ICOSEG

A Design Project Report
Presented to the Engineering Division of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering (Electrical)

by
Jason Wesley Lew
Project Advisor: Tsuhan Chen
Degree Date: May 2010

Abstract

Master of Electrical Engineering Program
Cornell University
Design Project Report

Project Title: iCoseg

Author: Jason Lew

Abstract:

As people continually turn to embedded devices for their multimedia needs, mobile applications are becoming increasingly pragmatic, powerful, and prolific. In order to provide users with the power to easily manipulate their images in a meaningful way, we implemented an image cosegmentation algorithm on the iPhone, iPod Touch, and iPad. Development of this application for the current iPhone OS was done in Apple's Cocoa API and programming environment, and testing was conducted with the iPhone Simulator as well as on live iPhone, iPod Touch, and iPad hardware. The result of this project was a mobile application which employed a user-friendly interface allowing users to manipulate their favorite images in a meaningful way by being able to segment different objects in the image through Multi-Touch gestures. This application has significance because it will allow the ordinary people to easily apply a powerful image processing algorithm in order to segment different objects in their their favorite photos.

Report Approved by

Project Advisor: _____ Date: _____

1 Executive Summary

In summary, an interactive cosegmentation application was created for the iPhone, iPod Touch, and iPad.

A user-friendly GUI was developed to navigate through the application. It took advantage of existing APIs and also implemented some customized UI.

A custom Cover Flow was developed to navigate between different images.

User input was detected from the touchscreen. Single touch gestures were used for UI navigation, drawing, and translation. Multi-Touch gestures were used for zooming.

The accelerometer was utilized as a form of user input. It was used to rotate the screen when the device orientation changed, and it was used to clear scribbles when the device was shaken.

A drawing algorithm was implemented. It took advantage of OpenGL's hardware acceleration capabilites to maximize drawing responsiveness to the user as well as maximize the number of scribble points captured.

An Apache Web Server was set up on a Mac Pro. PHP scripting was enabled on this server.

The algorithm was run using Matlab on the web server. PHP scripts were written to receive images from the client, receive scribbles from the client, and run the algorithm. The client and server were able to communicate effectively with one another.

The app was run successfully on the iPhone Simulator, iPad Simulator, iPhone, iPod Touch, and iPad.

2 Introduction

In the past few years, the mobile device has become a full-featured communcations and entertainment platform. The popularity and power of these do-it-all devices will only continue to grow in the years to come as developers shift their focus to the mobile applications marketplace. After Apple Inc.'s 2008 launch of the App Store–which allowed software developers to easily distribute their applications to the populace–users have increasingly been in search of innovative and powerful mobile applications. Our specific application area of interest is image processing, and we have implemented our exisiting image cosegmentation algorithm on the iPhone, iPod Touch, and iPad. Once the user has denoted an object and the background in an image through scribbles, the algorithm can accurately separate them. The iPhone, iPod Touch, and iPad are ideal devices for our application because their Multi-Touch systems allow users to easily draw on their favorite pictures; moreover, our application can effectively utilize other potent device features such as the camera, accelerometer, and Internet connectivity. The result of the project was a user-friendly application which everyday users can use to segment their favorite photos. Dr. Tsuhan Chen has advised the project; Ph.D. students Dhruv Batra and Adarsh Kowdle have developed the cosegmentation algorithm; and I was responsible for designing and developing the iPhone App and integrating it with the algorithm.

3 Background

3.1 Image Cosegmentation

The work of Batra, Kowdle, Parikh, Luo, and Chen presents an algorithm for interactive cosegmentation of a foreground object from a group of related images [13]. The algorithm allows users to decide what the foreground and background are by denoting each through scribbles. Having the user involved in the cosegmentation process allows for the use of many images per group. Their system can intelligently differentiate the foreground from the background, and the user can attain high-quality segmented images with much greater ease than if all cutouts had been exhaustively examined.

Figure 1: Interactive Cosegmentation of Topically-Related Images

3.2 iPhone App Development

3.2.1 iPhone SDK

The iPhone SDK is a software development kit developed by Apple, Inc. Third-party developers can use the iPhone SDK to develop native apps for iPhone OS. The SDK is free to download, meaning that developers can develop apps and run them in the provided simulator at no cost. However, in order to run an app on hardware or release an app to Apple's App Store, one must enroll in the iPhone Developer Program at a cost of $99 per year.

Figure 2: iPhone SDK

3.2.2 iPhone OS

iPhone OS as a mobile operating system developed by Apple. It runs on the iPhone, iPod Touch, and iPad. iPhone OS is derived from Apple's Mac OS X operating system. iPhone OS, like Mac OS X, has a Darwin OS foundation and is similar to UNIX. iPhone OS has four layers of abstraction: the Core OS layer, the Core Services layer, the Media layer, and the Cocoa Touch layer.

3.2.3 Cocoa Touch and Objective-C

Cocoa Touch is an API used to developed embedded software applications for the iPhone OS. Cocoa applications are developed primarily by using Xcode and

Interface Builder while using the Objective-C Language.

Objective-C is an object-oriented programming language which is a strict superset of the C programming language. It was created in the early 1980s by the company StepStone. In 1988, NeXT licensed Objective-C from StepStone. When Apple acquired NeXT in 1996, Apple incorporated Objective-C into its new operating syste, Mac OS X.

Figure 3: Cocoa Touch (Stanford CS193P)

4 Statement of Problem

4.1 Application Development

The major issue was choosing the right approach in developing the application. The application needed to be developed modularly so that working parts could be ultimately incorporated into the final application.

One particular challenge was learning Objective-C and iPhone App development while working on the project. I often had to spend extensive time researching how to implement certain features before I built them into the app.

The plan changed slightly when the iPad was released in April of 2010. We decided to also develop the app for the iPad because of its larger touchscreen.

4.2 Graphical User Interface (GUI)

A user-friendly GUI was of utmost importance to the project. The GUI needed to take advantage of exisiting APIs to create an interface which is familiar to users and could be easily used.

The UI for selecting an image to display was challenging since there was limited screen real estate. Displaying all images in full at once would mean that the images would be too small.

