Masters Project

LeapMotion for Kids

Author:
Christopher Desch

Supervisor:
Dr. Jerry Fails

Submitted in partial fulfilment of the requirements
for the degree of Masters of Computer Science

in the

Department of Computer Science
Montclair State University

May 2013

MONTCLAIR STATE UNIVERSITY

LeapMotion for Kids

by

Christopher Desch

A Masters Project Submitted to the Faculty of

Montclair State University

In Partial Fulfillment of the Culminating Experience For the Degree of

Master of Science in Computer Science

August 2012

College of Science and Mathematics, Department of Computer Science

The above project was presented on May 10th 2013

Was IRB review needed (check one) Yes No

If IRB was needed, provide the IRB Approval date and Protocol Number: _____

The Project Committee attended the project presentation, read the project report and verifies that the project can be accepted as a culminating experience in accordance with the departmental and university guidelines.

Project Sponsor: Dr. Jerry Alan Fails

Committee Member: Dr. Angel Gutierrez

Committee Member: Dr. John Jenq

MONTCLAIR STATE UNIVERSITY

Abstract

Department of Computer Science

Masters of Computer Science

LeapMotion for Kids

by Christopher DESCH

The LeapMotion Technology presents a new way of interacting with a computer system alternative to the customary keyboard and mouse interface. The goal of the project was to discover whether or not an alternative, "hands off" interface has the possibility of being as successful and reliable as the keyboard and mouse interface. The purpose of this paper is to describe the development process of building an application for this new interface in order to highlight the capabilities of the new interface. The application was designed by KidsTeam through cooperative inquiry techniques for the LeapMotion.

Acknowledgements

I would like to express my sincerest gratitude to my supervisor Dr. Jerry Fails for his constructive comments and insights throughout the process of creating this master project. His enthusiasm for his craft is both inspiring and contagious. I would like to thank Montclair State University for the opportunity to study as part of their learning community. I would like to thank the faculty and staff of the University for sharing their knowledge and their time. Furthermore, I would like to thank the members of KidsTeam for volunteering their time and energy to the work necessary to complete this project. I thank you for seeing a world of possibilities in the smallest of ideas. Finally, I would like to thank my family and friends for their unending love and support.

Contents

List of Figures

List of Tables

Abbreviations

API	**A**pplication **P**rogramming **I**nterface
HUD	**H**eads **U**p **D**isplay
SDK	**S**oftware **D**eveloper **K**it
RAD	**R**apid **A**pplication **D**evelopment
USB	**U**niversal **S**erial **B**us

Chapter 1

Overview

1.1 Introduction

For decades, the keyboard and mouse have been the standard interface mechanisms of input for computer systems. Their legacy is apparent in the applications that have been designed with the purpose of maximizing their efficiency and effectiveness. However, as technology advances, new mechanisms have been created in the hopes of making interactions with computer systems less rigid. While the keyboard and mouse interface are less physically demanding on the body in order to carry out commands, they do not allow for an expressive range of motion. The keyboard and mouse are only expressive in two dimensions as they are limited to up, down or right, left movements of the mouse across a computer screen. The keyboard is even less expressive than the mouse as the user must use key strokes to carry out commands, thus making the mouse slightly more fluid in motion than the keyboard but not by much. The interfaces have stayed relatively the same despite advances in computer software and hardware. With the advent of tablets and smart devices, the envelope had been pushed in terms of what could be considered "effective" means of input for a computer system. These devices allow for a user to carry out a command with a simple stroke of their finger directly on the screen of their computer system, thus making the command more expressive than the traditional keyboard and mouse interface. The devices with touchscreen capabilities

FIGURE 1.1: Interacting with the LeapMotion. [1]

have opened the door for opportunities to create a fresh interface.A new interface may have the ability to replace the use of a keyboard and mouse or augment the way we interact with digital information while offering an even greater, more expressive range of motion to the user in a more "hands-off" capacity. The term "hands-off" referring to the fact that the keyboard and mouse interface as well as the touchscreen interface both require the user to literally be touching a device of some sort in order to carry out a command. However, certain interface technologies were created with the intention of never having to place one's hands on a device.

With these ideas in mind, the project set out with the simple goal of developing an application for children by children using an interface with these specifications. The interface chosen was LeapMotion.

1.2 LeapMotion

The LeapMotion is a small device that can fit in the palm of your hand which sits in front of a monitor as seen in Figure 1.1 and captures motion in 3 cubic feet of space using a pair of cameras. The small cameras triangulate the positions of hands, fingers and tools in their relative space between the LeapMotion and the monitor relaying accurate position

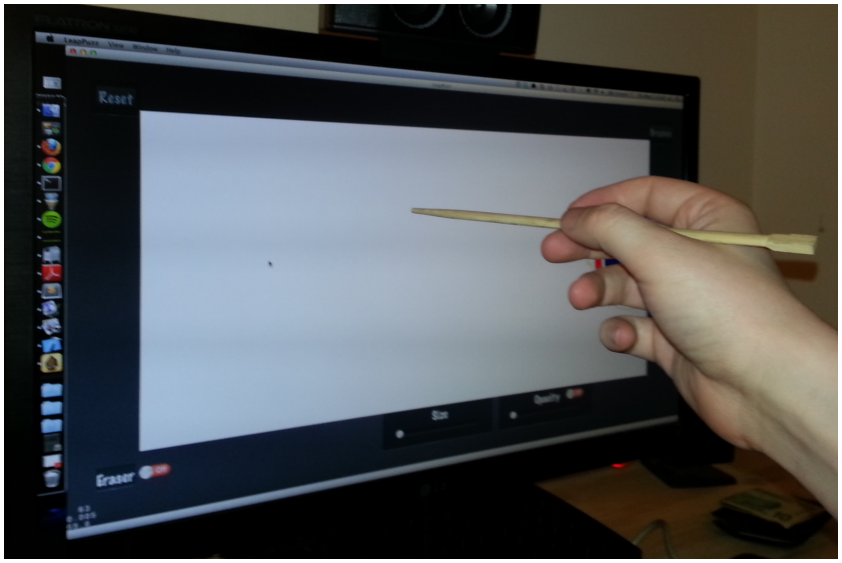


FIGURE 1.2: Using a chopstick as a pointer

and velocity data in real time. The data can then be used to control the application by driving the user interface of the system. [3]

This type of interaction with the computer system is different from the traditional keyboard and mouse because it does not require any physical contact with objects connected to the system and has the ability to sense a much wider range of input. The LeapMotion is also expressive in three dimensions as opposed to two.

In this paper and throughout the project we refer to a finger or tool interacting with LeapMotion as a "pointer[1]." The tool can be any object is "stick-like" such as a pen, pencil or chopstick. Throughout this project a chopstick was mostly used as seen in Figure 1.2 as the preferable tool. Initially it was intended for the application to track fingers although through development and testing it was found that tracking fingers was not as reliable as rigid tool. Tracking was less reliable for the children's smaller sporadically moved hands.

[1]The documentation refers to this as a "pointable"

1.3 Project Scope

Although we wanted to test if the keyboard and mouse could be replaced by the Leap-Motion, we recognized that there are some functions that would not be possible without such an interface as the keyboard and mouse. Tasks that require the user to type large amounts of information, such as data entry or word processing, are two such examples of exceptions we will not be focusing on in this project. The focus, therefore, would be on some of the other functions the keyboard may serve such as switching applications which can be performed by the keyboard shortcut ALT + TAB or with a mouse. The LeapMotion might be able to replace this functionality with a wave of a hand indicating to switch the applications automatically in carousel.

Chapter 2

Development Methodology

This project differed from traditional software development projects in that children participate in the process of designing the interactions and applications that were created. In pursuing and developing this research project, I worked with KidsTeam which is an intergenerational design team directed by Dr. Jerry Alan Fails where children and adult researchers work together as design partners in a collaborative and elaborative process. The children affiliated with this particular group participated in the application's design process. Because of this collaboration, the project's development process was different from the traditional projects in the way the phases were performed and required a very flexible model of development.

2.1 KidsTeam

The KidsTeam is a group of eight children ages 6 to 11, male and female, directed by Dr. Jerry Alan Fails at Montclair State University. The group meets twice a week for one and a half hour (4-5:30pm) sessions over the course of a semester. During that time, the KidsTeam worked on various projects which were facilitated by Dr. Fails along with several undergraduate and graduate students. The objective is to leverage the children's natural capacity for divergent thinking in a collaborative environment to develop an application for the LeapMotion. In order to achieve an optimal outcome, the professor

FIGURE 2.1: Reviewing the sticky notes using a spatial graph on a white board for easy comparison and analysis.

and students created a community of respect and rapport. The purpose is to create a safe space where the children felt free to explore and share their ideas. Therefore, it is imperative that the atmosphere remain relaxed. The atmosphere is purposefully not like school, and measures are taken to equalize the power structures that are generally inherent in adult-child relationships and instead build a relationship of mutual respect and trust, thus facilitating a fluid working relationship.

KidsTeam creates a dialog when designing and developing an application between the developers and the users by building off each others ideas. This intergenerational design team collaborates by exchanging ideas and giving opportunities to enhance every aspect of the application.

2.2 Collaboration Techniques

Several techniques were implemented in order to aid the development process. These techniques were chosen for their means to foster a community of cooperative inquiry where the children felt encouraged to contribute design ideas freely. [4][5]

2.2.1 Sticky Notes

Each child was given a pad of Post-It notes[1].During activities which required comment, as the children played with the cubes, the children would write comments on the PostIt notes[2]. These comments were categorized into like, dislike, or design idea and then stuck to the white board. A facilitator would then organize the notes based on the category and

[1]Small square of paper with adhesive on reverse side.
[2]One comment per sticky note.

the comment content to resemble a spatial graph where similar comments are grouped closer together, while outlying comments are spread farther apart. Comments relating to a specific function or component are arranged into the same row while the category of the comment will determine the column. At the end of the session the observer debriefs with the children about the session by reviewing the comments arranged on the white board and having the children comment about the session overall. This time allows for the observer and the children to summarize the session, gives the children extra time to comment and provide more specific feedback, and permits the observer to further clarify a child's reasoning for making certain comments. The result is a frequency analysis as seen in Figure 2.1 which feedback can be turned into specifications for the next design cycle. [6][7]

2.2.2 Bags of Stuff

Each child is given a Bag of Stuff that contains a variety of arts and crafts supplies such as Popsicle sticks, felt, construction paper, markers, etc. The children are then given a design concept by the observer and asked to use these materials to construct that concept. As the children build, they must explain in their own words how their concept works while the observer takes notes. [6][7]

2.2.3 Storyboarding

The children are split into groups. The groups were chosen at random and did not remain intact from session to session. Each group then receives one large piece of construction paper. They will use the paper, sticky notes, and markers to draw their particular game concept sequence of events. Any actions or changes in a scene must be drawn in the order that they occur. The children must use arrows to delineate the progression of events. In order to make sure their ideas are conveyed clearly, the children are asked to be as descriptive as possible while facilitators take notes as the children work. [6][7]

2.3 Development Model

Working with KidsTeam required a lot of flexibility in the development process and is why Rapid Application Development (RAD) model[3] for software development was chosen as the best fit model for this project. The RAD model uses rapid prototyping[4] and continuous iterative cycles which allowed for testing with KidsTeam on a weekly basis. Each session with the KidsTeam generated new requirements and fixes to be implemented within tight time constraints prior to the next session.

The model centered around five phases: Specification, Design, Code, Test and Review as seen in Figure 2.2 constituting a full cycle. Changes in the specification were then implemented and reflected in the application prototypes ready for the next session. Each session with KidsTeam marked the end of current cycle and start of the next in the development process. [8][9]

The children were most heavily involved in the specifications, design, test and review phases in each of the nine sessions. Some parts of the design and testing were done independently of the children but mostly cosisted of elaborating on either their designs and ideas or were performed to ensure the application was stable enough for testing with the children.

2.3.1 Specification

The specification is the set of requirements for the application's functional components and use cases. This includes what the application must be able to do and defines the parameters the application must operate within.

2.3.2 Design

The design takes the specification and frames a way to implement them in the next phase of coding. This includes how the user interface will be laid out, components will

[3] Iterative or incremental development process resembling an evolutionary pattern.
[4] Rapid Prototyping will produce a quick mockup for testing in each cycle.

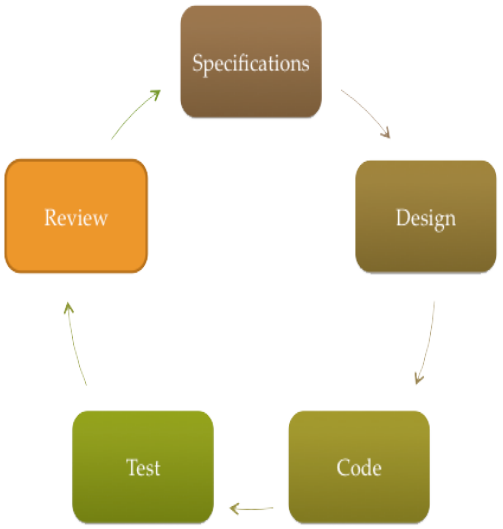


FIGURE 2.2: Rapid Application Development cycle [9]

function and the architecture of how each component will interface to become a working application that will fulfill the specification requirements.

2.3.3 Code

Coding is the process of taking the design and implementing it into functional units of code that run the application. The code will contain models of objects, actions, responders, delegates and flow control of application.

2.3.4 Test

Includes writing test cases for the functional components and debugging issues within the code such that the application meets the requirements in the specification. This process was done by the developer to ensure the application is stable enough for the children to perform fullstack testing of the application.

2.3.5 Review

After testing was performed by KidsTeam, the children provided direct feedback via the sticky notes collaboration technique. A frequency analysis of the sticky notes determined the next set of requirements for the specifications in the upcoming cycle.

2.4 Progress Updates

Throughout the course of the project updates were posted on a wiki and on youtube. Each wiki update summarized the session with the children including goals, observations and new requirements taken from the session along with pictures of the session. Videos posted on youtube mostly described major changes within the application itself and intermediate project updates.

Source code was kept in a public github repository[5] along with the project documentation. Any additional project documents were kept here.

[5]Due to the number of prototypes, the repositories were later consolidated into one repository.

Chapter 3

Design and Development

Throughout the project nine session were spent with KidsTeam. The initial sessions with the KidsTeam centered mostly around the introduction of the LeapMotion. The purpose was to help the children become comfortable with the technology through both brief explanation of how it works and independent exploration. It was important that the children spent time interacting with the technology as early on in the sessions as possible. The more comfortable the children were with using the LeapMotion, the more realistic their creations could be for the interface.

Later sessions focused on reviewing and improving the application. The application will not be able to make too many drastic design or architectural changes in these later phases.

3.1 Session 1: Introduction and Brainstorm

This first session with the children started with brainstorming ways to use a computer system or control an application without a keyboard and mouse and only using their hands. Using the Bags of Stuff 2.2.2 exercise the children explored designing applications that could only be controlled using gestures.

The children developed several application and game ideas including Pong, Fruit Ninja, Temple Run, Virtual Pet, Internet Explorer, Paint and Maps. The applications were accompanied by many controlling gestures such as a chopstick with button, waving, grabbing, pointing, dragging, pull and release[1], scratching and striking.

Afterward the children were allowed to play with a LeapMotion visualizer of the LeapMotion at the end of the session. The visualizer is 3d rendering of what the LeapMotion detects in realtime.

3.2 Session 2: Testing Breakout and Designing a Paint Application

The children tested a simple BreakOut game which allowed them to interact with the LeapMotion with an application and get a feeling of how it works. With that experience the children were then tasked with designing a painting application that can draw, choose colors and brushes only using gestures to control the interface of their application using the storyboarding 2.2.3 technique.

Along with the interface the children came up with several gestures in controlling a variety of user interface controls but most of the gesture controls relied on techniques similar to the way a mouse would function in picking up, dragging and dropping an icons on the screen. Other designs required an action to be performed while pointing at the targeted user interface control indicating a beginning action or ending action[2] instead of waving of a hand, drawing a figure eight in the air or a slashing motion.

3.3 Gesture Design Challenges

The immediate challenge that faced gestures requiring beginning and ending triggers is how to interpret when each trigger takes place to perform the action. In the mouse and touch screen interfaces the typical design paradigm consists of a beginning action,

[1] Similar to the bird launching interface in Angry Birds
[2] begin and end actions

(a) Open Thumb (b) Closed Thumb

FIGURE 3.1: Hand trigging actions

intermediate action and ending action respectively representing the state in which the input is currently in.

TABLE 3.1: Apple Mac OS X and iOS Application Programming Interface for standard methods of input [2]

Action	OS X	iOS
Begin	MouseDown	TouchesBegan
Moved	MouseDragged	TouchesMoved
End	MouseUp	TouchesEnded
Alternate	–	TouchesCancelled

With the LeapMotion there was no standard way of detecting when each action is to be performed. With this challenge the children came up with the concept of using two fingers to indicate when an action would be performed. Using the thumb as the control mechanism to indicate the action state to be performed and the index finger to indicate the targeted interface control to act upon, the interface control could triggered based on its functionality. The gesture consisted of pulling the thumb flush to the hand while pointing at the user interface control to begin the action and releasing the thumb to indicate the end of the action.With this simple system the a gesture could perform a BeginAction, MovedAction, and EndAction on user interface controls.

Use case include some of the following control operations

- Icon Movement pulling the thumb flush to the hand would trigger the BeginAction "pickup" the icon and start dragging it on the screen. The icon could then be placed anywhere on the screen until releasing icon and "dropping" at that position by moving the thumb to a relaxed position no longer flush with the hand performing the EndAction.

- Selecting a color. The BeginAction would present a popover that would remain open while the user pointed at the desired color until the EndAction. The EndAction would select the color that was pointed at when triggered.

- Radio Buttons. A quick BeginAction and EndAction trigger while pointing at the radio button would change the state of the selected item.

3.3.1 Implementation Challenges

In concept this idea seemed to be an ideal and natural way of interacting with the system but faced several challenges. When the thumb became flush with the hand the LeapMotion could no longer detect it as a finger. Additionally, there were several instances of occlusion occurring, for example when the hand was positioned with the thumb pointing upward the line of sight from the camera to the thumb was blocked. This unique challenge was difficult because the thumb could not be accounted for in all cases. An option might be to use the thumb's absence as a trigger although this is prone to many false positives[3] when the user's hands were on the boarder line of the LeapMotions visibility range or has lost track of the thumb due to occlusion[4]. Reversing the actions and performing the opposite gesture did not feel natural. Holding the thumb closely to the rest of the hand is not relaxed state and also did not mimic the act of grasping something or triggering an action as people are more commonly used to.

[3]False positive indicates a given condition is present when it is not.
[4]Occlusion occurs when the a surface is hidden by blocking the line of sight.

3.3.2 Unity 3D Engine Testing

Touching two or more fingers together rendered them not recognizable by the Leap-Motion and thus would not allow pinch gesture to be used. Alternatives might be calculating when two fingers are close to each other although the LeapMotion cannot differentiate the fingers into which is the index, middle, ring, pinky, and thumb fingers. The application would have to assume any two fingers were performing the gesture and would not allow for the natural separation between fingers unless all fingers were kept folded tight into the hand until needed.

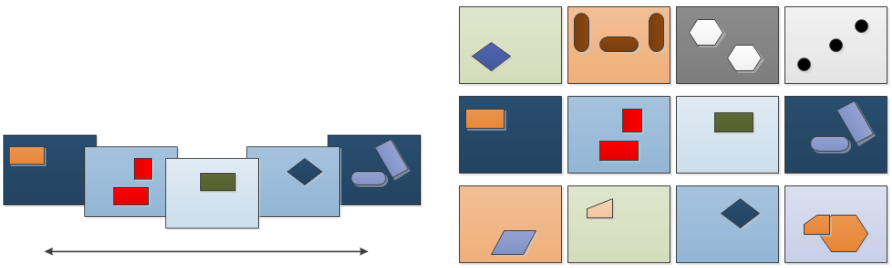
To observe this behavior a separate project was setup using the Unity 3D Engine to track the different pointers and their interactions in 3D space.[10]

It was clear after some lengthy prototyping and testing that some of the gestures developed by the children would not be possible and also raised some preliminary speculations that the LeapMotion may only be supplemental in its role except when in application specific situations. Different methods of interacting with these types of controls would need to be developed or would rely on the keyboard and mouse for their functionality.

3.4 Session 3: Interface Controls

Common user interface controls were printed out and the children were tasked with brainstorming how they might use LeapMotion to control the widgets or design their own widget that can reproduce the functionality. The user interface controls that we focused on included performing the following tasks were: buttons, drop downs, color palettes, check boxes, radio, buttons, sliders, toggles, steppers, wheels, trees and tabs.

The children did not come up with many new ways of interacting with the user interface controls or developing their own. The main method of interaction focused on pointing to the interface control triggering it using a hand signal with their fingers. The wheel controls could be controlled with flicking motions and dialogs could be dismissed with a wave indicating that the user is finished with the form. Selecting from a list could be done with a list in a carousel view style 3.4(a) where each of the items could be panned



(a) Carousel view style (b) Collection view style

FIGURE 3.2: Carousel view style maybe zoomed out using a pinch and pull gesture to a collection view.

through by waving or zoomed out into a collection view style 3.4(b) of items using pinch and pull gestures.

Observation of this session reinforced the suspicion that the LeapMotion may not be able to take a role as a dedicated device but as a supplemental device to the mouse an keyboard. Interface controls would also require more space between each of other and to be a of a larger size allowing for a margin of error. Feasible options may include iOS style of modal dialogs which take primary focus until the user in completed with the control.

3.5 Session 4: Testing a drawing application

Demonstration and hands on testing of a prototype drawing application developed based on the specifications the children had designed in the previous. The children provided feedback using the sticky notes exercise 2.2.1 review technique.

The one requirement that stood out the most early on was the necessity for a Heads Up Display (HUD) which would track where the a pointer is pointing on the screen and also display and interaction with the pointer. Additional features might include motion streaks or gesture animations to provide visual conformation of the action being performed or help the user track where the pointer has been.

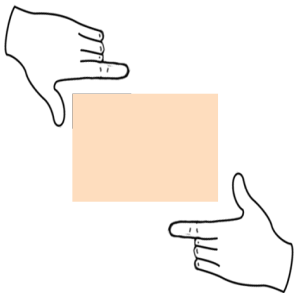


FIGURE 3.3: Indicating the size of a box by specifying two opposing corners with 'L' shape between the index and thumb fingers.

3.6 Session 5: Designing Gestures

The children were given some sample gestures and asked to design a way of using the gestures or generate their own for using Microsoft Paint application using the storyboarding 2.2.3 technique. The sample gestures were turning, tapping, swiping and hand wave. Gestures developed by the children were fairly consistent to previous session in pointing to a user interface control and selecting it to perform the action. In the group discussion an idea to use two fingers on two hands to draw the bounds of the box was generated. The finger tips could either represent each of the four corners of the box or two opposing corners by making an 'L' shape as seen in Figure 3.3.

This session reinforced the idea that the LeapMotion cannot be a dedicated device but a supplemental device.

3.7 Session 6: Testing Paint Prototype

The children were tasked with testing the painting prototype application dubbed Leap-Paint using the sticky notes exercise 2.2.1 review technique. This early prototype focused on adding a HUD to display the cursor and did not have all of the features in the initial requirements. Testing was focused on the interactions with the cursor. The While not testing they designed their own brushes, shapes and tools to be included in the next iteration of the tool. The overall consensus from this session was to implement better accuracy of the tool which required redesigning how the coordinate systems were translated from LeapMotion space into the application.

3.8 Session 7: Testing Improved Paint

The children were tasked with testing the painting application with the fixes based on the last session and providing feedback using the sticky notes 2.2.1 review technique. The new features focused on testing in the application were changing colors, erasing and an experimental method of triggering the drawing action.

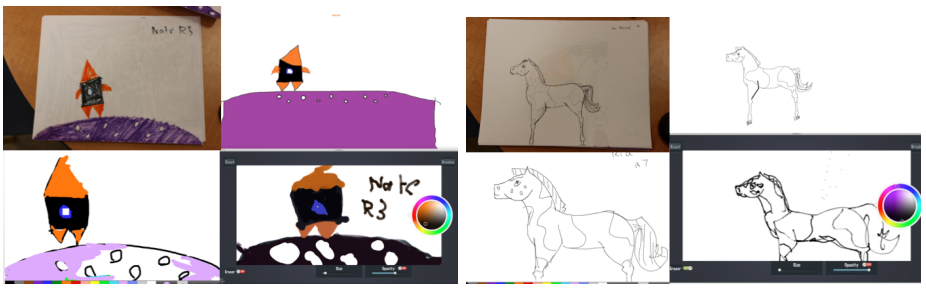
The two different drawing actions were tested with one using the space bar to indicate when to begin drawing. The other used the Z axis of the LeapMotion to begin drawing when breaking a certain plane of depth toward the screen. Children could switch between the modes by pressing the '1' key for space bar mode and the '2' for depth mode and compare each of them.

After testing the children noted that it was difficult to determine where the plane was that would start and stop the drawing actions using the depth mode. The space bar mode seemed to be the preferable option of the two.

To help the children determine whether they were drawing or not a ring indicator would later be added to show when drawing and not drawing around the cursor by changing from green to red. Additionally the depth mode's flat plane along the Z and X axis would have to calculated as a concave shape for better performance since painting near the edges of the drawing required moving the arm slightly further toward the screen then required in the center.

3.9 Session 8: Final Testing

The children were tasked with testing the painting application for the last time using the sticky notes 2.2.1 review technique. Testing for new features which included a depth opacity control. The opacity of the brush would become greater the closer the pointer was to the monitor simulating how a paintbrush or marker makes a darker line when pressed harder to paper.



(a) (b)

FIGURE 3.4: Drawings done by hand (top left), in Microsoft Paint (top right), on the iPad (bottom left) and with the LeapMotion (bottom right)

The children preferred the method of using depth to control the opacity over performing the manual adjustments with the mouse.

3.10 Session 9: Usability Comparison

The last session compared the LeapPaint application with a three different methods of drawing with different interfaces. The children were given the task of drawing the same picture of their choice on each system. Each child was given 10 minutes of time to attempt to reproduce a drawing by hand drawing using markers, Microsoft Paint using a mouse and keyboard, SimpleDraw on the iPad's touch interface and with LeapPaint using the LeapMotion.

The resulting drawings were mixed in comparison but showed that the drawing ability in the application may be related to the amount of time the child has had using the application. Drawing comparisons can be seen in Figure 3.4 by two different children.

Chapter 4

Application Architecture

The changing requirements caused architecture to change several times over the course of the project as components were first prototyped then tested.

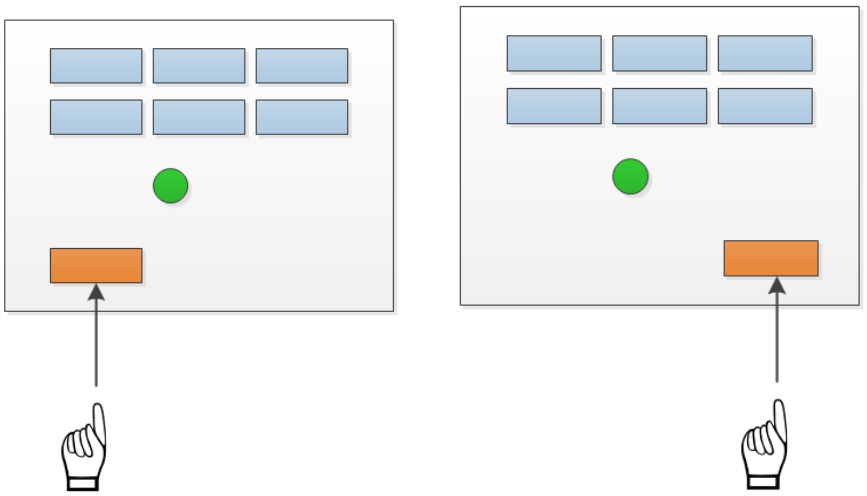
4.1 Prototyping

The initial prototypes included HelloWorld [1] application and simple game of BreakOut. HelloWorld and Breakout were both prototyped with the Cocos2d game engine because of the game engines ability to allow quick control over game objects. [11]

4.1.1 HelloWorld

The HelloWorld application tracked blocks across the screen using input data from the LeapMotion animating the interface. This application served as starting point for working with the LeapMotion SDK and also as a way of testing input received from the LeapMotion and resolving it to the coordinate space within the application. This code became a boiler plate interface to working with LeapMotion SDK later on in the project.

[1] HelloWorld is commonly a testbed to ensure the build environment and external libraries build correctly



(a) Left position (b) Right position

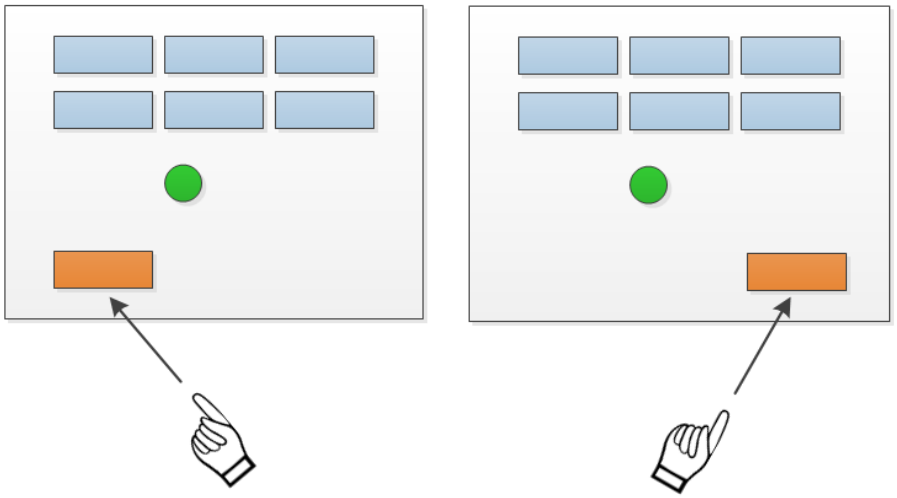
FIGURE 4.1: Left to right paddle control in breakout using the hand position relative to the screen

4.1.2 BreakOut

Furthering the HelloWorld application and testing the LeapMotion capabilities a simple game of BreakOut[2] was adapted for using the LeapMotion as the control mechanism for the paddle. This application was used in Session 2 3.2 with the children to show sample interactions of a complete system.

This prototype also showed through observation with the children how different methods of calculating the coordinates on the screen might be accomplished. The first method takes the position of the pointer in its coordinate space and translates it the coordinate space in the application as shown in Figure 4.1 despite the orientation of the pointer. The second method uses a vector pointing from the tip of the pointer and finds where that vector intersects with the screen as seen in Figure 4.2. Between the two methods the second method of finding the intersection on the screen appeared to be most natural.

[2]The breakout game was pre-built sample code[12]



(a) Left Position (b) Right position

FIGURE 4.2: Left to right paddle control in Breakout using the pointer intersection with the screen

4.1.3 Unity

One of the main gestures in the requirements used fingers to indicate beginning and ending actions as part of the gesture. Attempting to model these gestures required a 3D environment that could show multiple pointer interactions with game objects which was not possible with Cocos2d. The LeapMotion SDK provided a API with Unity 3D Game Engine which could perform the modeling required to begin recognizing the gestures. It was through this prototype that it was found that the LeapMotion would not be able to detect fingers touching each other[3]. [10]

4.1.4 Quartz 2D

The Cocos2d engine is not designed particularly for drawing and rendering textures to images. Apples Quartz 2D and Cocoa libraries are well suited for this task providing a large array of built in functionalities. These libraries were used to create the prototype drawing application using some of the boilerplate code from the earlier Helloworld 4.1.1

[3]A later released visualizer in the LeapMotion SDK would show the same result.

and Breakout 4.1.2 prototypes. The boilerplate code formed into a standard interface and coordinate system for working with the LeapMotion. [2]

This prototype worked well in testing in Session 4 3.5 and was generally well received by children although they had one major requirement of adding a cursor in addition to some minor features. The cursor would show where the pointer was on the screen at any given time so they could position it prior to painting. The minor features included selecting brushes, erasing, changing the brush size, changing the brush type and opacity.

4.1.4.1 Challenges

The cursor was very difficult to implement using the Quartz 2D and Cocoa because the libraries do not natively support layering and drawing simultaneously due to the way the rendering context functions as a single context. Creating independent views and transparent windows did not render correctly when attempting to simultaneously update each view. An alternative might be to put each functionality into separate applications running on different process threads although this was not possible because the LeapMotion can only be accessible from one process at a time. To create a cursor in HUD layer the application needed to access the LeapMotion, HUD and Drawing objects simultaneously. This was the defining factor in returning to the Cocos2d Game Library because it support independent layering of views. [2]

4.1.5 LeapPaint

The prototype dubbed "LeapPaint" by the children consisted of a combination components from the earlier Helloworld 4.1.1 and Breakout 4.1.2 prototypes managing input on a standard interface and output into different visible for drawing and the HUD. The layers were connected via a GameManager [4] passing the delegate actions between layers and interfaces.

[4] GameManager takes the role of the ApplicationDelegate

(a) Mock Up (b) Screenshot

FIGURE 4.3: Mock up compared to a screen shot of finished application with a drawing.

This prototype received the best reviews by the children in testing Session 6 3.7 despite lacking some of the features of available in the Quartz 2D 4.1.4. This architecture could be used going forward in adding features.

4.2 User Interface Layout

The interface layout of controls was based upon a combination of designs made by the children as seen in Figure 4.3 with the canvas to paint on centered and the user interface controls tucked to the sides of the canvas.

