
5-7	Unused	Reserved for further use

If SET flag is clear (0), the message is 12-byte long, specified as following:

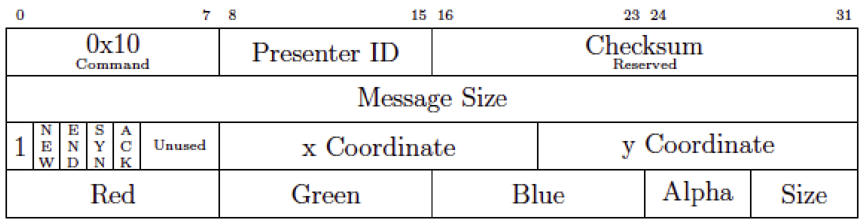
Command	0x10	
Direction	Client → Server / Server → Client	
Argument Count	3	
Argument 1	1 Byte	Flag bitmap
Argument 2	12 Bits	x-Coordinate for Point of Path, range: [0, 4095]
Argument 3	12 Bits	y-Coordinate for Point of Path, range: [0, 4095]



The x-coordinate and y-coordinate is limited in range [0, 4095]. As resolution of current machine or projector is at most 2048×1536 for iPad with retina display, and resolution of 2160p ultra HD is 3840×2160, and should be enough for next few years.

If the SET flag is set (1), there are four bytes of arguments appended at the end of message:

Command	`0x10`	
Direction	Client → Server / Server → Client	
Argument Count	8	
Argument 1 – 3	4 Bytes	*(Same as above)*
Argument 4	1 Byte	Red component of Color, range: [0, 255]
Argument 5	1 Byte	Green component of Color, range: [0, 255]
Argument 6	1 Byte	Blue component of Color, range: [0, 255]
Argument 7	4 Bits	Alpha of Color, range: [0, 127]
Argument 8	4 Bits	Size of Path (or Point3), range: [0, 127]



Regarding the SYN and ACK flag, we have implemented a SSH-liked echo mechanism. When a client send a message of path drawing to sever with SYN flag set, it do not immediate update, but wait for server process and update after receiving server echo message, of which the ACK is set.

Using echo mechanism, we can ensure the path drawing request and content is valid before the path appear in presenter's device. In addition, it can guarantee path seen from projector is same as that in presenter's device.

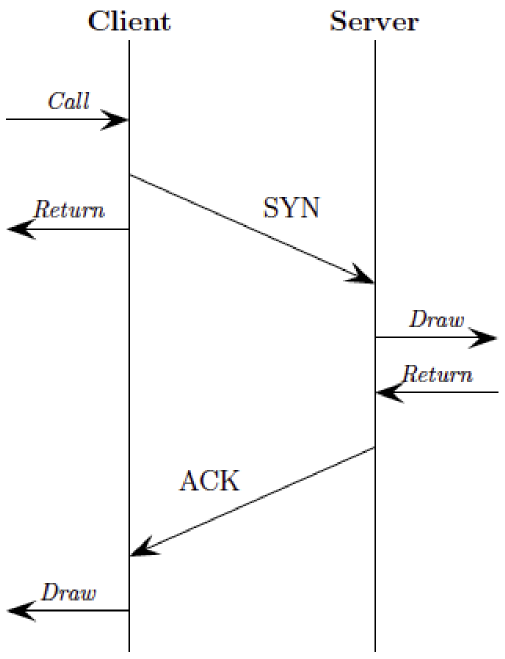


Fig.18 Message Send Chart for Path Drawing

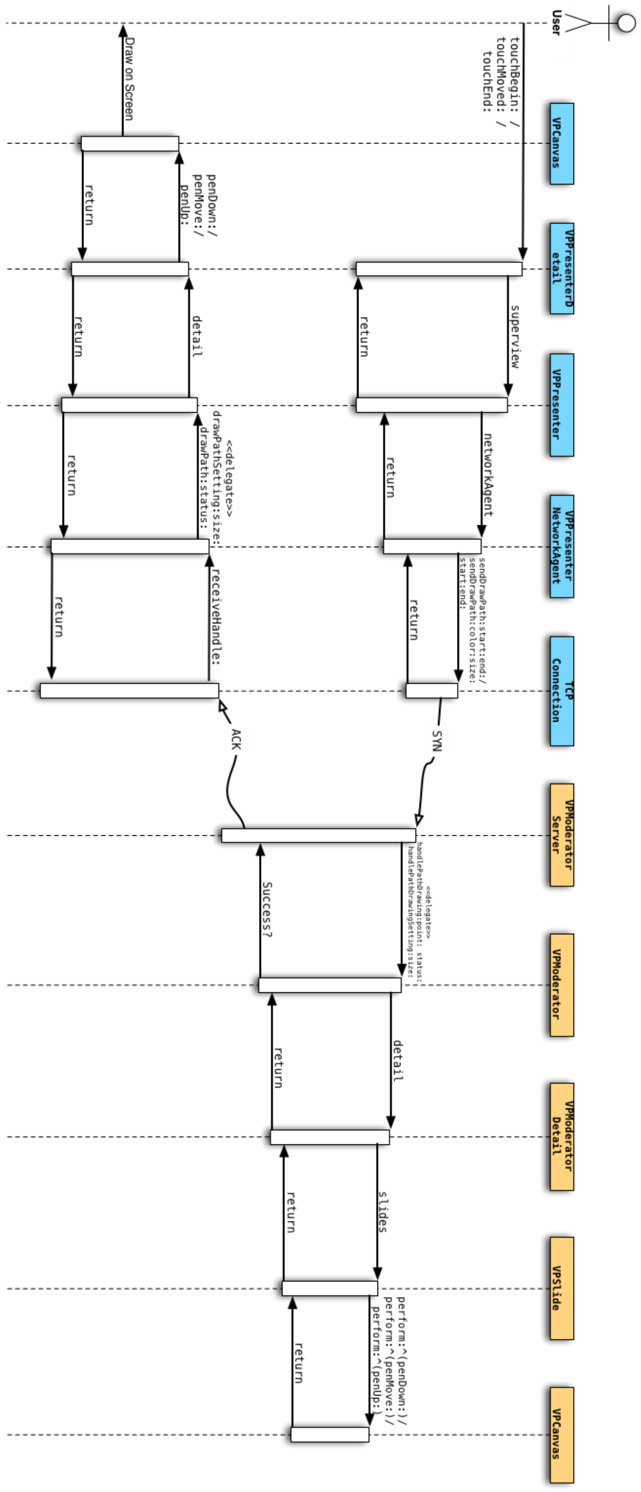


Fig.19 UML Sequence Diagram for Handling Path Drawing

Chapter 4. User Experience

4.1 Initial Approaches

During the earlier phases of the project, there was less design on the user interface. All interface items were displayed on one single main view, as shown in the following:

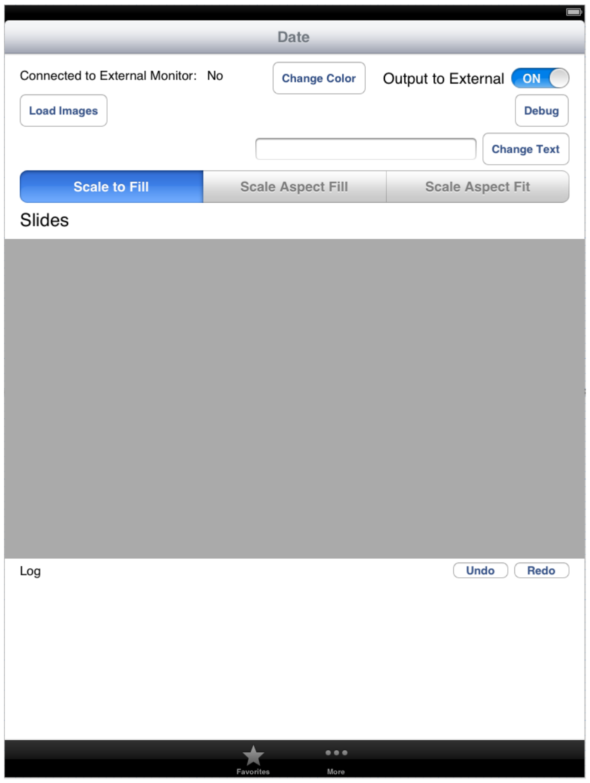


Fig.20 Initial Interface Design

Before long, as the functionality and system status increases, the interface layout become more and more messy and unorganized. So we need to redesign the interface by reordering all the items.