Additionally, the UI for drawing on an image needed to support various features. For example, it needed to support switching between drawing and multi-touch translation/zoom as well as switching between drawing on the foreground and background. Also, the Multi-Touch features needed to be implemented to resize and move the views.

4.3 Drawing Algorithm

Development of an effective drawing algorithm was necessary. The algorithm needed to accurately track the position of the person's finger on a given image. Also, the algorithm needed to be able to be able to account for an object and its background separately.

Speed was a key issue. The algorithm needed to be fast enough so that a non-jagged line could be drawn and that maximum scribble points could be stored.

4.4 Client-Server Model and Communication

The server needed to be able to run the algorithm, which was written in Matlab. The server needed to be abstracted as a web server since the iPhone API operates at the Application level of the ISO layers. The server needed to be accessible through WiFi.

The iPhone needed to be able to send the following to the server: images, scribble points, and a command to run the algorithm. It needed to retrieve from the server the segmented images.

4.5 Design Specifications

- Detect touchscreen input from the user
  - Detect single touch gestures for UI navigation and drawing
  - Detect Multi-Touch gestures for pinch-to-zoom
- Detect accelerometer events
  - Use to rotate the screen when the device orientation changes
  - Use to delete scribbles when the device is shaken
- Draw on the screen as the user
  - Real-time, as fast as possible

  - Hardware-accelerated

- Run on the iPhone, iPod Touch, and iPad
- Run the algorithm on a server
- Send images to server
- Trigger algorithm to run on server
- Get images back from server

5 Range of Solutions

5.1 Application Development

As far as development tools are concerned, there is no choice when developing apps for iPhone OS. I used the iPhone SDK, using Xcode to code in Objective-C as well as Interface Builder for some of the UI.

For testing, I could have tried to develop almost exclusively in the simulator and then run the app on live hardware later on. However, I opted to continually test with both the hardware and simulator during development. This allowed me the rectify any hardware-specific issues as they occurred.

I had access to my own iPhone throughout the project. I had perioidic access to Adarsh Kowdle's iPod Touch, and later had access to my own iPad. Having the different devices was useful in making sure that the app was robust on all types of compatible hardware.

5.2 Graphical User Interface (GUI)

When creating the UI, an iPhone developer has a choice between coding the UI in Objective-C in Xcode and laying it out in Interface Builder. The advantage of using Interface Builder is that it requires less coding and is more visual. However, for this application, I opted to implement most of the UI programatically since a great deal of it was customized. For instance, the Cover Flow interface and the Multi-Touch implementation could not be intuitively implemented in Interface Builder and were hence implemented programatically.

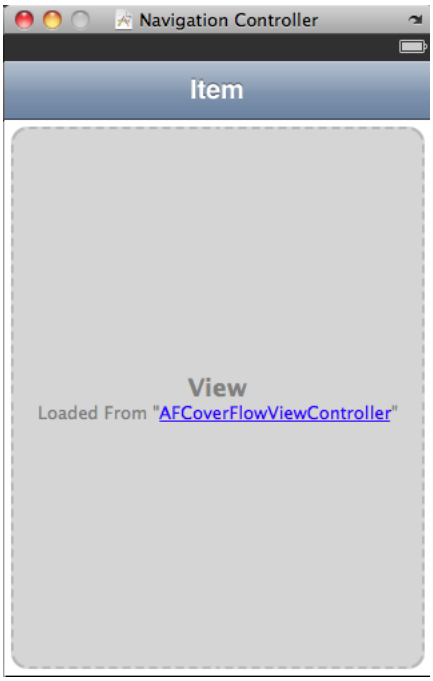

Figure 4: Interface Builder UI Layout

5.3 Drawing Algorithm

The original drawing algorithm used the iPhone's APIs, which are able to notify to program when and where a finger has been pressed onto the capacitive touch-screen, when and where it has moved, and when and where it has been released. The algorithm which I initially developed kept track of all recorded positions and continually redrew the scribble to the screen as the user moved his finger (Figure 5). Once the finger was released, since the API inevitably missed some of the points where the finger has moved, I extrapolated these missed points to create a connected line. An example of a final scribble can be seen in Figure 6.

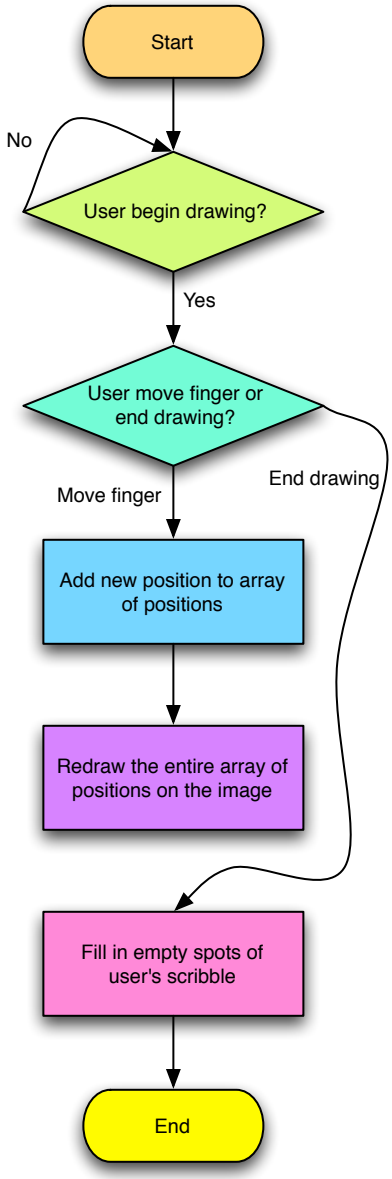

Figure 5: Original Drawing Algorithm Flowchart

This algorithm was scrapped once I discovered how to implement drawing using OpenGL, which was hardware-accelerated. The advantage of OpenGL was two-fold; it drew more quickly, meaning that points did not need to be filled in after the user had finished scribbling, and since it was faster, more scribble points could be captured since the device was able to sample finger movements at a

higher rate.

