```
    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:
   15:  Serves as an application wide callback object for events that affects the whole application, such as
low-memory, etc.
   16:  */
   17: @interface AppDelegate : NSObject <NSApplicationDelegate>
   18: {
   19:
   20:     NSWindow    *window_;  /**< window is the main window to be displayed */
   21:     CCGLView    *glView_;  /**< glView is the embedded view in which cocos2d will run inside the windo
w */
   22: }
   23: @property (strong) IBOutlet NSWindow *window;
   24: @property (strong) IBOutlet CCGLView *glView;
   25:
   26: /** RunGameSceen sets up the Cocos2d environment and runs it in the application.
   27:  */
   28: - (void)runGameScene;
   29:
   30: /** Toggles from a window to full screen view point
   31:  @param sender is the action sending the command
   32:  @return IBAction binding to interface builder
   33:  */
   34: - (IBAction)toggleFullScreen:(id)sender;
   35:
   36: @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: {
18:     // Insert code here to initialize your application
19:     [self runGameScene];
20: }
21:
22:
23: - (void)runGameScene{
24:
25:     CCDirectorMac *director = (CCDirectorMac*) [CCDirector sharedDirector];
26:
27:     //NSRect screensFrame = [[NSScreen mainScreen] frame];
28:     NSRect screensFrame = [[NSScreen mainScreen] visibleFrame];
29:
30:
31:     [glView_ setFrameSize:NSMakeSize(screensFrame.size.width,screensFrame.size.height)];
32:         // enable FPS and SPF
33:         [director setDisplayStats:YES];
34:
35:         // connect the OpenGL view with the director
36:         [director setView:glView_];
37:
38:         // EXPERIMENTAL stuff.
39:         // 'Effects' don't work correctly when autoscale is turned on.
40:         // Use kCCDirectorResize_NoScale if you don't want auto-scaling.
41:         [director setResizeMode:kCCDirectorResize_AutoScale];
42:
43:     //[glView_ setFrameSize:NSMakeSize(window_.frame.size.width,window_.frame.size.height-42)];
44:         // Enable "moving" mouse event. Default no.
45:         [window_ setAcceptsMouseMovedEvents:NO];
46:
47:         // Center main window
48:         [window_ center];
49:
50:         //CCScene *scene = [GameScene node];
51:
52:     CCScene* scene  = [GameScene scene];
53:         //[scene addChild:[GameScene node]];
54:
55:
56:
57:         [director runWithScene:scene];
58:
59: }
60:
61:
62:
63: #pragma mark AppDelegate - IBActions
64:
65: - (IBAction)toggleFullScreen: (id)sender
66: {
67:         CCDirectorMac *director = (CCDirectorMac*) [CCDirector sharedDirector];
68:         [director setFullScreen: ! [director isFullScreen] ];
69: }
70:
71:
72:
73:
74: @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: /**
   21:  Callback with a change in color of the brush
   22:  @param color is the new selected color value
   23:  */
   24: - (void)changeColorControl:(ccColor3B)color;
   25: /**
   26:  Callback with a change in thickness of the brush
   27:  @param value is the new selected color value
   28:  */
   29: - (void)changeThicknessControl:(float)value;
   30: /**
   31:  Callback with a change in brush texture
   32:  @param brushname is the new selected brush value
   33:  */
   34: - (void)changeBrushControl:(NSString*)brushname;
   35: /**
   36:  Callback with a change in opacity
   37:  @param value is the new selected opacity value
   38:  */
   39: - (void)changeOpacityControl:(float)value;
   40: /**
   41:  Callback to notify to clear the drawing
   42:  */
   43: - (void)clearDrawing;
   44: /**
   45:  Callback with a change in color
   46:  @param mode is the  toggled eraser mode
   47:  TODO: Turn off eraser mode when new color is selected
   48:  */
   49: - (void)eraserMode:(bool)mode;
   50:
   51:
   52: @end
   53:
   54: /** Controls Layer
   55:     User inferface controls for operating buttons, switches, sliders
   56:  */
   57:
   58: @interface ControlsLayer : CCLayer <BrushSelectionLayerDelegate>{
   59:
   60:     CCLabelTTF *colorLabel;         /**< colorLabel displays name of color in hash value */
   61:     CCLabelTTF *displayValueLabel;  /**< displayValueLabel displays coordinate */
   62:     GameSettings* gameSettings;     /**< gameSettings global reference to shared settings instance */
   63: }
   64: @property (nonatomic, strong) CCControlSlider   *slider;                   /**< slider is the thickne
ss control of the brush */
   65: @property (nonatomic, strong) CCControlSlider   *opacitySlider;            /**< opacitySlider is the
opacity contro of the brush*/
   66: @property (nonatomic, strong) CCControlSwitch   *opacitySwitchControl;      /**< opacitySwitchControl
is the control for setting automatic or manual opacity control */