4.2.1 pointer Tracking

Initially a tracking system was design to use the relative coordinates of a pointer in the LeapMotion coordinate space and translate them to coordinates within the application. Later with the an SDK update the LeapMotion could provide coordinates on the screen where a vector from the pointers tip would intersect. This required some screen calibration to be performed such that the LeapMotion would be able to track points on the screen.

4.2.2 Application Layers

The application was broken into component layers to manage and modularize different functionalities as seen in Figure 4.4. This allowed for some components to become

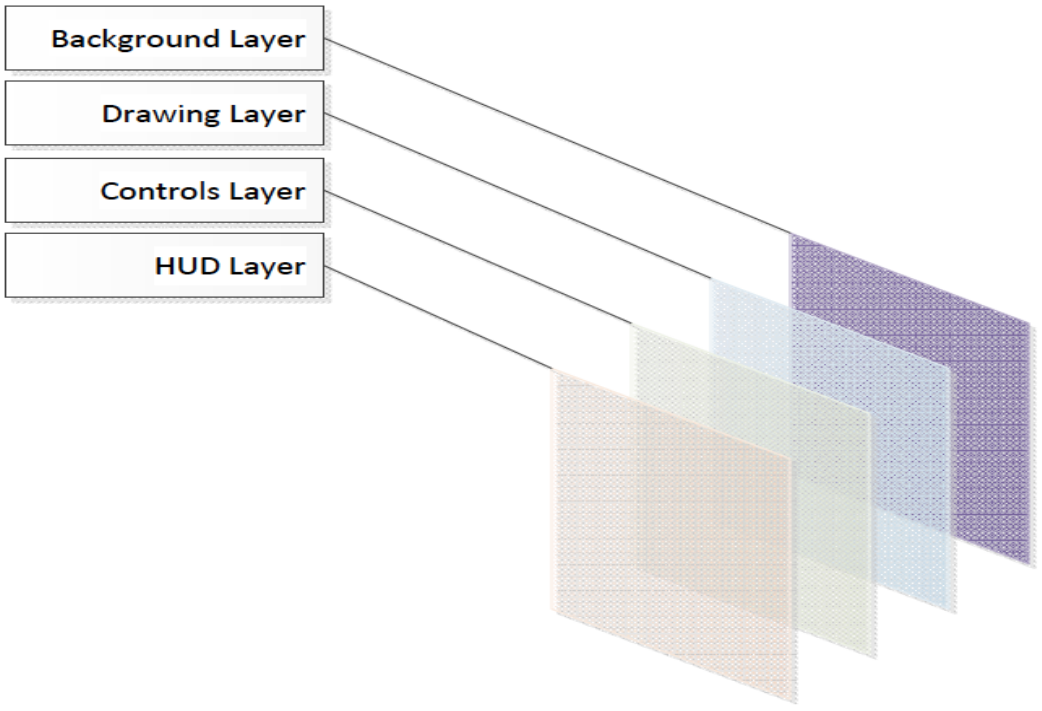


FIGURE 4.4: The ordered layers of visibility in the application.

reusable for other applications by providing a common application programming interface (API).

- Background Layer manages the background images and for the application. This practice is fairly standard and
- Drawing Layer is where the image will be edited and render.
- Controls Layer provides the user interface controls that for different aspects of the application.
- HUD Layer displays feedback information for the position of the pointer in relation to the screen and any different actions that might be taking place.

4.3 External Libraries

- LeapMotion SDK is the provided API for using with the LeapMotion.[3]

- Cocos2d is a game engine framework that allows simple animation and control of sprites and layers. [11]
- CCControlExtensions are user interface control elements built for the Cocos2d Engine. [13]
- Graphics Recognition Toolkit was used initially for doing gesture recognition with customized pipeline until a LeapMotion SDK update provided the same functionality. [14]

4.4 Testing

The project uses unit testing on all the class functions to verify their input and outputs correspond to their intended function. Using the simple methodology of making the test fail and then making the test pass with a variety of inputs and outputs helped modularize class functionality into smaller and more reusable sections. The testing framework OCUnit tests at class and bundle levels to produce full code test coverage. Performing this step on each function refined the design by directly questioning each sub components role and required functionality.[2]

Parts of the project unit testing could not cover were mostly involved at the interface level where the LeapMotion SDK provided data. Environmental factors effect the LeapMotions performance when in a different lighting conditions. The different types and positions of lighting caused erratic effects that often had to be compensated for when noticed.

4.5 Documentation

Code documentation is done with in-line comments and then automatically generated with Doxygen. This was an essential tool due to the ever changing state of the project during each cycle that the project be able to automatically reflect changes in the design.

4.6 Experimental Features

Several experimental interface designs were added for the children to test with since there were often more than one design approach to a features in the application. The approached attempted to leverage the 3D space available to the LeapMotion that is not available to the keyboard and mouse.

4.6.1 Drawing Modes

Two different drawing modes were tested with the children in Session 7 3.8. The first was a mode where the pointer would begin drawing when crossing a boundary on the Z axis toward the screen. The cursor could then be moved about the screen without interacting with the drawing by pulling the pointer back and then pushing forward when ready to begin drawing. A ring around the cursor icon would indicate weather or not the cursor would begin drawing based on the depth of pointer. The ring would change from red when not in not drawing state to green when beginning to draw. Further development might include a yellow color ring indicator when approaching the threshold to transition states.

The children did not like using the depth mode option of drawing compared to the spacebar mode bar of drawing as they had trouble becoming accustom to the threshold in which the application would begin and stop drawing. They did like that they could draw and change colors at the same time by using their free hand with the mouse to create continuous rainbow effects with the brush.

The second drawing mode was to begin drawing when pressing the space bar on the keyboard and disregarding the depth of the pointer. This proved to be the favorite method of input for the children to indicate their actions.

4.6.2 Depth Opacity

Another option explored was using the Z axis to control the opacity of the brush. The intent was to provide a pressure sensitive brush which would mimic drawing implements

in the real world where pressing harder on a paint brush or marker will draw a darker or thicker line.

The children did like this feature although and preferred it to manually adjusting the opacity with the mouse. This was Dependant on what they were drawing where it was preferable for background colors but not drawing lines. The feature could be improved with adding some brush stroke effects that vary based on the speed of the pointer when drawing.

Chapter 5

Summary

5.1 General Observations

The children would first sketch an outline of their drawing and then attempt to color in their outlines. This proved difficult in the case of the LeapMotion. Drawing the lines proved fairly easy for the children while shading in sections appeared more difficult. I found the opposite to be true of my own experience as the lines were harder to draw than shading in the areas.

5.2 Similar Applications

Comparison to industry competitors Corel's Painter Freestyle which will have many of the same features. In terms of interface design they chose a similar layout of control mechanisms. We haven't be able to to see this all quiet yet since it has not been released to perform a full comparison although from the initial details given on their website seems to lead that their application could be similar to ours in many respects. [**?**]

5.2.1 Google Earth

Examples of dedication are shown in some example applications where the LeapMotion has a specific purpose. Google Earth uses it for navigation only allowing the user to pan, rotate and elevated the camera in relation to the earth. Interpreting the hand motions as the path of airplane as the control mechanism with yaw, pitch, roll, bank and elevation.

The way of controlling the Google Earth is similar to the idea of controlling the Temple Run game brainstormed in from Session 1 3.1. This connection was not apparent when first considering the idea for controlling the game.

5.3 Conclusions

The LeapMotion is a great device for capturing the motions and positions of the hand in real time but is tough to consider as replacement for the keyboard and mouse. It is foreseeable that the main application for the LeapMotion will not be as a general use device but for specific applications which enable expressive movement. Games, drawing and music applications are good examples of where the LeapMotion can focus on specialized actions. Integration into general purpose applications will be difficult due to existing designs and interface paradigms.

Appendix A

LeapPaint

Contents

1 Main Page

`Project Home & Wiki`

#Requirements Specification

Interface

- HUD Requirement to render a cursor where the pointable is intersecting with the screen. The cursor should show the color that will be painting on the screen

- Ring and round cursor to indicate drawing or not drawing.

Features

- Change Colors
- Change Brushes
- Eraser
- Change size of brush
- Reset drawing
- Change Opacity of brushes

#Unit Tests

#Libraries & Sub Modules

- Cocos2d 2.0
- CCControlExtension
- #Build Settings

- Valid Architecture i386 x86_64
- Other Linker Flags -lz -ObjC
- C Language Dialect GNU99 -std=gnu99
- C ++ Language Dialect GNU++11 -std=gnu++11
- C ++ Standard Library libc++ (LLVM C++ standard lib)

- run script after build:

  echo TARGET_BUILD_DIR=${TARGET_BUILD_DIR} echo TARGET_NAME=${TARGET_NAME} cd ${TARGET_BUILD_DIR}/${TARGET_NAME}.app/Contents/MacOS ls -la install_name_tool -change /libLeap.dylib /../Resources/libLeap.dylib ${TARGET_NAME}

#Documentation

Documentation is done using Doxygen

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

CCLayer

3 Class Index

3.1 Class List

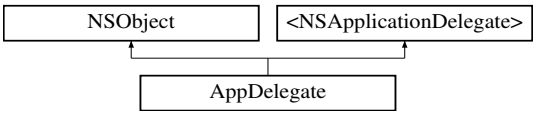
Here are the classes, structs, unions and interfaces with brief descriptions:

4 Class Documentation

4.1 AppDelegate Class Reference

`#import <AppDelegate.h>`

Inheritance diagram for AppDelegate:



Instance Methods

- (void) - runGameScene
- (IBAction) - toggleFullScreen:

Protected Attributes

- NSWindow * window_
- CCGLView * glView_

Properties

- IBOutlet NSWindow * window
- IBOutlet CCGLView * glView

4.1.1 Detailed Description

Application Delegate Creates app instance and binds libraries to interface builder xibs Serves as an application wide callback object for events that affects the whole application, such as low-memory, etc.

4.1.2 Method Documentation

4.1.2.1 - (void) runGameScene

RunGameSceen sets up the Cocos2d environment and runs it in the application.

4.1.2.2 - (IBAction) toggleFullScreen: (id) *sender*

Toggles from a window to full screen view point

Parameters

sender	is the action sending the command

Returns

IBAction binding to interface builder

4.1.3 Member Data Documentation

4.1.3.1 - (CCGLView*) glView_ [protected]

glView is the embedded view in which cocos2d will run inside the window

Referenced by runGameScene.

4.1.3.2 - (NSWindow*) window_ [protected]

window is the main window to be displayed

Referenced by runGameScene.

4.1.4 Property Documentation

4.1.4.1 - (IBOutlet CCGLView*) glView [read], [write], [atomic], [strong]

glView is the embedded view in which cocos2d will run inside the window

4.1.4.2 - (IBOutlet NSWindow*) window [read], [write], [atomic], [strong]

window is the main window to be displayed

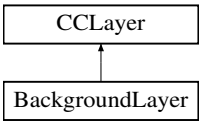
The documentation for this class was generated from the following files:

- LeapPaint/AppDelegate.h
- LeapPaint/AppDelegate.m

4.2 BackgroundLayer Class Reference

`#import <BackgroundLayer.h>`

Inheritance diagram for BackgroundLayer:



4.2.1 Detailed Description

Background Layer Displays a background image for the scene

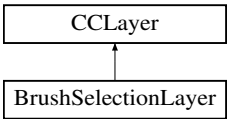
The documentation for this class was generated from the following file:

- LeapPaint/BackgroundLayer.h

4.3 BrushSelectionLayer Class Reference

`#import <BrushSelectionLayer.h>`

Inheritance diagram for BrushSelectionLayer:



Protected Attributes

- NSMutableDictionary * imageNamesDictionary

Properties

- id< BrushSelectionLayerDelegate > delegate
- bool layerHidden

4.3.1 Detailed Description

BrushSelectionLayer This user interface layer provides a collection view of all the available brushes that can be selected.

4.3.2 Member Data Documentation

4.3.2.1 - (NSMutableDictionary*) imageNamesDictionary [protected]

imageNamesDictionary is the list of brush names available for selection

4.3.3 Property Documentation

4.3.3.1 - (id<BrushSelectionLayerDelegate>) delegate [read], [write], [nonatomic], [weak]

delegate is the instance reference for triggering delegate call back functions

4.3.3.2 - (bool) layerHidden [read], [write], [nonatomic], [assign]

layerHidded tracks the visibility state of the layer

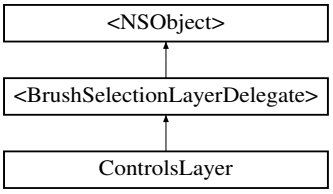
The documentation for this class was generated from the following file:

- LeapPaint/BrushSelectionLayer.h

4.4 <BrushSelectionLayerDelegate> Protocol Reference

`#import <BrushSelectionLayer.h>`

Inheritance diagram for <BrushSelectionLayerDelegate>:



Instance Methods

- (void) - hidePanel
- (void) - brushSelected:

4.4.1 Detailed Description

BrushSelectionLayer Delegate Provides a delegate interface for the layer to notify of actions

4.4.2 Method Documentation

4.4.2.1 - (void) brushSelected: (NSString *) *brushname*

Calls back to notify a new brushname has been selected

Parameters

brushname	is the name of the brush that has been selected.

4.4.2.2 - (void) hidePanel

Calls back to notify that the layer can be hidden

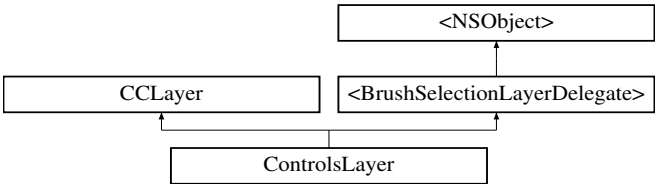
The documentation for this protocol was generated from the following file:

- LeapPaint/BrushSelectionLayer.h

4.5 ControlsLayer Class Reference

`#import <ControlsLayer.h>`

Inheritance diagram for ControlsLayer:



Instance Methods

- (void) - valueChanged:
- (void) - opacitySliderChanged:
- (void) - expandPanel
- (void) - collapsePanel
- (CCControlSwitch *) - makeControlSwitch
- (void) - switchValueChanged:
- (void) - updateOpacitySlider:

Protected Attributes

- CCLabelTTF * colorLabel
- GameSettings * gameSettings

Properties

- CCControlSlider * slider
- CCControlSlider * opacitySlider
- CCControlSwitch * opacitySwitchControl
- CCLabelTTF * opacitydisplayValueLabel
- id< ControlsLayerDelegate > delegate
- BrushSelectionLayer * brushSelection
- CCLabelTTF * displayValueLabel
- CCControlSwitch * switchControl

4.5.1 Detailed Description

Controls Layer User inferface controls for operating buttons, switches, sliders

4.5.2 Method Documentation

4.5.2.1 - (void) collapsePanel

Collapses Brushes Panel

4.5.2.2 - (void) expandPanel

Expands brushes panel

4.5.2.3 - (CCControlSwitch ∗) makeControlSwitch

Creates and returns a new CCControlSwitch.

Returns

a generate ControlSwitch

4.5.2.4 - (void) opacitySliderChanged: (CCControlSlider ∗) *sender*

Recieves opacitySliderControl delegate callbacks and updates values in the interface

Parameters

sender	is the object performing the callback

4.5.2.5 - (void) switchValueChanged: (CCControlSwitch ∗) *sender*

Callback for the change value.

Parameters

sender	is the object performing the callback

4.5.2.6 - (void) updateOpacitySlider: (float) *value*

Callback for opacity changing with the slider

Parameters

sender	is the object performing the callback

4.5.2.7 - (void) valueChanged: (CCControlSlider ∗) *sender*

Recieves brushSizeControl delegate callbacks and updates values in the interface

Parameters

sender	is the object performing the callback

4.5.3 Member Data Documentation

4.5.3.1 - (CCLabelTTF*) colorLabel [protected]

colorLabel displays name of color in hash value

4.5.3.2 - (GameSettings*) gameSettings [protected]

gameSettings global reference to shared settings instance

4.5.4 Property Documentation

4.5.4.1 - (BrushSelectionLayer*) brushSelection [read], [write], [nonatomic], [strong]

brushSelection layer expands as a drawer to allow for brush selection

4.5.4.2 - (id<ControlsLayerDelegate>) delegate [read], [write], [nonatomic], [weak]

delegate is the instance reference for triggering delegate call back functions

Referenced by GameScene::scene.

4.5.4.3 - (CCLabelTTF *) displayValueLabel [read], [write], [nonatomic], [strong]

displayValueLabel displays coordinate

displayValueLabel displays eraser toggle state

Referenced by switchValueChanged:.

4.5.4.4 - (CCLabelTTF*) opacitydisplayValueLabel [read], [write], [nonatomic], [strong]

opacitydisplayValueLabel shows the state of the opacitySwitchControl

4.5.4.5 - (CCControlSlider*) opacitySlider [read], [write], [nonatomic], [strong]

opacitySlider is the opacity contro of the brush

Referenced by updateOpacitySlider:.

4.5.4.6 - (CCControlSwitch*) opacitySwitchControl [read], [write], [nonatomic], [strong]

opacitySwitchControl is the control for setting automatic or manual opacity control

4.5.4.7 - (CCControlSlider*) slider [read], [write], [nonatomic], [strong]

slider is the thickness control of the brush

4.5.4.8 - (CCControlSwitch*) switchControl [read], [write], [nonatomic], [strong]

switchControl is the eraser toggle

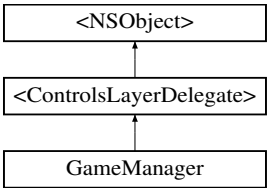
The documentation for this class was generated from the following files:

- LeapPaint/ControlsLayer.h
- LeapPaint/ControlsLayer.mm

4.6 <ControlsLayerDelegate> Protocol Reference

```
#import <ControlsLayer.h>
```

Inheritance diagram for <ControlsLayerDelegate>:



Instance Methods

- (void) - changeColorControl:
- (void) - changeThicknessControl:
- (void) - changeBrushControl:
- (void) - changeOpacityControl:
- (void) - clearDrawing
- (void) - eraserMode:

4.6.1 Detailed Description

Controls Layer Delegate Provides a delegate interface for the layer to notify of actions

4.6.2 Method Documentation

4.6.2.1 - (void) changeBrushControl: (NSString *) *brushname*

Callback with a change in brush texture

Parameters

brushname	is the new selected brush value

4.6.2.2 - (void) changeColorControl: (ccColor3B) *color*

Callback with a change in color of the brush

Parameters

color	is the new selected color value

4.6.2.3 - (void) changeOpacityControl: (float) *value*

Callback with a change in opacity

Parameters

value	is the new selected opacity value

4.6.2.4 - (void) changeThicknessControl: (float) *value*

Callback with a change in thickness of the brush

Parameters

value	is the new selected color value

4.6.2.5 - (void) clearDrawing

Callback to notify to clear the drawing

4.6.2.6 - (void) eraserMode: (bool) *mode*

Callback with a change in color

Parameters

mode	is the toggled eraser mode TODO: Turn off eraser mode when new color is selected

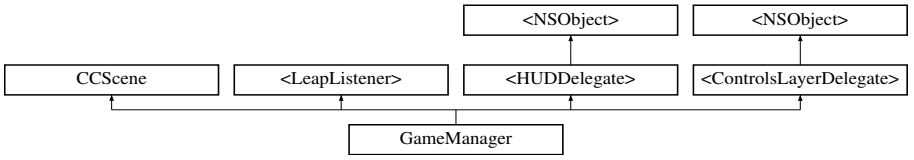
The documentation for this protocol was generated from the following file:

- LeapPaint/ControlsLayer.h

4.7 GameManager Class Reference

`#import <GameManager.h>`

Inheritance diagram for GameManager:



Instance Methods

- (float) - findPecentageDifference:withMin:withValue:
- (float) - opacityPercentage:

Protected Attributes

- InputMode inputMode
- LeapPointable * currentPointable
- CGPoint currentPoint
- BOOL painting
- GameSettings * gameSettings
- int lastTag
- SimplePoint * lastPoint
- int framesSinceLastFound

Properties

- HUDLayer * hudLayer
- SketchRenderTextureScene * textureScene
- BackgroundLayer * backgroundLayer
- ControlsLayer * controlsLayer
- LeapController * controller
- LeapScreen * leapScreen

4.7.1 Detailed Description

Core Application Management Provides interfaces and controls the various inputs, controls and outputs

4.7.2 Method Documentation

4.7.2.1 - (float) findPecentageDifference: (float) *max* withMin:(float) *min* withValue:(float) *value*

Finds the percentage of a number between two values If the number is greater or less than the range, that boundry of the range will be returned.

Parameters

max	is the top range value
min	is the bottom range value
value	is the number we are seeking the percentage from

Returns

the a percentage between 0 and 100%

Find the percentage between two numbers

Referenced by opacityPercentage:.

4.7.2.2 - (float) opacityPercentage: (float) *value*

Determines the opacity based upon the Z axis coordinate

Parameters

value	is the Z axis coordinate

Returns

the opacity value to set the brush at.

Return the Opacity value based on Z position

4.7.3 Member Data Documentation

4.7.3.1 - (CGPoint) currentPoint [protected]

colorLabel displays name of color in hash value

4.7.3.2 - (LeapPointable*) currentPointable [protected]

colorLabel displays name of color in hash value

4.7.3.3 - (int) framesSinceLastFound [protected]

framesSinceLastFound number of frames since last finding a LeapPointable

4.7.3.4 - (GameSettings*) gameSettings [protected]

gameSettings singleton to global seetings

4.7.3.5 - (InputMode) inputMode [protected]

colorLabel displays name of color in hash value

4.7.3.6 - (SimplePoint*) lastPoint [protected]

lastPoint is the last known point on the screen of the LeapPointable

4.7.3.7 - (int) lastTag [protected]

lastTag is the last tag value tracked of a LeapPointable

4.7.3.8 - (BOOL) painting [protected]

painting indicates wether or not the application is painting at that moment

4.7.4 Property Documentation

4.7.4.1 - (BackgroundLayer*) backgroundLayer [read], [write], [nonatomic], [strong]

backgroundLayer is the layer for setting up the background
Referenced by GameScene::scene.

4.7.4.2 - (LeapController*) controller [read], [write], [nonatomic], [strong]

controller is the leapController

4.7.4.3 - (ControlsLayer*) controlsLayer [read], [write], [nonatomic], [strong]

controlsLayer is the layer for managing interface controls
Referenced by GameScene::scene.

4.7.4.4 - (HUDLayer*) hudLayer [read], [write], [nonatomic], [strong]

hudLayer displays the icons for tracking where a leapPointable is pointing
Referenced by GameScene::scene.

4.7.4.5 - (LeapScreen*) leapScreen [read], [write], [nonatomic], [strong]

leapScreen references the screen being used on the system

4.7.4.6 - (SketchRenderTextureScene*) textureScene [read], [write], [nonatomic], [strong]

textureScene is the drawing layer

Referenced by GameScene::scene.

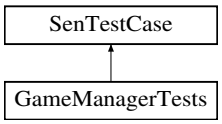
The documentation for this class was generated from the following files:

- LeapPaint/GameManager.h
- LeapPaint/GameManager.mm

4.8 GameManagerTests Class Reference

`#import <GameManagerTests.h>`

Inheritance diagram for GameManagerTests:



Protected Attributes

- NSString * testName
- GameManager * node

4.8.1 Detailed Description

Tests the GameManager object

4.8.2 Member Data Documentation

4.8.2.1 - (GameManager*) node [protected]

gameManager instance

4.8.2.2 - (NSString*) testName [protected]

testName is the name of the test

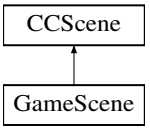
The documentation for this class was generated from the following file:

- LeapPaintTests/GameManagerTests.h

4.9 GameScene Class Reference

`#import <GameScene.h>`

Inheritance diagram for GameScene:



Class Methods

- (CCScene ∗) + scene

4.9.1 Detailed Description

GameScene Initializes and assembles all of the layers and gameobjects into the GameManager

4.9.2 Method Documentation

4.9.2.1 + (CCScene ∗) scene

Scene initializes each object and assigns interlinking pointers and delegates to each class

Returns

scene for CCDirector to begin running

Referenced by AppDelegate::runGameScene.

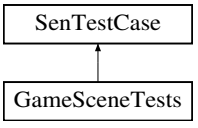
The documentation for this class was generated from the following files:

- LeapPaint/GameScene.h
- LeapPaint/GameScene.mm

4.10 GameSceneTests Class Reference

`#import <GameSceneTests.h>`

Inheritance diagram for GameSceneTests:



Protected Attributes

- NSString ∗ testName

4.10.1 Detailed Description

Tests the GameScene object

4.10.2 Member Data Documentation

4.10.2.1 - (NSString*) testName [protected]

testName is the name of the test

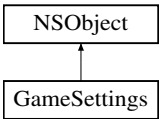
The documentation for this class was generated from the following file:

- LeapPaintTests/GameSceneTests.h

4.11 GameSettings Class Reference

`#import <GameSettings.h>`

Inheritance diagram for GameSettings:



Class Methods

- (GameSettings *) + sharedInstance

Properties

- BOOL depthOpacityMode
- BOOL painting
- BOOL eraserMode
- InputMode inputMode

4.11.1 Detailed Description

GameSettings is a globally shared class instance which tracks all the game settings. This class can be accessed by any object in the game.

4.11.2 Method Documentation

4.11.2.1 + (GameSettings *) sharedInstance

Singleton Intiailizes and Returns a shared instance of the class

Returns

sharedInstance of the class.

Singleton SharedInstance Intiailizes and Returns a shared instance of the class

4.11.3 Property Documentation

4.11.3.1 - (BOOL) depthOpacityMode [read], [write], [nonatomic], [assign]

depthOpacityMode controls use of z axis control of opacity

4.11.3.2 - (BOOL) eraserMode [read], [write], [nonatomic], [assign]

eraserMode controls erasing on drawing canvas

4.11.3.3 - (InputMode) inputMode [read], [write], [nonatomic], [assign]

inputMode controller input mode for leapmotion

4.11.3.4 - (BOOL) painting [read], [write], [nonatomic], [assign]

painting indicates wether or not the application is painting at that moment

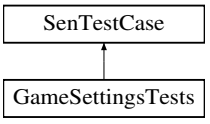
The documentation for this class was generated from the following files:

- LeapPaint/GameSettings.h
- LeapPaint/GameSettings.mm

4.12 GameSettingsTests Class Reference

`#import <GameSettingsTests.h>`

Inheritance diagram for GameSettingsTests:



Protected Attributes

- GameSettings * gameSettings

4.12.1 Detailed Description

Tests the GameSettings object

4.12.2 Member Data Documentation

4.12.2.1 - (GameSettings*) gameSettings [protected]

gameSettings singleton instance

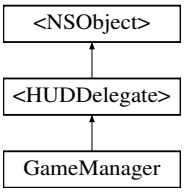
The documentation for this class was generated from the following file:

- LeapPaintTests/GameSettingsTests.h

4.13 <HUDDelegate> Protocol Reference

`#import <HUDLayer.h>`

Inheritance diagram for <HUDDelegate>:



Instance Methods

- (void) - changeMode:
- (void) - painting:

4.13.1 Detailed Description

HUD Delegate Protocol User inferface controls for operating buttons, switches, sliders

4.13.2 Method Documentation

4.13.2.1 - (void) changeMode: (InputMode) *mode*

Calls back to notify a new input mode has been selected by the keyboard interface

Parameters

mode	is the state of the input mode

4.13.2.2 - (void) painting: (BOOL) *paintingState*

Calls back to notify a new change in painting state

Parameters

paintingState	

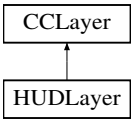
The documentation for this protocol was generated from the following file:

- LeapPaint/HUDLayer.h

4.14 HUDLayer Class Reference

`#import <HUDLayer.h>`

Inheritance diagram for HUDLayer:



Instance Methods

- (void) - toolMoved:toolID:
- (void) - startTrackingTool:toolID:
- (void) - moveTrackingTool:toolID:
- (void) - endTrackingTool
- (void) - **changeColor:**
- (void) - **changeBrush:**
- (void) - **changeScale:**
- (void) - **erasingMode:**

Protected Attributes

- NSString * primaryToolID
- LPTool * primaryTool
- InputMode inputMode
- ccColor3B lastColor
- ccColor3B previousColor
- NSString * lastBrush
- float lastScale
- CCSprite * paintingIndicator
- BOOL eraseMode
- GameSettings * gameSettings

Properties

- id< HUDDelegate > delegate
- CCLabelTTF * xyzcoords

4.14.1 Detailed Description

HUD Layer Tracks the position of the LeapCursor on the screen

4.14.2 Method Documentation

4.14.2.1 - (void) endTrackingTool

EndTracking tool singles the end of the tool being tracked. The tool may be lost or no longer drawing

4.14.2.2 - (void) moveTrackingTool: (CGPoint) *point* toolID:(NSString∗) *toolid*

MoveTrackingTool updates the position and path of a tool.

Parameters

point	is the coordinate location on the screen in which pointable interesects
toolid	is LeapSDK provided tool id of the tool moving

Referenced by toolMoved:toolID:.

4.14.2.3 - (void) startTrackingTool: (CGPoint) *point* toolID:(NSString∗) *toolid*

StartTrackingTool begins the process of tracking a tool starting with a new path

Parameters

point	is the coordinate location on the screen in which pointable interesects
toolid	is LeapSDK provided tool id of the tool moving

Referenced by toolMoved:toolID:.

4.14.2.4 - (void) toolMoved: (CGPoint) *point* toolID:(NSString∗) *toolid*

ToolMoved updates the last known tracked position of the tool.

Parameters

point	is the coordinate location on the screen in which pointable interesects
toolid	is LeapSDK provided tool id of the tool moving

4.14.3 Member Data Documentation

4.14.3.1 - (BOOL) eraseMode [protected]

eraseMode determines weather the pointable is painting or erasing

4.14.3.2 - (GameSettings∗) gameSettings [protected]

gameSettings singleton to global seetings

4.14.3.3 - (InputMode) inputMode [protected]

inputMode is the current mode of input

4.14.3.4 - (NSString∗) lastBrush [protected]

lastBrush is last brush to be selected

4.14.3.5 - (ccColor3B) lastColor [protected]

lastColor is the lastColor to be selected

4.14.3.6 - (float) lastScale [protected]

lastScale is last scale to be selected

4.14.3.7 - (CCSprite*) paintingIndicator [protected]

paintingIndicator shows the state at which the object is currently paintg

4.14.3.8 - (ccColor3B) previousColor [protected]

previousColor is the color before the lastcolor to be selected

4.14.3.9 - (LPTool*) primaryTool [protected]

primaryTool points to the current pointable object

Referenced by endTrackingTool, moveTrackingTool:toolID:, startTrackingTool:toolID:, and toolMoved:toolID:.

4.14.3.10 - (NSString*) primaryToolID [protected]

primaryToolID stores the id tag to the pointable in reference

4.14.4 Property Documentation

4.14.4.1 - (id<HUDDelegate>) delegate [read], [write], [nonatomic], [weak]

colorLabel displays name of color in hash value

Referenced by GameScene::scene.

4.14.4.2 - (CCLabelTTF*) xyzcoords [read], [write], [nonatomic], [strong]

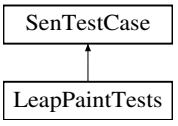
xyzcoords is the X,Y,Z coordinates in string form for displaying on the HUD in real-time for debugging

The documentation for this class was generated from the following files:

- LeapPaint/HUDLayer.h
- LeapPaint/HUDLayer.mm

4.15 LeapPaintTests Class Reference

Inheritance diagram for LeapPaintTests:



Protected Attributes

- NSString * testName

4.15.1 Member Data Documentation

4.15.1.1 - (NSString*) testName [protected]

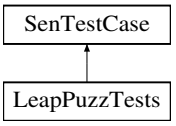
testName is the name of the test

The documentation for this class was generated from the following file:

- LeapPaintTests/LeapPaintTests.h

4.16 LeapPuzzTests Class Reference

Inheritance diagram for LeapPuzzTests:



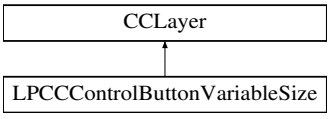
The documentation for this class was generated from the following file:

- LeapPaintTests/LeapPuzzTests.h

4.17 LPCCControlButtonVariableSize Class Reference

```
#import <LPCCControlButtonVariableSize.h>
```

Inheritance diagram for LPCCControlButtonVariableSize:



Instance Methods

- (CCControlButton ∗) - standardButtonWithTitle:

4.17.1 Detailed Description

LPCCControlButtonVariableSize Extends CCLayer to have a customizable control button interface

4.17.2 Method Documentation

4.17.2.1 - (CCControlButton ∗) standardButtonWithTitle: (NSString ∗) *title*

Creates and return a button with a default background and title color. Creates and return a button with a default background and title color.