4.2 Possible Solution

We have considered making use of the tab bar at the bottom of the screen, to organize the objects on the screen. The tab bar consists of some tab bar items, each of them was responsible for a category of functions, for example the status and switch for external monitors can group into the same group named "Screen".

These items would only be visible when the respective tab bar button was touched, then a pop up dialog box would appear and overlay on the screen. This dialog box would fade out if the user tab outside the window of the screen, as illustrated in the following figure:

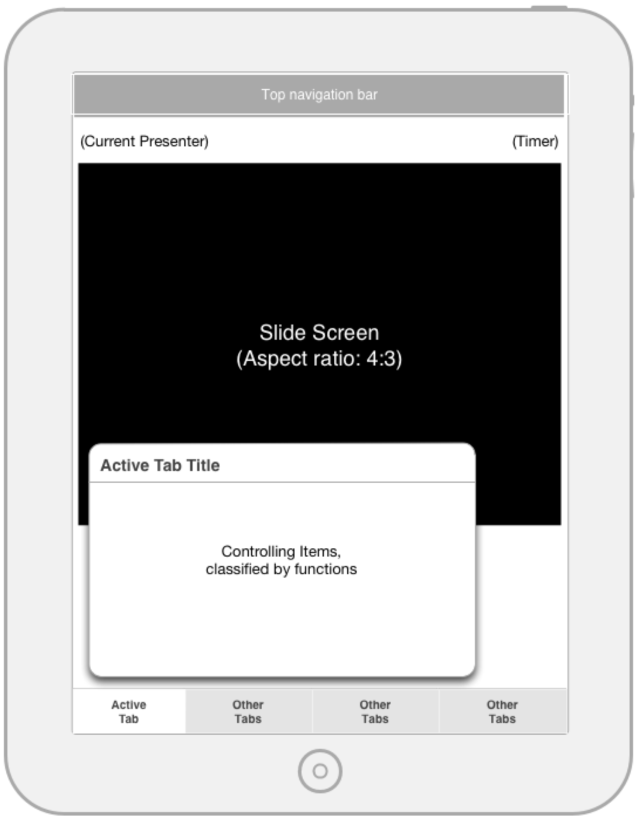


Fig.21 Interface design with Portrait

This approach can effectively solve the problem of unorganized controlling items and it improved the effectiveness of limited screen spaces. But soon we discovered a major problem of this interface design. The problem is that the dialog boxes of the active tabs would partially cover the slide screen. For the moderators, although it would not affect the view output to the external screen, it would definitely affect the effectiveness of the flow controlling ability. For example, if the current presenter displayed some inappropriate contents on the slide screen but the moderator was busy in handling some requests displaying on the dialog box of an active tab, the dialog box may covered the banned contents and the moderator was not noticed. Moreover, this is a portrait-based interface, which is not able to use when the orientation of the device is landscape. Due to the above issues, we need to redesign the interface.

4.3 Final Decisions

Finally, we decided to use the “split view” framework design. This kind of interface is officially supported by iOS and only available on iPad. This kind of user interface design is currently using by many iPad applications. Actually, we are inspired by the iPad version of Dropbox client. Split View (or Master-Detail View) is suitable for applying applications relating to file management. In the Dropbox client-side application, the master view can show file and folder lists of clients’ Dropbox, and also the display usage of Dropbox. On the other hand, detail view on the right display the content of the chosen file from the master view. When the iPad is holding on landscape orientation, performing actions to control the Dropbox files on the master view, the content displaying on the detail view are not neither affected nor covered.

This feature of split view is suitable for our application as the slides screen on the detail view is not favorable to be covered by some other windows or view. The following figure shows the general layout of iOS Split Views.

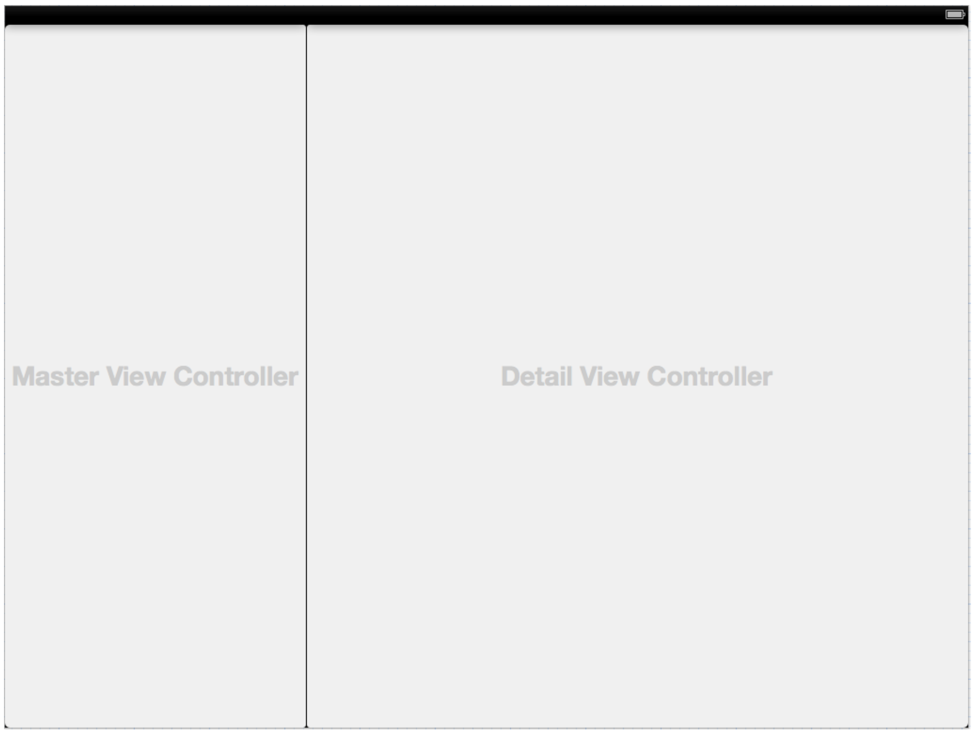
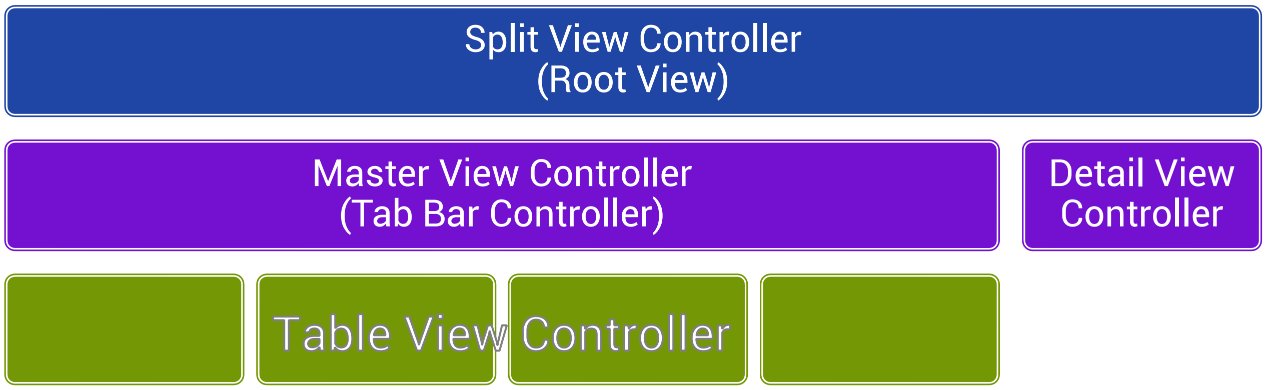