   67: @property (nonatomic, strong) CCLabelTTF        *opacitydisplayValueLabel;  /**<  opacitydisplayValueL
abel shows the state of the opacitySwitchControl*/
   68: @property (nonatomic, weak) id <ControlsLayerDelegate> delegate;           /**< delegate is the insta
nce reference for triggering delegate call back functions */
   69: @property (nonatomic,strong) BrushSelectionLayer *brushSelection;          /**< brushSelection layer
expands as a drawer to allow for brush selection */
   70: @property (nonatomic, strong) CCLabelTTF        *displayValueLabel;         /**< displayValueLabel dis
plays eraser toggle state */
   71: @property (nonatomic, strong) CCControlSwitch   *switchControl;            /**< switchControl is the
eraser toggle */
   72:
```

```
 73: /**
 74:  Recieves brushSizeControl delegate callbacks and updates values in the interface
 75:  @param sender is the object performing the callback
 76:  */
 77: - (void)valueChanged:(CCControlSlider *)sender;
 78:
 79: /**
 80:  Recieves opacitySliderControl delegate callbacks and updates values in the interface
 81:  @param sender is the object performing the callback
 82:  */
 83: - (void)opacitySliderChanged:(CCControlSlider *)sender;
 84:
 85: /** Expands brushes panel*/
 86: - (void)expandPanel;
 87: /** Collapses Brushes Panel */
 88: - (void)collapsePanel;
 89:
 90: /** Creates and returns a new CCControlSwitch.
 91:  @return a generate ControlSwitch
 92:  */
 93: - (CCControlSwitch *)makeControlSwitch;
 94: /** Callback for the change value.
 95:   @param sender is the object performing the callback*/
 96: - (void)switchValueChanged:(CCControlSwitch *)sender;
 97: /** Callback for opacity changing with the slider
 98:   @param sender is the object performing the callback
 99:  */
100: - (void)updateOpacitySlider:(float)value;
101:
102: @end
```

```
//
//  ControlsLayer.m
//  LeapPuzz
//
//  Created by cj on 4/9/13.
//
//

#import "ControlsLayer.h"

@implementation ControlsLayer
@synthesize slider;
@synthesize opacitySlider;
@synthesize opacitydisplayValueLabel;
@synthesize opacitySwitchControl;
@synthesize delegate;

@synthesize displayValueLabel;
@synthesize switchControl;
@synthesize brushSelection;
- (id)init
{
        if ((self = [super init]))
        {
                // Get window size
                CGSize screenSize = [[CCDirector sharedDirector] winSize];

        gameSettings = [GameSettings sharedInstance];

        //LayerBackground


        [self sliderinit];
        //[self buttoninit];
        //[self initEraserButton];
        [self initEraserSwitch];
        [self colorpickerinit];

        [self initResetButton];

        //Open button

        //Close Button
        self.brushSelection = [BrushSelectionLayer node];
        self.brushSelection.position = ccp(-screenSize.width,0);

        self.brushSelection.delegate = self;

        [self addChild:brushSelection z:10];
        [self initBrushSelectionButton];
        [self opacitySliderInit];
        [self initOpacitySwitch];
        //buttons

        // Color picker

        //


        }
        return self;
}


- (void)sliderinit{

    //Slideer
    CGSize screenSize = [[CCDirector sharedDirector] winSize];
    // Add the slider
    self.slider                     = [CCControlSlider sliderWithBackgroundFile:@"sliderTrack.png"
                                                                    progressFile:@"sliderProgress.png"
                                                                       thumbFile:@"sliderThumb.png"];
    self.slider.anchorPoint             = ccp(0.5f, 1.0f);
    self.slider.minimumValue            = 0.0f; // Sets the min value of range
    self.slider.maximumValue            = 5.0f; // Sets the max value of range
    self.slider.position                = ccp(screenSize.width / 2.0f, 100);


    CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Size" fontName:@"Marker Felt" fontSize:30]
;
```

```
   80:     titleButton.position =ccp(screenSize.width / 2.0f , 125);
   81:
   82:     // Add the black background
   83:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
   84:     [background setContentSize:CGSizeMake(350,100)];
   85:     [background setPosition:ccp(screenSize.width / 2.0f, 100)];
   86:     [self addChild:background];
   87:
   88:     [self addChild:titleButton];
   89:     // When the value of the slider will change, the given selector will be call
   90:     [self.slider addTarget:self action:@selector(valueChanged:) forControlEvents:CCControlEventValueCh
anged];
   91:
   92:     [self addChild:self.slider z:0 tag:1];
   93: }
   94:
   95: - (void)valueChanged:(CCControlSlider *)sender
   96: {
   97:         // Change value of label.