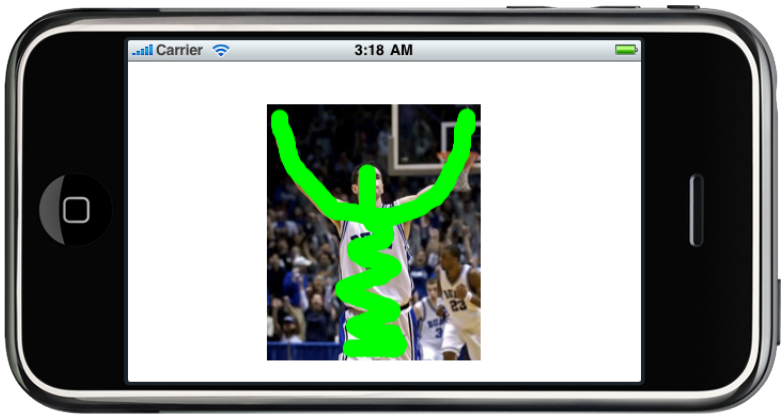

Figure 6: Original Finger Drawing

5.4 Client-Server Model and Communication

Designing and implementing the interface between the iPhone client and the server was perhaps the most challenging aspect of this project.

From the users' perspective, they could manipulate a photo, send it to the server, and it would come back segmented (Figure 7).

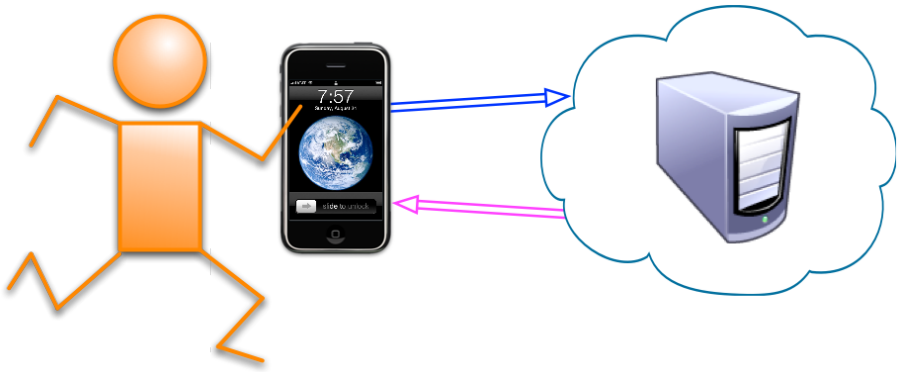

Figure 7: Client-Server Model

The iPhone (client) needed to send the following to the server:

- The images

- All of the points for the scribbles on the images
- A command to run the algorithm

The server needed to send the following back to the iPhone:

- The segmented images

Initially, I thought that FTP (File Transfer Protocol) was the best option for sending image files to and from the server. An FTP connection could be established using the iPhone's APIs. Later, though, I realized that the better solution was to use HTTP requests to communicate with a web server.

5.5 Additional Features

Additional features which were considered included:

- In the application, using the iPhone's camera to take a picture, which could then be scribbled upon and segmented
  - This feature was implemented successfully
- The choice to choose a picture from the iPhone's photo library
  - This was implemented successfully
- Being able to save the image to the photo library
  - This was implemented successfully
- The ability to send a segmented image to others through MMS (Multimedia Messaging Service)
  - This could not be implemented because the iPhone API did not allow access to the SMS application
- Sending a segmented image as an attachment to an email
  - This could not be implemented because the iPhone API did not allow access to the Mail application

6 Design and Implementation

6.1 Application Development

The application was developed for and tested for iPhone OS 3.0, 3.1, 3.1.3, and 3.2. As required by Apple to develop for the iPhone, the development tools used were those included in Apple's Cocoa API and programming environment (Figure 8).

Figure 8: Cocoa API and Programming Environment

Testing of the application was conducted with a combination of the iPhone Simulator, an iPhone 3GS, an iPod Touch, and an iPad (Figure 9) .

Figure 9: iPhone 3GS, iPod Touch, and iPad

The result was an application which can run on any iPhone, iPod Touch, or iPad with the current OS.

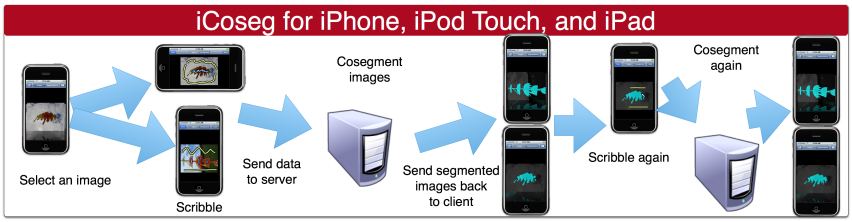

Figure 10: Application Flow

A useful practice was to test different parts of the app on their own before integrating them into the entire program. By doing this, I was able to limit the number of bugs which may have appeared later on.

6.2 Navigational GUI

The user interface was based on UIViews and UIView controllers. UIViews are displayed on the screen, and UIViewControllers control the UIViews.

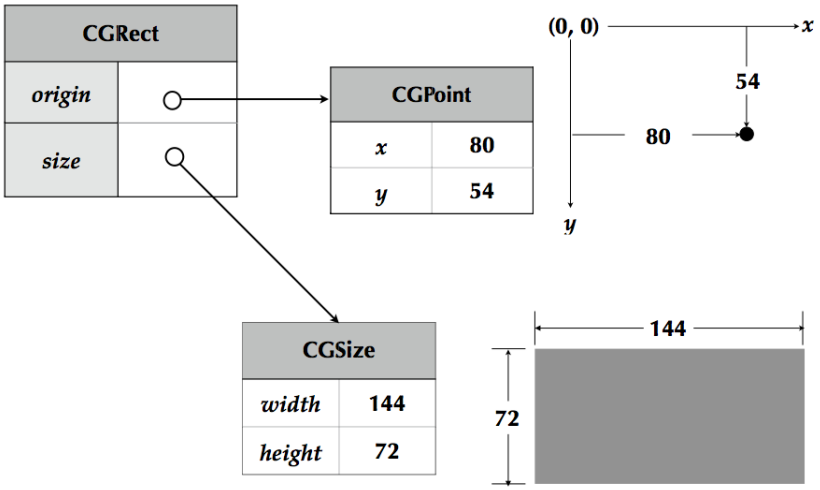

Figure 11: CGRects, CGPoints, and CGSizes (Stanford CS193P)

Each UIView is defined by a CGRect struct, which contains a CGPoint and a CGSize (Figure 11). The CGPoint gives the upper-left corner of the view, and the CGSize tells the height and width of the view.