Fig.22 Landscape view of a split view controller

Based on the split view controller framework, we have extended the view hierarchy as shown the following chart:



And the following is the final interface design:

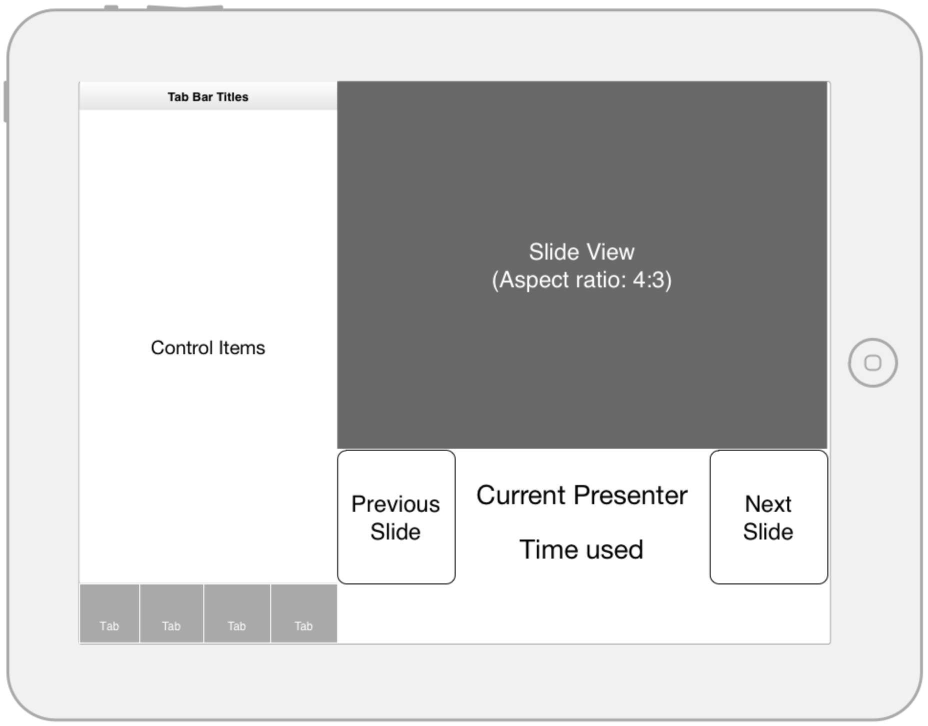


Fig.23 Overview of the user interface of our application

As shown in the figure, the monitor display is divided into two parts. In our design, master views are used to show the controlling options and show the system status. On the other hand, the detail view shows the slide contents and the items related to the presentation flow, such as the current presenter, timer, etc.

The master view controller is containing a tab bar controller, which allows multiple tabs to be displayed on the master view by selecting the respective tab bar items. The content in the master view controller in the moderator version application is different from that in the presenter version. The moderator version consists of the IP address and the listening port of itself, a request list, connected presenter list, and the external screen control, etc. The presenter version consists of the presenting file list, connection status, drawing pen style, etc.

Both versions of the application show the presentation status on the detail screen. For the server version, it will show the current presenter of the presentation (empty if no presenter is presenting), and show the total time used of that presenter. For the presenter version, it will also show the total time used and the connection status with the moderator.

Chapter 5. Implementation Detail

5.1 Overview

In Fall 2012, we have implemented a subset of function and feature. The goal of this semester is to demonstrate the concept of collaborative presentation, as well as implementing some simple function of presentation system.

5.2 Drawing Pad / Canvas

We did not implemented a full-function drawing pad in this stage. We have implemented only arbitrary path drawing with color and stoke size setting. We choose arbitrary path drawing as we have to test on iPad capability and computational power on large amount of data transmission, as well as frequent update of canvas.

The implementation of arbitrary path drawing is intuitive: when user start touch on canvas, it drawing a point on that position, and storing the point in object as private variable. When the user move his / her finger on canvas, it send message to canvas with parameter of the new point. Then the canvas draw a line between the previous point and the new point, and also store the new point into it private field. When user ends the touch, it clear the class variables storing previous point. The pseudo-code is given as following:

```
penDown: point
    join(point, point)
    prevPoint <- point

penMove: point
    join(prevPoint, point)
    prevPoint <- point

penUp: point
    join(prevPoint, point)
    prevPoint <- nil
```

The drawing is done using Core Graphics, library provided by Cocoa framework. The Core Graphics handle the rendering detail of drawing and provide some drawing utilities such as `CGContextAddLineToPoint()`, for drawing line between two points. In addition, the Core Graphics framework also provide utilities for serializing image to common image format such as JPEG and PNG. If have tried to use the utilities to store the drawing as PNG, then make a simple undo-redo function. However, we found saving image to PNG takes lots computational power and make the device freeze for a second. Therefore, we did not add the undo and redo function to current prototype, until we optimize it by storing the path and erasing current content.

5.3 External Display / Projector Connection

When implementing the external display, we have two concerns: detection of external display and drawing on external display.

To detect the external display, we, again, use the `NSNotificationCenter` for detecting broadcast. Registering `handleScreenDidConnectNotification` and `handleScreenDidDisconnectNotification`, we can handle the external screen immediately when it is connected or disconnected.

After the connection of external monitor, we have to manage the content of external monitor. The setup of external monitor involve two classes, `UIScreen` and `UIWindow`. `UIScreen` contain settings of the connected external monitor including resolution and brightness. In addition, `UIScreen` contains available resolution setting of connected monitor. Based on the resolution available, of which preferred resolution is used in prototype, we create an instance of `UIWindow`. Adding root view to the created `UIWindow` object, we are able to add subview on it, showing custom content to external screen.

The next step is to make content to show in external screen, synchronizing with device view. To manage those actions, we have create a new view controller, handling operations regarding external screen called `VPSlidesViewController`. Moreover, we are trying to hiding that there are two view hierarchies in the controller. Thus classes using this controller may only consider it as normal view controller. As discussed in design section302, we are going to make two operations to view: copying and performing.

Copying a view, we found a view may not be able to make deep copy and sometime may need custom copying in order to have a deep copy of view. To handle different class in `VPSlidesViewController` for copying, we have use introspection for determining class of an object, then handle each class differently. We have only handle few classes such as `UIImageView` and `VPCanvas` for copying, which is enough with currently implemented functionalities. But we will add support on more classes later.

To perform an operation on a view, we try to add a method which use block as parameter. A block of code is to abstract the programming logic in the block. The block basically having one parameter, the view, and the block content assume valid view is passed into it and able to perform operation on the block. However, we found having only one parameter is not enough. When the block is containing some coordinate and position related value, the position in device may not be the same as that in view, due to resizing and translating of view when fitting to external monitor. Therefore, we further write a method of block with two input, view and point, and do resizing and translating within controller, making the point is relative to external screen. Thus the position of point is still same as viewing in device.

```
- (void) perform: (void (^) (UIView *)) action onTag: (NSInteger) tag {
    action([self.deviceView viewWithTag: tag]);
    action([self.externalView viewWithTag: tag]);
}

- (void) perform: (void (^) (UIView *, CGPoint)) action onTag: (NSInteger) tag withPoint: (CGPoint) point {
    action([self.deviceView viewWithTag: tag], point);
    action([self.externalView viewWithTag: tag], CGPointMakeScale(point, 1, self.externalViewZoomRatio));
}
```

Remark 1: tag is used for retrieving a view from view hierarchy

Remark 2: `CGPointMakeScale()` map and scale a point with `prevRatio` and `newRatio`. Normally, `prevRatio` is set to 1

5.4 Network Connection

As discussed in design section 3.4.1, we are going to use TCP/IP connection.