   98:         //NSLog(@"slider value %@", [NSString stringWithFormat:@"Slider value = %.02f", sender.value])
;
   99:     [self.delegate changeThicknessControl:sender.value];
  100: }
  101:
  102:
  103: - (void)opacitySliderInit{
  104:
  105:     //Slideer
  106:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
  107:     CCNode *layer                       = [CCNode node];
  108:     layer.position                      =ccp(screenSize.width / 2.0f + 200, 100);
  109:     [self addChild:layer z:3];
  110:
  111:     double layer_width = 0;
  112:
  113:
  114:     // Add the black background
  115:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
  116:     [background setContentSize:CGSizeMake(350,100)];
  117:     [background setPosition:ccp(background.contentSize.width / 2.0f, 0)];
  118:     [layer addChild:background];
  119:     layer_width += background.contentSize.width;
  120:
  121:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Opacity" fontName:@"Marker Felt" fontSize:
30];
  122:     titleButton.position =ccp(layer_width / 2.0f , 25);
  123:
  124:
  125:     [layer addChild:titleButton];
  126:     // When the value of the slider will change, the given selector will be call
  127:
  128:     // Add the slider
  129:     self.opacitySlider                    = [CCControlSlider sliderWithBackgroundFile:@"sliderTrack.p
ng"
  130:                                                                                progressFile:@"sliderProgres
s.png"
  131:                                                                                   thumbFile:@"sliderThumb.p
ng"];
  132:     self.opacitySlider.anchorPoint              = ccp(0.5f, 1.0f);
  133:     self.opacitySlider.minimumValue             = 0.0f; // Sets the min value of range
  134:     self.opacitySlider.maximumValue             = 100.0f; // Sets the max value of range
  135:     self.opacitySlider.position                 = ccp(layer_width  / 2.0f, 0);
  136:
  137:     [self.opacitySlider addTarget:self action:@selector(opacitySliderChanged:) forControlEvents:CCCont
rolEventValueChanged];
  138:
  139:     [layer addChild:self.opacitySlider z:0 tag:2];
  140: }
  141:
  142: - (void)opacitySliderChanged:(CCControlSlider *)sender
  143: {
  144:
  145:         // Change value of label.