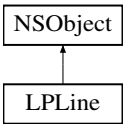
The documentation for this class was generated from the following files:

- LeapPaint/LPCCControlButtonVariableSize.h
- LeapPaint/LPCCControlButtonVariableSize.m

4.18 LPLine Class Reference

`#import <LPLine.h>`

Inheritance diagram for LPLine:



Properties

- NSMutableArray * points
- float width

4.18.1 Detailed Description

LPLine is tracks the points in one line from beginning to end

4.18.2 Property Documentation

4.18.2.1 - (NSMutableArray*) points `[read], [write], [nonatomic], [strong]`

points is a an array of points for the line

4.18.2.2 - (float) width `[read], [write], [nonatomic], [assign]`

width is a constant width for the line

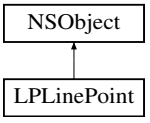
The documentation for this class was generated from the following file:

- LeapPaint/LPLine.h

4.19 LPLinePoint Class Reference

`#import <LPLinePoint.h>`

Inheritance diagram for LPLinePoint:



Instance Methods

- (id) - initWithPosition:

- (id) - initWithX:withY:
- (id) - initWithPosition:withWidth:
- (id) - initWithX:withY:withWidth:
- (CGPoint) - point

Properties

- float x
- float y
- float width

4.19.1 Detailed Description

LPLinePoint is a plotted point for drawing onto the canvas

4.19.2 Method Documentation

4.19.2.1 - (id) initWithPosition: (CGPoint) *p*

Init constructor with existing point to create with no width

Parameters

p	an point (x,y)

Returns

object instance

init 2d point with CGPoint

4.19.2.2 - (id) initWithPosition: (CGPoint) *p* withWidth:(float) *wVal*

Init constructor with existing point with width

Parameters

p	a point (x,y)
wVal	width of the point

Returns

object instance

Init point with CGPoint and width Value

4.19.2.3 - (id) initWithX: (float) *xVal* withY:(float) *yVal*

Init constructor with x and y values with no width

Parameters

xVal	coordinate value
yVal	coordinate value

Returns

object instance

Init Point with 2 separate values

4.19.2.4 - (id) initWithX: (float) *xVal* withY:(float) *yVal* withWidth:(float) *wVal*

Init constructor with x and y values with width

Parameters

xVal	coordinate value
yVal	coordinate value
wVal	width of the point

Returns

object instance

Init Point with x and y values with width

4.19.2.5 - (CGPoint) point

Returns point based on x and y

Returns

CGPoint

Return the CGPoint type from the object

4.19.3 Property Documentation

4.19.3.1 - (float) width [read], [write], [nonatomic], [assign]

width of the point

4.19.3.2 - (float) x [read], [write], [nonatomic], [assign]

x coordinate

Referenced by point.

4.19.3.3 - (float) y [read], [write], [nonatomic], [assign]

y coordinate

Referenced by point.

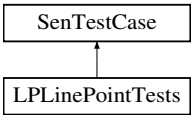
The documentation for this class was generated from the following files:

- LeapPaint/LPLinePoint.h
- LeapPaint/LPLinePoint.m

4.20 LPLinePointTests Class Reference

`#import <LPLinePointTests.h>`

Inheritance diagram for LPLinePointTests:



Protected Attributes

- NSString * testName
- LPLinePoint * pointNoWidth
- LPLinePoint * pointWithWidth

4.20.1 Detailed Description

Tests the LPLinePointTests object

4.20.2 Member Data Documentation

4.20.2.1 - (LPLinePoint*) pointNoWidth [protected]

pointNoWidth is a test point without width variable at init

4.20.2.2 - (LPLinePoint*) pointWithWidth [protected]

pointNoWidth is a test point width variable at init

4.20.2.3 - (NSString*) testName [protected]

testName is the name of the test

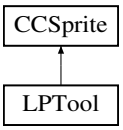
The documentation for this class was generated from the following file:

- LeapPaintTests/LPLinePointTests.h

4.21 LPTool Class Reference

`#import <LPTool.h>`

Inheritance diagram for LPTool:



Properties

- NSString * toolID
- BOOL updated

4.21.1 Detailed Description

Extends CCSprite object with two properties for tracking sprites with pointable objects

4.21.2 Property Documentation

4.21.2.1 - (NSString*) toolID [read], [write], [nonatomic], [strong]

toolID is the ID number assigned by the LeapMotion SDK

Referenced by HUDLayer::moveTrackingTool:toolID:.

4.21.2.2 - (BOOL) updated [read], [write], [nonatomic], [assign]

updated is if the sprite has been updated in that frame.

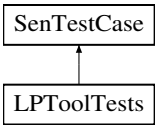
The documentation for this class was generated from the following file:

- LeapPaint/LPTool.h

4.22 LPToolTests Class Reference

`#import <LPToolTests.h>`

Inheritance diagram for LPToolTests:



Protected Attributes

- NSString * testName

4.22.1 Detailed Description

Tests the GameSettings object

4.22.2 Member Data Documentation

4.22.2.1 - (NSString*) testName [protected]

name of the test

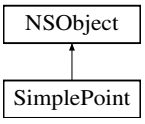
The documentation for this class was generated from the following file:

- LeapPaintTests/LPToolTests.h

4.23 SimplePoint Class Reference

`#import <SimplePoint.h>`

Inheritance diagram for SimplePoint:



Instance Methods

- (id) - initWithPosition:
- (id) - initWithX:withY:
- (id) - initWithPosition:withZ:
- (id) - initWithX:withY:withZ:
- (CGPoint) - point

Properties

- float x
- float y
- float z
- BOOL is3d

4.23.1 Detailed Description

2D or 3D space coordinate for temporarily maniulapting points

4.23.2 Method Documentation

4.23.2.1 - (id) initWithPosition: (CGPoint) *p*

Init constructor with existing point to create a 2d Point

Parameters

p	an point (x,y)

Returns

object instance

init 2d point with CGPoint

4.23.2.2 - (id) initWithPosition: (CGPoint) *p* withZ:(float) *zVal*

Init constructor with existing point to create a 3d Point

Parameters

p	a point (x,y)
zVal	coordinateValue

Returns

object instance

Init 3d point with CGPoint and z Value

4.23.2.3 - (id) initWithX: (float) *xVal* withY:(float) *yVal*

Init constructor with x and y values to create a 2d point

Parameters

xVal	coordinate value
yVal	coordinate value

Returns

object instance

Init 2d Point with 2 separate values

4.23.2.4 - (id) initWithX: (float) *xVal* withY:(float) *yVal* withZ:(float) *zVal*

Init constructor with x, y and z values to create 3D point

Parameters

xVal	coordinate value
yVal	coordinate value
zval	coordinate value

Returns

object instance

Init 3d Point with 3 separate values

4.23.2.5 - (CGPoint) point

Returns point based on x and y

Returns

CGPoint

Return the CGPoint type from the object

4.23.3 Property Documentation

4.23.3.1 - (BOOL) is3d [read], [write], [nonatomic], [assign]

is3d is 2d or 3d point type

4.23.3.2 - (float) x [read], [write], [nonatomic], [assign]

x coordinate

Referenced by point.

4.23.3.3 - (float) y [read], [write], [nonatomic], [assign]

y coordinate

Referenced by point.

4.23.3.4 - (float) z [read], [write], [nonatomic], [assign]

z coordinate

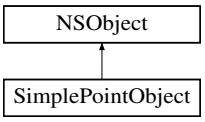
The documentation for this class was generated from the following files:

- LeapPaint/SimplePoint.h
- LeapPaint/SimplePoint.mm

4.24 SimplePointObject Class Reference

`#import <SimplePointObject.h>`

Inheritance diagram for SimplePointObject:



Instance Methods

- (id) - initWithPosition:
- (id) - initWithX:withY:

Properties

- CGPoint point

4.24.1 Detailed Description

2D space coordinate for temporarily maniulapting points

4.24.2 Method Documentation

4.24.2.1 - (id) initWithPosition: (CGPoint) *p*

Init constructor with existing point to create a 2d Point

Parameters

p	an point consisting of (x,y)

Returns

object instance

4.24.2.2 - (id) initWithX: (float) *x* withY:(float) *y*

Init constructor with existing point to create a 2d Point

Parameters

x	is x axis coordinate
y	is y axis coordinate

Returns

object instance

4.24.3 Property Documentation

4.24.3.1 - (CGPoint) point `[read]`, `[write]`, `[nonatomic]`, `[assign]`

point is the X and Y coordinates

Referenced by initWithPosition:.

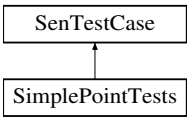
The documentation for this class was generated from the following files:

- LeapPaint/SimplePointObject.h
- LeapPaint/SimplePointObject.m

4.25 SimplePointTests Class Reference

`#import <SimplePointTests.h>`

Inheritance diagram for SimplePointTests:



Protected Attributes

- NSString * testName
- SimplePoint * twoValuePoint
- SimplePoint * threeValuePoint

4.25.1 Detailed Description

Tests the SimplePoint object

4.25.2 Member Data Documentation

4.25.2.1 - (NSString*) testName [protected]

name of the test

4.25.2.2 - (SimplePoint*) threeValuePoint [protected]

three coordinate point (x,y,z)

4.25.2.3 - (SimplePoint*) twoValuePoint [protected]

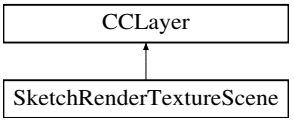
two coordinate point (x,y)

The documentation for this class was generated from the following file:

- LeapPaintTests/SimplePointTests.h

4.26 SketchRenderTextureScene Class Reference

Inheritance diagram for SketchRenderTextureScene:



Instance Methods

- (void) - **beginDraw:withZ:**
- (void) - **updateDraw:withZ:**
- (void) - **endDraw:**
- (void) - **changeColor:**

- (void) - **changeBrush:**
- (void) - **changeScale:**
- (void) - **changeOpacity:**
- (void) - **erasingMode:**
- (void) - **clearDrawing**

Protected Attributes

- CCSprite * **brush**
- NSMutableArray * **touches**
- ccColor3B **lastColor**
- ccColor3B **previousColor**
- NSString * **lastBrush**
- float **lastScale**
- bool **eraseMode**

Properties

- float **opacity**

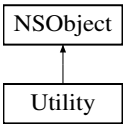
The documentation for this class was generated from the following files:

- LeapPaint/SketchRenderTextureScene.h
- LeapPaint/SketchRenderTextureScene.mm

4.27 Utility Class Reference

`#import <Utility.h>`

Inheritance diagram for Utility:



Class Methods

- (int) + getRandomNumberBetween:to:
- (int) + getRandomUniformNumberUnder:
- (int) + getRandomNumberUnder:

4.27.1 Detailed Description

Utility class provides common usage function throughout the application.

4.27.2 Method Documentation

4.27.2.1 + (int) getRandomNumberBetween: (int) *from* to:(int) *to*

Generates a random number between two designated integers

Parameters

from	is the bottom of the range
to	is the top of the range

Returns

a random number between the from and to parameters

returns random number within a range with defined upper and lower bounds

4.27.2.2 + (int) getRandomNumberUnder: (int) *to*

Generates a random number between 0 designated integer

Parameters

to	is the top of the range

Returns

a random number between 0 and to parameters

Returns a random number from 0 to an upper bound

4.27.2.3 + (int) getRandomUniformNumberUnder: (int) *to*

Generates a random number between 0 designated integer

Parameters

to	is the top of the range

Returns

a random number between 0 and to parameters

Returns a Uniform Random Number from 0 to an upper bound

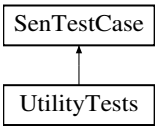
The documentation for this class was generated from the following files:

- LeapPaint/Utility.h
- LeapPaint/Utility.m

4.28 UtilityTests Class Reference

```
#import <UtilityTests.h>
```

Inheritance diagram for UtilityTests:



Protected Attributes

- NSString * testName

4.28.1 Detailed Description

Tests the GameSettings object

4.28.2 Member Data Documentation

4.28.2.1 - (NSString*) testName [protected]

testName is the name of the test

The documentation for this class was generated from the following file:

- LeapPaintTests/UtilityTests.h

Index

```
    1: //
    2: //  AppDelegate.h
    3: //  LeapPaint
    4: //
    5: //  Created by cj on 5/7/13.
    6: //  Copyright (c) 2013 cjdesch. All rights reserved.
    7: //
    8:
    9: #import <Cocoa/Cocoa.h>
   10: #import "cocos2d.h"
   11:
   12: /** Application Delegate
   13:  Creates app instance and binds libraries to interface builder xibs
   14:  Serves as an application wide callback object for events that affects the whole application, such as
low-memory, etc.
   15:  */
   16: @interface AppDelegate : NSObject <NSApplicationDelegate>
   17: {
   18:
   19:     NSWindow    *window_;  /**< window is the main window to be displayed */
   20:     CCGLView    *glView_;  /**< glView is the embedded view in which cocos2d will run inside the windo
w */
   21: }
   22: @property (strong) IBOutlet NSWindow *window;/**< window is the main window to be displayed */
   23: @property (strong) IBOutlet CCGLView *glView;/**< glView is the embedded view in which cocos2d will ru
n inside the window */
   24: /** RunGameSceen sets up the Cocos2d environment and runs it in the application.
   25:  */
   26: - (void)runGameScene;
   27: /** Toggles from a window to full screen view point
   28:  @param sender is the action sending the command
   29:  @return IBAction binding to interface builder
   30:  */
   31: - (IBAction)toggleFullScreen:(id)sender;
   32:
   33: @end
```

```
 1: //
 2: //  AppDelegate.m
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 5/7/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import "AppDelegate.h"
10: #import "GameScene.h"
11:
12: @implementation AppDelegate
13:
14: @synthesize window=window_, glView=glView_;
15:
16: - (void)applicationDidFinishLaunching:(NSNotification *)aNotification{
17:     // Insert code here to initialize your application
18:     [self runGameScene];
19: }
20: - (void)runGameScene{
21:
22:     CCDirectorMac *director = (CCDirectorMac*) [CCDirector sharedDirector];
23:
24:     //NSRect screensFrame = [[NSScreen mainScreen] frame];
25:     NSRect screensFrame = [[NSScreen mainScreen] visibleFrame];
26:     [glView_ setFrameSize:NSMakeSize(screensFrame.size.width,screensFrame.size.height)];
27:         // enable FPS and SPF
28:         [director setDisplayStats:YES];
29:         // connect the OpenGL view with the director
30:         [director setView:glView_];
31:         // EXPERIMENTAL stuff.
32:         // 'Effects' don't work correctly when autoscale is turned on.
33:         // Use kCCDirectorResize_NoScale if you don't want auto-scaling.
34:         [director setResizeMode:kCCDirectorResize_AutoScale];
35:
36:     //[glView_ setFrameSize:NSMakeSize(window_.frame.size.width,window_.frame.size.height-42)];
37:         // Enable "moving" mouse event. Default no.
38:         [window_ setAcceptsMouseMovedEvents:NO];
39:
40:         // Center main window
41:         [window_ center];
42:
43:     CCScene* scene  = [GameScene scene];
44:         [director runWithScene:scene];
45: }
46: #pragma mark AppDelegate - IBActions
47: - (IBAction)toggleFullScreen: (id)sender{
48:         CCDirectorMac *director = (CCDirectorMac*) [CCDirector sharedDirector];
49:         [director setFullScreen: ! [director isFullScreen] ];
50: }
51:
52:
53:
54:
55: @end
```

```
 1: //
 2: //  BackgroundLayer.h
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 4/9/13.
 6: //
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10: #import "cocos2d.h"
11:
12:
13: /** Background Layer
14:     Displays a background image for the scene
15:  */
16: @interface BackgroundLayer : CCLayer
17:
18: @end
```

```
 1: //
 2: //  BackgroundLayer.m
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 4/9/13.
 6: //
 7: //
 8:
 9: #import "BackgroundLayer.h"
10:
11: @implementation BackgroundLayer
12:
13: /** init */
14: - (id)init
15: {
16:         if ((self = [super init]))
17:         {
18:                 // Get window size
19:                 CGSize size = [[CCDirector sharedDirector] winSize];
20:
21:                 // Add a button which takes us back to HelloWorldScene
22:
23:         // Add the generated background
24:         CCSprite *background = [CCSprite spriteWithFile:@"background-fullscreen.png"];
25:         [background setPosition:ccp(size.width / 2, size.height / 2)];
26:
27:         [self addChild:background];
28:
29:         }
30:         return self;
31: }
32:
33: @end
```

```
    1: //
    2: //  BrushSelectionLayer.h
    3: //  LeapPuzz
    4: //
    5: //  Created by cj on 4/9/13.
    6: //
    7: //
    8:
    9:
   10: #import "cocos2d.h"
   11: #import "CCControlExtension.h"
   12:
   13:
   14: /** BrushSelectionLayer Delegate
   15:  Provides a delegate interface for the layer to notify of actions
   16:  */
   17: @protocol BrushSelectionLayerDelegate <NSObject>
   18: /**
   19:  Calls back to notify that the layer can be hidden
   20:  */
   21: - (void)hidePanel;
   22: /**
   23:  Calls back to notify a new brushname has been selected
   24:  @param brushname is the name of the brush that has been selected.
   25:  */
   26: - (void)brushSelected:(NSString*)brushname;
   27: @end
   28:
   29: /** BrushSelectionLayer
   30:  This user interface layer provides a collection view of all the available brushes that can be selecte
d.
   31:  */
   32: @interface BrushSelectionLayer : CCLayer{
   33:
   34:     NSMutableDictionary* imageNamesDictionary; /**< imageNamesDictionary is the list of brush names av
ailable for selection */
   35:
   36: }
   37:
   38: @property (nonatomic, weak) id <BrushSelectionLayerDelegate> delegate;/**< delegate is the instance re
ference for triggering delegate call back functions */
   39: @property (nonatomic, readwrite) bool layerHidden;/**< layerHidded tracks the visibility state of the
layer */
   40:
   41:
   42: @end
```

```
    1: //
    2: //  BrushSelectionLayer.m
    3: //  LeapPuzz
    4: //
    5: //  Created by cj on 4/9/13.
    6: //
    7: //
    8:
    9: #import "BrushSelectionLayer.h"
   10:
   11: @implementation BrushSelectionLayer
   12: @synthesize delegate;
   13: @synthesize layerHidden;
   14: - (id)init
   15: {
   16:         if ((self = [super init]))
   17:         {
   18:                 // Get window size
   19:                 CGSize size = [[CCDirector sharedDirector] winSize];
   20:
   21:                 // Add a button which takes us back to HelloWorldScene
   22:         CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"BrushSelectionLayer" fontName:@"Marker
 Felt" fontSize:30];
   23:         titleButton.position =ccp(size.width / 2.0f , 125);
   24:
   25:
   26:         [self addChild:titleButton];
   27:         // Add the generated background
   28:         CCSprite *background = [CCSprite spriteWithFile:@"background-fullscreen.png"];
   29:         [background setPosition:ccp(size.width / 2, size.height / 2)];
   30:         self.layerHidden = true;
   31:         [self addChild:background];
   32:
   33:         [self buttoninit];
   34:
   35:         int brushCount = 30;
   36:         //NSMutableArray* menuItems = [[NSMutableArray alloc] init];
   37:         imageNamesDictionary = [[NSMutableDictionary alloc] init];
   38:         CCMenu *starMenu = [CCMenu menuWithItems:nil];
   39:         for (int i =0; i < brushCount; i++){
   40:             NSString* imagename = [NSString stringWithFormat:@"brush_%d.png",i];
   41:             CCMenuItem *starMenuItem = [CCMenuItemImage
   42:                                         itemWithNormalImage:imagename selectedImage:imagename
   43:                                         target:self selector:@selector(brushSelectedAction:)];
   44:             //starMenuItem.position = ccp(size.width /2, size.height /2);
   45:             starMenuItem.tag = i;
   46:             [imageNamesDictionary setObject:imagename forKey:[NSString stringWithFormat:@"%d",i]];
   47:
   48:
   49:
   50:             [starMenu addChild:starMenuItem];
   51:         }
   52:
   53:         //[starMenu alignItemsHorizontally];
   54:         NSNumber* itemsPerRow = [NSNumber numberWithInt:5];
   55:         [starMenu alignItemsInColumns:itemsPerRow,itemsPerRow,itemsPerRow,itemsPerRow,itemsPerRow,item
sPerRow, nil];
   56:
   57:
   58:
   59:
   60:
   61:         starMenu.position = ccp(size.width / 2, size.height / 2);
   62:         [self addChild:starMenu];
   63:
   64:         }
   65:         return self;
   66: }
   67:
   68:
   69:
   70: - (void)buttoninit{
   71:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
   72:     // Add the button
   73:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
   74:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
   75:
   76: #ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
   77:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Touch Me!" fontName:@"HelveticaNeue-Bold"
```

```
fontSize:30];
   78: #elif __MAC_OS_X_VERSION_MAX_ALLOWED
   79:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Hide" fontName:@"Marker Felt" fontSize:30]
;
   80: #endif
   81:     [titleButton setColor:ccc3(159, 168, 176)];
   82:
   83:     CCControlButton *controlButton = [CCControlButton buttonWithLabel:titleButton
   84:                                                       backgroundSprite:backgroundButton];
   85:     [controlButton setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted]
;
   86:     [controlButton setTitleColor:ccWHITE forState:CCControlStateHighlighted];
   87:
   88:     controlButton.anchorPoint = ccp(0.5f, 1);
   89:     controlButton.position = ccp(screenSize.width / 2.0f, screenSize.height -100);
   90:     [self addChild:controlButton z:1];
   91:
   92:     // Add the black background
   93:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
   94:     [background setContentSize:CGSizeMake(300, 170)];
   95:     [background setPosition:ccp(screenSize.width / 2.0f, screenSize.height / 2.0f)];
   96:     //[self addChild:background];
   97:
   98:     // Sets up event handlers
   99:     [controlButton addTarget:self action:@selector(touchDownAction:) forControlEvents:CCControlEventTo
uchDown];
  100: }
  101:
  102: - (void)touchDownAction:(CCControlButton *)sender
  103: {
  104:
  105:
  106:     [self.delegate hidePanel];
  107: }
  108:
  109:
  110: - (void)brushSelectedAction:(id)sender
  111: {
  112:     NSLog(@"Selected Brush");
  113:
  114:     CCMenuItemImage* menuItem =     (CCMenuItemImage*)sender;
  115:     int i = menuItem.tag;
  116:     NSString* imageName = [imageNamesDictionary objectForKey:[NSString stringWithFormat:@"%d",i]];
  117:     [self.delegate brushSelected:imageName];
  118:     [self.delegate hidePanel];
  119:
  120: }
  121:
  122:
  123:
  124: @end
```

```
    1: //
    2: //  ControlsLayer.h
    3: //  LeapPuzz
    4: //
    5: //  Created by cj on 4/9/13.
    6: //
    7: //
    8:
    9: #import <Foundation/Foundation.h>
   10: #import "cocos2d.h"
   11: #import "CCControlExtension.h"
   12: #import "BrushSelectionLayer.h"
   13: #import "GameSettings.h"
   14: /**
   15:  Controls Layer Delegate
   16: Provides a delegate interface for the layer to notify of actions
   17:  */
   18: @protocol ControlsLayerDelegate <NSObject>
   19: /**
   20:  Callback with a change in color of the brush
   21:  @param color is the new selected color value
   22:  */
   23: - (void)changeColorControl:(ccColor3B)color;
   24: /**
   25:  Callback with a change in thickness of the brush
   26:  @param value is the new selected color value
   27:  */
   28: - (void)changeThicknessControl:(float)value;
   29: /**
   30:  Callback with a change in brush texture
   31:  @param brushname is the new selected brush value
   32:  */
   33: - (void)changeBrushControl:(NSString*)brushname;
   34: /**
   35:  Callback with a change in opacity
   36:  @param value is the new selected opacity value
   37:  */
   38: - (void)changeOpacityControl:(float)value;
   39: /**
   40:  Callback to notify to clear the drawing
   41:  */
   42: - (void)clearDrawing;
   43: /**
   44:  Callback with a change in color
   45:  @param mode is the  toggled eraser mode
   46:  TODO: Turn off eraser mode when new color is selected
   47:  */
   48: - (void)eraserMode:(bool)mode;
   49: @end
   50:
   51: /** Controls Layer
   52:     User inferface controls for operating buttons, switches, sliders
   53:  */
   54:
   55: @interface ControlsLayer : CCLayer <BrushSelectionLayerDelegate>{
   56:     CCLabelTTF *colorLabel;         /**< colorLabel displays name of color in hash value */
   57:     CCLabelTTF *displayValueLabel;  /**< displayValueLabel displays coordinate */
   58:     GameSettings* gameSettings;     /**< gameSettings global reference to shared settings instance */
   59: }
   60: @property (nonatomic, strong) CCControlSlider   *slider;                    /**< slider is the thickne
ss control of the brush */
   61: @property (nonatomic, strong) CCControlSlider   *opacitySlider;             /**< opacitySlider is the
opacity contro of the brush*/
   62: @property (nonatomic, strong) CCControlSwitch   *opacitySwitchControl;      /**< opacitySwitchControl
is the control for setting automatic or manual opacity control */
   63: @property (nonatomic, strong) CCLabelTTF        *opacitydisplayValueLabel;  /**<  opacitydisplayValueL
abel shows the state of the opacitySwitchControl*/
   64: @property (nonatomic, weak) id <ControlsLayerDelegate> delegate;            /**< delegate is the insta
nce reference for triggering delegate call back functions */
   65: @property (nonatomic,strong) BrushSelectionLayer *brushSelection;           /**< brushSelection layer
expands as a drawer to allow for brush selection */
   66: @property (nonatomic, strong) CCLabelTTF        *displayValueLabel;         /**< displayValueLabel dis
plays eraser toggle state */
   67: @property (nonatomic, strong) CCControlSwitch   *switchControl;             /**< switchControl is the
eraser toggle */
   68:
   69: /**
   70:  Recieves brushSizeControl delegate callbacks and updates values in the interface
   71:  @param sender is the object performing the callback
   72:  */
```

```
- (void)valueChanged:(CCControlSlider *)sender;

/**
 Recieves opacitySliderControl delegate callbacks and updates values in the interface
 @param sender is the object performing the callback
 */
- (void)opacitySliderChanged:(CCControlSlider *)sender;

/** Expands brushes panel*/
- (void)expandPanel;
/** Collapses Brushes Panel */
- (void)collapsePanel;

/** Creates and returns a new CCControlSwitch.
 @return a generate ControlSwitch
 */
- (CCControlSwitch *)makeControlSwitch;
/** Callback for the change value.
  @param sender is the object performing the callback*/
- (void)switchValueChanged:(CCControlSwitch *)sender;
/** Callback for opacity changing with the slider
  @param sender is the object performing the callback
 */
- (void)updateOpacitySlider:(float)value;

@end
```

```
    1: //
    2: //  ControlsLayer.m
    3: //  LeapPuzz
    4: //
    5: //  Created by cj on 4/9/13.
    6: //
    7: //
    8:
    9: #import "ControlsLayer.h"
   10:
   11: @implementation ControlsLayer
   12: @synthesize slider;
   13: @synthesize opacitySlider;
   14: @synthesize opacitydisplayValueLabel;
   15: @synthesize opacitySwitchControl;
   16: @synthesize delegate;
   17: @synthesize displayValueLabel;
   18: @synthesize switchControl;
   19: @synthesize brushSelection;
   20:
   21: - (id)init
   22: {
   23:         if ((self = [super init]))
   24:         {
   25:                 // Get window size
   26:                 CGSize screenSize = [[CCDirector sharedDirector] winSize];
   27:
   28:         gameSettings = [GameSettings sharedInstance];
   29:         [self sliderinit];
   30:         [self initEraserSwitch];
   31:         [self colorpickerinit];
   32:
   33:         [self initResetButton];
   34:         self.brushSelection = [BrushSelectionLayer node];
   35:         self.brushSelection.position = ccp(-screenSize.width,0);
   36:
   37:         self.brushSelection.delegate = self;
   38:
   39:         [self addChild:brushSelection z:10];
   40:         [self initBrushSelectionButton];
   41:         [self opacitySliderInit];
   42:         [self initOpacitySwitch];
   43:         }
   44:         return self;
   45: }
   46:
   47:
   48: - (void)sliderinit{
   49:
   50:     //Slideer
   51:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
   52:     // Add the slider
   53:     self.slider                     = [CCControlSlider sliderWithBackgroundFile:@"sliderTrack.png"
   54:                                                                    progressFile:@"sliderProgress.png"
   55:                                                                       thumbFile:@"sliderThumb.png"];
   56:     self.slider.anchorPoint             = ccp(0.5f, 1.0f);
   57:     self.slider.minimumValue            = 0.0f; // Sets the min value of range
   58:     self.slider.maximumValue            = 5.0f; // Sets the max value of range
   59:     self.slider.position                = ccp(screenSize.width / 2.0f, 100);
   60:
   61:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Size" fontName:@"Marker Felt" fontSize:30]
;
   62:     titleButton.position =ccp(screenSize.width / 2.0f , 125);
   63:
   64:     // Add the black background
   65:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
   66:     [background setContentSize:CGSizeMake(350,100)];
   67:     [background setPosition:ccp(screenSize.width / 2.0f, 100)];
   68:     [self addChild:background];
   69:
   70:     [self addChild:titleButton];
   71:     // When the value of the slider will change, the given selector will be call
   72:     [self.slider addTarget:self action:@selector(valueChanged:) forControlEvents:CCControlEventValueCh
anged];
   73:
   74:     [self addChild:self.slider z:0 tag:1];
   75: }
   76:
   77: - (void)valueChanged:(CCControlSlider *)sender{
   78:         // Change value of label.
```

```
   79:         //NSLog(@"slider value %@", [NSString stringWithFormat:@"Slider value = %.02f", sender.value])
;
   80:     [self.delegate changeThicknessControl:sender.value];
   81: }
   82:
   83: - (void)opacitySliderInit{
   84:
   85:     //Slideer
   86:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
   87:     CCNode *layer                       = [CCNode node];
   88:     layer.position                      =ccp(screenSize.width / 2.0f + 200, 100);
   89:     [self addChild:layer z:3];
   90:
   91:     double layer_width = 0;
   92:
   93:
   94:     // Add the black background
   95:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
   96:     [background setContentSize:CGSizeMake(350,100)];
   97:     [background setPosition:ccp(background.contentSize.width / 2.0f, 0)];
   98:     [layer addChild:background];
   99:     layer_width += background.contentSize.width;
  100:
  101:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Opacity" fontName:@"Marker Felt" fontSize:
30];
  102:     titleButton.position =ccp(layer_width / 2.0f , 25);
  103:
  104:
  105:     [layer addChild:titleButton];
  106:     // When the value of the slider will change, the given selector will be call
  107:
  108:     // Add the slider
  109:     self.opacitySlider                    = [CCControlSlider sliderWithBackgroundFile:@"sliderTrack.p
ng"
  110:                                                                                progressFile:@"sliderProgres
s.png"
  111:                                                                                   thumbFile:@"sliderThumb.p
ng"];
  112:     self.opacitySlider.anchorPoint              = ccp(0.5f, 1.0f);
  113:     self.opacitySlider.minimumValue             = 0.0f; // Sets the min value of range
  114:     self.opacitySlider.maximumValue             = 100.0f; // Sets the max value of range
  115:     self.opacitySlider.position                 = ccp(layer_width  / 2.0f, 0);
  116:
  117:     [self.opacitySlider addTarget:self action:@selector(opacitySliderChanged:) forControlEvents:CCCont
rolEventValueChanged];
  118:
  119:     [layer addChild:self.opacitySlider z:0 tag:2];
  120: }
  121:
  122: - (void)opacitySliderChanged:(CCControlSlider *)sender{
  123:         // Change value of label.
  124:         //NSLog(@"slider value %@", [NSString stringWithFormat:@"Slider value = %.02f", sender.value])
;
  125:     [self.delegate changeOpacityControl:sender.value];
  126: }
  127:
  128: - (void)updateOpacitySlider:(float)value{
  129:     //ensure the value is within its bounds
  130:     if(value > self.opacitySlider.maximumValue){
  131:         //Max Value
  132:         self.opacitySlider.value = self.opacitySlider.maximumValue;
  133:     }else if(value < self.opacitySlider.minimumValue){
  134:         //Min Value
  135:         self.opacitySlider.value = self.opacitySlider.minimumValue;
  136:     }else{
  137:         self.opacitySlider.value = value;
  138:     }
  139: }
  140:
  141:
  142: - (void)initOpacitySwitch{
  143:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
  144:
  145:     CCNode *layer               = [CCNode node];
  146:     //layer.position              = ccp (screenSize.width / 2, screenSize.height / 2);
  147:     layer.position = ccp(screenSize.width / 2.0f + 400, 125);
  148:     [self addChild:layer z:5];
  149:
  150:     double layer_width = 0;
  151:
```

```
   152:     // Add the black background for the text
   153:     CCScale9Sprite *background  = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
   154:     background.contentSize      = CGSizeMake(80, 50);
   155:     background.position         = ccp(layer_width + background.contentSize.width / 2.0f, 0);
   156:     //[layer addChild:background];
   157:
   158:     layer_width += background.contentSize.width;
   159:     self.opacitydisplayValueLabel      = [CCLabelTTF labelWithString:@"Auto" fontName:@"Marker Felt" f
ontSize:30];
   160:
   161:     self.opacitydisplayValueLabel.position  = background.position;
   162:     //[layer addChild:self.opacitydisplayValueLabel];
   163:
   164:     // Create the switch
   165:     self.opacitySwitchControl          = [self makeControlSwitch];
   166:     self.opacitySwitchControl.position      = ccp (layer_width + 10 + self.opacitySwitchControl.conten
tSize.width / 2, 0);
   167:     self.opacitySwitchControl.on            = NO;
   168:     [layer addChild:self.opacitySwitchControl];
   169:
   170:     [self.opacitySwitchControl addTarget:self action:@selector(opacitySwitchValueChanged:) forControlE
vents:CCControlEventValueChanged];
   171:
   172:     // Set the layer size
   173:     layer.contentSize           = CGSizeMake(layer_width, 0);
   174:     layer.anchorPoint           = ccp (0.5f, 0.5f);
   175:
   176: }
   177:
   178:
   179:
   180:
   181: - (void)opacitySwitchValueChanged:(CCControlSwitch *)sender{
   182:     if ([sender isOn]){
   183:         //displayValueLabel.string    = @"Eraser";
   184:         gameSettings.depthOpacityMode  = true;
   185:     } else{
   186:         //displayValueLabel.string    = @"Eraser";
   187:         gameSettings.depthOpacityMode  = false;
   188:     }
   189: }
   190:
   191: #pragma mark - button
   192:
   193: - (void)buttoninit{
   194:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
   195:     // Add the button
   196:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
   197:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
   198:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Touch Me!" fontName:@"Marker Felt" fontSiz
e:30];
   199:
   200:     [titleButton setColor:ccc3(159, 168, 176)];
   201:
   202:     CCControlButton *controlButton = [CCControlButton buttonWithLabel:titleButton
   203:                                                       backgroundSprite:backgroundButton];
   204:     [controlButton setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted]
;
   205:     [controlButton setTitleColor:ccWHITE forState:CCControlStateHighlighted];
   206:
   207:     controlButton.anchorPoint = ccp(0.5f, 1);
   208:     controlButton.position = ccp(screenSize.width / 2.0f, screenSize.height / 2.0f);
   209:     [self addChild:controlButton z:1];
   210:
   211:     // Add the black background
   212:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
   213:     [background setContentSize:CGSizeMake(300, 170)];
   214:     [background setPosition:ccp(screenSize.width / 2.0f, screenSize.height / 2.0f)];
   215:     [self addChild:background];
   216:
   217:     // Sets up event handlers
   218:     [controlButton addTarget:self action:@selector(touchDownAction:) forControlEvents:CCControlEventTo
uchDown];
   219: }
   220:
   221: - (void)touchDownAction:(CCControlButton *)sender
   222: {
   223:     NSLog(@"button value %@", [NSString stringWithFormat:@"Touch Down"]);
   224: }
```

```
225:
226: - (void)initEraserButton{
227:     // Add the button
228:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
229:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
230:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Eraser" fontName:@"Marker Felt" fontSize:3
0];
231:
232:     [titleButton setColor:ccc3(159, 168, 176)];
233:
234:     CCControlButton *controlButton = [CCControlButton buttonWithLabel:titleButton
235:                                                       backgroundSprite:backgroundButton];
236:     [controlButton setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted]
;
237:     [controlButton setTitleColor:ccWHITE forState:CCControlStateHighlighted];
238:
239:     controlButton.anchorPoint = ccp(0.5f, 1);
240:     controlButton.position = ccp(100, 100);
241:     [self addChild:controlButton z:1];
242:
243:     // Add the black background
244:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
245:     background.anchorPoint =  ccp(0, 0);
246:     [background setContentSize:CGSizeMake(100, 75)];
247:     [background setPosition: ccp(50, 50)];
248:     [self addChild:background];
249:
250:     // Sets up event handlers
251:     [controlButton addTarget:self action:@selector(eraserAction:) forControlEvents:CCControlEventTouch
Down];
252: }
253:
254: - (void)eraserAction:(CCControlButton *)sender
255: {
256:     [self.delegate changeColorControl:ccWHITE];
257:
258: }
259: - (void)initResetButton{
260:
261:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
262:     // Add the button
263:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
264:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
265:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Reset" fontName:@"Marker Felt" fontSize:30
];
266:
267:     [titleButton setColor:ccc3(159, 168, 176)];
268:
269:     CCControlButton *controlButton = [CCControlButton buttonWithLabel:titleButton
270:                                                       backgroundSprite:backgroundButton];
271:     [controlButton setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted]
;
272:     [controlButton setTitleColor:ccWHITE forState:CCControlStateHighlighted];
273:
274:     controlButton.anchorPoint = ccp(0.5f, 1);
275:     controlButton.position = ccp(100, screenSize.height -100);
276:     [self addChild:controlButton z:1];
277:
278:     // Add the black background
279:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
280:     [background setContentSize:CGSizeMake(100, 75)];
281:     [background setPosition:ccp(100, screenSize.height -125)];
282:     [self addChild:background];
283:
284:     // Sets up event handlers
285:     [controlButton addTarget:self action:@selector(resetAction:) forControlEvents:CCControlEventTouchD
own];
286: }
287:
288:
289: - (void)resetAction:(CCControlButton*)sender{
290:     [self.delegate clearDrawing];
291: }
292:
293: - (CCControlButton *)standardButtonWithTitle:(NSString *)title
294: {
295:     /** Creates and return a button with a default background and title color. */
296:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
```

```
297:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
298:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:title fontName:@"Marker Felt" fontSize:30];
299:
300:     [titleButton setColor:ccc3(159, 168, 176)];
301:
302:     CCControlButton *button = [CCControlButton buttonWithLabel:titleButton backgroundSprite:background
Button];
303:     [button setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted];
304:     [button setTitleColor:ccWHITE forState:CCControlStateHighlighted];
305:
306:     return button;
307: }
308:
309:
310: #pragma mark - ColorPicker
311:
312: - (void)colorpickerinit{
313:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
314:     CCNode *layer                       = [CCNode node];
315:     layer.position                      = ccp (screenSize.width -300 , screenSize.height / 2);
316:     [self addChild:layer z:1];
317:
318:     double layer_width = 0;
319:
320:     // Add the black background for the text
321:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
322:     [background setContentSize:CGSizeMake(150, 50)];
323:     [background setPosition:ccp(layer_width + background.contentSize.width / 2.0f, 0)];
324:     //[layer addChild:background];
325:
326:     layer_width += background.contentSize.width;
327:     colorLabel = [CCLabelTTF labelWithString:@"#color" fontName:@"Marker Felt" fontSize:30];
328:     colorLabel.position = background.position;
329:     //[layer addChild:colorLabel];
330:
331:     // Create the colour picker
332:     CCControlColourPicker *colourPicker = [CCControlColourPicker colourPickerWithHueFile:@"hueBackgrou
nd.png"
333:                                                                                 tintBackgroundFile:@"tintBackgro
und.png"
334:                                                                                    tintOverlayFile:@"tintOverlay
.png"
335:                                                                                         pickerFile:@"picker.png"
336:                                                                                          arrowFile:@"arrow.png"]
;
337:     colourPicker.color                  = ccc3(37, 46, 252);
338:     colourPicker.position               = ccp (layer_width + colourPicker.contentSize.width / 2, 0);
339:     colourPicker.arrowDirection         = CCControlColourPickerArrowDirectionLeft;
340:
341:     // Add it to the layer
342:     [layer addChild:colourPicker];
343:
344: #if NS_BLOCKS_AVAILABLE
345:     // Add block for value changed event
346:     [colourPicker setBlock:^(id sender, CCControlEvent event)
347:      {
348:          [self colourValueChanged:sender];
349:      }
350:           forControlEvents:CCControlEventValueChanged];
351: #else
352:     // Add the target-action pair
353:     [colourPicker addTarget:self action:@selector(colourValueChanged:) forControlEvents:CCControlEvent
ValueChanged];
354: #endif
355:
356:     layer_width += colourPicker.contentSize.width;
357:
358:     // Set the layer size
359:     layer.contentSize                   = CGSizeMake(layer_width, 0);
360:     layer.anchorPoint                   = ccp (0.5f, 0.5f);
361:
362:     // Update the color text
363:     [self colourValueChanged:colourPicker];
364: }
365:
366: - (void)colourValueChanged:(CCControlColourPicker *)sender
367: {
368:     colorLabel.string   = [NSString stringWithFormat:@"#%02X%02X%02X",sender.color.r, sender.color.g,
sender.color.b];
```

```
   369:
   370:     [self.delegate changeColorControl:sender.color];
   371: }
   372: #pragma mark - Window Controls
   373:
   374:
   375: - (void)initEraserSwitch{
   376:
   377:
   378:
   379:     CCNode *layer               = [CCNode node];
   380:     //layer.position             = ccp (screenSize.width / 2, screenSize.height / 2);
   381:     layer.position = ccp(100, 100);
   382:     [self addChild:layer z:1];
   383:
   384:     double layer_width = 0;
   385:
   386:     // Add the black background for the text
   387:     CCScale9Sprite *background  = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
   388:     background.contentSize      = CGSizeMake(80, 50);
   389:     background.position         = ccp(layer_width + background.contentSize.width / 2.0f, 0);
   390:     [layer addChild:background];
   391:
   392:     layer_width += background.contentSize.width;
   393:     self.displayValueLabel      = [CCLabelTTF labelWithString:@"Eraser" fontName:@"Marker Felt" fontSi
ze:30];
   394:
   395:     displayValueLabel.position  = background.position;
   396:     [layer addChild:displayValueLabel];
   397:
   398:     // Create the switch
   399:     self.switchControl          = [self makeControlSwitch];
   400:     switchControl.position      = ccp (layer_width + 10 + switchControl.contentSize.width / 2, 0);
   401:     switchControl.on            = NO;
   402:     [layer addChild:switchControl];
   403:
   404:     [switchControl addTarget:self action:@selector(switchValueChanged:) forControlEvents:CCControlEven
tValueChanged];
   405:
   406:     // Set the layer size
   407:     layer.contentSize           = CGSizeMake(layer_width, 0);
   408:     layer.anchorPoint           = ccp (0.5f, 0.5f);
   409:
   410:
   411: }
   412:
   413:
   414: - (CCControlSwitch *)makeControlSwitch
   415: {
   416:     return [CCControlSwitch switchWithMaskSprite:[CCSprite spriteWithFile:@"switch-mask.png"]
   417:                                        onSprite:[CCSprite spriteWithFile:@"switch-on.png"]
   418:                                       offSprite:[CCSprite spriteWithFile:@"switch-off.png"]
   419:                                     thumbSprite:[CCSprite spriteWithFile:@"switch-thumb.png"]
   420:                                         onLabel:[CCLabelTTF labelWithString:@"On" fontName:@"Arial-Bo
ldMT" fontSize:16]
   421:                                        offLabel:[CCLabelTTF labelWithString:@"Off" fontName:@"Arial-B
oldMT" fontSize:16]];
   422: }
   423:
   424:
   425: - (void)switchValueChanged:(CCControlSwitch *)sender{
   426:     if ([sender isOn]){
   427:         displayValueLabel.string    = @"Eraser";
   428:         [self.delegate eraserMode:true];
   429:     } else{
   430:         displayValueLabel.string    = @"Eraser";
   431:         [self.delegate eraserMode:false];
   432:     }
   433: }
   434:
   435:
   436: #pragma mark- Brush Selection Delegate
   437:
   438: - (void)initBrushSelectionButton{
   439:
   440:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
   441:     // Add the button
   442:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
   443:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
```

```
   444:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Brushes" fontName:@"Marker Felt" fontSize:
30];
   445:
   446:     [titleButton setColor:ccc3(159, 168, 176)];
   447:
   448:     CCControlButton *controlButton = [CCControlButton buttonWithLabel:titleButton
   449:                                                       backgroundSprite:backgroundButton];
   450:     [controlButton setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted]
;
   451:     [controlButton setTitleColor:ccWHITE forState:CCControlStateHighlighted];
   452:
   453:     controlButton.anchorPoint = ccp(0.5f, 1);
   454:     controlButton.position = ccp(screenSize.width -100, screenSize.height -100);
   455:     [self addChild:controlButton z:1];
   456:
   457:     // Add the black background
   458:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
   459:     [background setContentSize:CGSizeMake(100, 75)];
   460:     [background setPosition:ccp(screenSize.width -100, screenSize.height -125)];
   461:     [self addChild:background];
   462:
   463:     // Sets up event handlers
   464:     [controlButton addTarget:self action:@selector(brushButtonAction:) forControlEvents:CCControlEvent
TouchDown];
   465: }
   466:
   467:
   468: - (void)brushButtonAction:(CCControlButton*)sender{
   469:     if (self.brushSelection.layerHidden) {
   470:         [self showBrushSelectionPanel];
   471:     }else{
   472:         [self hideBrushSelectionPanel];
   473:     }
   474:
   475: }
   476:
   477: - (void)showBrushSelectionPanel{
   478:     self.brushSelection.layerHidden = false;
   479:     [self.brushSelection runAction:[CCMoveTo actionWithDuration:2 position:ccp(0,0)]];
   480: }
   481:
   482: - (void)hideBrushSelectionPanel{
   483:     // Get window size
   484:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
   485:
   486:     self.brushSelection.layerHidden = true;
   487:     [self.brushSelection runAction:[CCMoveTo actionWithDuration:2 position:ccp(-screenSize.width,0)]];
   488: }
   489:
   490: - (void)hidePanel{
   491:     [self hideBrushSelectionPanel];
   492: }
   493:
   494:
   495: - (void)brushSelected:(NSString *)brushname{
   496:     [self.delegate changeBrushControl:brushname];
   497: }
   498:
   499:
   500: @end
```

```
    1: //
    2: //  GameManager.h
    3: //  LeapPuzz
    4: //
    5: //  Created by cj on 4/2/13.
    6: //
    7: //
    8:
    9: #import <Foundation/Foundation.h>
   10: #import "cocos2d.h"
   11: #import "LeapObjectiveC.h"
   12: #import "HUDLayer.h"
   13:
   14:
   15: #import "SketchRenderTextureScene.h"
   16: #import "BackgroundLayer.h"
   17: #import "ControlsLayer.h"
   18: #import "GameSettings.h"
   19: #import "SimplePoint.h"
   20:
   21: /**
   22:  Core Application Management
   23:  Provides interfaces and controls the various inputs, controls and outputs
   24:
   25:  */
   26: @interface GameManager : CCScene <LeapListener, HUDDelegate, ControlsLayerDelegate>
   27: {
   28:     InputMode inputMode;    /**< colorLabel displays name of color in hash value */
   29:     LeapPointable* currentPointable; /**< colorLabel displays name of color in hash value */
   30:     CGPoint currentPoint;   /**< colorLabel displays name of color in hash value */
   31:     //Settings
   32:     BOOL painting;      /**< painting indicates wether or not the application is painting at that mome
nt*/
   33:
   34:     GameSettings* gameSettings; /**< gameSettings singleton to global seetings*/
   35:
   36:
   37:     int lastTag;                /**< lastTag is the last tag value tracked of a LeapPointable */
   38:     SimplePoint* lastPoint;     /**< lastPoint is the last known point on the screen of the LeapPointa
ble */
   39:     int framesSinceLastFound;   /**< framesSinceLastFound number of frames since last finding a LeapPo
intable */
   40:
   41:
   42: }
   43:
   44: @property (nonatomic,strong) HUDLayer* hudLayer;                   /**< hudLayer displays the icons f
or tracking where a leapPointable is pointing */
   45: @property (nonatomic,strong) SketchRenderTextureScene* textureScene;/**< textureScene is the drawing l
ayer */
   46: @property (nonatomic,strong) BackgroundLayer* backgroundLayer;      /**< backgroundLayer is the layer
for setting up the background */
   47: @property (nonatomic,strong) ControlsLayer* controlsLayer;          /**< controlsLayer is the layer fo
r managing interface controls */
   48:
   49: @property (nonatomic,strong) LeapController* controller;            /**< controller is the leapControl
ler */
   50: @property (nonatomic,strong) LeapScreen* leapScreen;                /**< leapScreen references the scr
een being used on the system */
   51:
   52: /**
   53:  Finds the percentage of a number between two values
   54:  If the number is greater or less than the range, that boundry of the range will be returned.
   55:  @param max is the top range value
   56:  @param min is the bottom range value
   57:  @param value is the number we are seeking the percentage from
   58:  @return the a percentage between 0 and 100%
   59:  */
   60: - (float)findPecentageDifference:(float)max withMin:(float)min withValue:(float)value;
   61:
   62: /**
   63:  Determines the opacity based upon the Z axis coordinate
   64:  @param value is the Z axis coordinate
   65:  @return the opacity value to set the brush at.
   66:  */
   67: - (float)opacityPercentage:(float)value;
   68:
   69: @end
```

```
    1: //
    2: //  GameManager.m
    3: //  LeapPuzz
    4: //
    5: //  Created by cj on 4/2/13.
    6: //
    7: //
    8:
    9: #import "GameManager.h"
   10:
   11: @implementation GameManager
   12:
   13: @synthesize hudLayer;
   14: @synthesize textureScene;
   15: @synthesize backgroundLayer;
   16: @synthesize controlsLayer;
   17: @synthesize controller;
   18: @synthesize leapScreen;
   19:
   20:
   21: // On "init" you need to initialize your instance
   22: -(id) init
   23: {
   24:         // always call "super" init
   25:         // Apple recommends to re-assign "self" with the "super" return value
   26:         if( (self=[super init])) {
   27:
   28:
   29:                 // create and initialize a Label
   30:                 CCLabelTTF *label = [CCLabelTTF labelWithString:@"Leap Paint" fontName:@"Marker Felt"
fontSize:64];
   31:
   32:                 // ask director the the window size
   33:                 CGSize size = [[CCDirector sharedDirector] winSize];
   34:
   35:         NSLog(@"Window size (pixels)-- Width: %0.0f Height: %0.0f",size.width, size.height);
   36:
   37:                 // position the label on the center of the screen
   38:                 label.position =  ccp( size.width /2 , size.height - 25 );
   39:                 // add the label as a child to this Layer
   40:
   41:                 [self addChild: label];
   42:         [self run];
   43:
   44:         inputMode = kPressKeyMode;
   45:         painting = false;
   46:
   47:         gameSettings = [GameSettings sharedInstance];
   48:
   49:         lastTag = -1;
   50:         lastPoint = [[SimplePoint alloc] initWithX:0.0f withY:0.0f withZ:0.0f];
   51:         framesSinceLastFound = 0;
   52:
   53:         }
   54:         return self;
   55: }
   56:
   57: #pragma mark - SampleDelegate Callbacks
   58:
   59: /**
   60:  LeapMotion SDK Delegate Callback
   61:  Init's a LeapMotion instance to initiate connection and tracking with the LeapMotion and assigns the
delegate or listener for the controller
   62:  */
   63: - (void)run
   64: {
   65:     controller = [[LeapController alloc] init];
   66:     [controller addListener:self];
   67:
   68: }
   69: /**
   70:  LeapMotion SDK Delegate Callback
   71:  Initialize
   72:  Verifies the LeapMotion has been initialized and any additional steps for setup can continue.
   73:  */
   74:
   75: - (void)onInit:(NSNotification *)notification{
   76:     NSLog(@"Leap: Initialized");
   77: }
   78: /**
```

```
 79:  LeapMotion SDK Delegate Callback
 80:  Connect
 81:  Verifies the LeapMotion is connected and additional steps for setup can continue.
 82:  Sets up the screens to be track intersecting vectors from pointables.
 83:  */
 84: - (void)onConnect:(NSNotification *)notification{
 85:     NSLog(@"Leap: Connected");
 86:     NSArray* screens = controller.locatedScreens;
 87:     if ([screens count] > 0){
 88:         leapScreen = [screens objectAtIndex:0];
 89:         NSLog(@"Screens: %0.0ld", (unsigned long)[screens count]);
 90:     }else{
 91:         NSLog(@"No Screens");
 92:     }
 93: }
 94: /**
 95:  LeapMotion SDK Delegate Callback
 96:  Disconnect
 97:  Notifies the application that the LeapMotion has been disconnected and hold or release any processes
in regard to the LeapMotion
 98:  */
 99: - (void)onDisconnect:(NSNotification *)notification{
100:     NSLog(@"Leap: Disconnected");
101: }
102:
103: /**
104:  LeapMotion SDK Delegate Callback
105:  Exits
106:  Releases memory and sets object instances to nil (null)
107:  */
108: - (void)onExit:(NSNotification *)notification{
109:     NSLog(@"Leap: Exited");
110: }
111: /**
112:  LeapMotion SDK Delegate Callback
113:  OnFrame Event notifies the application that an incoming frame has been processed and the data can be
used to control the application
114:  */
115: - (void)onFrame:(NSNotification *)notification{
116:     LeapController *aController = (LeapController *)[notification object];
117:     // Get the most recent frame and report some basic information
118:     LeapFrame *frame = [aController frame:0];
119:
120:     //Try and find the same one as last time.
121:     if ([[frame pointables] count] != 0) {
122:         NSArray* leapPointables = [frame pointables];
123:
124:         LeapPointable* tool;
125:         if (lastTag != -1){
126:             for (LeapPointable* pointable in leapPointables){
127:                 if (lastTag == pointable.id){
128:                     tool = pointable;
129:                     lastTag = pointable.id;
130:                     break;
131:                 }
132:             }
133:
134:             //Find a new point able
135:             if (tool == nil){
136:                 tool = [self pointableClosestToScreen:leapPointables];
137:                 lastTag = tool.id;
138:             }
139:         }else{
140:             //Find a new pointable
141:             tool = [self pointableClosestToScreen:leapPointables];
142:             lastTag = tool.id;
143:         }
144:
145:         //Get the screen
146:         LeapVector* normalized = [leapScreen intersect:tool normalize:YES clampRatio:2.0];
147:
148:         if ([leapScreen isValid]){
149:             double x = normalized.x * [leapScreen widthPixels];
150:             double y = normalized.y * [leapScreen heightPixels];
151:
152:             CGPoint pointer = CGPointMake(x, y);
153:
154:             //Convert to Local coordinates from Screen Coordinates
155:             CCDirector* director = [CCDirector sharedDirector];
156:             NSPoint var = [director.view.window convertScreenToBase:pointer];
```

```
157:
158:            //Logging
159:            //NSLog(@"x %0.0f y %0.0f z %0.0f Pointer: x %0.0f y %0.0f ", x, y,tool.tipPosition.z, var
.x, var.y);
160:            //SimplePoint* simplePoint = [[SimplePoint alloc] initWithPosition:var withZ:tool.tipPosit
ion.z];
161:            //[[NSNotificationCenter defaultCenter] postNotificationName:@"CoordHUDUpdate" object:simp
lePoint];
162:
163:            if (gameSettings.depthOpacityMode){
164:                float opacity = [self opacityPercentage:tool.tipPosition.z];
165:                //Update the controls
166:                [controlsLayer updateOpacitySlider:opacity];
167:            }
168:
169:            if (inputMode == kDepthMode){
170:
171:                if (tool.tipPosition.z > 0){
172:                    painting = FALSE;
173:                }else{
174:                    painting = TRUE;
175:                }
176:            }
177:
178:            //Update the HUD View
179:            [self.hudLayer toolMoved:var toolID:[NSString stringWithFormat:@"%0.0d",tool.id]];
180:            if (painting){
181:                [self movedToolTexture:var tool:tool];
182:            }else{
183:               // NSLog(@"Not Painting");
184:            }
185:
186:        }else{
187:            NSLog(@"Leap Screen is invalid");