As Objective-C is superset of C, and iOS has implemented interface on Berkeley sockets, as known as BSD sockets, we can use the recompile codes written in C and developed based on Linux. However, we tried to avoid some low level referencing and try using Core Framework calls and `NSFileHandle`.

Core Framework of iOS provides family of `CFSocket*` methods for some socket operations. We have used including `CFSocketCreate()`, `CFSocketSetAddress()`, `CFSocketInvalidate()` and `CFRelease()` for replacing system calls such as `socket()`, `bind()`, `connect()`, `listen()`, `close()` etc. In addition, we have wrap the socket file descriptor to `NSFileHandle`, providing more utilities for operations. Therefore we could avoid storing variables in low level, making use of object oriented paradigm and API provided in Cocoa foundation.

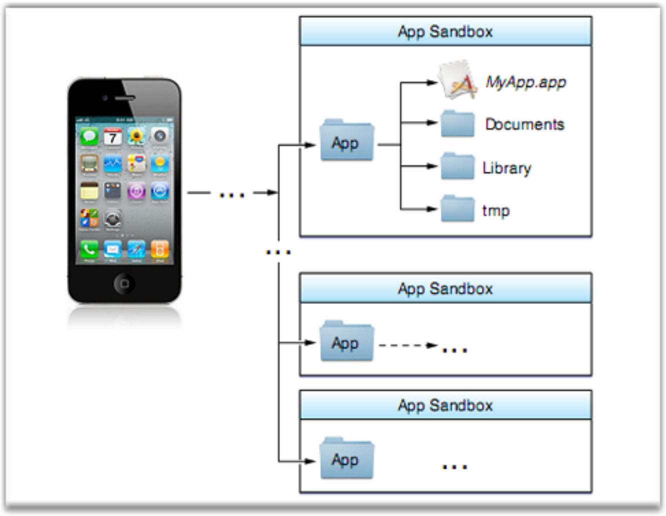
Wrapping the file descriptor to `NSFileHandle`, we can use more API of Coca foundation and avoid complicated codes. In Linux environment with C implementation, we have to use `read()` for reading data from foreign host. As a blocking call, `read()` in socket usage always come with thread programing using POSIX `pthread_*()` and some mutex or semaphore to prevent race condition. Those implementation make code complex and difficult in development. Using `NSFileHandle` together with `NSNotificationCenter`, a global broadcasting class within program. By sending message `waitForDataInBackgroundAndNotify` and adding `NSFileHandleDataAvailableNotification` to notification center, we can wait data in background. Once the run loop found there is data available, it would send selector message to observer and therefore we can wait data in background, and handle data with another method.

We have write a class of `TCPServer` and `TCPConnection`. Both class only making method such as `connectTo:`, `listenTo:` public to call. Classes use `TCPServer` or `TCPConnection` do not need to consider the internal implementation and simply register callback when receiving data.

During testing, we found and problem with waiting data in background with notification center: the handing method is called for each packet received. Therefore we cannot get data in the method if the message size is larger than one packet, and the data we received is at most 1448 bytes. This is a serious problem it is impossible to limited the slide image be 1448 bytes. Therefore we tried to implement a buffer for storing incomplete message. The message handling method will be called only if the message is completed.

5.5 File Read / Write

Comparing to other mobile device OS like Andriod, iOS adopts the “Sandbox” file storage approaches for each application, which is more secure because the file imported for one application will not affect other application installed on the same iOS machine. But definitely, for iOS applications developers, this file storage approaches limits the flexibility of file handling.



[2]

Fig.24 An illustration of the iOS Sandbox storage approaches

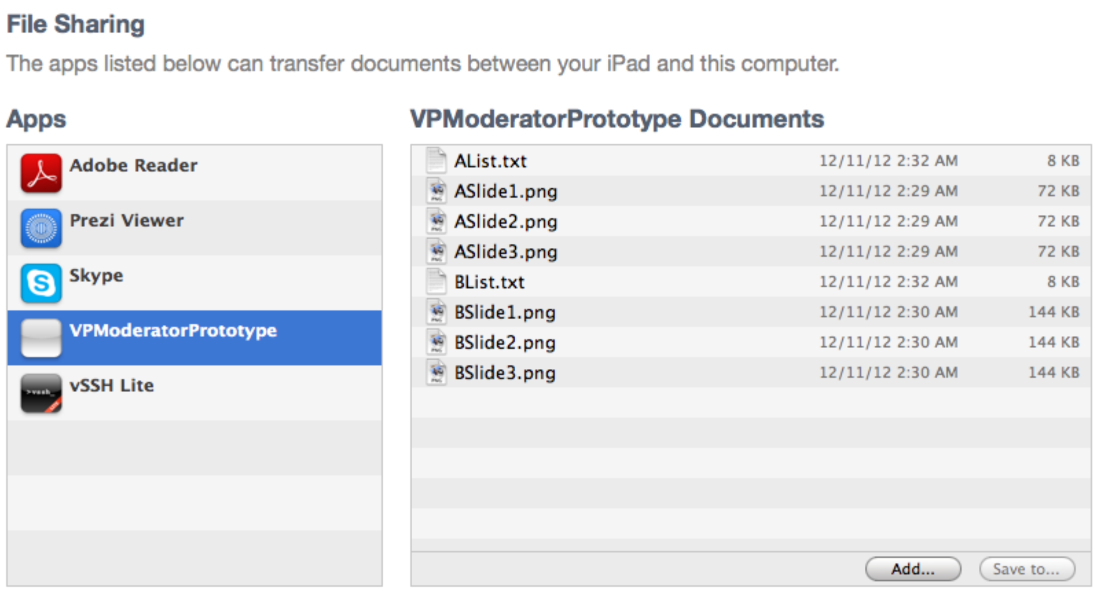
On the Internet, one of the most frequently asked questions made by iOS application developers is: How can we interact (import/export) data files with the application? The answer would be through iTunes developed by Apple.

First of all, a variable is added into the preference list (one of the project file with file extension "`.plist`"):

Application supports iTunes file sharing = YES

This property triggered out the file sharing ability of the iOS application. Then the iTunes will show the connected iPad and allow to put files into the application, as shown below

[2] *Image Source: http://www.iphoneroots.com*



File Sharing

The apps listed below can transfer documents between your iPad and this computer.

Apps

- Adobe Reader
- Prezi Viewer
- Skype
- VPModeratorPrototype
- vSSH Lite

VPModeratorPrototype Documents

AList.txt	12/11/12 2:32 AM	8 KB
ASlide1.png	12/11/12 2:29 AM	72 KB
ASlide2.png	12/11/12 2:29 AM	72 KB
ASlide3.png	12/11/12 2:29 AM	72 KB
BList.txt	12/11/12 2:32 AM	8 KB
BSlide1.png	12/11/12 2:30 AM	144 KB
BSlide2.png	12/11/12 2:30 AM	144 KB
BSlide3.png	12/11/12 2:30 AM	144 KB

Fig.25 File sharing interface in iTunes

But, this file sharing approaches has a great limitation on it. That is users are only allow to add files into the document folder, but neither the users nor the developers can add, create or access any sub-folders in this file sharing folder; neither through this interface, nor through program coding. Things getting worse is, developers are also not allowed to create any folders at the application home folder.