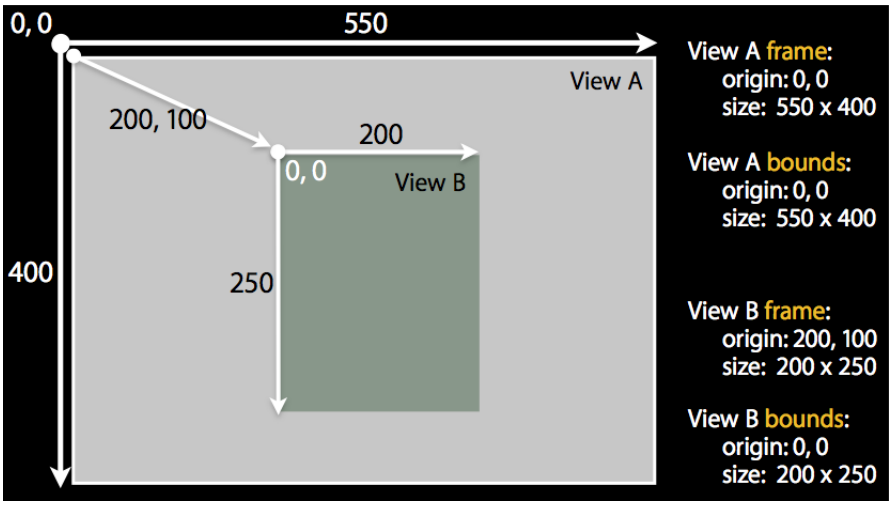

Figure 12: UIView (Stanford CS193P)

A UIView can contain what are known as subviews (Figure 12). Subviews are instantiated in terms of their superview.

One way for the user to navigate between views is by using a UINavigationController (Figure 13). A UINavigationController acts like a stack data structure. To display a different view, that view's ViewController is pushed onto the UINavigationController. To display the previous view, the current view controller is popped from the NavigationController stack.

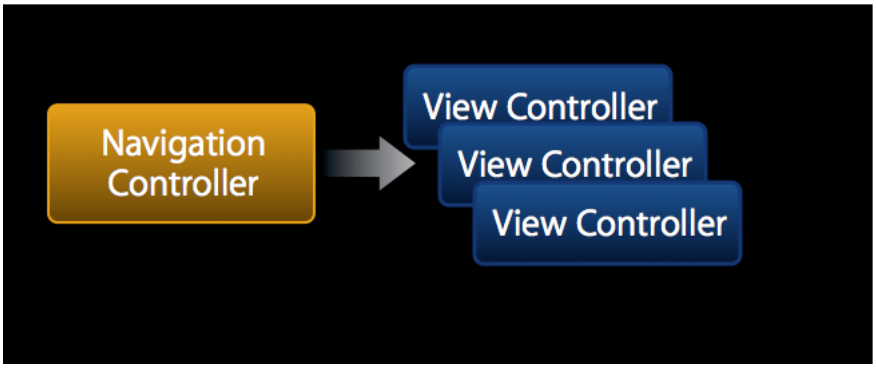

Figure 13: UINavigationController (Stanford CS193P)

For this application, a UINavigationController was used, with a AFOpenFlowViewController as the base ViewController. When an image was selected, a FingerDrawViewController was pushed onto the NavigationController.

6.3 Cover Flow

Cover Flow is visually-pleasing feature used on the iPhone to navigate between music albums (Figure 14). The user use single-finger touch gestures to swipe through the different albums, and then selects an album by pressing on it.

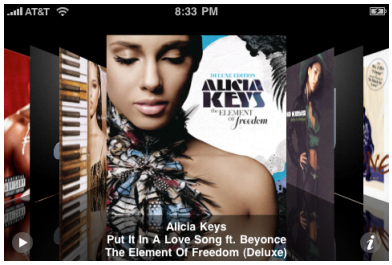

Figure 14: iPhone Album Cover Flow

I decided to implement Cover Flow as part of the iCoseg app as a way to navigate between images. Unfortunately, Cover Flow is not included in the developer's

API. An open-source Cover Flow created by Alex Fajkowski was used as a basis for my custom Cover Flow (Figure 15).

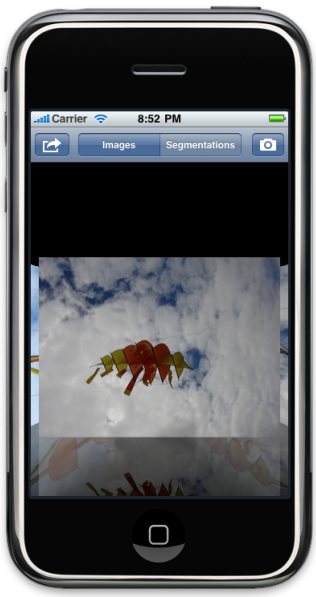

Figure 15: Custom Cover Flow

6.4 Finger Drawing

6.4.1 Scribbling

The first step in scribbling was detecting user touches. User touches were detected by overriding the touchesBegan, touchesMoved, and touchesEnded methods in the FingerDrawView. touchesBegan is called when a user places his finger on the screen. touchesMoved is called when an existing touch changes position. touchesEnded is called when the user lifts his finger off the screen, ending the Touch.