188:        }
189:    }else{
190:        NSLog(@"No frame");
191:        //Remove the marker from the HUD view
192:        if (currentPointable != nil) {
193:
194:            [self endLineDrawingTexture:currentPoint tool:currentPointable];
195:            [self.hudLayer endTrackingTool];
196:        }
197:
198:        lastTag = -1;
199:
200:        framesSinceLastFound ++;
201:        if (framesSinceLastFound > kMaxFrames){
202:
203:            framesSinceLastFound = 0;
204:        }
205:    }
206: }
207:
208: #pragma mark - TextureScene
209:
210: /** Moves the tool on the screen when painting */
211: - (void)movedToolTexture:(CGPoint)point tool:(LeapPointable*)pointable{
212:
213:    if (currentPointable != nil){
214:
215:        [self moveLineDrawingTexture:point tool:pointable];
216:        currentPointable = pointable;
217:    }else{
218:        [self beginLineDrawingTexture:point tool:pointable];
219:        currentPointable = pointable;
220:    }
221: }
222:
223: /** Begin drawing to the canvas
224:  @param point is the current coordinate the LeapPointable is interescting with the screen
225:  @param pointable is a reference to the pointable currently drawing
226:  */
227: - (void)beginLineDrawingTexture:(CGPoint)point tool:(LeapPointable*)pointable{
228:
229:    [self.textureScene beginDraw:point withZ:pointable.tipPosition.z];
230:    currentPoint = point;
231: }
232: /** Update drawing with a moved image on the canvas
233:   @param point is the current coordinate the LeapPointable is interescting with the screen
```

```
234:   @param pointable is a reference to the pointable currently drawing
235:  */
236: - (void)moveLineDrawingTexture:(CGPoint)point tool:(LeapPointable*)pointable{
237:
238:     [self.textureScene updateDraw:point withZ:pointable.tipPosition.z];
239:     currentPoint = point;
240: }
241: /** End the drawing
242:   @param point is the current coordinate the LeapPointable is interescting with the screen
243:   @param pointable is a reference to the pointable currently drawing
244:  */
245: - (void)endLineDrawingTexture:(CGPoint)point tool:(LeapPointable*)pointable{
246:     [self.textureScene endDraw:point];
247:     currentPointable = nil;
248: }
249:
250: #pragma mark - Keyboard Events
251:
252: /** Change Input Mode */
253: - (void)changeMode:(InputMode)mode{
254:     //NSLog(@"Changemode");
255:     inputMode = mode;
256:     gameSettings.inputMode = mode;
257: }
258: /** Change Paiting state
259:  @param paintState changes the painting sate
260:  */
261: - (void)painting:(BOOL)paintingState{
262:     painting = paintingState;
263:     gameSettings.painting = paintingState;
264: }
265: #pragma mark - ControlsDelegate
266:
267: /** Change the color of the brush
268:  Updates the HUD Layer and the Texture Layer
269:   @param color is the color to be changed
270:  */
271:
272: - (void)changeColorControl:(ccColor3B)color{
273:     [self.hudLayer changeColor:color];
274:     [self.textureScene changeColor:color];
275:
276: }
277: /** Change the thickness of the brush
278:  Updates the HUD Layer and the Texture Layer
279:  @param value is the thinkness(width) value
280:  */
281: - (void)changeThicknessControl:(float)value{
282:     [self.hudLayer changeScale:value];
283:     [self.textureScene changeScale:value];
284: }
285: /** Change the brush type
286:  Updates the HUD Layer and the Texture Layer
287:  @param brushname is the name of the brush to be changed
288:  */
289: - (void)changeBrushControl:(NSString *)brushname{
290:
291:     [self.hudLayer changeBrush:brushname];
292:     [self.textureScene changeBrush:brushname];
293: }
294: /** Change the opacity of the brush
295:  Updates the HUD Layer and the Texture Layer
296:  @param value is the opacity value
297:  */
298: - (void)changeOpacityControl:(float)value{
299:     [self.textureScene changeOpacity:value];
300: }
301:
302: /** Clears the drawing */
303: - (void)clearDrawing{
304:     [self.textureScene clearDrawing];
305:     //**Turns off eraser mode if it is on
306:     if (gameSettings.eraserMode){
307:         gameSettings.eraserMode = false;
308:
309:         //Update texture mode and update Controls layer
310:     }
311: }
312: /** Update the eraser mode
313:  Updates the HUD Layer and the Texture Layer
```

```
 314:  @param mode is the current state of the eraser
 315:  */
 316: - (void)eraserMode:(bool)mode{
 317:
 318:     [self.hudLayer erasingMode:mode];
 319:     [self.textureScene erasingMode:mode];
 320:
 321: }
 322:
 323: /** Return the Opacity value based on Z position */
 324: - (float)opacityPercentage:(float)value{
 325:     //NSLog(@"value %0.0f", value);
 326:     if (value < kOpMinRange){
 327:         return kOpMax;
 328:     }else if(value > kOpMaxRange){
 329:         return kOpMin;
 330:     }else {
 331:         float percentage = [self findPecentageDifference:kOpMaxRange withMin:kOpMinRange withValue:val
ue];
 332:         percentage = 100 - percentage;
 333:         return percentage;
 334:     }
 335: }
 336: /** Find the percentage between two numbers */
 337: - (float)findPecentageDifference:(float)max withMin:(float)min withValue:(float)value{
 338:     return (value - min)/(max - min)*100;
 339: }
 340: /**
 341:  Using all the pointables, gets the closest one to the screen
 342:  @param pointables is an array of pointables currently observered by the LeapMotion
 343:  @return pointable that is closest by the screen
 344:  */
 345: - (LeapPointable*)pointableClosestToScreen:(NSArray*)pointables{
 346:     LeapPointable* closestPointable;
 347:     for (LeapPointable*pointable in pointables){
 348:
 349:         //Check for the first iteration that the closest is not equal to nil
 350:         if (closestPointable != nil){
 351:             if (closestPointable.tipPosition.z > pointable.tipPosition.z){
 352:                 closestPointable = pointable;
 353:             }
 354:         }else{
 355:             closestPointable = pointable;
 356:         }
 357:     }
 358:     return closestPointable;
 359: }
 360:
 361: /**
 362:  Find the closest LeapPointable to the current last vector
 363:  @param leapVector is the position of the last pointbale
 364:  @param pointables is an array of pointables currently observered by the LeapMotion
 365:  @return LeapPointable closest to a leapVector
 366:  */
 367: - (LeapPointable*)pointableClosestToVector:(LeapVector*)leapVector withPointables:(NSArray*)pointables
{
 368:     LeapPointable* closestPointable;
 369:     //Check to make sure there is atleast one object in the array
 370:
 371:     //if the array is empty, throw an exception
 372:     if ([pointables count] == 0){
 373:         NSLog(@"Cannot pass item 0 array");
 374:         return nil;
 375:     }
 376:     //If there is only one object in the array, return it
 377:     else if ([pointables count] == 1){
 378:         return [pointables objectAtIndex:0];
 379:     }else{
 380:         //Get the distance for the first point
 381:         float minDistance = 0;
 382:         closestPointable = [pointables objectAtIndex:0];
 383:         minDistance = [leapVector distanceTo:closestPointable.tipPosition];
 384:
 385:         for (int i = 1; i < [pointables count]; i++){
 386:
 387:             LeapPointable* point = [pointables objectAtIndex:i];
 388:             float distance = [leapVector distanceTo:point.tipPosition];
 389:             if ( distance < minDistance){
 390:                 minDistance = distance;
 391:                 closestPointable = point;
```

```
            }
        }
    }
    return closestPointable;
}
@end
```

```
 1: //
 2: //  GameScene.h
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 4/1/13.
 6: //
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10: #import "cocos2d.h"
11: #import "HUDLayer.h"
12: #import "GameManager.h"
13: #import "LeapObjectiveC.h"
14: #import "SketchRenderTextureScene.h"
15: #import "BackgroundLayer.h"
16: #import "ControlsLayer.h"
17:
18: /**
19:  GameScene
20:  Initializes and assembles all of the layers and gameobjects into the GameManager
21:  */
22: @interface GameScene : CCScene
23: /**
24:  Scene initializes each object and assigns interlinking pointers and delegates to each class
25:  @return scene for CCDirector to begin running
26:  */
27: +(CCScene *) scene;
28: @end
```

```
 1: //
 2: //  GameScene.m
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 4/1/13.
 6: //
 7: //
 8:
 9: #import "GameScene.h"
10:
11: @implementation GameScene
12:
13: +(CCScene *) scene
14: {
15:         // 'scene' is an autorelease object.
16:     GameManager*scene = [GameManager node];
17:
18:         HUDLayer* hudLayer = [HUDLayer node];
19:     BackgroundLayer* backgroundLayer = [BackgroundLayer node];
20:     ControlsLayer* controlsLayer = [ControlsLayer node];
21:     SketchRenderTextureScene* textureScene = [SketchRenderTextureScene node];
22:
23:     //setup delegates
24:     hudLayer.delegate = scene;
25:     controlsLayer.delegate = scene;
26:
27:     // add layer as a child to scene
28:     [scene addChild:backgroundLayer z:0];
29:     [scene addChild:controlsLayer z:3];
30:         [scene addChild:hudLayer z:5];
31:     [scene addChild:textureScene z:2];
32:     scene.hudLayer = hudLayer;
33:     scene.backgroundLayer = backgroundLayer;
34:     scene.controlsLayer = controlsLayer;
35:     scene.textureScene = textureScene;
36:
37:         // return the scene
38:         return scene;
39: }
40: @end
```

```
    1: //
    2: //  GameSettings.h
    3: //  LeapPuzz
    4: //
    5: //  Created by cj on 4/16/13.
    6: //
    7: //
    8:
    9: #import <Foundation/Foundation.h>
   10:
   11: #define kVelMax 1000
   12: #define kVelMin 0
   13: #define kOpMinRange -80
   14: #define kOpMaxRange 120
   15: #define kOpMin 0
   16: #define kOpMax 100
   17: #define kNormalizedVelMax 15
   18: #define kNormalizedVelMin 0
   19: #define kMaxFrames 1000
   20: extern int const BLOCK_SIZE;
   21:
   22: typedef enum {
   23:     kPressKeyMode,
   24:     kDepthMode,
   25: } InputMode;
   26:
   27: /**
   28:  GameSettings is a globally shared class instance which tracks all the game settings.
   29:  This class can be accessed by any object in the game.
   30:  */
   31: @interface GameSettings : NSObject
   32: @property (nonatomic,readwrite) BOOL depthOpacityMode;      /**< depthOpacityMode controls use of z ax
is control of opacity */
   33: @property (nonatomic,readwrite) BOOL painting;      /**< painting indicates wether or not the applicat
ion is painting at that moment*/
   34: @property (nonatomic,readwrite) BOOL eraserMode;           /**< eraserMode controls erasing on drawin
g canvas */
   35: @property (nonatomic,readwrite) InputMode inputMode;       /**< inputMode controller input mode for l
eapmotion */
   36: /** Singleton
   37:  Intiailizes and Returns a shared instance of the class
   38:  @return sharedInstance of the class.
   39:  */
   40: + (GameSettings *)sharedInstance;
   41: @end
```

```
 1: //
 2: //  GameSettings.m
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 4/16/13.
 6: //
 7: //
 8:
 9: #import "GameSettings.h"
10:
11: //Constants
12: int const  BLOCK_SIZE = 128;
13:
14: @implementation GameSettings
15: @synthesize depthOpacityMode;
16: @synthesize eraserMode;
17: @synthesize inputMode;
18: @synthesize painting;
19:
20: /** Singleton SharedInstance
21:  Intiailizes and Returns a shared instance of the class
22:  */
23: + (GameSettings *)sharedInstance{
24:     static GameSettings *sharedInstance;
25:     @synchronized(self)
26:     {
27:         if (!sharedInstance)
28:             sharedInstance = [[GameSettings alloc] init];
29:         return sharedInstance;
30:     }
31: }
32:
33: /**
34:  Initialize the class and sets the default values
35:  */
36: - (id)init{
37:     if (self = [super init]) {
38:         // Init Defaults
39:         self.depthOpacityMode = false;
40:         self.painting = false;
41:     }
42:     return self;
43: }
44: @end
```

```
    1: //
    2: //  HUDLayer.h
    3: //  LeapPuzz
    4: //
    5: //  Created by cj on 4/1/13.
    6: //
    7: //
    8:
    9: #import <Foundation/Foundation.h>
   10: #import "cocos2d.h"
   11: #import "LPTool.h"
   12: #import "LeapObjectiveC.h"
   13: #import "SimplePoint.h"
   14: #import "GameSettings.h"
   15:
   16:
   17: /** HUD Delegate Protocol
   18:  User inferface controls for operating buttons, switches, sliders
   19:  */
   20: @protocol HUDDelegate <NSObject>
   21: /**
   22:  Calls back to notify a new input mode has been selected by the keyboard interface
   23:  @param mode is the state of the input mode
   24:  */
   25: - (void)changeMode:(InputMode)mode;
   26: /**
   27:  Calls back to notify a new change in painting state
   28:  @param paintingState
   29:  */
   30: - (void)painting:(BOOL)paintingState;
   31:
   32: @end
   33:
   34: /** HUD Layer
   35:  Tracks the position of the LeapCursor on the screen
   36:  */
   37: @interface HUDLayer : CCLayer  {
   38:     NSString* primaryToolID;    /**< primaryToolID stores the id tag to the pointable in reference*/
   39:     LPTool* primaryTool;        /**< primaryTool points to the current pointable  object*/
   40:     InputMode inputMode;        /**< inputMode is the current mode of input*/
   41:
   42:     ccColor3B lastColor;        /**< lastColor is the lastColor to be selected */
   43:     ccColor3B previousColor;    /**< previousColor is the color before the lastcolor to be selected */
   44:     NSString* lastBrush;        /**< lastBrush is last brush to be selected */
   45:     float lastScale;            /**< lastScale is last scale to be selected */
   46:
   47:     CCSprite* paintingIndicator; /**< paintingIndicator shows the state at which the object is current
ly paintg */
   48:     BOOL eraseMode;             /**< eraseMode determines weather the pointable is painting or erasing
 */
   49:
   50:
   51:     GameSettings* gameSettings; /**< gameSettings singleton to global seetings*/
   52:
   53:
   54: }
   55:
   56: @property (nonatomic, weak) id <HUDDelegate> delegate; /**< colorLabel displays name of color in hash
value */
   57: @property (nonatomic, strong) CCLabelTTF* xyzcoords; /**< xyzcoords is the X,Y,Z coordinates in string
 form for displaying on the HUD in real-time for debugging */
   58:
   59: /**
   60:  ToolMoved updates the last known tracked position of the tool.
   61:  @param point is the coordinate location on the screen in which pointable interesects
   62:  @param toolid is LeapSDK provided tool id of the tool moving
   63:  */
   64: - (void)toolMoved:(CGPoint)point toolID:(NSString*)toolid;
   65: /**
   66:  StartTrackingTool begins the process of tracking a tool starting with a new path
   67:  @param point is the coordinate location on the screen in which pointable interesects
   68:  @param toolid is LeapSDK provided tool id of the tool moving
   69:  */
   70: - (void)startTrackingTool:(CGPoint)point toolID:(NSString*)toolid;
   71: /**
   72:  MoveTrackingTool updates the position and path of a tool.
   73:  @param point is the coordinate location on the screen in which pointable interesects
   74:  @param toolid is LeapSDK provided tool id of the tool moving
   75:  */
   76: - (void)moveTrackingTool:(CGPoint)point toolID:(NSString*)toolid;
```

```
77: /**
78:  EndTracking tool singles the end of the tool being tracked.
79:  The tool may be lost or no longer drawing
80:  */
81: - (void)endTrackingTool;
82:
83:
84: - (void)changeColor:(ccColor3B)color;
85: - (void)changeBrush:(NSString*)brushname;
86: - (void)changeScale:(float)size;
87: - (void)erasingMode:(BOOL)mode;
88:
89: @end
```

```
    1: //
    2: //  HUDLayer.m
    3: //  LeapPuzz
    4: //
    5: //  Created by cj on 4/1/13.
    6: //
    7: //
    8:
    9: #import "HUDLayer.h"
   10:
   11: @implementation HUDLayer
   12: @synthesize delegate;
   13: @synthesize xyzcoords;
   14: - (id)init
   15: {
   16:         if ((self = [super init]))
   17:         {
   18:                 // Get window size
   19:                 CGSize size = [[CCDirector sharedDirector] winSize];
   20:
   21:                 // Add a button which takes us back to HelloWorldScene
   22:
   23:                 // Create a label with the text we want on the button
   24:                 CCLabelTTF *label = [CCLabelTTF labelWithString:@"Tap Here" fontName:@"Helvetica" font
Size:32.0];
   25:
   26:                 // Create a button out of the label, and tell it to run the "switchScene" method
   27:                 CCMenuItem *button = [CCMenuItemLabel itemWithLabel:label target:self selector:@select
or(testing:)];
   28:
   29:                 // Add the button to a menu - "nil" terminates the list of items to add
   30:                 CCMenu *menu = [CCMenu menuWithItems:button, nil];
   31:
   32:                 // Place the menu in center of screen
   33:                 [menu setPosition:ccp(size.width / 2, size.height / 2)];
   34:
   35:         lastColor = ccWHITE;
   36:         lastBrush = @"roundbrush.png";
   37:         lastScale = 1.0;
   38:
   39:         eraseMode = false;
   40:
   41:                 // Finally add the menu to the layer
   42:                 //[self addChild:menu];
   43: #ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
   44:                 self.isTouchEnabled = YES;
   45:                 self.isAccelerometerEnabled = YES;
   46: #elif defined(__MAC_OS_X_VERSION_MAX_ALLOWED)
   47:                 self.isMouseEnabled = YES;
   48:         self.isKeyboardEnabled= YES;
   49: #endif
   50:         inputMode = kDepthMode;
   51:
   52:         /*
   53:         self.xyzcoords = [CCLabelTTF labelWithString:@"Coords" fontName:@"Helvetica" fontSize:16.0];
   54:         self.xyzcoords.position = ccp(size.width / 2, 50);
   55:         [self addChild:self.xyzcoords];
   56:
   57:         [[NSNotificationCenter defaultCenter] addObserver:self
   58:                                                  selector:@selector(handleHUDCoordUpdate:)
   59:                                                      name:@"CoordHUDUpdate"
   60:                                                    object:nil];
   61:
   62:          */
   63:
   64:         }
   65:         return self;
   66: }
   67:
   68:
   69: //Add the sprite hud
   70: - (LPTool*)addLPTool:(CGPoint)p objectID:(NSString*)objectID withBrushName:(NSString*)brushname{
   71:
   72:         LPTool *sprite = [LPTool spriteWithFile:brushname];
   73:
   74:     [self addChild:sprite];
   75:
   76:     sprite.updated = TRUE;
   77:     sprite.toolID = objectID;
   78:     [sprite setScale:lastScale];
```

```
    sprite.position = ccp( p.x, p.y);
    sprite.color = lastColor;

    return sprite;
}

/* Tool Moved */
- (void)toolMoved:(CGPoint)point toolID:(NSString*)toolid{

    if (primaryTool == nil){
        [self startTrackingTool:point toolID:toolid];
    }else{
        [self moveTrackingTool:point toolID:toolid];
    }
}

/* Start Tracking Tool */
- (void)startTrackingTool:(CGPoint)point toolID:(NSString*)toolid{
    if (primaryTool == nil){
        primaryTool = [self addLPTool:point objectID:toolid withBrushName:lastBrush];
    }
}

/* Move Tracking Tool*/
- (void)moveTrackingTool:(CGPoint)point toolID:(NSString*)toolid{

    //Create tool if it does not exist
    if (primaryTool == nil){
        primaryTool = [self addLPTool:point objectID:toolid withBrushName:lastBrush];
    }else{
        //Update since it does exist
        primaryTool.position =  point;
        if ([toolid isNotEqualTo:primaryTool.toolID]){
            primaryTool.toolID = toolid;
        }
    }
}

/* End Trackingn Tool */
- (void)endTrackingTool{
    if (primaryTool != nil){
        [self removeChild:primaryTool cleanup:YES];
        primaryTool = nil;
    }
}


//Key up event
-(BOOL) ccKeyUp:(NSEvent*)event{

    unichar ch = [event keyCode];

    if (inputMode == kPressKeyMode){
        if ( ch == 49){
            [self.delegate painting:FALSE];
        }
    }

    if ( ch == 18){
        //change to space bar press mode
        inputMode = kPressKeyMode;
        [self.delegate changeMode:inputMode];
    }else if(ch == 19){
        //Change to depth mode
        inputMode = kDepthMode;
        [self.delegate changeMode:inputMode];
    }

    return YES;
}
//Key down event
-(BOOL) ccKeyDown:(NSEvent*)event{
    unichar ch = [event keyCode];

    if (inputMode == kPressKeyMode){
        if ( ch == 49){
            [self.delegate painting:TRUE];
        }
    }
    return YES;
```

```
159: }
160:
161: - (void)changeColor:(ccColor3B)color{
162:
163:
164:
165:     if(primaryTool != nil){
166:
167:         [primaryTool setColor:color];
168:
169:     }
170:     lastColor = color;
171: }
172:
173: - (void)changeBrush:(NSString*)brushname{
174:
175:     lastBrush = brushname;
176:     if (primaryTool != nil){
177:         //Save important data
178:         CGPoint lastlocation = primaryTool.position;
179:         NSString* toolid = [primaryTool.toolID copy];
180:
181:         //Remove Tool
182:         [self removeChild:primaryTool cleanup:YES];
183:
184:         //Add it back
185:         primaryTool = [self addLPTool:lastlocation objectID:toolid withBrushName:lastBrush];
186:     }
187:
188: }
189:
190:
191: - (void)changeScale:(float)size{
192:
193:     lastScale = size;
194:     if(primaryTool != nil){
195:
196:         [primaryTool setScale:size];
197:     }
198: }
199:
200:
201: - (void)erasingMode:(BOOL)mode{
202:
203:     eraseMode = mode;
204:
205:     //turn Erasing Mode on
206:     if (mode){
207:         previousColor = lastColor;
208:         lastColor = ccRED;
209:         [primaryTool setColor:ccRED];
210:     }else{
211:         //Turn erasing mode off
212:         lastColor = previousColor;
213:         [primaryTool setColor:lastColor];
214:     }
215:
216: }
217:
218: - (void)updateCoordsHUDWithX:(float)x withY:(float)y withZ:(float)z{
219:
220:     self.xyzcoords.string = [NSString stringWithFormat:@"Coords x: %0.0f, y %0.0f, z %0.0f",x,y,z];
221:
222: }
223:
224: - (void)handleHUDCoordUpdate:(id)sender{
225:
226:     NSNotification* note = sender;
227:     //LEDColor* ledColor = note.object;
228:
229:     SimplePoint* point = note.object;
230:
231:     [self updateCoordsHUDWithX:point.x withY:point.y withZ:point.z];
232:
233:
234: }
235:
236:
237: @end
```

```
 1: //
 2: //  LPCCControlButtonVariableSize.h
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 4/9/13.
 6: //
 7: //
 8: #import "cocos2d.h"
 9: #import "CCControlExtension.h"
10: /**
11:  LPCCControlButtonVariableSize Extends CCLayer to have a customizable control button interface
12:  */
13: @interface LPCCControlButtonVariableSize : CCLayer
14:
15: /** Creates and return a button with a default background and title color. */
16: - (CCControlButton *)standardButtonWithTitle:(NSString *)title;
17: @end
```

```
    1: //
    2: //  LPCCControlButtonVariableSize.m
    3: //  LeapPuzz
    4: //
    5: //  Created by cj on 4/9/13.
    6: //
    7: //
    8:
    9: #import "LPCCControlButtonVariableSize.h"
   10:
   11: @implementation LPCCControlButtonVariableSize
   12: - (id)init
   13: {
   14:     if ((self = [super init]))
   15:     {
   16:         CGSize screenSize = [[CCDirector sharedDirector] winSize];
   17:
   18:         // Defines an array of title to create buttons dynamically
   19:         NSArray *stringArray = [NSArray arrayWithObjects:@"Hello",@"Variable",@"Size",@"!", nil];
   20:
   21:         CCNode *layer = [CCNode node];
   22:         [self addChild:layer z:1];
   23:
   24:         double total_width = 0, height = 0;
   25:
   26:         // For each title in the array
   27:         for (NSString *title in stringArray)
   28:         {
   29:             // Creates a button with this string as title
   30:             CCControlButton *button = [self standardButtonWithTitle:title];
   31:             [button setPosition:ccp (total_width + button.contentSize.width / 2, button.contentSize.he
ight / 2)];
   32:             [layer addChild:button];
   33:
   34:             // Compute the size of the layer
   35:             height = button.contentSize.height;
   36:             total_width += button.contentSize.width;
   37:         }
   38:
   39:         [layer setAnchorPoint:ccp (0.5, 0.5)];
   40:         [layer setContentSize:CGSizeMake(total_width, height)];
   41:         [layer setPosition:ccp(screenSize.width / 2.0f, screenSize.height / 2.0f)];
   42:
   43:         // Add the black background
   44:         CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
   45:         [background setContentSize:CGSizeMake(total_width + 14, height + 14)];
   46:         [background setPosition:ccp(screenSize.width / 2.0f, screenSize.height / 2.0f)];
   47:         [self addChild:background];
   48:     }
   49:     return self;
   50: }
   51:
   52: #pragma mark -
   53: #pragma CCControlButtonTest_HelloVariableSize Public Methods
   54: #pragma CCControlButtonTest_HelloVariableSize Private Methods
   55:
   56: - (CCControlButton *)standardButtonWithTitle:(NSString *)title{
   57:     /** Creates and return a button with a default background and title color. */
   58:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
   59:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
   60:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:title fontName:@"Marker Felt" fontSize:30];
   61:     [titleButton setColor:ccc3(159, 168, 176)];
   62:     CCControlButton *button = [CCControlButton buttonWithLabel:titleButton backgroundSprite:background
Button];
   63:     [button setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted];
   64:     [button setTitleColor:ccWHITE forState:CCControlStateHighlighted];
   65:
   66:     return button;
   67: }
   68:
   69:
   70: @end
```

```
 1: //
 2: //  LPLine.h
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 4/2/13.
 6: //
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10: /**
11:  LPLine is tracks the points in one line from beginning to end
12:  */
13: @interface LPLine : NSObject {
14:
15: }
16:
17: @property (nonatomic, strong) NSMutableArray* points;  /**< points is a an array of points for the lin
e */
18: @property (nonatomic, readwrite) float width;    /**< width is a constant width for the line */
19:
20: @end
```

```
 1: //
 2: //  LPLine.m
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 4/2/13.
 6: //
 7: //
 8:
 9: #import "LPLine.h"
10:
11: @implementation LPLine
12:
13: @synthesize points;
14: @synthesize width;
15:
16: - (id)init
17: {
18:         if ((self = [super init]) == nil) {
19:         self.points = [[NSMutableArray alloc] init];
20:                 self.width = 1.0f;
21:         }
22:     return self;
23: }
24:
25: @end
```

```objc
//
//  LPPoint.h
//  LeapPaint
//
//  Created by cj on 5/8/13.
//  Copyright (c) 2013 cjdesch. All rights reserved.
//

#import <Foundation/Foundation.h>



/**
 LPLinePoint is a plotted point for drawing onto the canvas
 */
@interface LPLinePoint : NSObject

@property (nonatomic, readwrite) float x; /**< x coordinate */
@property (nonatomic, readwrite) float y; /**< y coordinate */
@property (nonatomic, readwrite) float width; /**< width of the point */

/**
 * Init constructor with existing point to create with no width
 * @param p an point (x,y)
 * @return object instance
 */
- (id)initWithPosition:(CGPoint)p;
/**
 * Init constructor with x and y values with no width
 * @param xVal coordinate value
 * @param yVal coordinate value
 * @return object instance
 */
- (id)initWithX:(float)xVal withY:(float)yVal;
/**
 * Init constructor with existing point with width
 * @param p a point (x,y)
 * @param wVal width of the point
 * @return object instance
 */
- (id)initWithPosition:(CGPoint)p withWidth:(float)wVal;
/**
 * Init constructor with x and y values with width
 * @param xVal coordinate value
 * @param yVal coordinate value
 * @param wVal width of the point
 * @return object instance
 */
- (id)initWithX:(float)xVal withY:(float)yVal withWidth:(float)wVal;

/**
 * Returns point based on x and y
 * @return CGPoint
 */
- (CGPoint)point;
@end
```

```objc
//
//  LPPoint.m
//  LeapPaint
//
//  Created by cj on 5/8/13.
//  Copyright (c) 2013 cjdesch. All rights reserved.
//

#import "LPLinePoint.h"

@implementation LPLinePoint
@synthesize x;
@synthesize y;
@synthesize width;

/** init 2d point with CGPoint */
- (id)initWithPosition:(CGPoint)p{
    if (self = [super init]) {

        self.x = p.x;
        self.y = p.y;
        self.width = 0.0f;


    }
    return self;
}
/** Init Point with 2 separate values */
- (id)initWithX:(float)xVal withY:(float)yVal{
    if (self = [super init]) {

        self.x = xVal;
        self.y = yVal;
        self.width = 0.0f;
    }
    return self;
}
/** Init  point with CGPoint and width Value */
- (id)initWithPosition:(CGPoint)p withWidth:(float)wVal{
    if (self = [super init]) {
        self.x = p.x;
        self.y = p.y;
        self.width = wVal;
    }
    return self;
}
/** Init Point with x and y values with width*/
- (id)initWithX:(float)xVal withY:(float)yVal withWidth:(float)wVal{
    if (self = [super init]) {
        self.x = xVal;
        self.y = yVal;
        self.width = wVal;
    }
    return self;
}
/** Return the CGPoint type from the object */
- (CGPoint)point{
    return CGPointMake(self.x, self.y);
}
@end
```

```
 1: //
 2: //  LPTool.h
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 3/29/13.
 6: //
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10: #import "cocos2d.h"
11:
12: /**
13:  Extends CCSprite object with two properties for tracking sprites with pointable objects
14:  */
15: @interface LPTool : CCSprite
16: @property (nonatomic, strong) NSString* toolID; /**< toolID is the ID number assigned by the LeapMotio
n SDK */
17: @property (nonatomic, readwrite) BOOL updated; /**< updated is if the sprite has been updated in that
frame.*/
18: @end
```

```
 1: //
 2: //  LPTool.m
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 3/29/13.
 6: //
 7: //
 8:
 9: #import "LPTool.h"
10:
11: @implementation LPTool
12: @end
```

```
 1: //
 2: //  main.m
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 5/7/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Cocoa/Cocoa.h>
10:
11: int main(int argc, char *argv[])
12: {
13:     return NSApplicationMain(argc, (const char **)argv);
14: }
```

```
 1: //
 2: //  SimplePoint.h
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 2/19/13.
 6: //
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10: #import "cocos2d.h"
11: /**
12:  * 2D or 3D space coordinate for temporarily maniulapting points
13:  *
14:  */
15: @interface SimplePoint : NSObject
16: @property (nonatomic, readwrite) float x; /**< x coordinate */
17: @property (nonatomic, readwrite) float y; /**< y coordinate */
18: @property (nonatomic, readwrite) float z; /**< z coordinate */
19: @property (nonatomic, readwrite) BOOL is3d; /**< is3d is 2d or 3d point type */
20: /**
21:  * Init constructor with existing point to create a 2d Point
22:  * @param p an point (x,y)
23:  * @return object instance
24:  */
25: - (id)initWithPosition:(CGPoint)p;
26: /**
27:  * Init constructor with x and y values to create a 2d point
28:  * @param xVal coordinate value
29:  * @param yVal coordinate value
30:  * @return object instance
31:  */
32: - (id)initWithX:(float)xVal withY:(float)yVal;
33: /**
34:  * Init constructor with existing point to create a 3d Point
35:  * @param p a point (x,y)
36:  * @param zVal coordinateValue
37:  * @return object instance
38:  */
39: - (id)initWithPosition:(CGPoint)p withZ:(float)zVal;
40: /**
41:  * Init constructor with x, y and z values to create 3D point
42:  * @param xVal coordinate value
43:  * @param yVal coordinate value
44:  * @param zval coordinate value
45:  * @return object instance
46:  */
47: - (id)initWithX:(float)xVal withY:(float)yVal withZ:(float)zVal;
48:
49: /**
50:  * Returns point based on x and y
51:  * @return CGPoint
52:  */
53: - (CGPoint)point;
54: @end
```

```objc
//
//  SimplePoint.m
//  LeapPuzz
//
//  Created by cj on 2/19/13.
//
//

#import "SimplePoint.h"

@implementation SimplePoint

@synthesize x,y,z;
@synthesize is3d;


/** init 2d point with CGPoint */
- (id)initWithPosition:(CGPoint)p{
    if (self = [super init]) {

        self.x = p.x;
        self.y = p.y;
        self.z = 0.0f;
        self.is3d = false;

    }
    return self;
}

/** Init 2d Point with 2 separate values */
- (id)initWithX:(float)xVal withY:(float)yVal{

    if (self = [super init]) {

        self.x = xVal;
        self.y = yVal;
        self.z = 0.0f;
        self.is3d = false;

    }
    return self;
}


/** Init 3d point with CGPoint and z Value */
- (id)initWithPosition:(CGPoint)p withZ:(float)zVal{
    if (self = [super init]) {

        self.x = p.x;
        self.y = p.y;
        self.z = zVal;
        self.is3d = true;

    }
    return self;
}


/** Init 3d Point with 3 separate values */
- (id)initWithX:(float)xVal withY:(float)yVal withZ:(float)zVal{

    if (self = [super init]) {

        self.x = xVal;
        self.y = yVal;
        self.z = zVal;
        self.is3d = true;

    }
    return self;
}

/** Return the CGPoint type from the object */
- (CGPoint)point{
    return CGPointMake(self.x, self.y);
}


@end
```

```
 1: //
 2: //  Point.h
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 2/19/13.
 6: //
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10:
11: /**
12:  * 2D space coordinate for temporarily maniulapting points
13:  *
14:  */
15: @interface SimplePointObject : NSObject
16:
17: @property (nonatomic, readwrite) CGPoint point; /**< point is the X and Y coordinates */
18: /**
19:  * Init constructor with existing point to create a 2d Point
20:  * @param p an point consisting of (x,y)
21:  * @return object instance
22:  */
23: - (id)initWithPosition:(CGPoint)p;
24: /**
25:  * Init constructor with existing point to create a 2d Point
26:  * @param x is x axis coordinate
27:  * @param y is y axis coordinate
28:  * @return object instance
29:  */
30: - (id)initWithX:(float)x withY:(float)y;
31:
32: @end
```

```
 1: //
 2: //  Point.m
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 2/19/13.
 6: //
 7: //
 8:
 9: #import "SimplePointObject.h"
10:
11: @implementation SimplePointObject
12:
13:
14: - (id)initWithPosition:(CGPoint)p{
15:     if (self = [super init]) {
16:
17:         self.point = p;
18:
19:     }
20:     return self;
21: }
22:
23: - (id)initWithX:(float)x withY:(float)y{
24:
25:     if (self = [super init]) {
26:
27:         self.point = CGPointMake(x, y);
28:
29:     }
30:     return self;
31: }
32: @end
```

```
 1: //
 2: //  SketchRenderTextureScene.h
 3: //  Cocos2D-CCRenderTexture-Demo
 4: //
 5: //  Copyright (c) 2011 Steffen Itterheim.
 6: //      Distributed under MIT License.
 7: //
 8:
 9: #import "cocos2d.h"
10: #import "SimplePoint.h"
11: #import "GameSettings.h"
12: @interface SketchRenderTextureScene : CCLayer
13: {
14:         CCSprite* brush;
15:         NSMutableArray* touches;
16:
17:     ccColor3B lastColor;
18:     ccColor3B previousColor;
19:     NSString* lastBrush;
20:     float lastScale;
21:     bool eraseMode;
22: }
23:
24: @property (nonatomic,readwrite) float opacity;
25:
26: - (void)beginDraw:(CGPoint)point withZ:(float)z;
27: - (void)updateDraw:(CGPoint)point withZ:(float)z;
28: - (void)endDraw:(CGPoint)point;
29: - (void)changeColor:(ccColor3B)color;
30: - (void)changeBrush:(NSString*)brushname;
31: - (void)changeScale:(float)size;
32: - (void)changeOpacity:(float)o;
33: - (void)erasingMode:(BOOL)mode;
34: - (void)clearDrawing;
35: @end
```

```
    1: //
    2: //  SketchRenderTextureScene.m
    3: //  Cocos2D-CCRenderTexture-Demo
    4: //
    5: //  Copyright (c) 2011 Steffen Itterheim.
    6: //      Distributed under MIT License.
    7: //
    8:
    9: #import "SketchRenderTextureScene.h"
   10:
   11:
   12: @implementation SketchRenderTextureScene
   13: @synthesize opacity;
   14:
   15: -(id) init
   16: {
   17:         if ((self = [super init]))
   18:         {
   19:                 // create a simple rendertexture node and clear it with the color white
   20:
   21:         //target = [CCRenderTexture renderTextureWithWidth:s.width height: s.height pixelFormat:kCCTex
ture2DPixelFormat_RGBA8888];
   22:                 CGSize s = [CCDirector sharedDirector].winSize;
   23:
   24:         CCDirector* sharedDirector =[CCDirector sharedDirector];
   25:         CGSize frameSize = sharedDirector.view.frame.size;
   26:
   27:
   28:
   29:         float topbottombar = 300;
   30:         float sidebars = 300;
   31:
   32:
   33:
   34:
   35:         //CCSprite* imageBackground = [CCSprite spriteWithFile:@"squarebrush.png"] ;
   36:         //imageBackground set
   37:
   38:
   39:         CCRenderTexture* rtx = [CCRenderTexture renderTextureWithWidth:frameSize.width-sidebars height
: frameSize.height-topbottombar];
   40:                 [rtx clear:1.0f
   41:                                 g:1.0f
   42:                                 b:1.0f
   43:                                 a:1.0f];
   44:
   45:                 rtx.position = CGPointMake(s.width/2, s.height/2);
   46:                 [self addChild:rtx z:0 tag:1];
   47:
   48:
   49:
   50:
   51:                 //CCLabelTTF* label = [CCLabelTTF labelWithString:@"Drawing onto CCRenderTexture witho
ut clear" fontName:@"Arial" fontSize:16];
   52:                 //label.position = CGPointMake(240, 15);
   53:                 //label.color = ccGRAY;
   54:                 //[self addChild:label];
   55:
   56:                 // create and retain the brush sprite, but don't add it as child
   57:
   58:         lastColor = ccWHITE;
   59:         lastBrush = @"roundbrush.png";
   60:         lastScale = 1.0;
   61:
   62:         eraseMode = false;
   63:         self.opacity = 10;
   64:
   65:         [self addBrush:lastBrush];
   66:
   67:
   68:
   69:                 //brush.scale = 0.5f;
   70:
   71:                 // create the array holding the touches
   72:                 touches = [[NSMutableArray alloc] init];
   73:
   74:                 //[CCTouchDispatcher sharedDispatcher] addTargetedDelegate:self priority:0 swallowsTou
ches:NO];
   75:
   76:                 [self scheduleUpdate];
```

```
77:
78:         }
79:         return self;
80: }
81:
82: - (void)addBrush:(NSString*)brushName{
83:
84:     brush = [CCSprite spriteWithFile:brushName] ;
85:     [brush setScale:lastScale];
86:
87:
88:     if(eraseMode){
89:         //[brush setBlendFunc:(ccBlendFunc) { GL_ZERO,GL_ONE_MINUS_SRC_ALPHA }];
90:         [brush setBlendFunc:(ccBlendFunc) { GL_ONE,GL_ONE }];
91:
92:
93:
94:                 [brush setOpacity:80];
95:     }else{
96:         brush.color = lastColor;
97:         brush.opacity = opacity;
98:     }
99: }
100:
101: -(void) cleanup
102: {
103:         brush = nil;
104:         touches = nil;
105:
106:         [super cleanup];
107: }
108:
109: - (void)beginDraw:(CGPoint)point withZ:(float)z{
110:     //NSLog(@"Begin Draw");
111:     SimplePoint* simplePoint = [[SimplePoint alloc] initWithPosition:point withZ:z];
112:     [touches addObject:simplePoint];
113:
114: }
115: - (void)updateDraw:(CGPoint)point withZ:(float)z{
116:
117:       //  NSLog(@"update Draw");
118:     SimplePoint* simplePoint = [[SimplePoint alloc] initWithPosition:point withZ:z];
119:     [touches addObject:simplePoint];
120:
121: }
122: - (void)endDraw:(CGPoint)point {
123:         [touches removeAllObjects];
124: }
125:
126:
127: /*
128: -(BOOL) ccTouchBegan:(UITouch *)touch withEvent:(UIEvent *)event
129: {
130:         // add new touches to the array as they come in
131:         [touches addObject:touch];
132:         return YES;
133: }
134:
135: -(void) ccTouchEnded:(UITouch *)touch withEvent:(UIEvent *)event
136: {
137:         // must remove the touches that have ended or where cancelled
138:         [touches removeObject:touch];
139: }
140:
141: -(void) ccTouchCancelled:(UITouch *)touch withEvent:(UIEvent *)event
142: {
143:         [self ccTouchEnded:touch withEvent:event];
144: }
145:
146: */
147: -(void) setBrushColor:(int)color
148: {
149:         switch (color)
150:         {
151:                 default:
152:                 case 0:
153:                         brush.color = ccWHITE;
154:                         break;
155:                 case 1:
156:                         brush.color = ccGREEN;
```

```
  157:                         break;
  158:                 case 2:
  159:                         brush.color = ccRED;
  160:                         break;
  161:                 case 3:
  162:                         brush.color = ccc3(0, 255, 255);
  163:                         break;
  164:                 case 4:
  165:                         brush.color = ccBLUE;
  166:                         break;
  167:         }
  168: }
  169:
  170: -(void) update:(ccTime)delta
  171: {
  172:
  173:         CCRenderTexture* rtx = (CCRenderTexture*)[self getChildByTag:1];
  174:
  175:         // explicitly don't clear the rendertexture
  176:         [rtx begin];
  177:
  178:         //int color = 0;
  179:
  180:         // Since we store all current touches in an array, we can render a sprite at each touch locati
on
  181:         // even if the touch isn't moving. That way a continued press will increase the opacity of the
 sprite
  182:         // simply because the sprite is drawn repeatedly with low opacity at the same location.
  183:     NSArray* tempTouches = [[NSArray alloc] initWithArray:touches];
  184:         for (SimplePoint* touch in tempTouches)
  185:         {
  186:                 //CGPoint touchLocation = [director convertToGL:[touch locationInView:director.openGLV
iew]];
  187:         CGPoint touchLocation = [touch point];
  188:
  189:                 // the location must be converted to the rendertexture sprite's node space
  190:                 touchLocation = [rtx.sprite convertToNodeSpace:touchLocation];
  191:
  192:                 // because the rendertexture sprite is flipped along its Y axis the Y coordinate must
be flipped:
  193:                 touchLocation.y = rtx.sprite.contentSize.height - touchLocation.y;
  194:
  195:                 //CCLOG(@"touch: %.0f, %.0f", touchLocation.x, touchLocation.y);
  196:
  197:                 // set the brush at that location and render it
  198:                 brush.position = touchLocation;
  199:                 //[self setBrushColor:color++];
  200:                 [brush visit];
  201:         }
  202:
  203:
  204:
  205:         [rtx end];
  206:
  207:     [touches removeAllObjects];
  208: }
  209:
  210: - (void)changeColor:(ccColor3B)color{
  211:
  212:
  213:     if(brush != nil){
  214:
  215:         brush.color = color;
  216:
  217:     }
  218:     lastColor = color;
  219:
  220: }
  221: - (void)changeBrush:(NSString*)brushname{
  222:
  223:     lastBrush = brushname;
  224:     if (brush != nil){
  225:         //Save important data
  226:         CGPoint lastlocation = brush.position;
  227:         [self addBrush:lastBrush];
  228:         brush.position = lastlocation;
  229:     }
  230:
  231: }
  232:
```

```
233: - (void)changeScale:(float)size{
234:
235:     lastScale = size;
236:     if(brush != nil){
237:
238:         [brush setScale:size];
239:
240:     }
241: }
242:
243: - (void)changeOpacity:(float)o{
244:
245:     self.opacity = o;
246:     if (brush != nil){
247:
248:         brush.opacity = self.opacity;
249:     }
250:
251: }
252:
253: - (void)clearDrawing{
254:
255:         CCRenderTexture* rtx = (CCRenderTexture*)[self getChildByTag:1];
256:
257:         // explicitly don't clear the rendertexture
258: //      [rtx begin];
259: //    glClearColor(r, g, b, a);
260:   //  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
261:         //get rid of the mask
262:         // glColorMask(TRUE, TRUE, TRUE, FALSE);
263: //    [rtx end];
264:
265:     [rtx clear:1 g:1 b:1 a:0];
266:
267: }
268:
269:
270:
271: - (void)erasingMode:(BOOL)mode{
272:
273:     eraseMode = mode;
274:
275:     //turn Erasing Mode on
276:     if (mode){
277:         previousColor = lastColor;
278:         lastColor = ccRED;
279:
280:         CGPoint lastlocation = brush.position;
281:         [self addBrush:lastBrush];
282:         brush.position = lastlocation;
283:
284:     }else{
285:         //Turn erasing mode off
286:         lastColor = previousColor;
287:         CGPoint lastlocation = brush.position;
288:         [self addBrush:lastBrush];
289:         brush.position = lastlocation;
290:
291:     }
292:
293: }
294:
295:
296:
297:
298:
299: @end
```

```
 1: //
 2: //  Utility.h
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 4/24/13.
 6: //
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10:
11:
12:
13: /**
14:  Utility class provides common usage function throughout the application.
15:  */
16:
17: @interface Utility : NSObject {
18: }
19:
20: /**
21:  Generates a random number between two designated integers
22:  @param from is the bottom of the range
23:  @param to is the top of the range
24:  @return a random number between the from and to parameters
25:  */
26: + (int)getRandomNumberBetween:(int)from to:(int)to;
27: /**
28:  Generates a random number between 0 designated integer
29:  @param to is the top of the range
30:  @return a random number between 0 and to parameters
31:  */
32: + (int)getRandomUniformNumberUnder:(int)to;
33: /**
34:  Generates a random number between 0 designated integer
35:  @param to is the top of the range
36:  @return a random number between 0 and to parameters
37:  */
38: + (int)getRandomNumberUnder:(int)to;
39: //- (void) initRandomSeed(long firstSeed);
40: //float nextRandomFloat();
41: @end
```

```
 1: //
 2: //  Utility.m
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 4/24/13.
 6: //
 7: //
 8:
 9: #import "Utility.h"
10:
11: @implementation Utility
12:
13: /** returns random number within a range with defined upper and lower bounds */
14: + (int)getRandomNumberBetween:(int)from to:(int)to {
15:
16:     //Check that one isn't greater than the other
17:     //if so, flip them
18:
19:     return (int)from + arc4random() % (to-from+1);
20: }
21:
22: /** Returns a random number from 0 to an upper bound */
23: + (int)getRandomNumberUnder:(int)to{
24:     return (arc4random() % to);
25: }
26:
27:
28: /** Returns a Uniform Random Number from 0 to an upper bound */
29: + (int)getRandomUniformNumberUnder:(int)to{
30:     //Check if uniform available
31:     if (arc4random_uniform != NULL)
32:         return  arc4random_uniform (to);
33:     else
34:         return  (arc4random() % to);
35: }
36:
37:
38:
39:
40:
41: /*
42: static unsigned long seed;
43:
44: void initRandomSeed(long firstSeed)
45: {
46:     seed = firstSeed;
47: }
48:
49: float nextRandomFloat()
50: {
51:     return (((seed= 1664525*seed + 1013904223)>>16) / (float)0x10000);
52: }*/
53:
54:
55:
56: @end
```