Nevertheless, it highly limited the file accessibility of iOS application. Still there is an old saying: “When God closes a door, somewhere he opens a window”. In our case, that is “When Apple disables the sub-folder storage of the document folder, they allow another fully functional folder to use.” Actually, each application will have a special folder to use, named “Library” in iOS 5.1.1. This folder is not visible to the users, since they will not be able to see the contents of these folders through iTunes. But it can be fully controlled by the application developers through program codes.

The following shows the program code to get the shared folder path of the current application:

```
//---init a file manager---
NSFileManager * filemgr = [NSFileManager defaultManager];
//---get the path of the Documents folder---
NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
NSString *documentsDirectory = [paths objectAtIndex:0];
```

Then, for our implementation, we can make use of the file sharing ability provided by iOS and iTunes, to allow the presenters to import their prepared presentation materials. Then we can copy it into the "Library" folder and to make further processing without changing the original contents imported by the users.

5.6 User Interface

5.6.1 Split View

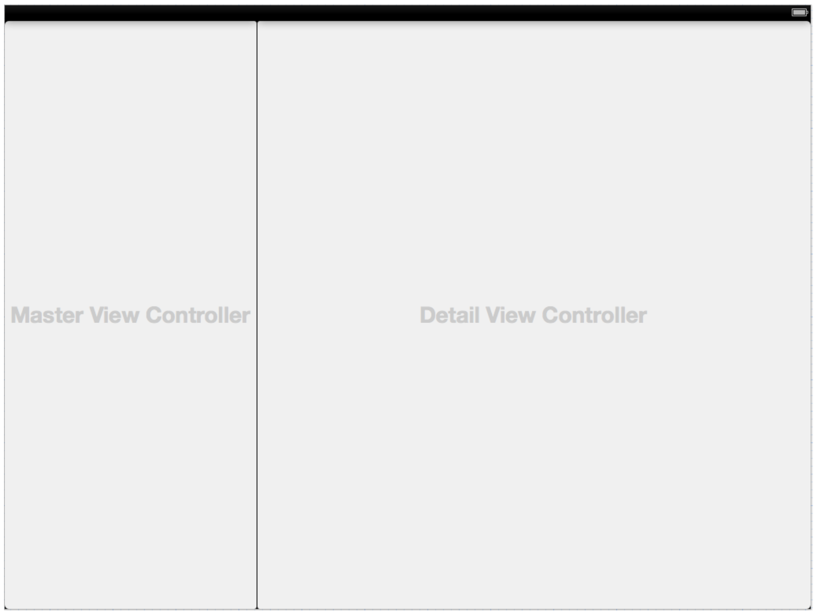


Fig.26 Split View Representation display in Xcode interface builder

As mentioned at the previous sections, split view controller is one of the famous user interface design framework, widely using by the application on iPad. This framework itself have already support and pre-implemented lots of animations and segue actions concerning the two main subviews of it, namely Master View and Detail View, from left to right respectively. As defined in class `UISplitViewController`, the two subviews are stored in an `NSArray`, with object index `0` and `1` respectively.

Because of this design, the reference of both subview controllers can be obtained easily by accessing the property of the split view controller. The following code demonstrate how to the reference of the subview controllers:

```
// in subclass of UISplitViewController
// Getting the master view controller
UIViewController* mvc = [self.viewController objectAtIndex :0];
// Getting the detail view controller
UIViewController* dvc = [self.viewController objectAtIndex :1];
```

On the other hand, the subviews will also holds a property to the reference of the nearest split view controller ancestor in the view hierarchy. That means the subview controllers can easily access the property of the split view controller. This is essential for different view controllers to communicate and passing values and variable as we often use the controlling items in the master view to control the interface items possessed by another view controller.

Other than the references of view controllers, a number of inherited method from the `UISplitViewController` and the delegate methods after implemented the `UISplitViewControllerDelegate` protocol are worth mentioning. These methods involve to control the interface behaviors when the device is in various orientation, the action of the master view controller, etc. In our implementation, since we only support landscape orientation layout, the major task carry out by the `UISplitViewControllerDelegate` protocol is minimal because we do not need to reorder or resize the interface items on the detail view.

One of the major problems for iOS developers who are unfamiliar with the split view controller framework is to control the action of the application when the orientation changes. We have spent a significant among of working hours to figure out the working principle.

```
-(BOOL)shouldAutorotateToInterfaceOrientation:
(UIInterfaceOrientation)toInterfaceOrientation
```

This method have to be implemented to all view controllers which are under the view hierarchy of the split view controllers before the whole application can perform rotation correctly.

5.6.2 Tab Bar

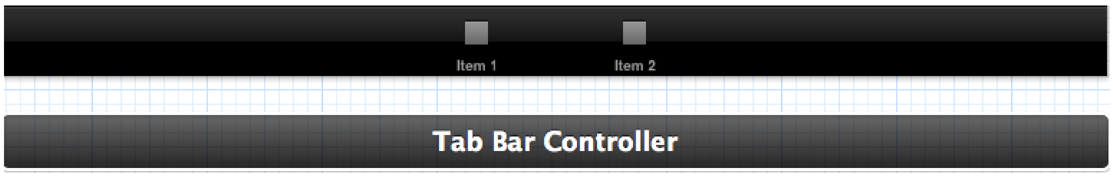


Fig.27 Tab Bar View display in Xcode interface builder

Tab bar is used to divide the same view space into different pages. It can be achieved by linking the respective tab bar items with respective view controllers. Similar to the split view controllers, the tab bar controller can contains its subviews in an `NSArray`, with object index from `0` to `(n-1)` when there is `n` tabs defined.

Again, the subview controllers will also hold a property of object reference of their nearest tab bar controller ancestor. One of the features we found this useful is to change the title view of the top navigation bar when switching between tabs. Also, it is common that there are different layouts of navigation bar view, and it can be changed easily by directly change the navigation bar behaviors.

5.6.3 Table View

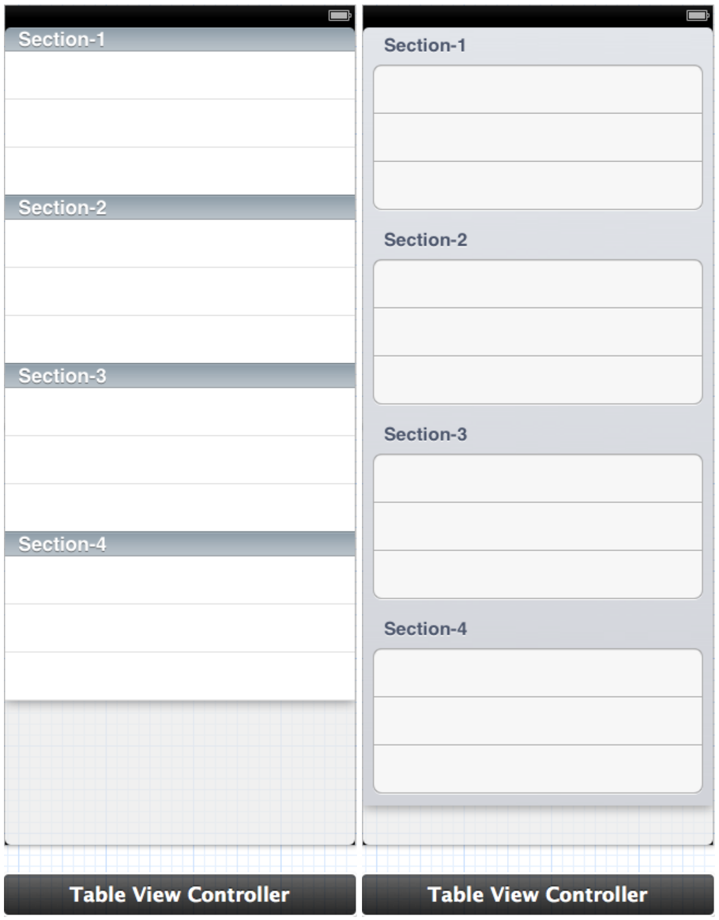


Fig.28 Table View display in Xcode interface builder

Here shows two style of table view, from left to right, namely plain style and grouped style respectively.