Algorithm 1 touchesBegan

- Convert the point of the touch from the UIView coordinate system to the OpenGL coordinate system (flip it upside-down)
- Change the color of the line drawn according to whether the user has selected to draw on the foreground or background

Algorithm 2 touchesMoved

- Convert the point of the touch from the UIView coordinate system to the OpenGL coordinate system (flip it upside down)
- Set the color of the OpenGL context depending on whether the user is currently selecting to draw the foreground or background
- Have OpenGL render the stroke from the previous point to the current point
- Create a a SimplePoint object to wrap the x and y coordinates of the touch
- Add the point to the mutable array holding foreground points or background points, depending on what the user has currently selected

Algorithm 3 touchesEnded

- Convert the point of the touch from the UIView coordinate system to the OpenGL coordinate system (flip it upside down)
- Have OpenGL render the stroke from the previous point to the current point

Originally, CoreGraphics was used to draw scribbles. As mentioned previously, this was way to laggy. Performance was acceptable in the simulator but struggled on the iPhone and really struggled on the iPod Touch.

Ultimately, OpenGL was used to draw the scribbles on the screen. iPhone OS supports OpenGL ES (Embedded Systems), a version of OpenGL designed specifically for use on embedded systems. The advantage of using OpenGL is that it is hardware-accelerated, meaning that it uses the device's GPU instead of the main processor. If an image was able to be drawn quickly, that meant that there was less latency between touchesMoved being called.

Drawing a line using OpenGL involved repeatedly drawing a brush image. Every time that touchesMoved was called, the image from Figure 16 was drawn on the screen at a new position. This image was given a blue color if the user was currently scribbling on the foreground, and it was given a yellow color if the user was currently scribbling on the background.

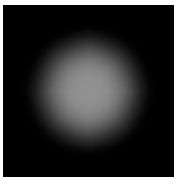

Figure 16: Brush Image used to Draw Lines in OpenGL

Upon close examination of a scribble, the original brush image can clearly be seen (Figure 17). However, at a normal zoom factor, it appears as a line with a black border (Figure 18).

Figure 17: Zoomed in on a Scribble

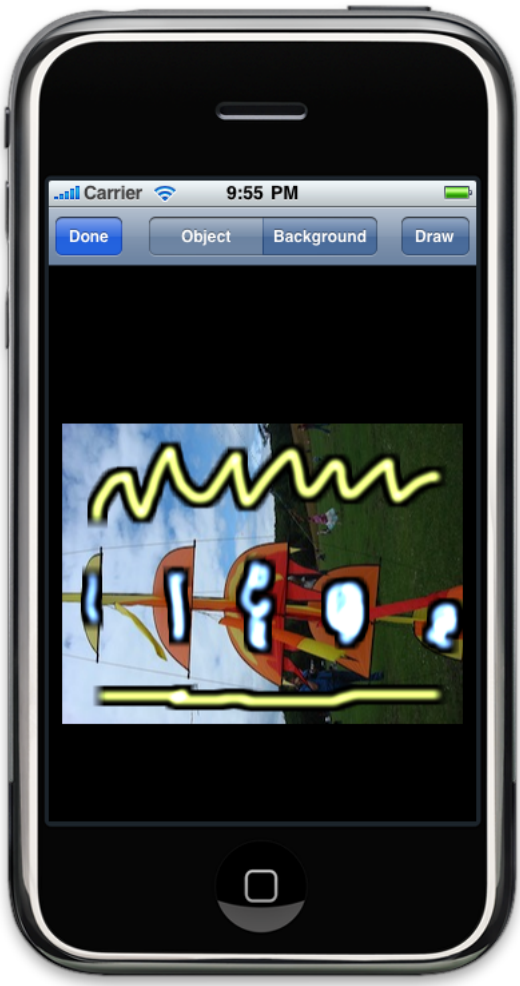

Figure 18: Scribbles

6.4.2 Multi-Touch Zoom and Translation

Multi-Touch zoom and translation were enabled by implementing a UIScrollView (Figure 19). A UIScrollView provides support for displaying content that is larger than the size of the application's window. It enables users to scroll within that content by making swiping gestures, and to zoom in and back from portions of the content by making two-fingered pinching gestures. The scrollView had a MyContainerView (a custom UIView cubclass) called containerView as a subview. myContainerView had two subviews: a UIImageView called imageView containing the image on which to scribble, and the Finger-

DrawView fingerDrawView, which accepted the user scribble input. Figure 19 shows the View structure of the Multi-Touch implementation; an arrow pointing to a View means that that View is a subview of the View from which the arrow emanates.

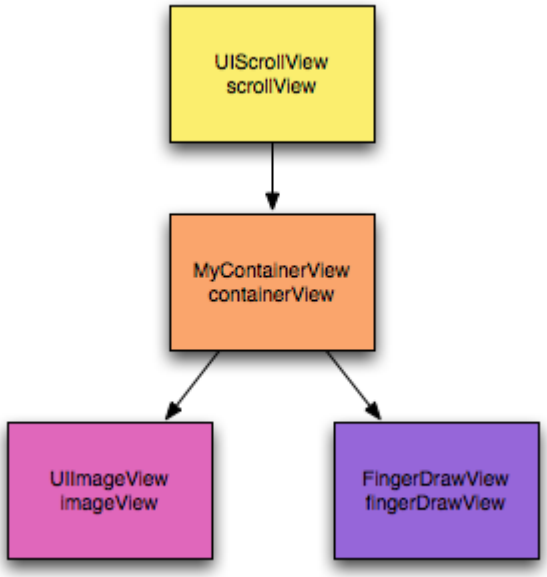

Figure 19: Multi-Touch UIView Structure

Once the subviews of the UIScrollView were organized, the key to being able to translate and zoom were to set the properties of the UIScrollView to approprite values.

The frame of a UIScrollView describes the location of what is visible on the screen at any given time. The frame of scrollView was therefore set to the entire screen minus the top navigation bar.

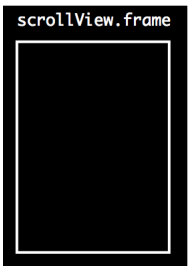

Figure 20: Frame of a UIScrollView (Stanford CS193P)

The contentSize of a ScrollView describes the size of the entire content that the frame of the ScrollView can display. The contentSize was set to the size of containerView.