Appendix B

Quartz2DPaint

Contents

```
 1: //
 2: //  AppDelegate.h
 3: //  Quartz2DPaint
 4: //
 5: //  Created by cj on 5/7/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Cocoa/Cocoa.h>
10:
11: @interface AppDelegate : NSObject <NSApplicationDelegate>
12:
13: @property (assign) IBOutlet NSWindow *window;
14:
15: @end
```

```
 1: //
 2: //  AppDelegate.m
 3: //  Quartz2DPaint
 4: //
 5: //  Created by cj on 5/7/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import "AppDelegate.h"
10:
11: @implementation AppDelegate
12:
13: - (void)applicationDidFinishLaunching:(NSNotification *)aNotification
14: {
15:     // Insert code here to initialize your application
16: }
17:
18: @end
```

```
 1: //
 2: //  Brush.h
 3: //  Paint
 4: //
 5:
 6:
 7:
 8:
 9: @class Canvas;
10:
11: @interface Brush : NSObject {
12:         // Information about the brush that's always used
13:         float                           mRadius;
14:         CGMutablePathRef        mShape;
15:         CGColorRef                      mColor;
16:         float                           mSoftness; // 0.0 - 1.0, 0.0 being hard, 1.0 be all soft
17:         BOOL                            mHard; // should have a hard edge?
18:
19:         // Cached information that's only used when actually tracking/drawing
20:         CGImageRef                      mMask;
21:         NSPoint                         mLastPoint;
22:         float                           mLeftOverDistance;
23: }
24:
25: - (void) mouseDown:(NSEvent *)theEvent inView:(NSView *)view onCanvas:(Canvas *)canvas;
26: - (void) mouseDragged:(NSEvent *)theEvent inView:(NSView *)view onCanvas:(Canvas *)canvas;
27: - (void) mouseUp:(NSEvent *)theEvent inView:(NSView *)view onCanvas:(Canvas *)canvas;
28:
29: @end
```

```
    1: //
    2: //  Brush.m
    3: //  Paint
    4: //
    5:
    6:
    7: #import "Brush.h"
    8: #import "Canvas.h"
    9: #import <QuartzCore/QuartzCore.h>
   10:
   11: @interface Brush (Private)
   12:
   13: - (NSPoint) canvasLocation:(NSEvent *)theEvent view:(NSView *)view;
   14: - (void) stampStart:(NSPoint)startPoint end:(NSPoint)endPoint inView:(NSView *)view onCanvas:(Canvas *
)canvas;
   15:
   16: - (CGContextRef) createBitmapContext;
   17: - (void) disposeBitmapContext:(CGContextRef)bitmapContext;
   18: - (CGImageRef) createShapeImage;
   19:
   20: @end
   21:
   22: @implementation Brush
   23:
   24: - (id) init
   25: {
   26:         self = [super init];
   27:
   28:         if ( self ) {
   29:                 mRadius = 10.0;
   30:
   31:                 // Create the shape of the tip of the brush. Code currently assumes the bounding
   32:                 //      box of the shape is square (height == width)
   33:                 mShape = CGPathCreateMutable();
   34:                 CGPathAddEllipseInRect(mShape, nil, CGRectMake(0, 0, 2 * mRadius, 2 * mRadius));
   35:                 //CGPathAddRect(mShape, nil, CGRectMake(0, 0, 2 * mRadius, 2 * mRadius));
   36:
   37:                 // Create the color for the brush
   38:                 CGColorSpaceRef colorspace = CGColorSpaceCreateWithName(kCGColorSpaceGenericRGB);
   39:                 float components[] = { 0.0, 0.0, 1.0, 1.0 }; // I like blue
   40:                 mColor = CGColorCreate(colorspace, components);
   41:                 CGColorSpaceRelease(colorspace);
   42:
   43:                 // The "softness" of the brush edges
   44:                 mSoftness = 0.5;
   45:                 mHard = NO;
   46:
   47:                 // Initialize variables that will be used during tracking
   48:                 mMask = nil;
   49:                 mLastPoint = NSZeroPoint;
   50:                 mLeftOverDistance = 0.0;
   51:         }
   52:
   53:         return self;
   54: }
   55:
   56: - (void) dealloc
   57: {
   58:         // Clean up our shape and color
   59:         CGPathRelease(mShape);
   60:         CGColorRelease(mColor);
   61:
   62:
   63: }
   64:
   65: - (void) mouseDown:(NSEvent *)theEvent inView:(NSView *)view onCanvas:(Canvas *)canvas
   66: {
   67:     NSLog(@"BrushDown");
   68:         // Translate the event point location into a canvas point
   69:         NSPoint currentPoint = [self canvasLocation:theEvent view:view];
   70:
   71:         // Initialize all the tracking information. This includes creating an image
   72:         //      of the brush tip
   73:         mMask = [self createShapeImage];
   74:         mLastPoint = currentPoint;
   75:         mLeftOverDistance = 0.0;
   76:
   77:         // Since this is a mouse down, we want to stamp the brush's image not matter
   78:         //      what.
   79:         [canvas stampMask:mMask at:currentPoint];
```

```
   80:
   81:         // This isn't very efficient, but we need to tell the view to redraw. A better
   82:         //      version would have the canvas itself to generate an invalidate for the view
   83:         //      (since it knows exactly where the bits changed).
   84:         [view setNeedsDisplay:YES];
   85: }
   86:
   87: - (void) mouseDragged:(NSEvent *)theEvent inView:(NSView *)view onCanvas:(Canvas *)canvas
   88: {
   89:
   90:         NSLog(@"Brushdragg");
   91:         // Translate the event point location into a canvas point
   92:         NSPoint currentPoint = [self canvasLocation:theEvent view:view];
   93:
   94:         // Stamp the brush in a line, from the last mouse location to the current one
   95:         [self stampStart:mLastPoint end:currentPoint inView:view onCanvas:canvas];
   96:
   97:         // Remember the current point, so that next time we know where to start
   98:         //      the line
   99:         mLastPoint = currentPoint;
  100: }
  101:
  102: - (void) mouseUp:(NSEvent *)theEvent inView:(NSView *)view onCanvas:(Canvas *)canvas
  103: {
  104:         // Translate the event point location into a canvas point
  105:         NSPoint currentPoint = [self canvasLocation:theEvent view:view];
  106:
  107:         // Stamp the brush in a line, from the last mouse location to the current one
  108:         [self stampStart:mLastPoint end:currentPoint inView:view onCanvas:canvas];
  109:
  110:         // This is a mouse up, so we are done tracking. Use this opportunity to clean
  111:         //      up all the tracking information, including the brush tip image.
  112:         CGImageRelease(mMask);
  113:         mMask = nil;
  114:         mLastPoint = NSZeroPoint;
  115:         mLeftOverDistance = 0.0;
  116: }
  117:
  118: @end
  119:
  120: @implementation Brush (Private)
  121:
  122: - (NSPoint) canvasLocation:(NSEvent *)theEvent view:(NSView *)view
  123: {
  124:         // Currently we assume that the NSView here is a CanvasView, which means
  125:         //      that the view is not scaled or offset. i.e. There is a one to one
  126:         //      correlation between the view coordinates and the canvas coordinates.
  127:         NSPoint eventLocation = [theEvent locationInWindow];
  128:         return [view convertPoint:eventLocation fromView:nil];
  129: }
  130:
  131: - (void) stampStart:(NSPoint)startPoint end:(NSPoint)endPoint inView:(NSView *)view onCanvas:(Canvas *
)canvas
  132: {
  133:         // We need to ask the canvas to draw a line using the brush. Keep track
  134:         //      of the distance left over that we didn't draw this time (so we draw
  135:         //      it next time).
  136:         mLeftOverDistance = [canvas stampMask:mMask from:startPoint to:endPoint leftOverDistance:mLeft
OverDistance];
  137:
  138:         // This isn't very efficient, but we need to tell the view to redraw. A better
  139:         //      version would have the canvas itself to generate an invalidate for the view
  140:         //      (since it knows exactly where the bits changed).
  141:         [view setNeedsDisplay:YES];
  142: }
  143:
  144: - (CGContextRef) createBitmapContext
  145: {
  146:         // Create the offscreen bitmap context that we can draw the brush tip into.
  147:         //      The context should be the size of the shape bounding box.
  148:         CGRect boundingBox = CGPathGetBoundingBox(mShape);
  149:
  150:         size_t width = CGRectGetWidth(boundingBox);
  151:         size_t height = CGRectGetHeight(boundingBox);
  152:         size_t bitsPerComponent = 8;
  153:         size_t bytesPerRow = ((width * 4) + 0x0000000F) & ~0x0000000F; // 16 byte aligned is good
  154:         size_t dataSize = bytesPerRow * height;
  155:         void* data = calloc(1, dataSize);
  156:         CGColorSpaceRef colorspace = CGColorSpaceCreateWithName(kCGColorSpaceGenericRGB);
  157:
```

```
158:         CGContextRef bitmapContext = CGBitmapContextCreate(data, width, height, bitsPerComponent,
159:
 bytesPerRow, colorspace,
160:
 kCGImageAlphaPremultipliedFirst);
161:
162:         CGColorSpaceRelease(colorspace);
163:
164:         // Clear the context to transparent, 'cause we'll be using transparency
165:         CGContextClearRect(bitmapContext, CGRectMake(0, 0, width, height));
166:
167:         return bitmapContext;
168: }
169:
170: - (void) disposeBitmapContext:(CGContextRef)bitmapContext
171: {
172:         // Free up the offscreen bitmap
173:         void * data = CGBitmapContextGetData(bitmapContext);
174:         CGContextRelease(bitmapContext);
175:         free(data);
176: }
177:
178: - (CGImageRef) createShapeImage
179: {
180:         // Create a bitmap context to hold our brush image
181:         CGContextRef bitmapContext = [self createBitmapContext];
182:
183:         // If we're not going to have a hard edge, set the alpha to 50% (using a
184:         //      transparency layer) so the brush strokes fade in and out more.
185:         if ( !mHard )
186:                 CGContextSetAlpha(bitmapContext, 0.5);
187:         CGContextBeginTransparencyLayer(bitmapContext, nil);
188:
189:         // I like a little color in my brushes
190:         CGContextSetFillColorWithColor(bitmapContext, mColor);
191:
192:         // The way we acheive "softness" on the edges of the brush is to draw
193:         //      the shape full size with some transparency, then keep drawing the shape
194:         //      at smaller sizes with the same transparency level. Thus, the center
195:         //      builds up and is darker, while edges remain partially transparent.
196:
197:         // First, based on the softness setting, determine the radius of the fully
198:         //      opaque pixels.
199:         int innerRadius = (int)ceil(mSoftness * (0.5 - mRadius) + mRadius);
200:         int outerRadius = (int)ceil(mRadius);
201:         int i = 0;
202:
203:         // The alpha level is always proportial to the difference between the inner, opaque
204:         //      radius and the outer, transparent radius.
205:         float alphaStep = 1.0 / (outerRadius - innerRadius + 1);
206:
207:         // Since we're drawing shape on top of shape, we only need to set the alpha once
208:         CGContextSetAlpha(bitmapContext, alphaStep);
209:
210:         for (i = outerRadius; i >= innerRadius; --i) {
211:                 CGContextSaveGState(bitmapContext);
212:
213:                 // First, center the shape onto the context.
214:                 CGContextTranslateCTM(bitmapContext, outerRadius - i, outerRadius - i);
215:
216:                 // Second, scale the the brush shape, such that each successive iteration
217:                 //      is two pixels smaller in width and height than the previous iteration.
218:                 float scale = (2.0 * (float)i) / (2.0 * (float)outerRadius);
219:                 CGContextScaleCTM(bitmapContext, scale, scale);
220:
221:                 // Finally, actually add the path and fill it
222:                 CGContextAddPath(bitmapContext, mShape);
223:                 CGContextEOFillPath(bitmapContext);
224:
225:                 CGContextRestoreGState(bitmapContext);
226:         }
227:
228:         // We're done drawing, composite the tip onto the context using whatever
229:         //      alpha we had set up before BeginTransparencyLayer.
230:         CGContextEndTransparencyLayer(bitmapContext);
231:
232:         // Create the brush tip image from our bitmap context
233:         CGImageRef image = CGBitmapContextCreateImage(bitmapContext);
234:
235:         // Free up the offscreen bitmap
```

```
236:         [self disposeBitmapContext:bitmapContext];
237:
238:         return image;
239: }
240:
241:
242: @end
```

```
    1: //
    2: //  Canvas.h
    3: //  Paint
    4: //
    5:
    6: @interface Canvas : NSObject {
    7:         // The canvas is simply backed by a bitmap context. A CGLayerRef would be
    8:         //      better, but you have to know your destination context before you create
    9:         //      it (which we don't).
   10:         CGContextRef    mBitmapContext;
   11: }
   12:
   13: // Constructor that creates a canvas at the specified size. Canvas cannot be resized.
   14: - (id) initWithSize:(NSSize)size;
   15:
   16: // Draws the contents of the canvas into the specified context. Handy for views
   17: //      that host a canvas.
   18: - (void)drawRect:(NSRect)rect inContext:(NSGraphicsContext*)context;
   19:
   20: // Graphics privimites for the canvas. The first draws a line given the brush
   21: //      image, and the second, draws a point given the brush image.
   22: - (float)stampMask:(CGImageRef)mask from:(NSPoint)startPoint to:(NSPoint)endPoint leftOverDistance:(fl
oat)leftOverDistance;
   23: - (void)stampMask:(CGImageRef)mask at:(NSPoint)point;
   24:
   25: @end
```

```
    1: //
    2: //  Canvas.m
    3: //  Paint
    4: //
    5:
    6:
    7: #import "Canvas.h"
    8:
    9: @implementation Canvas
   10:
   11: - (id) initWithSize:(NSSize)size
   12: {
   13:         self = [super init];
   14:
   15:         if ( self ) {
   16:                 // Create a bitmap context for the canvas. To keep things simple
   17:                 //      we're going to use a 32-bit ARGB format.
   18:                 size_t width = size.width;
   19:                 size_t height = size.height;
   20:                 size_t bitsPerComponent = 8;
   21:                 size_t bytesPerRow = ((width * 4) + 0x0000000F) & ~0x0000000F; // 16-byte aligned is g
ood
   22:                 size_t dataSize = bytesPerRow * height;
   23:                 void* data = calloc(1, dataSize);
   24:                 CGColorSpaceRef colorspace = CGColorSpaceCreateWithName(kCGColorSpaceGenericRGB);
   25:
   26:                 mBitmapContext = CGBitmapContextCreate(data, width, height, bitsPerComponent,
   27:                                                                                         bytesPerRow,
 colorspace,
   28:                                                                                         kCGImageAlph
aPremultipliedFirst);
   29:
   30:                 CGColorSpaceRelease(colorspace);
   31:
   32:                 // Paint on a white background so the user has something to start with.
   33:                 CGContextSaveGState(mBitmapContext);
   34:
   35:                 CGRect fillRect = CGRectMake(0, 0, CGBitmapContextGetWidth(mBitmapContext),
   36:                                                                         CGBitmapContextGetHeight(mBi
tmapContext));
   37:
   38:                 CGContextSetRGBFillColor(mBitmapContext, 1.0, 1.0, 1.0, 1.0);
   39:                 CGContextFillRect(mBitmapContext, fillRect);
   40:
   41:                 CGContextRestoreGState(mBitmapContext);
   42:         }
   43:
   44:         return self;
   45: }
   46:
   47: - (void) dealloc
   48: {
   49:         // Free up our bitmap context
   50:         void* data = CGBitmapContextGetData(mBitmapContext);
   51:         CGContextRelease(mBitmapContext);
   52:         free(data);
   53:
   54: }
   55:
   56: - (void)drawRect:(NSRect)rect inContext:(NSGraphicsContext*)context
   57: {
   58:         // Here we simply want to render our bitmap context into the view's
   59:         //      context. It's going to be a straight forward bit blit. First,
   60:         //      create an image from our bitmap context.
   61:         CGImageRef imageRef = CGBitmapContextCreateImage(mBitmapContext);
   62:
   63:         // Grab the destination context
   64:         CGContextRef contextRef = [context graphicsPort];
   65:         CGContextSaveGState(contextRef);
   66:
   67:         // Composite on the image at the bottom left of the context
   68:         CGRect imageRect = CGRectMake(0, 0, CGBitmapContextGetWidth(mBitmapContext),
   69:                                                                 CGBitmapContextGetHeight(mBitmapCont
ext));
   70:         CGContextDrawImage(contextRef, imageRect, imageRef);
   71:
   72:         CGImageRelease(imageRef);
   73:
   74:         CGContextRestoreGState(contextRef);
   75: }
```

```
   76:
   77: - (float)stampMask:(CGImageRef)mask from:(NSPoint)startPoint to:(NSPoint)endPoint leftOverDistance:(fl
oat)leftOverDistance
   78: {
   79:         // Set the spacing between the stamps. By trail and error, I've
   80:         //      determined that 1/10 of the brush width (currently hard coded to 20)
   81:         //      is a good interval.
   82:         float spacing = CGImageGetWidth(mask) * 0.1;
   83:
   84:         // Anything less that half a pixel is overkill and could hurt performance.
   85:         if ( spacing < 0.5 )
   86:                 spacing = 0.5;
   87:
   88:         // Determine the delta of the x and y. This will determine the slope
   89:         //      of the line we want to draw.
   90:         float deltaX = endPoint.x - startPoint.x;
   91:         float deltaY = endPoint.y - startPoint.y;
   92:
   93:         // Normalize the delta vector we just computed, and that becomes our step increment
   94:         //      for drawing our line, since the distance of a normalized vector is always 1
   95:         float distance = sqrt( deltaX * deltaX + deltaY * deltaY );
   96:         float stepX = 0.0;
   97:         float stepY = 0.0;
   98:         if ( distance > 0.0 ) {
   99:                 float invertDistance = 1.0 / distance;
  100:                 stepX = deltaX * invertDistance;
  101:                 stepY = deltaY * invertDistance;
  102:         }
  103:
  104:         float offsetX = 0.0;
  105:         float offsetY = 0.0;
  106:
  107:         // We're careful to only stamp at the specified interval, so its possible
  108:         //      that we have the last part of the previous line left to draw. Be sure
  109:         //      to add that into the total distance we have to draw.
  110:         float totalDistance = leftOverDistance + distance;
  111:
  112:         // While we still have distance to cover, stamp
  113:         while ( totalDistance >= spacing ) {
  114:                 // Increment where we put the stamp
  115:                 if ( leftOverDistance > 0 ) {
  116:                         // If we're making up distance we didn't cover the last
  117:                         //      time we drew a line, take that into account when calculating
  118:                         //      the offset. leftOverDistance is always < spacing.
  119:                         offsetX += stepX * (spacing - leftOverDistance);
  120:                         offsetY += stepY * (spacing - leftOverDistance);
  121:
  122:                         leftOverDistance -= spacing;
  123:                 } else {
  124:                         // The normal case. The offset increment is the normalized vector
  125:                         //      times the spacing
  126:                         offsetX += stepX * spacing;
  127:                         offsetY += stepY * spacing;
  128:                 }
  129:
  130:                 // Calculate where to put the current stamp at.
  131:                 NSPoint stampAt = NSMakePoint(startPoint.x + offsetX, startPoint.y + offsetY);
  132:
  133:                 // Ka-chunk! Draw the image at the current location
  134:                 [self stampMask:mask at: stampAt];
  135:
  136:                 // Remove the distance we just covered
  137:                 totalDistance -= spacing;
  138:         }
  139:
  140:         // Return the distance that we didn't get to cover when drawing the line.
  141:         //      It is going to be less than spacing.
  142:         return totalDistance;
  143: }
  144:
  145: - (void)stampMask:(CGImageRef)mask at:(NSPoint)point
  146: {
  147:         // When we stamp the image, we want the center of the image to be
  148:         //      at the point specified.
  149:         CGContextSaveGState(mBitmapContext);
  150:
  151:         // So we can position the image correct, compute where the bottom left
  152:         //      of the image should go, and modify the CTM so that 0, 0 is there.
  153:         CGPoint bottomLeft = CGPointMake( point.x - CGImageGetWidth(mask) * 0.5,
  154:                                                                         point.y - CGImageGetHeight(m
```

```
ask) * 0.5 );
  155:          CGContextTranslateCTM(mBitmapContext, bottomLeft.x, bottomLeft.y);
  156:
  157:          // Now that it's properly lined up, draw the image
  158:          CGRect maskRect = CGRectMake(0, 0, CGImageGetWidth(mask), CGImageGetHeight(mask));
  159:          CGContextDrawImage(mBitmapContext, maskRect, mask);
  160:
  161:          CGContextRestoreGState(mBitmapContext);
  162: }
  163:
  164: @end
```

```
 1: //
 2: //  CanvasView.h
 3: //  Paint
 4: //
 5:
 6:
 7: @class Canvas;
 8: @class Brush;
 9:
10: @interface CanvasView : NSView {
11:         // The canvas we will render to the screen
12:         Canvas          *canvas;
13:         // The brush that we will pass events to
14:         Brush           *brush;
15: }
16:
17: @end
```

```
 1: //
 2: //  CanvasView.m
 3: //  Paint
 4: //
 5:
 6:
 7: #import "CanvasView.h"
 8: #import "Canvas.h"
 9: #import "Brush.h"
10:
11: @implementation CanvasView
12:
13: - (id)initWithFrame:(NSRect)frame {
14:     self = [super initWithFrame:frame];
15:     if (self) {
16:         // Create both the canvas and the brush. Create the canvas
17:                 //      the same size as our initial size. Note that the canvas will not
18:                 //      resize along with us.
19:                 canvas = [[Canvas alloc] initWithSize:frame.size];
20:                 brush = [[Brush alloc] init];
21:     }
22:     return self;
23: }
24:
25:
26:
27: - (void)drawRect:(NSRect)rect {
28:     NSLog(@"Canvas DrawRect");
29:         // Simply ask the canvas to draw into the current context, given the
30:         //      rectangle specified. A more sophisticated view might draw a border
31:         //      around the canvas, or a pasteboard in the case that the view was
32:         //      bigger than the canvas.
33:         NSGraphicsContext* context = [NSGraphicsContext currentContext];
34:
35:         [canvas drawRect:rect inContext:context];
36: }
37:
38: - (void)mouseDown:(NSEvent *)theEvent
39: {
40:     NSLog(@"Canvas Mouse Down");
41:         // Simply pass the mouse event to the brush. Also give it the canvas to
42:         //      work on, and a reference to ourselves, so it can translate the mouse
43:         //      locations.
44:         [brush mouseDown:theEvent inView:self onCanvas:canvas];
45:
46:     [self setNeedsDisplay:YES];
47:
48:
49: }
50:
51: - (void)mouseDragged:(NSEvent *)theEvent
52: {
53:     NSLog(@"Canvas Mouse Dragged");
54:         // Simply pass the mouse event to the brush. Also give it the canvas to
55:         //      work on, and a reference to ourselves, so it can translate the mouse
56:         //      locations.
57:         [brush mouseDragged:theEvent inView:self onCanvas:canvas];
58: }
59:
60: - (void)mouseUp:(NSEvent *)theEvent
61: {
62:         // Simply pass the mouse event to the brush. Also give it the canvas to
63:         //      work on, and a reference to ourselves, so it can translate the mouse
64:         //      locations.
65:         [brush mouseUp:theEvent inView:self onCanvas:canvas];
66:
67:
68: }
69:
70: @end
```

```
 1: //
 2: //  DrawingTool.h
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 3/17/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10: #import "ToolSettings.h"
11:
12: @interface DrawingTool : NSObject
13:
14:
15: @property (strong) NSImage *drawingSnapshot;
16: @property (strong) NSImageView *drawingImageView;
17: @property (weak) ToolSettings *settings;
18: @property (assign) CGPoint firstPoint;
19: @property (copy) NSString *imageName;
20: @property (copy) NSString *toolName;
21:
22: - (void)inputBegan:(NSEvent *)theEvent locationInView:(CGPoint)currentPoint  withSettings:(ToolSetting
s *)settings;
23: - (void)inputMoved:(NSEvent *)theEvent locationInView:(CGPoint)currentPoint;
24: - (void)inputEnded:(NSEvent *)theEvent locationInView:(CGPoint)currentPoint;
25:
26:
27: @property (copy) dispatch_block_t completion;
28:
29: - (DrawingTool*)initWithCompletion:(dispatch_block_t)completion;
30:
31: @end
```

```
 1: //
 2: //  DrawingTool.m
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 3/17/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import "DrawingTool.h"
10:
11: @implementation DrawingTool
12:
13:
14: - (DrawingTool*)initWithCompletion:(dispatch_block_t)completion {
15:
16:     if (self = [super init])
17:     {
18:         self.completion = completion;
19:     }
20:     return self;
21:
22: }
23:
24: //override to disable double-buffered drawing (e.g. for eraser)
25: - (BOOL)doubleBufferDrawing
26: {
27:     return YES;
28: }
29:
30:
31: - (void)inputBegan:(NSEvent *)theEvent locationInView:(CGPoint)currentPoint withSettings:(ToolSettings
*)settings{
32:     self.settings = settings;
33:     self.firstPoint = currentPoint;
34: }
35:
36: - (void)inputMoved:(NSEvent *)theEvent locationInView:(CGPoint)currentPoint{
37:
38: }
39:
40: - (void)inputEnded:(NSEvent *)theEvent locationInView:(CGPoint)currentPoint{
41:
42:     if (self.completion) {
43:         self.completion();
44:     }
45: }
46:
47:
48: @end
```

```
 1: //
 2: //  LPBrushTool.h
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 3/19/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10: #import <QuartzCore/QuartzCore.h>
11: #import "LPDrawingTool.h"
12:
13: @interface LPBrushTool : LPDrawingTool {
14:
15:     NSString* templateImageName;
16:     NSImage* templateImage;
17: }
18:
19:
20:
21: @end
```

```
    1: //
    2: //  LPBrushTool.m
    3: //  LeapPaint
    4: //
    5: //  Created by cj on 3/19/13.
    6: //  Copyright (c) 2013 cjdesch. All rights reserved.
    7: //
    8:
    9: #import "LPBrushTool.h"
   10:
   11: @implementation LPBrushTool
   12:
   13: - (void)inputBegan:(NSEvent *)theEvent  withSettings:(ToolSettings *)settings{
   14:     self.settings = settings;
   15:     templateImage = [NSImage imageNamed:templateImageName];
   16:     CGSize size = CGSizeMake(self.lineWidth, self.lineWidth);
   17:     templateImage = [self imageResize:templateImage newSize:NSSizeFromCGSize(size)];
   18:     //templateImage = self adjustImage:templateImage withHue:<#(float)#>
   19:
   20: }
   21:
   22: - (void)inputMoved:(NSEvent *)theEvent {
   23:
   24: }
   25:
   26: - (void)inputEnded:(NSEvent *)theEvent {
   27:
   28: }
   29:
   30:
   31: - (void)toolDidLoad{
   32:
   33:     //NSLog(@"Tool Did Load Subclass");
   34:    // templateImage = [NSImage imageNamed:templateImageName];
   35: //     self.template = [[NSImage imageNamed:templateImageName withTint:self.primaryColor] resizedImage
:CGSizeMake(self.lineWidth), self.lineWidth) interpolationQuality:kCGInterpolationDefault];
   36: }
   37:
   38: - (NSImage *)imageResize:(NSImage*)anImage
   39:                  newSize:(NSSize)newSize
   40: {
   41:     NSImage *sourceImage = anImage;
   42:     [sourceImage setScalesWhenResized:YES];
   43:
   44:     // Report an error if the source isn't a valid image
   45:     if (![sourceImage isValid])
   46:     {
   47:         NSLog(@"Invalid Image");
   48:     } else
   49:     {
   50:         NSImage *smallImage = [[NSImage alloc] initWithSize: newSize] ;
   51:         [smallImage lockFocus];
   52:         [sourceImage setSize: newSize];
   53:         [[NSGraphicsContext currentContext] setImageInterpolation:NSImageInterpolationHigh];
   54:         [sourceImage compositeToPoint:NSZeroPoint operation:NSCompositeCopy];
   55:         [smallImage unlockFocus];
   56:         return smallImage;
   57:     }
   58:     return nil;
   59: }
   60:
   61: - (NSImage*) adjustImage:(NSImage*)img withHue:(float)hue {
   62:
   63:     CIImage *inputImage = [[CIImage alloc] initWithData:[img TIFFRepresentation]];
   64:
   65:     CIFilter *hueAdjust = [CIFilter filterWithName:@"CIHueAdjust"];
   66:     [hueAdjust setValue: inputImage forKey: @"inputImage"];
   67:     [hueAdjust setValue: [NSNumber numberWithFloat: hue]
   68:                  forKey: @"inputAngle"];
   69:     CIImage *outputImage = [hueAdjust valueForKey: @"outputImage"];
   70:
   71:     NSImage *resultImage = [[NSImage alloc] initWithSize:[outputImage extent].size];
   72:     NSCIImageRep *rep = [NSCIImageRep imageRepWithCIImage:outputImage];
   73:     [resultImage addRepresentation:rep];
   74:
   75:     return resultImage;
   76:
   77: }
   78:
   79: @end
```

```
    1: //
    2: //  LPDrawingTool.h
    3: //  LeapPaint
    4: //
    5: //  Created by cj on 3/18/13.
    6: //  Copyright (c) 2013 cjdesch. All rights reserved.
    7: //
    8:
    9: #import <Foundation/Foundation.h>
   10: #import "ToolSettings.h"
   11:
   12:
   13: @interface LPDrawingTool : NSObject{
   14:
   15:
   16: }
   17:
   18:
   19: @property (copy) NSString *drawingTool;
   20: @property (assign) int lineWidth;
   21: @property (assign) int transparency;
   22: @property (assign) int fontSize;
   23: @property (strong) NSColor *primaryColor;
   24: @property (strong) NSColor *secondaryColor;
   25: @property (strong) NSMutableArray* points;
   26: @property (strong) NSString* brushTextureName;
   27: @property (weak) ToolSettings *settings;
   28:
   29:
   30: - (id)initWithWidth:(int)w withTransparency:(int)t withPrimaryColor:(NSColor*)p withSecondaryColor:(NS
Color*)s;
   31:
   32: - (void)toolDidLoad;
   33: @end
```

```
 1: //
 2: //  LPDrawingTool.m
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 3/18/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import "LPDrawingTool.h"
10:
11: @implementation LPDrawingTool
12: @synthesize lineWidth;
13: @synthesize transparency;
14:
15: @synthesize primaryColor;
16: @synthesize secondaryColor;
17: @synthesize points;
18: @synthesize drawingTool;
19:
20:
21: - (id)initWithWidth:(int)w withTransparency:(int)t withPrimaryColor:(NSColor*)p withSecondaryColor:(NS
Color*)s{
22:     if (self = [super init]){
23:
24:         self.points = [[NSMutableArray alloc] init];
25:         self.primaryColor = p;
26:         self.secondaryColor = s;
27:
28:         self.lineWidth = w;
29:         self.transparency = t;
30:
31:         [self toolDidLoad];
32:
33:     }
34:
35:     return self;
36: }
37:
38: - (void)inputBegan:(NSEvent *)theEvent  withSettings:(ToolSettings *)settings{
39:     self.settings = settings;
40:
41: }
42:
43: - (void)inputMoved:(NSEvent *)theEvent {
44:
45: }
46:
47: - (void)inputEnded:(NSEvent *)theEvent {
48:
49: }
50:
51:
52:
53: - (void)toolDidLoad{
54:     //NSLog(@"ToolDidLoad");
55:
56: }
57:
58: @end
```

```
 1: //
 2: //  LPPenTool.h
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 3/19/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10: #import "LPDrawingTool.h"
11: @interface LPPenTool : LPDrawingTool
12:
13: @end
```

```
 1: //
 2: //  LPPenTool.m
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 3/19/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import "LPPenTool.h"
10:
11: @implementation LPPenTool
12:
13: @end
```

```
 1: //
 2: //  main.m
 3: //  Quartz2DPaint
 4: //
 5: //  Created by cj on 5/7/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Cocoa/Cocoa.h>
10:
11: int main(int argc, char *argv[])
12: {
13:     return NSApplicationMain(argc, (const char **)argv);
14: }
```

```
 1: //
 2: //  MyPoint.h
 3: //  010-NSView
 4: //
 5: // wrapper for NSPoint
 6: #import <Cocoa/Cocoa.h>
 7:
 8: @interface MyPoint : NSObject {
 9:         NSPoint myNSPoint;
10: }
11: - (id) initWithNSPoint:(NSPoint)pNSPoint;
12: - (NSPoint) myNSPoint;
13: - (float)x;
14: - (float)y;
15:
16: @end
```

```
 1: //
 2: //  MyPoint.m
 3: //  010-NSView
 4: //
 5: #import "MyPoint.h"
 6:
 7: @implementation MyPoint
 8:
 9: - (id) initWithNSPoint:(NSPoint)pNSPoint;
10: {
11:         if ((self = [super init]) == nil) {
12:                 return self;
13:         } // end if
14:
15:         myNSPoint.x     = pNSPoint.x;
16:         myNSPoint.y     = pNSPoint.y;
17:
18:     return self;
19:
20: } // end initWithNSPoint
21:
22: - (NSPoint) myNSPoint;
23: {
24:         return myNSPoint;
25: } // end myNSPoint
26:
27: - (float)x;
28: {
29:         return myNSPoint.x;
30: } // end x
31:
32: - (float)y;
33: {
34:         return myNSPoint.y;
35: } // end y
36:
37:
38:
39:
40:
41: @end
```

```
 1: //
 2: //  PaintCanvasView.h
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 3/17/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Cocoa/Cocoa.h>
10: #import "MyPoint.h"
11: #import "DrawingTool.h"
12: #import "ToolSettings.h"
13: #import "PenTool.h"
14: #import "LPDrawingTool.h"
15: #import "LeapObjectiveC.h"
16: #import "TrackedFinger.h"
17: #import "SimplePoint.h"
18:
19: @interface PaintCanvasView : NSView <LeapListener>{
20:
21:     LeapController *controller;
22:
23:     NSMutableArray  * myDrawingObjects;
24:     NSMutableArray      * myMutaryOfBrushStrokes;
25:         NSMutableArray  * myMutaryOfPoints;
26:     LPDrawingTool* drawingTool;
27:
28:
29:
30:     NSString* selectedTool;
31:
32:     NSColor* currentColor;
33:     float currentLineWidth;
34:     NSString* brushType;
35:     NSMutableDictionary* trackableList;
36:
37:
38:
39: }
40:
41: @property (nonatomic, strong)     NSMutableArray* tools;
42:
43: - (void)clearScreen;
44: - (void)setColor:(NSColor*)color;
45: - (void)setLineWidth:(float)w;
46: - (void)setTool:(NSString*)tool;
47:
48: @end
```

```
//
//  PaintCanvasView.m
//  LeapPaint
//
//  Created by cj on 3/17/13.
//  Copyright (c) 2013 cjdesch. All rights reserved.
//

#import "PaintCanvasView.h"


@interface PaintCanvasView()

@property (nonatomic, strong) ToolSettings *toolSettings;
@property (nonatomic, strong) NSMutableArray *drawingTools;

#pragma mark - Undo stack

@property (assign) int undoStackLocation;
@property (assign) int undoStackCount;

@end

@implementation PaintCanvasView

@synthesize drawingTools;
@synthesize toolSettings;

@synthesize tools;

- (id)initWithFrame:(NSRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        // Initialization code here.

        NSLog(@"Init");

        myMutaryOfBrushStrokes  = [[NSMutableArray alloc] init];
        myDrawingObjects = [[NSMutableArray alloc] init];

        drawingTools = [[NSMutableArray alloc] init];



        self.toolSettings = [[ToolSettings alloc] init];
         [self.toolSettings loadFromUserDefaults];
        [self initializeTools];

        //NSColorPanel* colorPanel = [NSColorPanel sharedColorPanel];

        currentColor = [NSColor blueColor];
        currentLineWidth = 1.0;

        trackableList = [[NSMutableDictionary alloc] init];

        [self run];

        self.tools = [[NSMutableArray alloc] initWithObjects:@"Pen",@"Brush", nil];
        selectedTool = @"Pen";


        // initialise random numebr generator used in drawRect for creating colours etc.
        srand(time(NULL));
    }

    return self;
}

#pragma mark - SampleDelegate Callbacks
- (void)run
{
    controller = [[LeapController alloc] init];
    [controller addListener:self];
    NSLog(@"running");
}

- (void)onInit:(NSNotification *)notification{
    NSLog(@"Leap: Initialized");
}
```

```
   81:
   82: - (void)onConnect:(NSNotification *)notification;
   83: {
   84:     NSLog(@"Leap: Connected");
   85:     LeapController *aController = (LeapController *)[notification object];
   86:     //[aController enableGesture:LEAP_GESTURE_TYPE_CIRCLE enable:YES];
   87:     //[aController enableGesture:LEAP_GESTURE_TYPE_KEY_TAP enable:YES];
   88:     //[aController enableGesture:LEAP_GESTURE_TYPE_SCREEN_TAP enable:YES];
   89:     //[aController enableGesture:LEAP_GESTURE_TYPE_SWIPE enable:YES];
   90: }
   91:
   92: - (void)onDisconnect:(NSNotification *)notification{
   93:     NSLog(@"Leap: Disconnected");
   94: }
   95:
   96: - (void)onExit:(NSNotification *)notification{
   97:     NSLog(@"Leap: Exited");
   98: }
   99:
  100:
  101: - (void)onFrame:(NSNotification *)notification{
  102:
  103:     //NSLog(@"OnFrame");
  104:     LeapController *aController = (LeapController *)[notification object];
  105:     // Get the most recent frame and report some basic information
  106:     LeapFrame *frame = [aController frame:0];
  107:
  108:
  109:     if ([[frame hands] count] != 0) {
  110:         // Get the first hand
  111:         LeapHand *hand = [[frame hands] objectAtIndex:0];
  112:
  113:         // Check if the hand has any fingers
  114:         NSArray *fingers = [hand fingers];
  115:
  116:         if ([fingers count] != 0) {
  117:
  118:             // Calculate the hand's average finger tip position
  119:             LeapVector *avgPos = [[LeapVector alloc] init];
  120:             for (int i = 0; i < [fingers count]; i++) {
  121:                 LeapFinger *finger = [fingers objectAtIndex:i];
  122:                 avgPos = [avgPos plus:[finger tipPosition]];
  123:
  124:                 if (avgPos.z < 0){
  125:                     NSString* fingerID = [NSString stringWithFormat:@"%d", finger.id];
  126:
  127:                     //Check if the Finger ID exists in the list already
  128:                     if ([trackableList objectForKey:fingerID]) {
  129:
  130:                         //If it does exist update the position on the screen
  131:                         TrackedFinger* sprite = [trackableList objectForKey:fingerID];
  132:                         sprite.position = [self covertLeapCoordinates:CGPointMake(finger.tipPosition.x
, finger.tipPosition.y)];
  133:                         sprite.updated = TRUE;
  134:
  135:                         [self updateFingerDraw:sprite];
  136:
  137:                         //SimplePoint* simplePoint = [[SimplePoint alloc] initWithPosition:sprite.posi
tion];
  138:
  139:
  140:                     }else{
  141:
  142:                         //NSLog(@"x %0.0f y %0.0f z %0.0f", finger.tipPosition.x, finger.tipPosition.y
, finger.tipPosition.z);
  143:                         // CGPoint point = [[CCDirector sharedDirector] convertEventToGL:event];
  144:                         //CGPoint mouseLocation = [self convertToNodeSpace:point];
  145:
  146:                         //Add it to the dictionary
  147:                         //TrackedFinger* redDot = [self addRedDot:[self covertLeapCoordinates:CGPointM
ake(finger.tipPosition.x, finger.tipPosition.y)] finger:fingerID];
  148:                         TrackedFinger* redDot = [[TrackedFinger alloc] initWithID:fingerID withPositio
n:[self covertLeapCoordinates:CGPointMake(finger.tipPosition.x, finger.tipPosition.y)]];
  149:                         [trackableList setObject:redDot forKey:fingerID];
  150:
  151:                         [self beginFingerDraw:redDot];
  152:
  153:                     }
  154:                 }
  155:
```

```
156:             }
157:
158:             avgPos = [avgPos divide:[fingers count]];
159:
160:             //NSLog(@"Hand has %ld fingers, average finger tip position %@", [fingers count], avgPos);
161:             for (LeapFinger* finger in fingers){
162:
163:                 //NSLog(@"Finger ID %d %ld", finger.id, (unsigned long)[finger hash]);
164:             }
165:
166:         }
167:
168:         //
169:         [self checkFingerExists];
170:     }
171:
172: }
173:
174:
175: /*
176: - (void)drawRect:(NSRect)dirtyRect
177: {
178:     NSLog(@"drawRect");
179:     // Drawing code here.
180:     // colour the background white
181:
182:         [[NSColor whiteColor] set];             // this is Cocoa
183:         NSRectFill( dirtyRect );
184:
185:         if ([myMutaryOfBrushStrokes count] == 0) {
186:                 return;
187:         } // end if
188:
189:         // This is Quartz
190:         NSGraphicsContext       *       tvarNSGraphicsContext   = [NSGraphicsContext currentContext];
191:         CGContextRef                    tvarCGContextRef                = (CGContextRef) [tvarNSGraphi
csContext graphicsPort];
192:
193:         NSUInteger tvarIntNumberOfStrokes       = [myMutaryOfBrushStrokes count];
194:
195:         NSUInteger i;
196:         for (i = 0; i < tvarIntNumberOfStrokes; i++) {
197:
198:                 CGContextSetRGBStrokeColor(tvarCGContextRef,[self randVar],[self randVar],[self randVa
r],[self randVar]);
199:                 CGContextSetLineWidth(tvarCGContextRef, (1.0 + ([self randVar] * 10.0)) );
200:
201:                 myMutaryOfPoints        = [myMutaryOfBrushStrokes objectAtIndex:i];
202:
203:                 NSUInteger tvarIntNumberOfPoints        = [myMutaryOfPoints count];
      // always >= 2
204:                 MyPoint * tvarLastPointObj                      = [myMutaryOfPoints objectAtIndex:0];
205:                 CGContextBeginPath(tvarCGContextRef);
206:                 CGContextMoveToPoint(tvarCGContextRef,[tvarLastPointObj x],[tvarLastPointObj y]);
207:
208:                 NSUInteger j;
209:                 for (j = 1; j < tvarIntNumberOfPoints; j++) {  // note the index starts at 1
210:                         MyPoint * tvarCurPointObj                       = [myMutaryOfPoints objectAtIn
dex:j];
211:                         CGContextAddLineToPoint(tvarCGContextRef,[tvarCurPointObj x],[tvarCurPointObj
y]);
212:                 } // end for
213:
214:                 CGContextDrawPath(tvarCGContextRef,kCGPathStroke);
215:
216:         } // end for
217:
218:
219: }
220:
221:  */
222:
223: - (void)drawRect:(NSRect)dirtyRect
224: {
225:     //NSLog(@"drawRect");
226:     // Drawing code here.
227:     // colour the background white
228:
229:         [[NSColor whiteColor] set];             // this is Cocoa
230:         NSRectFill( dirtyRect );
```

```
  231:
  232:         if ([myMutaryOfBrushStrokes count] == 0) {
  233:                 return;
  234:         } // end if
  235:
  236:         // This is Quartz
  237:         NSGraphicsContext       *       tvarNSGraphicsContext   = [NSGraphicsContext currentContext];
  238:         CGContextRef                    tvarCGContextRef                = (CGContextRef) [tvarNSGraphi
csContext graphicsPort];
  239:
  240:         NSUInteger tvarIntNumberOfStrokes       = [myMutaryOfBrushStrokes count];
  241:
  242:         NSUInteger i;
  243:         for (i = 0; i < tvarIntNumberOfStrokes; i++) {
  244:
  245:
  246:                 drawingTool     = [myMutaryOfBrushStrokes objectAtIndex:i];
  247:         CGFloat red, green, blue, a;
  248:         [drawingTool.primaryColor getRed:&red green:&green blue:&blue alpha:&a];
  249:
  250:                 CGContextSetRGBStrokeColor(tvarCGContextRef, red,green,blue,a);
  251:                 CGContextSetLineWidth(tvarCGContextRef, drawingTool.lineWidth );
  252:
  253:
  254:                 NSUInteger tvarIntNumberOfPoints        = [drawingTool.points count];//[myMutaryOfPoin
ts count];                              // always >= 2
  255:                 MyPoint * tvarLastPointObj                      = [drawingTool.points objectAtIndex:0]
;
  256:                 CGContextBeginPath(tvarCGContextRef);
  257:                 CGContextMoveToPoint(tvarCGContextRef,[tvarLastPointObj x],[tvarLastPointObj y]);
  258:
  259:                 NSUInteger j;
  260:                 for (j = 1; j < tvarIntNumberOfPoints; j++) {  // note the index starts at 1
  261:                         MyPoint * tvarCurPointObj                       = [drawingTool.points objectAt
Index:j];
  262:                         CGContextAddLineToPoint(tvarCGContextRef,[tvarCurPointObj x],[tvarCurPointObj
y]);
  263:                 } // end for
  264:
  265:                 CGContextDrawPath(tvarCGContextRef,kCGPathStroke);
  266:
  267:         } // end for
  268:
  269:
  270: }
  271:
  272: /*
  273: -(void)mouseDown:(NSEvent *)pTheEvent {
  274:
  275:     //Selected Tool
  276:     //NSLog(@"Mouse Down");
  277:
  278:     DrawingTool *drawingTool = [self activeTool];
  279:     if (drawingTool) {
  280:
  281:         NSPoint tvarMousePointInWindow  = [pTheEvent locationInWindow];
  282:         NSPoint tvarMousePointInView    = [self convertPoint:tvarMousePointInWindow fromView:self];
  283:         [drawingTool inputBegan:pTheEvent locationInView:tvarMousePointInView withSettings:self.toolSe
ttings];
  284:     }
  285:
  286:
  287:
  288:
  289: } // end mouseDown
  290:
  291: -(void)mouseDragged:(NSEvent *)pTheEvent {
  292:
  293:     //NSLog(@"Mouse Dragg");
  294:
  295:     DrawingTool *drawingTool = [self activeTool];
  296:     if (drawingTool) {
  297:
  298:         NSPoint tvarMousePointInWindow  = [pTheEvent locationInWindow];
  299:         NSPoint tvarMousePointInView    = [self convertPoint:tvarMousePointInWindow fromView:self];
  300:         [drawingTool inputMoved:pTheEvent locationInView:tvarMousePointInView];
  301:
  302:     }
  303:
  304:
```

```
} // end mouseDragged

-(void)mouseUp:(NSEvent *)pTheEvent {

    //NSLog(@"Mouse up");

    DrawingTool *drawingTool = [self activeTool];
    if (drawingTool) {


        NSPoint tvarMousePointInWindow  = [pTheEvent locationInWindow];
        NSPoint tvarMousePointInView    = [self convertPoint:tvarMousePointInWindow fromView:self];
        [drawingTool inputEnded:pTheEvent locationInView:tvarMousePointInView];

    }


} // end mouseUp

 */



/*
-(void)mouseDown:(NSEvent *)pTheEvent {

        myMutaryOfPoints        = [[NSMutableArray alloc]init];
        [myMutaryOfBrushStrokes addObject:myMutaryOfPoints];

        NSPoint tvarMousePointInWindow  = [pTheEvent locationInWindow];
        NSPoint tvarMousePointInView    = [self convertPoint:tvarMousePointInWindow fromView:nil];
        MyPoint * tvarMyPointObj                = [[MyPoint alloc]initWithNSPoint:tvarMousePointInView
];

        [myMutaryOfPoints addObject:tvarMyPointObj];

} // end mouseDown

-(void)mouseDragged:(NSEvent *)pTheEvent {

        NSPoint tvarMousePointInWindow  = [pTheEvent locationInWindow];
        NSPoint tvarMousePointInView    = [self convertPoint:tvarMousePointInWindow fromView:nil];
        MyPoint * tvarMyPointObj                = [[MyPoint alloc]initWithNSPoint:tvarMousePointInView
];

        [myMutaryOfPoints addObject:tvarMyPointObj];

        [self setNeedsDisplay:YES];

} // end mouseDragged

-(void)mouseUp:(NSEvent *)pTheEvent {

        NSPoint tvarMousePointInWindow  = [pTheEvent locationInWindow];
        NSPoint tvarMousePointInView    = [self convertPoint:tvarMousePointInWindow fromView:nil];
        MyPoint * tvarMyPointObj                = [[MyPoint alloc]initWithNSPoint:tvarMousePointInView
];

        [myMutaryOfPoints addObject:tvarMyPointObj];

        [self setNeedsDisplay:YES];

} // end mouseUp

*/


- (void)beginFingerDraw:(id)sender{

    TrackedFinger* trackedFinger = (TrackedFinger*)sender;
    [self beginDraw:trackedFinger.position withFinger:trackedFinger];

}

- (void)updateFingerDraw:(id)sender{
    TrackedFinger* trackedFinger = (TrackedFinger*)sender;
    [self updateDraw:trackedFinger.position withFinger:trackedFinger];

}