In our implementation, the table view controllers under each tab bar item are stored in the tab bar controller as mentioned above. The interface builder in Xcode can in fact generate a static type of table view with static contents, number of cells and sections. But that is definitely not suitable for our implementation, as the contents in the table views are used to display the system status, which the data needed to be obtained and updated in real time.

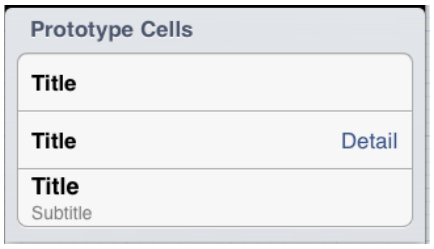


Fig.29 Prototype cells defined in the interface builder

Then the table needed to be configured into dynamic type table view, which that all the contents, number of rows and sections are defined by program code. Although the program code can define all the cell layout properties such as fonts and colors, the interface builder also support to build "prototype cells" to configure the cell layout.

Every prototype cell have to assign a unique "reuse identifier" for the table view controller to maintain a "pool" of reusable cells. Cells that use the same identifier will be generated based on the same prototype cells.

`UISwitch` and `UITextField`

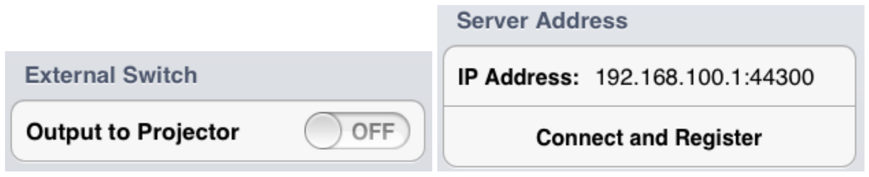


Fig.30 A controlling item added as a subview of a grouped cell.

Here shows two types of them, from left to right, a `UISwitch` and `UITextField` (the upper cell) respectively.

This kind of cell is actually very common on the set up screens and scenarios, which is one or more grouped cells with the titles on the left, and the switch on the right. This modification is not possible to be defined on the interface builder. So the control items have to be defined and added to the cell by program code. `UITableViewCell` objects have a property called accessory view, which can accept any instance created by subclass of `UIView`. The `UISwitch` can be added by this method.

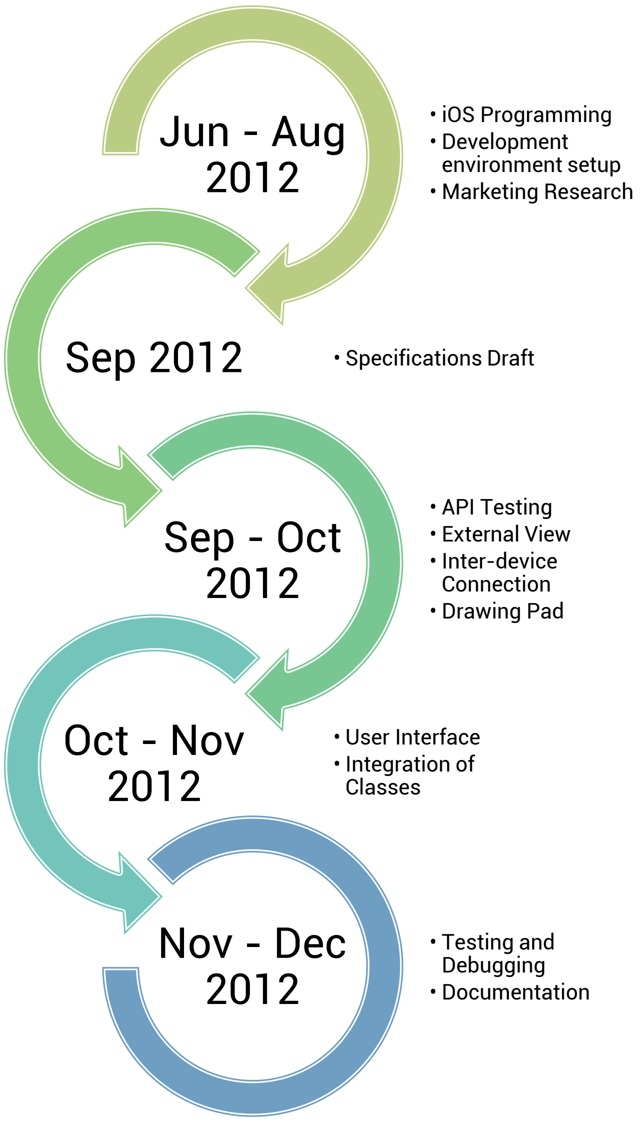
The method to add the text field is a little bit different from that of the switch, since `UITextField` itself does not have a standard size, so the text field need to be first initialize with a frame which used to define the size of it, then add the text field into the subview of the cell. Here is the code snippet how we implemented this:

```
// init. of text field
UITextField *filenameTextField = [[UITextField alloc]
initWithFrame:CGRectMake(130, 12, 185, 30)];
// add text field to the cell
[aCell addSubview:filenameTextField];
```

Chapter 6. Progress and Evaluation

6.1 Progress Report

The project was started from June 2012. As we are newbie to iOS programming and application development, we have to learn the syntax of Objective-C, design pattern of iOS application, user interface as well as APIs of Cocoa framework and NextStep API (NS*). We study iOS programming according to Apple's developer site and manual, online tutorial and course of Stanford University, which is available in iTunes U. In addition to study iOS programming, we set up environment of development in summer, and also work on marketing research about presentation software and systems.



Starting from September, we draft the specification of project, including features and functions being included, and plan the schedule of semester.

With draft of specification, we make some demonstrations with use of APIs provided. Demonstrations include the use of external screens, making connection between devices and also drawing on screen as path. Each demonstration make use of UIKit and user interface items of iOS, and also making us conformable with development environment and Objective-C syntax. In addition, the demonstration make classes or model separate from user interface, allowing us to import the classes for prototype without modification of class. The development is modularized and bottom-up approach, solving small problems then integrate to whole solution.

After testing of APIs with demonstration, we integrate classes done in each demonstration as well as developing user interface with integrating to classes. The user interface items, provided by UIKit, use lots of delegation and design pattern of Cocoa framework. In addition, we evolve the user interface a few times in order to making the application more user friendly. Therefore this progress used more time than expected.

After integrating classes and features into the applications, we have tested the application with correctness, also optimizing some code for better performance. In addition, we have to tidy up documents written before.

6.2 Outcome

In this semester, we have developed two prototypes of iOS application to demonstrate the basic idea of collaborative presentation. The two prototypes are moderator and presenter. The two prototypes able to communicate with each other under local area network, and practically be able to communicate via internet.

Regarding the moderator, it support making slides show display on external screen with scaling to best fit, and also can handle messages from presenter. Messages able to handle including registration of presenter, un-registration of presenter, requesting control of presentation, slides data and path drawing command. In addition, the moderator itself is also able to load slides and do presentation with timer and full presentation control.