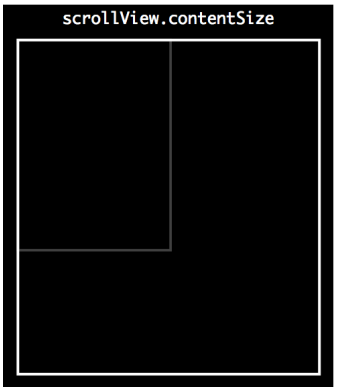

Figure 21: Content Size of a UIScrollView

The contentInset of a ScrollView describes additional scrolling area around the content of a UIScrollView. The contentInset was set to the width of the entire screen on each side and to the height of the screen minus the navigation bar on the top and bottom.

As a result of the view hierarchy which I created as well as the setting of parameters, the image could be translated and zoomed.

6.4.3 Accelerometer

The accelerometer is a very useful sensor which can be used as a form of user input. In my case, I implemented the accelerometer to detect changes in device orientation as well as the device being shaken. When the orientation changed, the screen rotated to fit the new orientation. When the device was shaken, scribbles were erased.

The way to most precisely read the acclerometer is to use the UIAccelerometer class and then have a class implement the UIAccelerometerDelegate protocol. The class implementing the UIAccelerometerDelegateProtocol can then implement the accelerometer:didAccelerate: method, which is called at a sampling rate set by the user (with the maximum sampling rate limited by the accelerometer hardware). A UIAcceleration object is passed into the accelerometer:didAccelerate: method; the UIAcceleration object contains the x, y, and z acclerations as well as a timestamp of the reading. Figure 22 shows the defined axes with respect to the device.

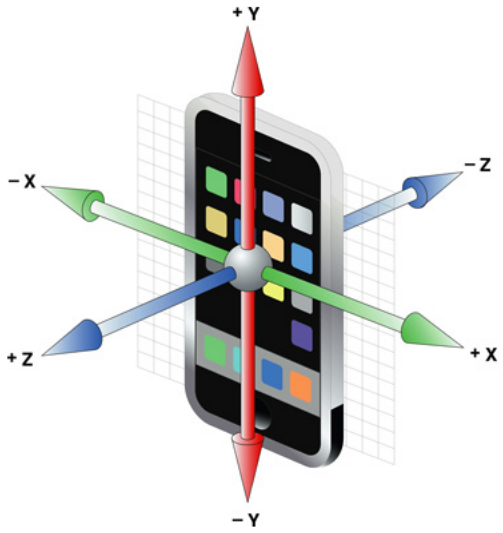

Figure 22: Accelerometer Orientation (iPhone API)

However, there are other ways to utilize the accelerometer, which proved to be more fit for this application. First, to detect a change in device orientation, UIViewControllers can override the methods shouldAutorotateToInterfaceOrientation and didRotateFromInterfaceOrientation. The first method tells the view controller which device orientations will call the second method. The second method should contain whatever changes should be made upon rotation. In the case of my application, this mainly involved re-centering the views with respect to the new orientation.

To detect the device being shaken, a UIResponder (from which UIView inherits) can utilize the motionEnded method. Within this method, I cleared the visible scribbles on the screen as well as the arrays holding the scribble points. It is important to note that before using the motionEnded method, the canBecomeFirstResponderMethod needs to be overriden to return a Boolean TRUE.

6.5 Web Server

A server needed to be set up to run the algorithm. The server needed to act as a web server since the iPhone deals with the Application Layer of the OSI model. Mac OS X has a built-in Apache server, so I used a Mac Pro in the lab as the server.

To set up a web server in Mac OS X, select Sharing under System Preferences.

Figure 23: Sharing in System Preferences (Mac OS X)

Then check the Web Sharing Box to turn on web sharing. It is also useful to utilize the Computer Name. The Computer Name is useful because it is independent of the local network. For instance, in our lab, the server was named "jwl246-1.ece.cornell.edu/", but it could be accessed at "iCoseg.local" both in the lab and on other networks.

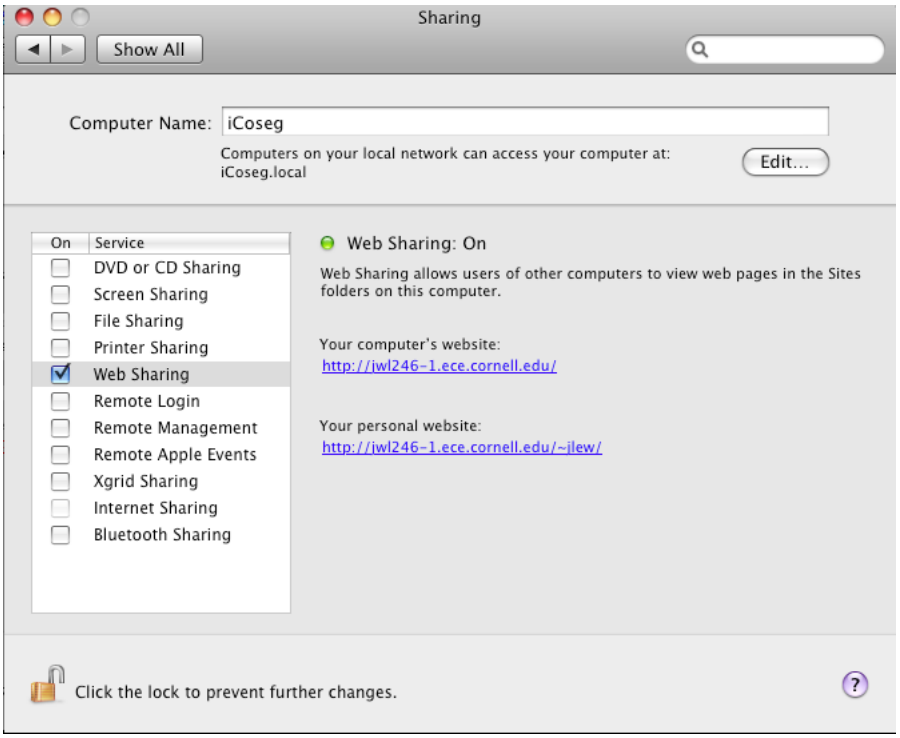

Figure 24: Enabling Web Sharing (Mac OS X)

Once the web server was set up, the next step was to be able to control executables on the server from the App. At first, I looked into CGI scripts. I was able to run CGI scripts on the server, but I was unable to run executables through the CGI scripts, likely because of a file permissions issue.