```

```
382: - (void)endFingerDraw:(id)sender{
383:     TrackedFinger* trackedFinger = (TrackedFinger*)sender;
384:     [self endDraw:trackedFinger.position withFinger:trackedFinger];
385: }
386:
387:
388: //The further negative, the thicker the line.
389: - (void)beginDraw:(CGPoint)point withFinger:(TrackedFinger*)finger{
390:
391:     //if there is no finger, we are drawing with the mouse
392:     if(finger == nil){
393:         drawingTool     = [[LPDrawingTool alloc] initWithWidth:currentLineWidth withTransparency:1 wit
hPrimaryColor:currentColor withSecondaryColor:[NSColor redColor]];
394:
395:         [myMutaryOfBrushStrokes addObject:drawingTool];
396:         MyPoint * tvarMyPointObj                = [[MyPoint alloc]initWithNSPoint:point];
397:         [drawingTool.points addObject:tvarMyPointObj];
398:     }else{
399:         drawingTool     = [[LPDrawingTool alloc] initWithWidth:currentLineWidth withTransparency:1 wit
hPrimaryColor:currentColor withSecondaryColor:[NSColor redColor]];
400:         finger.tool = drawingTool;
401:         [myMutaryOfBrushStrokes addObject:drawingTool];
402:         MyPoint * tvarMyPointObj                = [[MyPoint alloc]initWithNSPoint:point];
403:         [drawingTool.points addObject:tvarMyPointObj];
404:     }
405:
406: }
407:
408: - (void)updateDraw:(CGPoint)point withFinger:(TrackedFinger*)finger{
409:
410:     if(finger == nil){
411:         MyPoint * tvarMyPointObj                = [[MyPoint alloc]initWithNSPoint:point];
412:         [drawingTool.points addObject:tvarMyPointObj];
413:         [self setNeedsDisplay:YES];
414:     }else{
415:         MyPoint * tvarMyPointObj                = [[MyPoint alloc]initWithNSPoint:point];
416:         drawingTool     = finger.tool;
417:         [drawingTool.points addObject:tvarMyPointObj];
418:         [self setNeedsDisplay:YES];
419:     }
420: }
421:
422: - (void)endDraw:(CGPoint)point withFinger:(TrackedFinger*)finger {
423:
424:    if(finger == nil){
425:        MyPoint * tvarMyPointObj         = [[MyPoint alloc]initWithNSPoint:point];
426:        [drawingTool.points addObject:tvarMyPointObj];
427:        [self setNeedsDisplay:YES];
428:
429:    }else{
430:        MyPoint * tvarMyPointObj         = [[MyPoint alloc]initWithNSPoint:point];
431:        drawingTool      = finger.tool;
432:        [drawingTool.points addObject:tvarMyPointObj];
433:        [self setNeedsDisplay:YES];
434:    }
435:
436: }
437:
438:
439: -(void)mouseDown:(NSEvent *)pTheEvent {
440:     NSPoint tvarMousePointInWindow      = [pTheEvent locationInWindow];
441:         NSPoint tvarMousePointInView    = [self convertPoint:tvarMousePointInWindow fromView:nil];
442:
443:     [self beginDraw:tvarMousePointInView withFinger:nil];
444:
445: } // end mouseDown
446:
447: -(void)mouseDragged:(NSEvent *)pTheEvent {
448:
449:         NSPoint tvarMousePointInWindow  = [pTheEvent locationInWindow];
450:         NSPoint tvarMousePointInView    = [self convertPoint:tvarMousePointInWindow fromView:nil];
451:
452:     [self updateDraw:tvarMousePointInView withFinger:nil];
453:
454:
455: } // end mouseDragged
456:
457: -(void)mouseUp:(NSEvent *)pTheEvent {
458:
459:         NSPoint tvarMousePointInWindow  = [pTheEvent locationInWindow];
```

```
460:         NSPoint tvarMousePointInView    = [self convertPoint:tvarMousePointInWindow fromView:nil];
461:
462:     [self endDraw:tvarMousePointInView withFinger:nil];
463:
464:
465: } // end mouseUp
466:
467:
468: - (float)randVar;
469: {
470:         return ( (float)(rand() % 10000 ) / 10000.0);
471: } // end randVar
472:
473:
474: - (DrawingTool*)activeTool {
475:
476:     for (DrawingTool *tool in self.drawingTools) {
477:         if ([tool.toolName isEqualToString:self.toolSettings.drawingTool]) {
478:             return tool;
479:         }
480:     }
481:     return nil;
482: }
483:
484:
485: - (LPDrawingTool*)getActiveTool{
486:
487:     if (selectedTool )
488:
489:
490:     return nil;
491: }
492:
493: - (void)initializeTools {
494:
495:     self.drawingTools = [[NSMutableArray alloc] init];
496:
497:     //pen tool
498:     DrawingTool *tool = [[PenTool alloc] initWithCompletion:^{
499:
500:         //[self addDrawingToUndoStack];
501:         NSLog(@"Tool Completion");
502:
503:     }];
504:     tool.toolName = @"Pen";
505:     tool.imageName = @"pen-ink-mini.png";
506:     [self.drawingTools addObject:tool];
507:
508:
509:
510:
511: }
512:
513:
514: - (void)clearScreen{
515:
516:     [myMutaryOfBrushStrokes removeAllObjects];
517:     [self setNeedsDisplay:YES];
518: }
519:
520: - (void)setColor:(NSColor*)color{
521:
522:     currentColor = color;
523: }
524:
525: - (void)setLineWidth:(float)w{
526:
527:     currentLineWidth = w;
528:
529: }
530:
531: - (void)setTool:(NSString*)tool{
532:     selectedTool =  tool;
533: }
534:
535: - (CGPoint)covertLeapCoordinates:(CGPoint)p{
536:
537:     CGRect rect = NSRectToCGRect([self bounds]);
538:     CGSize s = rect.size;
539:     float screenCenter = 0.0f;
```

```
540:     float xScale = 1.75f;
541:     float yScale = 1.25f;
542:     return CGPointMake((s.width/2)+ (( p.x - screenCenter) * xScale), p.y * yScale);
543: }
544:
545: - (void)checkFingerExists{
546:
547:     for (id key in [trackableList allKeys]) {
548:         TrackedFinger* sprite = [trackableList objectForKey:key];
549:         if (sprite.updated) {
550:             sprite.updated = FALSE;
551:             //return;
552:         }else{
553:             [self endFingerDraw:sprite];
554:             [trackableList removeObjectForKey:key];
555:
556:         }
557:     }
558: }
559:
560: @end
```

```
 1: //
 2: //  PaintView.h
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 3/17/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Cocoa/Cocoa.h>
10: #import <OpenGL/OpenGL.h>
11: #include <OpenGL/gl.h>
12: #import "Brush.h"
13: #import "Canvas.h"
14: //CONSTANTS:
15:
16: #define kBrushOpacity           (1.0 / 3.0)
17: #define kBrushPixelStep         3
18: #define kBrushScale                     2
19: #define kLuminosity                     0.75
20: #define kSaturation                     1.0
21:
22:
23: @interface PaintView : NSOpenGLView {
24:     NSTrackingArea* trackingArea;
25: @private
26:
27:     // The pixel dimensions of the backbuffer
28:         GLint backingWidth;
29:         GLint backingHeight;
30:         CGImageRef              brushImage;
31:
32:     // OpenGL names for the renderbuffer and framebuffers used to render to this view
33:         GLuint viewRenderbuffer, viewFramebuffer;
34:
35:         GLuint  brushTexture;
36:         CGPoint location;
37:         CGPoint previousLocation;
38:         Boolean firstTouch;
39:         Boolean needsErase;
40:
41:     Brush* brush;
42:     Canvas* canvas;
43:
44: }
45:
46: @property(nonatomic, readwrite) CGPoint location;
47: @property(nonatomic, readwrite) CGPoint previousLocation;
48:
49: - (void)erase;
50: - (void)setBrushColorWithRed:(CGFloat)red green:(CGFloat)green blue:(CGFloat)blue;
51:
52:
53:
54: @end
```

```
    1: //
    2: //  PaintView.m
    3: //  LeapPaint
    4: //
    5: //  Created by cj on 3/17/13.
    6: //  Copyright (c) 2013 cjdesch. All rights reserved.
    7: //
    8:
    9: #import "PaintView.h"
   10:
   11: @implementation PaintView
   12:
   13: @synthesize  location;
   14: @synthesize  previousLocation;
   15:
   16:
   17: - (id)initWithFrame:(NSRect)frame
   18: {
   19:     NSMutableArray*     recordedPaths;
   20:         CGImageRef              brushImage;
   21:         CGContextRef    brushContext;
   22:         GLubyte                 *brushData;
   23:         size_t                  width, height;
   24:
   25:     self = [super initWithFrame:frame];
   26:     if (self) {
   27:         // Initialization code here.
   28:         canvas = [[Canvas alloc] initWithSize:frame.size];
   29:         brush = [[Brush alloc] init];
   30:
   31:         [[NSNotificationCenter defaultCenter] addObserver:self
   32:                                                  selector:@selector(_surfaceNeedsUpdate:)
   33:                                                      name:NSViewGlobalFrameDidChangeNotification
   34:                                                    object:self];
   35:
   36:         // Create a texture from an image
   37:                 // First create a UIImage object from the data in a image file, and then extract the C
ore Graphics image
   38:                 brushImage = [self nsImageToCGImageRef:[NSImage imageNamed:@"Particle.png"]];
   39:
   40:
   41:                 // Get the width and height of the image
   42:                 width = CGImageGetWidth(brushImage);
   43:                 height = CGImageGetHeight(brushImage);
   44:
   45:                 // Texture dimensions must be a power of 2. If you write an application that allows us
ers to supply an image,
   46:                 // you'll want to add code that checks the dimensions and takes appropriate action if
they are not a power of 2.
   47:
   48:                 // Make sure the image exists
   49:                 if(brushImage) {
   50:                         // Allocate  memory needed for the bitmap context
   51:                         brushData = (GLubyte *) calloc(width * height * 4, sizeof(GLubyte));
   52:                         // Use  the bitmatp creation function provided by the Core Graphics framework.
   53:                         brushContext = CGBitmapContextCreate(brushData, width, height, 8, width * 4, C
GImageGetColorSpace(brushImage), kCGImageAlphaPremultipliedLast);
   54:                         // After you create the context, you can draw the  image to the context.
   55:                         CGContextDrawImage(brushContext, CGRectMake(0.0, 0.0, (CGFloat)width, (CGFloat
)height), brushImage);
   56:                         // You don't need the context at this point, so you need to release it to avoi
d memory leaks.
   57:                         CGContextRelease(brushContext);
   58:                         // Use OpenGL ES to generate a name for the texture.
   59:                         glGenTextures(1, &brushTexture);
   60:                         // Bind the texture name.
   61:                         glBindTexture(GL_TEXTURE_2D, brushTexture);
   62:                         // Set the texture parameters to use a minifying filter and a linear filer (we
ighted average)
   63:                         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
   64:                         // Specify a 2D texture image, providing the a pointer to the image data in me
mory
   65:                         glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, GL_RGBA, GL_UNSIGNED
_BYTE, brushData);
   66:                         // Release  the image data; it's no longer needed
   67:             free(brushData);
   68:
   69:
   70:         }
   71:
```

```
   72:         // Set the view's scale factor
   73:
   74:                 //self.contentScaleFactor = 1.0;
   75:         GLfloat scale = 1.0;
   76:
   77:                 // Setup OpenGL states
   78:                 glMatrixMode(GL_PROJECTION);
   79:                 CGRect frame = self.bounds;
   80:                 //CGFloat scale = self.contentScaleFactor;
   81:                 // Setup the view port in Pixels
   82:
   83:                 //glOrthof(0, frame.size.width * scale, 0, frame.size.height * scale, -1, 1);
   84:                 glViewport(0, 0, frame.size.width * scale, frame.size.height * scale);
   85:                 glMatrixMode(GL_MODELVIEW);
   86:
   87:                 glDisable(GL_DITHER);
   88:                 glEnable(GL_TEXTURE_2D);
   89:                 glEnableClientState(GL_VERTEX_ARRAY);
   90:
   91:             glEnable(GL_BLEND);
   92:                 // Set a blending function appropriate for premultiplied alpha pixel data
   93:                 glBlendFunc(GL_ONE, GL_ONE_MINUS_SRC_ALPHA);
   94:
   95:                 //glEnable(GL_POINT_SPRITE_OES);
   96:                 //glTexEnvf(GL_POINT_SPRITE_OES, GL_COORD_REPLACE_OES, GL_TRUE);
   97:         glEnable(GL_POINT_SPRITE);
   98:         glTexEnvf(GL_POINT_SPRITE, GL_COORD_REPLACE, GL_TRUE);
   99:                 glPointSize(width / kBrushScale);
  100:
  101:                 // Make sure to start with a cleared buffer
  102:                 needsErase = YES;
  103:
  104:                 // Playback recorded path, which is "Shake Me"
  105:                 recordedPaths = [NSMutableArray arrayWithContentsOfFile:[[NSBundle mainBundle] pathFor
Resource:@"Recording" ofType:@"data"]];
  106:                 if([recordedPaths count])
  107:                         [self performSelector:@selector(playback:) withObject:recordedPaths afterDelay
:0.2];
  108:
  109:
  110:         trackingArea = [[NSTrackingArea alloc] initWithRect:self.bounds
  111:                                                     options: (NSTrackingMouseEnteredAndExited | NSTrac
kingMouseMoved | NSTrackingActiveInKeyWindow )
  112:                                                       owner:self userInfo:nil];
  113:         [self addTrackingArea:trackingArea];
  114:
  115:     }
  116:
  117:     return self;
  118: }
  119:
  120:
  121: - (void)updateTrackingAreas {
  122:     NSRect eyeBox;
  123:     [self removeTrackingArea:trackingArea];
  124:
  125:     trackingArea = [[NSTrackingArea alloc] initWithRect:self.bounds
  126:                                                 options: (NSTrackingMouseEnteredAndExited | NSTracking
MouseMoved | NSTrackingActiveInKeyWindow)
  127:                                                   owner:self userInfo:nil];
  128:     [self addTrackingArea:trackingArea];
  129:
  130: }
  131:
  132: -(void) drawRect: (NSRect) dirtyRect
  133: {
  134:
  135:
  136:
  137:
  138:     NSLog(@"Draw dirty rect");
  139:     glClearColor(0, 0, 0, 0);
  140:     glClear(GL_COLOR_BUFFER_BIT);
  141:     drawAnObject();
  142:     glFlush();
  143:
  144:
  145:
  146:     NSGraphicsContext* context = [NSGraphicsContext currentContext];
  147:
```

```
148:         [canvas drawRect:dirtyRect inContext:context];
149:
150: }
151:
152:
153: static void drawAnObject ()
154: {
155:     glColor3f(1.0f, 0.85f, 0.35f);
156:     glBegin(GL_TRIANGLES);
157:     {
158:         glVertex3f(  0.0,  0.6, 0.0);
159:         glVertex3f( -0.2, -0.3, 0.0);
160:         glVertex3f(  0.2, -0.3 ,0.0);
161:     }
162:     glEnd();
163: }
164:
165:
166: - (void) _surfaceNeedsUpdate:(NSNotification*)notification
167: {
168:     [self update];
169: }
170: /*
171: - (BOOL)createFramebuffer
172: {
173:         // Generate IDs for a framebuffer object and a color renderbuffer
174:         glGenFramebuffersOES(1, &viewFramebuffer);
175:         glGenRenderbuffersOES(1, &viewRenderbuffer);
176:
177:
178:         //glBindFramebufferOES(GL_FRAMEBUFFER_OES, viewFramebuffer);
179:     glBindFramebuffer(GL_FRAMEBUFFER, viewFramebuffer);
180:
181: //      glBindRenderbufferOES(GL_RENDERBUFFER_OES, viewRenderbuffer);
182:     glBindRenderbuffer(GL_RENDERBUFFER, viewRenderbuffer);
183:         // This call associates the storage for the current render buffer with the EAGLDrawable (our C
AEAGLLayer)
184:         // allowing us to draw into a buffer that will later be rendered to screen wherever the layer
is (which corresponds with our view).
185:
186:
187:
188:         //[context renderbufferStorage:GL_RENDERBUFFER_OES fromDrawable:(id<EAGLDrawable>)self.layer];
189:     self
190:     [self.openGLContext renderbufferStorage:GL_RENDERBUFFER fromDrawable:(id<EAGLDrawable>)self.layer]
;
191:
192:         glFramebufferRenderbufferOES(GL_FRAMEBUFFER_OES, GL_COLOR_ATTACHMENT0_OES, GL_RENDERBUFFER_OES
, viewRenderbuffer);
193:
194:         glGetRenderbufferParameterivOES(GL_RENDERBUFFER_OES, GL_RENDERBUFFER_WIDTH_OES, &backingWidth)
;
195:         glGetRenderbufferParameterivOES(GL_RENDERBUFFER_OES, GL_RENDERBUFFER_HEIGHT_OES, &backingHeigh
t);
196:
197:         // For this sample, we also need a depth buffer, so we'll create and attach one via another re
nderbuffer.
198:         glGenRenderbuffersOES(1, &depthRenderbuffer);
199:         glBindRenderbufferOES(GL_RENDERBUFFER_OES, depthRenderbuffer);
200:         glRenderbufferStorageOES(GL_RENDERBUFFER_OES, GL_DEPTH_COMPONENT16_OES, backingWidth, backingH
eight);
201:         glFramebufferRenderbufferOES(GL_FRAMEBUFFER_OES, GL_DEPTH_ATTACHMENT_OES, GL_RENDERBUFFER_OES,
 depthRenderbuffer);
202:
203:         if(glCheckFramebufferStatusOES(GL_FRAMEBUFFER_OES) != GL_FRAMEBUFFER_COMPLETE_OES)
204:         {
205:                 NSLog(@"failed to make complete framebuffer object %x", glCheckFramebufferStatusOES(GL
_FRAMEBUFFER_OES));
206:                 return NO;
207:         }
208:
209:         return YES;
210: }
211:
212:
213: // Clean up any buffers we have allocated.
214: - (void)destroyFramebuffer
215: {
216:         glDeleteFramebuffersOES(1, &viewFramebuffer);
217:         viewFramebuffer = 0;
```

```
  218:         glDeleteRenderbuffersOES(1, &viewRenderbuffer);
  219:         viewRenderbuffer = 0;
  220:
  221:         if(depthRenderbuffer)
  222:         {
  223:                 glDeleteRenderbuffersOES(1, &depthRenderbuffer);
  224:                 depthRenderbuffer = 0;
  225:         }
  226: }
  227:
  228: */
  229: // Erases the screen
  230: - (void) erase
  231: {
  232:     NSLog(@"Earse");
  233:
  234:         //[EAGLContext setCurrentContext:context];
  235:
  236:         // Clear the buffer
  237:         //glBindFramebufferOES(GL_FRAMEBUFFER_OES, viewFramebuffer);
  238:     glBindFramebuffer(GL_FRAMEBUFFER, viewFramebuffer);
  239:         glClearColor(0.0, 0.0, 0.0, 0.0);
  240:         glClear(GL_COLOR_BUFFER_BIT);
  241:
  242:         // Display the buffer
  243:         //glBindRenderbufferOES(GL_RENDERBUFFER_OES, viewRenderbuffer);
  244:     glBindRenderbuffer(GL_RENDERBUFFER, viewRenderbuffer);
  245:         //[context presentRenderbuffer:GL_RENDERBUFFER_OES];
  246:     //
  247: }
  248:
  249: // Drawings a line onscreen based on where the user touches
  250: - (void) renderLineFromPoint:(CGPoint)start toPoint:(CGPoint)end
  251: {
  252:         static GLfloat*         vertexBuffer = NULL;
  253:         static NSUInteger       vertexMax = 64;
  254:         NSUInteger                      vertexCount = 0,
  255:     count,
  256:     i;
  257:
  258:         //[EAGLContext setCurrentContext:context];
  259:         //glBindFramebufferOES(GL_FRAMEBUFFER_OES, viewFramebuffer);
  260:     glBindFramebuffer(GL_FRAMEBUFFER, viewFramebuffer);
  261:
  262:
  263:         // Convert locations from Points to Pixels
  264:         //CGFloat scale = self.contentScaleFactor;
  265:     CGFloat scale = 1.0;
  266:         start.x *= scale;
  267:         start.y *= scale;
  268:         end.x *= scale;
  269:         end.y *= scale;
  270:
  271:         // Allocate vertex array buffer
  272:         if(vertexBuffer == NULL)
  273:                 vertexBuffer = malloc(vertexMax * 2 * sizeof(GLfloat));
  274:
  275:         // Add points to the buffer so there are drawing points every X pixels
  276:         count = MAX(ceilf(sqrtf((end.x - start.x) * (end.x - start.x) + (end.y - start.y) * (end.y - s
tart.y)) / kBrushPixelStep), 1);
  277:         for(i = 0; i < count; ++i) {
  278:                 if(vertexCount == vertexMax) {
  279:                         vertexMax = 2 * vertexMax;
  280:                         vertexBuffer = realloc(vertexBuffer, vertexMax * 2 * sizeof(GLfloat));
  281:                 }
  282:
  283:                 vertexBuffer[2 * vertexCount + 0] = start.x + (end.x - start.x) * ((GLfloat)i / (GLflo
at)count);
  284:                 vertexBuffer[2 * vertexCount + 1] = start.y + (end.y - start.y) * ((GLfloat)i / (GLflo
at)count);
  285:                 vertexCount += 1;
  286:         }
  287:
  288:         // Render the vertex array
  289:         glVertexPointer(2, GL_FLOAT, 0, vertexBuffer);
  290:         glDrawArrays(GL_POINTS, 0, vertexCount);
  291:
  292:         // Display the buffer
  293:         //glBindRenderbufferOES(GL_RENDERBUFFER_OES, viewRenderbuffer);
  294:     glBindRenderbuffer(GL_RENDERBUFFER, viewRenderbuffer);
```

```
295:         //[context presentRenderbuffer:GL_RENDERBUFFER_OES];
296:     [self setNeedsDisplay:YES];
297: }
298:
299: // Reads previously recorded points and draws them onscreen. This is the Shake Me message that appears
 when the application launches.
300: - (void) playback:(NSMutableArray*)recordedPaths
301: {
302:         NSData*                         data = [recordedPaths objectAtIndex:0];
303:         CGPoint*                        point = (CGPoint*)[data bytes];
304:         NSUInteger                      count = [data length] / sizeof(CGPoint),
305:     i;
306:
307:         // Render the current path
308:         for(i = 0; i < count - 1; ++i, ++point)
309:                 [self renderLineFromPoint:*point toPoint:*(point + 1)];
310:
311:         // Render the next path after a short delay
312:         [recordedPaths removeObjectAtIndex:0];
313:         if([recordedPaths count])
314:                 [self performSelector:@selector(playback:) withObject:recordedPaths afterDelay:0.01];
315: }
316:
317: /*
318: // Handles the start of a touch
319: - (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
320: {
321:         CGRect                          bounds = [self bounds];
322:     UITouch*    touch = [[event touchesForView:self] anyObject];
323:         firstTouch = YES;
324:         // Convert touch point from UIView referential to OpenGL one (upside-down flip)
325:         location = [touch locationInView:self];
326:         location.y = bounds.size.height - location.y;
327: }
328: */
329:
330: // Handles the start of a touch
331: - (void)mouseDown:(NSEvent *)theEvent
332: {
333:         NSLog(@"Mouse up");
334:         CGRect                          bounds = [self bounds];
335:
336:     //UITouch*  touch = [[event touchesForView:self] anyObject];
337:
338:         firstTouch = YES;
339:     NSPoint mouseLocation = [self convertPoint:[theEvent locationInWindow] fromView:nil];
340:         // Convert touch point from UIView referential to OpenGL one (upside-down flip)
341:         //location = [touch locationInView:self];
342:         location.y = bounds.size.height - mouseLocation.y;
343:
344:         [brush mouseDown:theEvent inView:self onCanvas:canvas];
345: }
346:
347:
348:
349: /*
350: // Handles the continuation of a touch.
351: - (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
352: {
353:
354:         CGRect                          bounds = [self bounds];
355:         UITouch*                        touch = [[event touchesForView:self] anyObject];
356:
357:         // Convert touch point from UIView referential to OpenGL one (upside-down flip)
358:         if (firstTouch) {
359:                 firstTouch = NO;
360:                 previousLocation = [touch previousLocationInView:self];
361:                 previousLocation.y = bounds.size.height - previousLocation.y;
362:         } else {
363:                 location = [touch locationInView:self];
364:             location.y = bounds.size.height - location.y;
365:                 previousLocation = [touch previousLocationInView:self];
366:                 previousLocation.y = bounds.size.height - previousLocation.y;
367:         }
368:
369:         // Render the stroke
370:         [self renderLineFromPoint:previousLocation toPoint:location];
371: }*/
372:
373: // Handles the continuation of a touch.
```

```
374: - (void)mouseDragged:(NSEvent *)theEvent{
375:         NSLog(@"Mouse Dragged");
376:     CGRect                          bounds = [self bounds];
377:         //UITouch*                  touch = [[event touchesForView:self] anyObject];
378:     NSPoint mouseLocation = [self convertPoint:[theEvent locationInWindow] fromView:nil];
379:
380:
381:         // Convert touch point from UIView referential to OpenGL one (upside-down flip)
382:         if (firstTouch) {
383:                 firstTouch = NO;
384:
385:                 //previousLocation = [touch previousLocationInView:self];
386:         previousLocation = location;
387:                 previousLocation.y = bounds.size.height - previousLocation.y;
388:         } else {
389:                 //location = [touch locationInView:self];
390:         previousLocation = location;
391:                 previousLocation.y = bounds.size.height - previousLocation.y;
392:         location = mouseLocation;
393:             location.y = bounds.size.height - location.y;
394:
395:         }
396:
397:         // Render the stroke
398:         //[self renderLineFromPoint:previousLocation toPoint:location];
399:     //[self drawSomething];
400:
401:         [brush mouseDragged:theEvent inView:self onCanvas:canvas];
402: }
403:
404: - (void)drawSomething{
405:
406:
407: }
408:
409: /*
410: // Handles the end of a touch event when the touch is a tap.
411: - (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
412: {
413:         CGRect                          bounds = [self bounds];
414:     UITouch*    touch = [[event touchesForView:self] anyObject];
415:         if (firstTouch) {
416:                 firstTouch = NO;
417:                 previousLocation = [touch previousLocationInView:self];
418:                 previousLocation.y = bounds.size.height - previousLocation.y;
419:                 [self renderLineFromPoint:previousLocation toPoint:location];
420:         }
421: }
422: */
423: - (void)mouseUp:(NSEvent *)theEvent{
424:
425:     NSLog(@"Mouse up");
426:     CGRect                          bounds = [self bounds];
427:     //UITouch*  touch = [[event touchesForView:self] anyObject];
428:
429:     NSPoint mouseLocation = [self convertPoint:[theEvent locationInWindow] fromView:nil];
430:
431:         if (firstTouch) {
432:                 firstTouch = NO;
433:                 //previousLocation = [touch previousLocationInView:self];
434:                 //previousLocation.y = bounds.size.height - previousLocation.y;
435:         location = mouseLocation;
436:                 //[self renderLineFromPoint:previousLocation toPoint:location];
437:         }
438:
439:         [brush mouseUp:theEvent inView:self onCanvas:canvas];}
440: /*
441: // Handles the end of a touch event.
442: - (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event
443: {
444:         // If appropriate, add code necessary to save the state of the application.
445:         // This application is not saving state.
446: }*/
447:
448: // Handles the end of a touch event.
449: - (void)mouseExited:(NSEvent *)theEvent
450: {
451:         // If appropriate, add code necessary to save the state of the application.
452:         // This application is not saving state.
453: }
```

```
454:
455:
456: - (void)setBrushColorWithRed:(CGFloat)red green:(CGFloat)green blue:(CGFloat)blue
457: {
458:         // Set the brush color using premultiplied alpha values
459:         glColor4f(red   * kBrushOpacity,
460:                           green * kBrushOpacity,
461:                           blue  * kBrushOpacity,
462:                           kBrushOpacity);
463: }
464:
465:
466: //Move to utilities
467: - (CGImageRef)nsImageToCGImageRef:(NSImage*)image;
468: {
469:     NSData * imageData = [image TIFFRepresentation];
470:     CGImageRef imageRef;
471:     if(!imageData) return nil;
472:     //Bridge
473:     CGImageSourceRef imageSource = CGImageSourceCreateWithData((__bridge CFDataRef)imageData, NULL);
474:     imageRef = CGImageSourceCreateImageAtIndex(imageSource, 0, NULL);
475:     return imageRef;
476: }
477:
478:
479: void ManipulateImagePixelData(CGImageRef inImage)
480: {
481:     // Create the bitmap context
482:     CGContextRef cgctx = CreateARGBBitmapContext(inImage);
483:     if (cgctx == NULL)
484:     {
485:         // error creating context
486:         return;
487:     }
488:
489:     // Get image width, height. We'll use the entire image.
490:     size_t w = CGImageGetWidth(inImage);
491:     size_t h = CGImageGetHeight(inImage);
492:     CGRect rect = {{0,0},{w,h}};
493:
494:     // Draw the image to the bitmap context. Once we draw, the memory
495:     // allocated for the context for rendering will then contain the
496:     // raw image data in the specified color space.
497:     CGContextDrawImage(cgctx, rect, inImage);
498:
499:     // Now we can get a pointer to the image data associated with the bitmap
500:     // context.
501:     void *data = CGBitmapContextGetData (cgctx);
502:     if (data != NULL)
503:     {
504:
505:         // **** You have a pointer to the image data ****
506:
507:         // **** Do stuff with the data here ****
508:
509:     }
510:
511:     // When finished, release the context
512:     CGContextRelease(cgctx);
513:     // Free image data memory for the context
514:     if (data)
515:     {
516:         free(data);
517:     }
518:
519: }
520:
521: CGContextRef CreateARGBBitmapContext (CGImageRef inImage)
522: {
523:     CGContextRef    context = NULL;
524:     CGColorSpaceRef colorSpace;
525:     void *          bitmapData;
526:     int             bitmapByteCount;
527:     int             bitmapBytesPerRow;
528:
529:     // Get image width, height. We'll use the entire image.
530:     size_t pixelsWide = CGImageGetWidth(inImage);
531:     size_t pixelsHigh = CGImageGetHeight(inImage);
532:
533:     // Declare the number of bytes per row. Each pixel in the bitmap in this
```

```
534:     // example is represented by 4 bytes; 8 bits each of red, green, blue, and
535:     // alpha.
536:     bitmapBytesPerRow   = (pixelsWide * 4);
537:     bitmapByteCount     = (bitmapBytesPerRow * pixelsHigh);
538:
539:     // Use the generic RGB color space.
540:     colorSpace = CGColorSpaceCreateWithName(kCGColorSpaceGenericRGB);
541:     if (colorSpace == NULL)
542:     {
543:         fprintf(stderr, "Error allocating color space\n");
544:         return NULL;
545:     }
546:
547:     // Allocate memory for image data. This is the destination in memory
548:     // where any drawing to the bitmap context will be rendered.
549:     bitmapData = malloc( bitmapByteCount );
550:     if (bitmapData == NULL)
551:     {
552:         fprintf (stderr, "Memory not allocated!");
553:         CGColorSpaceRelease( colorSpace );
554:         return NULL;
555:     }
556:
557:     // Create the bitmap context. We want pre-multiplied ARGB, 8-bits
558:     // per component. Regardless of what the source image format is
559:     // (CMYK, Grayscale, and so on) it will be converted over to the format
560:     // specified here by CGBitmapContextCreate.
561:     context = CGBitmapContextCreate (bitmapData,
562:                                      pixelsWide,
563:                                      pixelsHigh,
564:                                      8,      // bits per component
565:                                      bitmapBytesPerRow,
566:                                      colorSpace,
567:                                      kCGImageAlphaPremultipliedFirst);
568:     if (context == NULL)
569:     {
570:         free (bitmapData);
571:         fprintf (stderr, "Context not created!");
572:     }
573:
574:     // Make sure and release colorspace before returning
575:     CGColorSpaceRelease( colorSpace );
576:
577:     return context;
578: }
579:
580: @end
```

```
 1: //
 2: //  PenTool.h
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 3/17/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10: #import "DrawingTool.h"
11:
12: @interface PenTool : DrawingTool
13:
14: @property (assign) float pathDistance;
15: @property (assign) CGPoint previousPoint1;
16: @property (assign) CGPoint previousPoint2;
17: @property (assign) CGMutablePathRef path;
18: @property (assign) CGPoint lastPoint;
19:
20:
21: - (void)createPath;
22:
23:
24: @end
```

```
  1: //
  2: //  PenTool.m
  3: //  LeapPaint
  4: //
  5: //  Created by cj on 3/17/13.
  6: //  Copyright (c) 2013 cjdesch. All rights reserved.
  7: //
  8:
  9: #import "PenTool.h"
 10:
 11: @implementation PenTool
 12:
 13: static const CGFloat kPointMinDistance = 5;
 14: static const CGFloat kPointMinDistanceSquared = kPointMinDistance * kPointMinDistance;
 15:
 16: - (void)inputBegan:(NSEvent *)theEvent locationInView:(CGPoint)currentPoint withSettings:(ToolSettings
*)settings{
 17:
 18:     //[super touchBegan:touch inImageView:imageView withSettings:settings];
 19:     [super inputBegan:theEvent locationInView:currentPoint withSettings:settings];
 20:
 21:     [self createPath];
 22:
 23:     //self.previousPoint1 = [touch previousLocationInView:self.drawingImageView];
 24:     //self.previousPoint2 = [touch previousLocationInView:self.drawingImageView];
 25:
 26:     self.previousPoint1 = currentPoint;
 27:     self.previousPoint2 = currentPoint;
 28:
 29:
 30:     [self inputMoved:theEvent locationInView:currentPoint];
 31:
 32: }
 33:
 34: - (void)inputMoved:(NSEvent *)theEvent locationInView:(CGPoint)currentPoint{
 35:
 36:     //CGPoint currentPoint = [touch locationInView:self.drawingImageView];
 37:     //NSPoint tvarMousePointInWindow    = [theEvent locationInWindow];
 38:
 39:
 40:     /* check if the point is farther than min dist from previous */
 41:     CGFloat dx = currentPoint.x - self.lastPoint.x;
 42:     CGFloat dy = currentPoint.y - self.lastPoint.y;
 43:
 44:     if (dx * dx + dy * dy < kPointMinDistanceSquared) {
 45:         return;
 46:     }
 47:
 48:     //call touch moved after the above check - image will be restored
 49:     [super inputMoved:theEvent locationInView:currentPoint];
 50:
 51:     [self drawCurveEndingAtTouch:theEvent locationInView:currentPoint];
 52:
 53:     if (self.pathDistance > 100) {
 54:         //commit changes and recreate path, faster
 55:         self.drawingSnapshot = self.drawingImageView.image;
 56:         [self createPath];
 57:     }
 58:
 59:     self.lastPoint = currentPoint;
 60:
 61: }
 62:
 63: - (void)inputEnded:(NSEvent *)theEvent locationInView:(CGPoint)currentPoint{
 64:
 65:
 66:
 67:     //first reset the image, we're still adjusting/drawing the curve
 68:     //self.drawingImageView.image = self.drawingSnapshot;
 69:
 70:     //finish the curve along to the final touch
 71:     [self drawCurveEndingAtTouch:theEvent locationInView:currentPoint];
 72:
 73:     [self releasePath];
 74:
 75:     [super inputEnded:theEvent locationInView:currentPoint];
 76:
 77: }
 78:
 79: // drawing curves using quadratic beziers based on https://github.com/levinunnink/Smooth-Line-View
```

```
 80: - (void)drawCurveEndingAtTouch:(NSEvent *)theEvent locationInView:(CGPoint)currentPoint{
 81:
 82:     [self addTouchToPath:theEvent locationInView:currentPoint];
 83:
 84:     [self drawCurveForPath];
 85:
 86: }
 87:
 88:
 89: - (void)addTouchToPath:(NSEvent *)theEvent  locationInView:(CGPoint)currentPoint{
 90:
 91:
 92:     self.previousPoint2 = self.previousPoint1;
 93:     self.previousPoint1 = currentPoint;
 94:
 95:     CGPoint mid1 = midPoint(self.previousPoint1, self.previousPoint2);
 96:     CGPoint mid2 = midPoint(currentPoint, self.previousPoint1);
 97:     CGMutablePathRef subpath = CGPathCreateMutable();
 98:
 99:     CGPathMoveToPoint(subpath, NULL, mid1.x, mid1.y);
100:     CGPathAddQuadCurveToPoint(subpath, NULL, self.previousPoint1.x, self.previousPoint1.y, mid2.x, mid
2.y);
101:
102:     CGPathAddPath(self.path, NULL, subpath);
103:
104:     CGFloat dx = currentPoint.x - self.lastPoint.x;
105:     CGFloat dy = currentPoint.y - self.lastPoint.y;
106:     float lastDistance = sqrt(dx * dx + dy * dy);
107:     self.pathDistance += lastDistance;
108:
109:     CGPathRelease(subpath);
110:
111: }
112:
113: - (void)drawCurveForPath {
114:
115:     //[self setupImageContextForDrawing];
116:     //NSGraphicsContext* theContext = [NSGraphicsContext currentContext];
117:     //CGContextRef context = UIGraphicsGetCurrentContext();
118:     CGContextRef context = (CGContextRef)[[NSGraphicsContext currentContext] graphicsPort];
119:
120:     CGContextAddPath(context, self.path);
121:     CGContextStrokePath(context);
122:     //self.drawingImageView.image = UIGraphicsGetImageFromCurrentImageContext();
123:
124:
125:     CGContextFlush(context);
126:
127:
128: }
129:
130: CGPoint midPoint(CGPoint p1, CGPoint p2) {
131:     return CGPointMake((p1.x + p2.x) * 0.5, (p1.y + p2.y) * 0.5);
132: }
133:
134: - (void)releasePath {
135:
136:     if (_path != nil) {
137:         CGPathRelease(_path);
138:         _path = nil;
139:     }
140:     self.pathDistance = 0.0;
141:
142: }
143:
144: - (void)createPath {
145:
146:     [self releasePath];
147:
148:     //analyzer will report this leaks - this is released in the above method
149:     self.path = CGPathCreateMutable();
150:     self.pathDistance = 0.0;
151:
152: }
153:
154:
155:
156: @end
```

```
 1: //
 2: //  SimplePoint.h
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 2/19/13.
 6: //
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10:
11:
12: @interface SimplePoint : NSObject {
13:
14: }
15:
16: @property (nonatomic, readwrite) float x;
17: @property (nonatomic, readwrite) float y;
18: //@property (nonatomic, readwrite) float z;
19: - (id)initWithPosition:(CGPoint)p;
20: - (id)initWithX:(float)x withY:(float)y;
21:
22:
23: @end
```

```
 1: //
 2: //  SimplePoint.m
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 2/19/13.
 6: //
 7: //
 8:
 9: #import "SimplePoint.h"
10:
11: @implementation SimplePoint
12:
13: - (id)initWithPosition:(CGPoint)p{
14:     if (self = [super init]) {
15:
16:         self.x = p.x;
17:         self.y = p.y;
18:
19:     }
20:     return self;
21: }
22:
23: - (id)initWithX:(float)x withY:(float)y{
24:
25:     if (self = [super init]) {
26:
27:         self.x = x;
28:         self.y = y;
29:
30:     }
31:     return self;
32: }
33:
34:
35: @end
```

```
 1: //
 2: //  ToolSettings.h
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 3/18/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10:
11: @interface ToolSettings : NSObject
12:
13:
14: @property (copy) NSString *drawingTool;
15: @property (assign) int lineWidth;
16: @property (assign) int transparency;
17: @property (assign) int fontSize;
18: @property (strong) NSColor *primaryColor;
19: @property (strong) NSColor *secondaryColor;
20:
21: - (void)loadFromUserDefaults;
22: - (void)saveToUserDefaults;
23:
24: @end
```

```
    1: //
    2: //  ToolSettings.m
    3: //  LeapPaint
    4: //
    5: //  Created by cj on 3/18/13.
    6: //  Copyright (c) 2013 cjdesch. All rights reserved.
    7: //
    8:
    9: #import "ToolSettings.h"
   10:
   11: @implementation ToolSettings
   12:
   13: NSString* const kSDSettingsColor1RedKey      =  @"DRAWING_COLOR1_RED";
   14: NSString* const kSDSettingsColor1BlueKey     =  @"DRAWING_COLOR1_BLUE";
   15: NSString* const kSDSettingsColor1GreenKey    =  @"DRAWING_COLOR1_GREEN";
   16: NSString* const kSDSettingsColor1AlphaKey    =  @"DRAWING_COLOR1_ALPHA";
   17: NSString* const kSDSettingsColor2RedKey      =  @"DRAWING_COLOR2_RED";
   18: NSString* const kSDSettingsColor2BlueKey     =  @"DRAWING_COLOR2_BLUE";
   19: NSString* const kSDSettingsColor2GreenKey    =  @"DRAWING_COLOR2_GREEN";
   20: NSString* const kSDSettingsColor2AlphaKey    =  @"DRAWING_COLOR2_ALPHA";
   21: NSString* const kSDSettingsLineWidth         =  @"DRAWING_LINE_WIDTH";
   22: NSString* const kSDSettingsTransparency      =  @"DRAWING_TRANSPARENCY";
   23: NSString* const kSDSettingsDrawingTool       =  @"DRAWING_TOOL";
   24: NSString* const kSDSettingsFontSize          =  @"DRAWING_FONT_SIZE";
   25:
   26:
   27: - (void)loadFromUserDefaults {
   28:
   29:     NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
   30:
   31:     if ([defaults objectForKey:kSDSettingsColor1AlphaKey]) {
   32:         self.primaryColor = [NSColor colorWithCalibratedRed:[defaults floatForKey:kSDSettingsColor1Red
Key] green:[defaults floatForKey:kSDSettingsColor1GreenKey] blue:[defaults floatForKey:kSDSettingsColor1BlueK
ey] alpha:[defaults floatForKey:kSDSettingsColor1AlphaKey]];
   33:     } else {
   34:         self.primaryColor = [NSColor blueColor];
   35:     }
   36:
   37:     if ([defaults objectForKey:kSDSettingsColor2AlphaKey]) {
   38:
   39:
   40:         self.secondaryColor = [NSColor colorWithCalibratedRed:[defaults floatForKey:kSDSettingsColor2R
edKey] green:[defaults floatForKey:kSDSettingsColor2GreenKey] blue:[defaults floatForKey:kSDSettingsColor2Blu
eKey] alpha:[defaults floatForKey:kSDSettingsColor2AlphaKey]];
   41:     } else {
   42:         self.secondaryColor = [NSColor redColor];
   43:     }
   44:
   45:     if ([defaults objectForKey:kSDSettingsLineWidth]) {
   46:         self.lineWidth = [defaults integerForKey:kSDSettingsLineWidth];
   47:     } else {
   48:         self.lineWidth = 10;
   49:     }
   50:
   51:     if ([defaults objectForKey:kSDSettingsTransparency]) {
   52:         self.transparency = [defaults integerForKey:kSDSettingsTransparency];
   53:     } else {
   54:         self.transparency = 0;
   55:     }
   56:
   57:     if ([defaults objectForKey:kSDSettingsDrawingTool]) {
   58:         self.drawingTool = [defaults stringForKey:kSDSettingsDrawingTool];
   59:         //fix old stored setting (Brush #1, Brush #2, etc)
   60:         if ([self.drawingTool rangeOfString:@"Brush #"].location != NSNotFound) {
   61:             self.drawingTool = @"Brush";
   62:         }
   63:     } else {
   64:         self.drawingTool = @"Pen";
   65:     }
   66:
   67:     if ([defaults objectForKey:kSDSettingsFontSize]) {
   68:         self.fontSize = [defaults integerForKey:kSDSettingsFontSize];
   69:     } else {
   70:         self.fontSize = 50;
   71:     }
   72:
   73: }
   74:
   75: - (void)saveToUserDefaults {
   76:
```

```
77:         NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
78:
79:     CGFloat red = 0.0, green = 0.0, blue = 0.0, alpha =0.0;
80:     [self.primaryColor getRed:&red green:&green blue:&blue alpha:&alpha];
81:
82:     [defaults setFloat:red forKey:kSDSettingsColor1RedKey];
83:     [defaults setFloat:green forKey:kSDSettingsColor1GreenKey];
84:     [defaults setFloat:blue forKey:kSDSettingsColor1BlueKey];
85:     [defaults setFloat:alpha forKey:kSDSettingsColor1AlphaKey];
86:
87:     [self.secondaryColor getRed:&red green:&green blue:&blue alpha:&alpha];
88:
89:     [defaults setFloat:red forKey:kSDSettingsColor2RedKey];
90:     [defaults setFloat:green forKey:kSDSettingsColor2GreenKey];
91:     [defaults setFloat:blue forKey:kSDSettingsColor2BlueKey];
92:     [defaults setFloat:alpha forKey:kSDSettingsColor2AlphaKey];
93:
94:     [defaults setInteger:self.lineWidth forKey:kSDSettingsLineWidth];
95:     [defaults setInteger:self.transparency forKey:kSDSettingsTransparency];
96:     [defaults setObject:self.drawingTool forKey:kSDSettingsDrawingTool];
97:     [defaults setInteger:self.fontSize forKey:kSDSettingsFontSize];
98:
99:     [defaults synchronize];
100:
101: }
102:
103: @end
```

```
 1: //
 2: //  TrackedFinger.h
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 2/8/13.
 6: //
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10: #import "LPDrawingTool.h"
11: @interface TrackedFinger : NSObject
12:
13: @property (nonatomic, strong) NSString* fingerID;
14: @property (nonatomic, readwrite) BOOL updated;
15: @property (nonatomic, readwrite) CGPoint position;
16: @property (weak) LPDrawingTool* tool;
17:
18: - (id)initWithID:(NSString*)finger withPosition:(CGPoint)p;
19: @end
```

```
 1: //
 2: //  TrackedFinger.m
 3: //  LeapPuzz
 4: //
 5: //  Created by cj on 2/8/13.
 6: //
 7: //
 8:
 9: #import "TrackedFinger.h"
10:
11: @implementation TrackedFinger
12: @synthesize tool;
13: - (id)initWithID:(NSString*)finger withPosition:(CGPoint)p{
14:     if (self = [super init]) {
15:         self.fingerID = finger;
16:         self.updated = TRUE;
17:         self.position = p;
18:     }
19:     return self;
20: }
21: @end
```