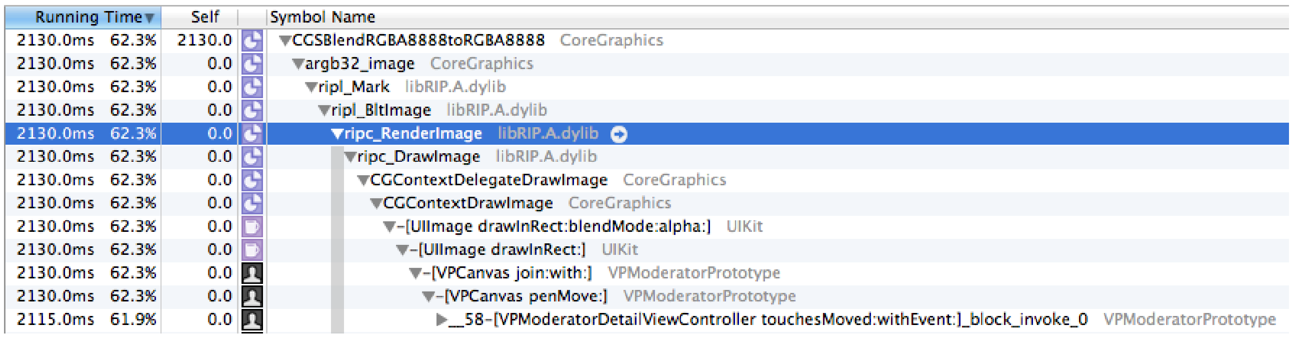
On the presenter side, it is able to connect to moderator, and also send message and present with moderator. After presenter register to moderator, it can request presentation control permission. Receiving control grant message, the slide data can be send to moderator and show in external monitor, as well as synchronizing drawing with echoing mechanism.

In conclude, we can make a wireless presentation will multiple presenters with synchronizing slides data and path drawing. Multiple presenters and seamless presentation is done under moderator control and moderator itself also able to present.

6.3 Issues

The current prototype are having some issues not yet been solved. Issues including

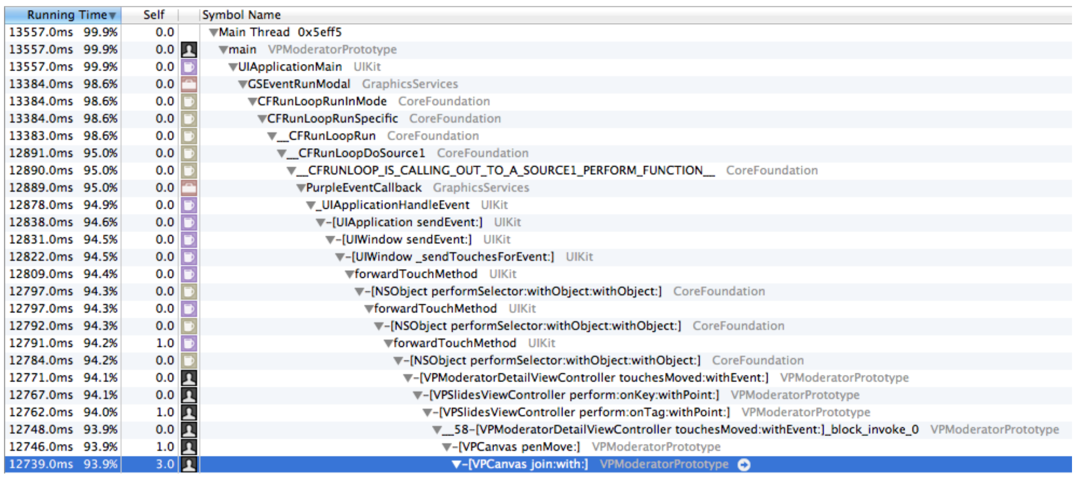
When a path is becoming long and position change rapidly, the drawing become lag and some point is missing. Observing the issue, we use Instruments, a debugging and analyzer of iOS application, to analyze the applications during such case. In time profiler of Instruments, we found the problem is due to huge CPU time in drawing path between two points in Core Graphics. We believe this is because Core Graphics drawing operation is computationally expensive. When the drawing involve rendering of large area and rapid changes, the CPU time shapely increase.



```
Running Time▼       Self     Symbol Name
2130.0ms  62.3%   2130.0   ▼CGSBlendRGBA8888toRGBA8888  CoreGraphics
2130.0ms  62.3%      0.0    ▼argb32_image  CoreGraphics
2130.0ms  62.3%      0.0     ▼ripl_Mark  libRIP.A.dylib
2130.0ms  62.3%      0.0      ▼ripl_BltImage  libRIP.A.dylib
2130.0ms  62.3%      0.0       ▼ripc_RenderImage  libRIP.A.dylib
2130.0ms  62.3%      0.0        ▼ripc_DrawImage  libRIP.A.dylib
2130.0ms  62.3%      0.0         ▼CGContextDelegateDrawImage  CoreGraphics
2130.0ms  62.3%      0.0          ▼CGContextDrawImage  CoreGraphics
2130.0ms  62.3%      0.0           ▼-[UIImage drawInRect:blendMode:alpha:]  UIKit
2130.0ms  62.3%      0.0            ▼-[UIImage drawInRect:]  UIKit
2130.0ms  62.3%      0.0             ▼-[VPCanvas join:with:]  VPModeratorPrototype
2130.0ms  62.3%      0.0              ▼-[VPCanvas penMove:]  VPModeratorPrototype
2115.0ms  61.9%      0.0               ▶__58-[VPModeratorDetailViewController touchesMoved:withEvent:]_block_invoke_0  VPModeratorPrototype
```

Fig.31 Time Profile of Testing Path Drawing in Local Device Screen

The problem become more apparent when the device is connected to external monitor. We believe the reason is the application have to render two views and, the scaling function for each point involve many floating point calculation. When connecting to external monitors, the CPU time increase much more and the path is not smooth when we disable the echo mechanism.



```
Running Time▼         Self   Symbol Name
13557.0ms  99.9%      0.0    ▼Main Thread  0x5eff5
13557.0ms  99.9%      0.0     ▼main  VPModeratorPrototype
13557.0ms  99.9%      0.0      ▼UIApplicationMain  UIKit
13384.0ms  98.6%      0.0       ▼GSEventRunModal  GraphicsServices
13384.0ms  98.6%      0.0        ▼CFRunLoopRunInMode  CoreFoundation
13384.0ms  98.6%      0.0         ▼CFRunLoopRunSpecific  CoreFoundation
13383.0ms  98.6%      0.0          ▼__CFRunLoopRun  CoreFoundation
12891.0ms  95.0%      0.0           ▼__CFRunLoopDoSource1  CoreFoundation
12890.0ms  95.0%      0.0            ▼__CFRUNLOOP_IS_CALLING_OUT_TO_A_SOURCE1_PERFORM_FUNCTION__  CoreFoundation
12889.0ms  95.0%      0.0             ▼PurpleEventCallback  GraphicsServices
12878.0ms  94.9%      0.0              ▼_UIApplicationHandleEvent  UIKit
12838.0ms  94.6%      0.0               ▼-[UIApplication sendEvent:]  UIKit
12831.0ms  94.5%      0.0                ▼-[UIWindow sendEvent:]  UIKit
12822.0ms  94.5%      0.0                 ▼-[UIWindow _sendTouchesForEvent:]  UIKit
12809.0ms  94.4%      0.0                  ▼forwardTouchMethod  UIKit
12797.0ms  94.3%      0.0                   ▼-[NSObject performSelector:withObject:withObject:]  CoreFoundation
12797.0ms  94.3%      0.0                    ▼forwardTouchMethod  UIKit
12792.0ms  94.3%      0.0                     ▼-[NSObject performSelector:withObject:withObject:]  CoreFoundation
12791.0ms  94.2%      1.0                      ▼forwardTouchMethod  UIKit
12784.0ms  94.2%      0.0                       ▼-[NSObject performSelector:withObject:withObject:]  CoreFoundation
12771.0ms  94.1%      0.0                        ▼-[VPModeratorDetailViewController touchesMoved:withEvent:]  VPModeratorPrototype
12767.0ms  94.1%      0.0                         ▼-[VPSlidesViewController perform:onKey:withPoint:]  VPModeratorPrototype
12762.0ms  94.0%      1.0                          ▼-[VPSlidesViewController perform:onTag:withPoint:]  VPModeratorPrototype
12748.0ms  93.9%      0.0                           ▼__58-[VPModeratorDetailViewController touchesMoved:withEvent:]_block_invoke_0  VPModeratorPrototype
12746.0ms  93.9%      1.0                            ▼-[VPCanvas penMove:]  VPModeratorPrototype
12739.0ms  93.9%      3.0                             ▼-[VPCanvas join:with:]  VPModeratorPrototype
```