I then switched to using PHP scripts, which provided similar functionality. Credit goes to Dun-Yu Hsiao from the University of Washington for this suggestion. In order for the Apache server to run PHP scripts, its "httpd.conf" file needed to be modified. The ensuing line needed to be uncommented to enable php:

```
LoadModule php5_module libexec/apache2/libphp5.so
```

6.6 App-Server Communication

Pressing the top-left button in the Cover Flow view called the method sendImagesAndScribblesAndRunAlgorithm, which triggered client-server communication. The following describe the steps that the App needed to take to talk to the server.

Algorithm 4 Send Images to Server

- For each image:
  - Create an NSData object from the UIImage
  - Set the target URL to "http://iCoseg.local/~jlew/image_upload.php"
  - Set the HTTP Method to a HTTP POST
  - Add some header info to the POST
  - Add the body of the post
    - Include the name of the photo file in the "filename" parameter
  - Send a synchronous request
  - Store the returned string from the request

Algorithm 5 Send Scribbles to Server

- For each image (each image has 1 set of foreground scribbles and 1 set of background scribbles):
  - Set the target URL to "http://iCoseg.local/~jlew/scribbles.php"
  - Set the HTTP Method to a HTTP POST
  - Add some header info to the POST
  - Add the body of the post
    - Include the name of the text file in the "filename" parameter
    - Add the text of the file, which includes the image name, width, height, number of foreground points, foreground points, number of background points, and background points
  - Send a synchronous request
  - Store the returned string from the request

The format of the text file containing the scribbles is as follows:

```
<image name>
<width> <height>
<number of foreground points>
<foreground point 1 x-coord> <foreground point 1 y-coord>
.
.
.
<foreground point n x-coord> <foreground point n y-coord>
<number of background points>
```

```
<background point 1 x-coord> <background point 1 y-coord>
.
.
.
<background point n x-coord> <background point n y-coord>
```

Algorithm 6 Run the Cosegmentation Algorithm

- Set the target URL to "http://iCoseg.local/~jlew/matlab_run_script.php"
- Send a synchronous HTTP GET request to the URL
- Store the returned string from the request

Algorithm 7 Get Segmented Images from Server

- For each image
  - Set the target URL to the image to retrieve
  - Send a synchronous HTTP GET request to the URL
  - Store the results of the request into an NSData Object
  - Create a UIImage from the NSData Object

7 Results

Please see the Appendices for detailed documentation of results.

8 Conclusion

Ultimately, the project resulted in the creation of a novel application for the iPhone, iPod Touch, and iPad which had pragmatic use for the average person. Users were able to segment their own images using an intuitive user interface. Beyond the app itself, I am now adept at mobile app development. Knowing how to develop mobile applications will be a key skill to have for my future as an engineer. Mobile app developers will be increasingly in demand as the industry continues to grow. Being able to work on this project–applying a powerful algorithm to premier mobile devices–was a great experience.

References

[1] 8 Great Resources For Learning Iphone OpenGL ES | iPhone and iPad SDK Development Tutorials and Programming Tips. <http://maniacdev.com/2009/04/8-great-resources-for-learning-iphone-opengl-es/>

[2] CGI Made Really Easy. <http://www.jmarshall.com/easy/cgi/>

[3] CGI Matlab. <http://www.mathworks.com/support/tech-notes/1600/1608.html#Section_4>

[4] CGI Programming 101: Chapter 1: Getting Started. <http://www.cgi101.com/book/ch1/text.html>

[5] CGI Programming With Apache and Perl on Mac OS X. <http://www.cgi101.com/learn/connect/mac.html>

[6] Client Server iPhone App - Stack Overflow. <http://stackoverflow.com/questions/1123903/client-server-iphone-app>

[7] Cocoa with Love: Easy custom UITableView drawing. <http://cocoawithlove.com/2009/04/easy-custom-uitableview-drawing.html>

[8] codza >> how to debug EXC_BAD_ACCESS on iPhone. <http://www.codza.com/how-to-debug-exc_bad_access-on-iphone>

[9] Connecting Apple's iPhone to Google's cloud computing offerings. <http://www.ibm.com/developerworks/web/library/wa-aj-iphone/>

[10] CS 193P iPhone Application Development. <http://www.stanford.edu/class/cs193p/cgi-bin/drupal/>

[11] Custom Delegates in Cocoa. <http://ekle.us/index.php/2007/01/custom_delegates_in_cocoa>

[12] D. Batra, A. Kowdle, D. Parikh, K. Tang, and T. Chen. <http://amp.ece.cornell.edu/projects/touch- coseg/>. 2009. Interactive Cosegmentation by Touch.

[13] D. Batra, A. Kowdle, D. Parikh, Jiebo Luo, and T. Chen. iCoseg: Interactive Co-segmentation with Intelligent Scribble Guidance.

[14] Enabling PHP in Mac OS X 10.5. <http://foundationphp.com/tutorials/php_leopard.php>

[15] EXC_BAD_ACCESS signal received - Stack Overflow. <http://stackoverflow.com/questions/327082/excbadaccess-signal-received>

[16] Executing Linux / UNIX commands from web page. <http://www.cyberciti.biz/tips/executing-linuxunix-commands-from-web-page-part-i.html>

[17] FTP vs. HTTP. <http://daniel.haxx.se/docs/ftp-vs-http.html>

[18] how do i call an .exe file from perl script - Perl. <http://www.daniweb.com/forums/thread30038.html#>

[19] How do I detect when someone shakes an iPhone? - Stack Overflow. <http://stackoverflow.com/questions/150446/how-do-i-detect-when-someone-shakes-an-iphone>

[20] How to edit httpd.conf - The macosxhints Forums. <http://forums.macosxhints.com/showthread.php?t=3640>

[21] How To Make your Mac a Web Server | Mac|Life. <http://www.maclife.com/article/howtos/how_make_your_mac_web_server>

[22] How to Market iPhone Apps. <http://www.onlinemarketingrant.com/how-to-market-iphone-apps>