Bibliography

[1] David Pierce. A look inside leap motion, the 3d gesture control that's like kinect on steroids, June 26 2012. URL http://www.theverge.com/2012/6/26/3118592/leap-motion-gesture-controls.

[2] Apple Computer Inc. Mac OS and iOS API, 2013. URL http://developer.apple.com.

[3] Leap Motion Inc. Leapmotion, April 2013. URL http://www.leapMotion.com.

[4] Stacey D. Scott, Regan L. Mandryk, and Kori Inkpen. Understanding children's collaborative interactions in shared environments. *J. Comp. Assisted Learning*, 19(2): 220–228, 2003. URL http://dx.doi.org/10.1046/j.0266-4909.2003.00022.x.

[5] Vanessa Colella, Richard Borovoy, and Mitchel Resnick. Participatory simulations: Using computational objects to learn about dynamic systems. In *Proceedings of ACM CHI 98 Conference on Human Factors in Computing Systems*, 1998.

[6] Allison Druin. Cooperative inquiry: developing new technologies for children with children. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 592–599, New York, NY, USA, 1999. ACM. ISBN 0-201-48559-1. doi: 10.1145/302979.303166. URL http://doi.acm.org/10.1145/302979.303166.

[7] Allison Druin. The role of children in the design of new technology. *Behaviour and Information Technology*, 21:1–25, 2002.

[8] Shari Lawrence Pfleeger and Joanne M. Atlee. *Software Engineering: Theory and Practice (4th Edition)*. Prentice Hall, 2009. ISBN 0136061699. URL

http://www.amazon.com/Software-Engineering-Theory-Practice-Edition/dp/0136061699%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0136061699.

[9] Nayan B. Ruparelia. Software development lifecycle models. *SIGSOFT Softw. Eng. Notes*, 35(3):8–13, May 2010. ISSN 0163-5948. doi: 10.1145/1764810.1764814. URL http://doi.acm.org/10.1145/1764810.1764814.

[10] Unity Technologies. Unity SDK, 2013. URL http://unity3d.com/.

[11] Community Project. Cocos2d game engine framework. URL http://www.cocos2d-iphone.org/.

[12] Ray Wenderlich. Breakout. URL http://www.raywenderlich.com/28604/how-to-create-a-breakout-game-with-box2d-and-cocos2d-2-x-tutorial-part-1.

[13] Yannick Loriot. Cccontrolextension. URL https://github.com/YannickL/CCControlExtension.

[14] Nick Gillian. Gesture recognition toolkit. URL https://code.google.com/p/gesture-recognition-toolkit/.