Fig.32 Time Profile of Testing Path Drawing when Synchronizing with External Screen

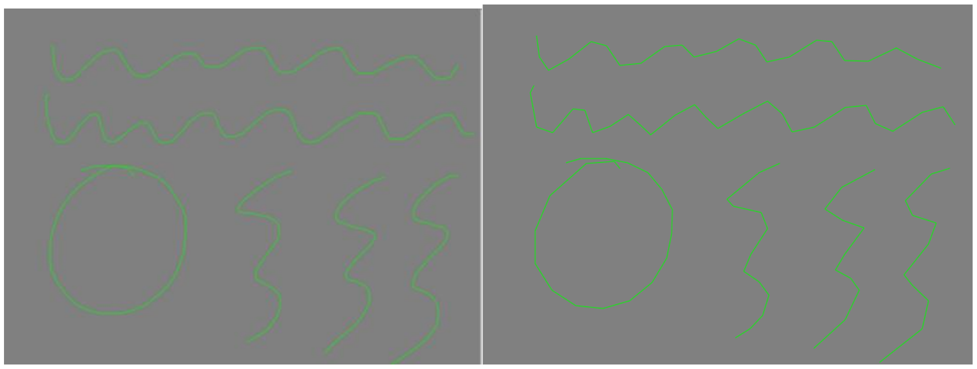


Fig.33 Canvas on Presenter (Left) and Moderator (Right)

We have disables the echoing mechanism, drawing directly to screen and ignoring the echo message in presenter. Therefore the presenter view shows all points detected and send to moderator. However, it is obvious in Fig.33 that the left canvas, which is device view of presenter, is smooth but the canvas on moderator, showing at the right of Fig.33, is not smooth.

Another issue is when we use presenter to write large amount of points in short period of time, the moderator would force quit. The problem is, again, due to high computational time in Core Graphics. In such case, the moderator is not able to handle the large amount of data is short time. The received message is kept in buffer but not able to process. When the large amount of point is sent quickly, the memory usage of application raise rapidly and receive memory warning, then force quit due to memory management of kernel. We use instrument to monitor the memory usage as well as network connection, with result in Fig.34. At the end the graph, the application force quitted.

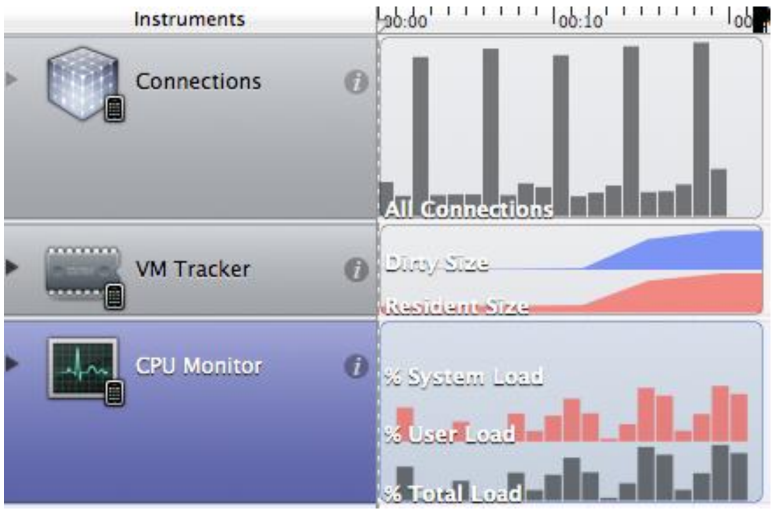


Fig.34 Connection Monitor, VM Tracker showing Memory Usage and CPU Monitor about CPU loading of Moderator when Receive Large Amount of Point

Addressing the issue, we now handle every three points touched instead of handling each point. Thus the lag issue and force quit issue is solved. However, we are still worry about when more shape and content is available and need to handle.

Chapter 7. Conclusion

In the semester, we have defined collaborative presentation, a new concept of presentation style. The collaborative presentation development is still at starting phase, and we are still improving the concept with more rigid definition, examples and features.

Moreover, we have implemented two prototypes for demonstrating collaborative presentation, for moderator and presenter respectively. Despite we have only implement a subset of functions stated in our design, we are able to demonstrate the idea of collaborative presentation, as well as proof the feasibility of implementing collaborative presentation application on iOS.

Apart from works related to collaborative presentation, we also learn and used to develop iOS application and the development environment. We get used to Xcode, storyboard, Objective-C and APIs provided by Cocoa framework. As a newbie of iOS programming from Summer 2012, we are now able to write an application with network communication on iOS. We are also able to use Core Graphics APIs to draw and render on iOS. Our learning ability have been improved, as well as the problem-solving skills.

Finally, we tried designing applications starting from zero. As application development involve many aspect including software design, user interface and experience designing, implementation of application, testing, debugging and also profiling of application, aiming at optimization, we have an experience of whole software engineering cycle. This is a valuable experience for us.

The prototypes are still not ready to market, in terms of performance, user experience and functionality. We will continue the development in the next semester.

Chapter 8. Future Development

We are going to continue the project in next semester. Our work and target of next semester can be categorize to two group: improving and optimizing existing functions; and also adding new features into our applications.

Regarding optimizing existing function, the critical issue would be computation-heavy Core Graphics rendering causing lag and force quit of application. In order to solve the problem, we planned to implement the canvas using OpenGL ES, a subset of OpenGL for mobile devices. As OpenGL is supported by GPU of iPad, we hope the performance could be improved by re-implementing with OpenGL.

The second issue being addressed is transmitting path-drawing point one-by-one using TCP connection. Using TCP to transmit little amount of data frequently is not efficient, and the TCP overhead cause performance issue and therefore slow down the synchronizing latency, as well as over-loading the kernel of iPad. We planned to modify the implementation, transmitting some small amount data with UDP, the best-effort protocol without guarantee of correctness and successful delivery. We need to use more time on considering the balance of performance and correctness.

Regarding new features, there are features stated in the design but not yet been implemented, such as recoding, blanking the screen and support of other presentation slides type. Those would be in our schedule of next semester.

Last but not least, we still have to implement another applications for viewer other than moderator and presenter. We also have to implement a back-end server for handling large scale conference.

Chapter 9. Acknowledgement

Firstly, we would like to thank our supervisor Prof. Michael R. Lyu. He provided us valuable comments and guidelines throughout the whole project.

Secondly, we would like to thank the researchers from ViewLab, Mr. Edward Yau Hon Hei and Mr. Un Tze Lung provided great amount of hardware supports and remarkably ideas to make our project become more fruitful and interesting.

Last but not least, we would like to thank our lab technicians to provide the accessibility of using the developing tools of Mac OS machines in our department computer laboratory.

Chapter 10. Reference

[1] Apple Inc. iOS Developer Library [Online]. Available: https://developer.apple.com/library/ios/navigation/index.html

[2] Object Management Group. UML Specification [Online]. Available: http://www.omg.org/spec/UML/

[3] Opencast Matterhorn http://opencast.org/matterhorn/

[4] Echo 360. Higher Education's First Active Learning Platform | Echo360 [Online]. Available: http://echo360.com/

[5] Microsoft. Microsoft PowerPoint [Online]. Available: http://office.microsoft.com/zh-hk/powerpoint/

[6] Stack Overflow [Online]. Available: http://stackoverflow.com/