[23] How to run a shell script from a Perl program?. http://www.perlmonks.org/?node_id=78523>

[24] How to write a shell script. <http://www.tinker.ncsu.edu/LEGO/shell_help.html>

[25] Internal Server Errors. <http://websitehelpers.com/perl/>

[26] iPhone Dev Center - Apple Developer <http://developer.apple.com/iphone/index.action>

[27] iPhone Development: OpenGL ES From the Ground Up, Part 1: Basic Concepts. <http://iphonedevelopment.blogspot.com/2009/04/opengl-es-from-ground-up-part-1-basic.html>

[28] iPhone in Action: UIActionSheet, Part One: Alternate User Input. <http://iphoneinaction.manning.com/iphone_in_action/2009/06/uiactionsheet-part-one-alternate-user-input.html>

[29] iPhone - Tips for building great server/client apps | blog.sallarp.com. <http://blog.sallarp.com/iphone-server-client-tips-techniques/>

[30] iPhone UIView Animation Best Practice - Stack Overflow. <http://stackoverflow.com/questions/630265/iphone-uiview-animation-best-practice>

[31] Lessons from developing an Iphone App + Server backend. <http://www.slideshare.net/sujee/lessons-from-developing-an-iphone-app-server-backend>

[32] Objective C – HTTP POSTor GET Data. <http://blog.timeister.com/2009/08/14/objective-c-http-post-get-data/>

[33] Open Source iPhone Software - The best open source apps on the iPhone. <http://opensourceiphonesoftware.com/>

[34] OpenFlow: a CoverFlow API replacement for the iPhone Alex Fajkowski. <http://fajkowski.com/blog/2009/08/02/openflow-a-coverflow-api-replacement-for-the-iphone/>

[35] Mac Developer Tips >> Objective-C: Initializers. <http://macdevelopertips.com/objective-c/objective-c-initializers.html>

[36] Making a simple web server in Python. <http://fragments.turtlemeat.com/pythonwebserver.php>

[37] MATLAB Compiler - MATLAB. <http://www.mathworks.com/products/compiler/>

[38] Matlab on the WWW. <http://www.mathworks.com/support/solutions/en/data/1-WK5DJ/index.html?solution=1-WK5DJ>

[39] Merging content of UIImageView and EAGLview - iPhone Dev SDK Forum. <http://www.iphonedevsdk.com/forum/iphone-sdk-development/20081-merging-content-uiimageview-eaglview.html>

[40] Modifying Linux, Unix, and Mac file permissions | drupal.org. <http://drupal.org/node/202483>

[41] Multi Touch Tutorial 3 - iPhone SDK Articles. <http://www.iphonesdkarticles.com/2008/09/multi-touch-tutorial-3.html>

[42] Network Programming - iPhone SDK Application Development - O'Reilly Media. http://oreilly.com/iphone/excerpts/iphone-sdk/network-programming.html>

[43] Optimising a drawing app using glCopyTexSubImage2D - iPhone Dev SDK Forum. <http://www.iphonedevsdk.com/forum/iphone-sdk-game-development/17412-optimising-drawing-app-using-glcopytexsubimage2d.html>

[44] PHP: Installation and Configuration - Manual. <http://php.net/manual/en/install.php>

[45] PHP: Program execution Functions - Manual. <http://www.php.net/manual/en/ref.exec.php>

[46] PHP vs CGI. <http://www.javascriptkit.com/howto/phpcgi.shtml>

[47] Post a UIImage to the web. <http://iphone.zcentric.com/2008/08/29/post-a-uiimage-to-the-web/>

[48] Resize a UIImage the right way - Trevor's Bike Shed. <http://vocaro.com/trevor/blog/2009/10/12/resize-a-uiimage-the-right-way/>

[49] Running a shell command inside a cgi/perl script :: ASPN Mail Archive :: modperl. <http://aspn.activestate.com/ASPN/Mail/Message/modperl/929525>

[50] Running Perl CGI on the Mac OS X Apache Web Server. <http://www.editrocket.com/articles/perl_apache_mac.html>

[51] Run shell script from web page. <http://www.cyberciti.biz/faq/run-shell-script-from-web-page/>

[52] Server-Side Programming for iPhone app? - Mac Forums. <http://forums.macrumors.com/showthread.php?t=585199>

[53] standalone CGI Matlab. <http://www.mathworks.com/support/solutions/en/data/1-30REEY/index.html>

[54] The Apache Software Foundation. <http://www.apache.org/>

[55] The beautifully detailed art of Mac OS X app icons. <http://www.tuaw.com/2009/09/05/the-beautifully-detailed-art-of-mac-os-x-app-icons/>

[56] trying to get started with iPhone to server communication. - Mac Forums. <http://forums.macrumors.com/showthread.php?t=531895>

[57] UISegmentedControl. <http://howtomakeiphoneapps.com/2009/09/here-is-how-you-use-the-segmented-control-uisegmentedcontrol/>

[58] UIView to UIImage. <http://pastie.org/244916>

[59] UNIX / Linux Bourne / Bash Shell Scripting Tutorial [steve-parker.org]. <http://steve-parker.org/sh/sh.shtml>

[60] Using C for CGI Programming. <http://www.linuxjournal.com/article/6863>

[61] Web Publishing using HTTP PUT. <http://servers.digitaldaze.com/extensions/put/>

[62] Web Server Programming. <http://www.uow.edu.au/~nabg/WebServer/>

[63] Welcome! - The Apache HTTP Server Project. <http://httpd.apache.org/>

[64] What does the property “Nonatomic”; mean? - Stack Overflow. <http://stackoverflow.com/questions/821692/what-does-the-property-nonatomic-mean>

[65] Working with PHP 5 in Mac OS X 10.5 (Leopard) - Professional PHP. <http://www.procata.com/blog/archives/2007/10/28/working-with-php-5-in-mac-os-x-105/>

[66] Xcode iPhone OS 3.0.1 Error in Organizer - technolosophy. <http://www.technolosophy.com/2009/08/xcode-iphone-os-301-error-in-o.html>

Appendix

<Add Software Documentation>

<Add Screenshots of App Working>