  146:         //NSLog(@"slider value %@", [NSString stringWithFormat:@"Slider value = %.02f", sender.value])
;
  147:     [self.delegate changeOpacityControl:sender.value];
  148: }
  149:
  150: - (void)updateOpacitySlider:(float)value{
  151:
```

```
  152:
  153:     //ensure the value is within its bounds
  154:     if(value > self.opacitySlider.maximumValue){
  155:         //Max Value
  156:         self.opacitySlider.value = self.opacitySlider.maximumValue;
  157:     }else if(value < self.opacitySlider.minimumValue){
  158:         //Min Value
  159:         self.opacitySlider.value = self.opacitySlider.minimumValue;
  160:     }else{
  161:         self.opacitySlider.value = value;
  162:     }
  163: }
  164:
  165:
  166: - (void)initOpacitySwitch{
  167:
  168:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
  169:
  170:     CCNode *layer               = [CCNode node];
  171:     //layer.position              = ccp (screenSize.width / 2, screenSize.height / 2);
  172:     layer.position = ccp(screenSize.width / 2.0f + 400, 125);
  173:     [self addChild:layer z:5];
  174:
  175:     double layer_width = 0;
  176:
  177:     // Add the black background for the text
  178:     CCScale9Sprite *background  = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
  179:     background.contentSize      = CGSizeMake(80, 50);
  180:     background.position         = ccp(layer_width + background.contentSize.width / 2.0f, 0);
  181:     //[layer addChild:background];
  182:
  183:     layer_width += background.contentSize.width;
  184:
  185: #ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
  186:     self.displayValueLabel      = [CCLabelTTF labelWithString:@"on" fontName:@"HelveticaNeue-Bold" fon
tSize:30];
  187: #elif __MAC_OS_X_VERSION_MAX_ALLOWED
  188:     self.opacitydisplayValueLabel      = [CCLabelTTF labelWithString:@"Auto" fontName:@"Marker Felt" f
ontSize:30];
  189: #endif
  190:     self.opacitydisplayValueLabel.position  = background.position;
  191:     //[layer addChild:self.opacitydisplayValueLabel];
  192:
  193:     // Create the switch
  194:     self.opacitySwitchControl          = [self makeControlSwitch];
  195:     self.opacitySwitchControl.position      = ccp (layer_width + 10 + self.opacitySwitchControl.conten
tSize.width / 2, 0);
  196:     self.opacitySwitchControl.on            = NO;
  197:     [layer addChild:self.opacitySwitchControl];
  198:
  199:     [self.opacitySwitchControl addTarget:self action:@selector(opacitySwitchValueChanged:) forControlE
vents:CCControlEventValueChanged];
  200:
  201:     // Set the layer size
  202:     layer.contentSize           = CGSizeMake(layer_width, 0);
  203:     layer.anchorPoint           = ccp (0.5f, 0.5f);
  204:
  205:
  206: }
  207:
  208:
  209:
  210:
  211: - (void)opacitySwitchValueChanged:(CCControlSwitch *)sender
  212: {
  213:     if ([sender isOn])
  214:     {
  215:         //displayValueLabel.string    = @"Eraser";
  216:
  217:         gameSettings.depthOpacityMode  = true;
  218:     } else
  219:     {
  220:         //displayValueLabel.string    = @"Eraser";
  221:         gameSettings.depthOpacityMode  = false;
  222:     }
  223: }
  224:
  225:
  226:
  227: #pragma mark - button
```

```
  228:
  229: - (void)buttoninit{
  230:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
  231:     // Add the button
  232:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
  233:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
  234:
  235: #ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
  236:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Touch Me!" fontName:@"HelveticaNeue-Bold"
fontSize:30];
  237: #elif __MAC_OS_X_VERSION_MAX_ALLOWED
  238:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Touch Me!" fontName:@"Marker Felt" fontSiz
e:30];
  239: #endif
  240:     [titleButton setColor:ccc3(159, 168, 176)];
  241:
  242:     CCControlButton *controlButton = [CCControlButton buttonWithLabel:titleButton
  243:                                                       backgroundSprite:backgroundButton];
  244:     [controlButton setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted]
;
  245:     [controlButton setTitleColor:ccWHITE forState:CCControlStateHighlighted];
  246:
  247:     controlButton.anchorPoint = ccp(0.5f, 1);
  248:     controlButton.position = ccp(screenSize.width / 2.0f, screenSize.height / 2.0f);
  249:     [self addChild:controlButton z:1];
  250:
  251:     // Add the black background
  252:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
  253:     [background setContentSize:CGSizeMake(300, 170)];
  254:     [background setPosition:ccp(screenSize.width / 2.0f, screenSize.height / 2.0f)];
  255:     [self addChild:background];
  256:
  257:     // Sets up event handlers
  258:     [controlButton addTarget:self action:@selector(touchDownAction:) forControlEvents:CCControlEventTo
uchDown];
  259: }
  260:
  261:
  262:
  263:
  264:
  265: - (void)touchDownAction:(CCControlButton *)sender
  266: {
  267:     NSLog(@"button value %@", [NSString stringWithFormat:@"Touch Down"]);
  268: }
  269:
  270: - (void)initEraserButton{
  271:
  272:
  273:     // Add the button
  274:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
  275:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
  276:
  277: #ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
  278:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Touch Me!" fontName:@"HelveticaNeue-Bold"
fontSize:30];
  279: #elif __MAC_OS_X_VERSION_MAX_ALLOWED
  280:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Eraser" fontName:@"Marker Felt" fontSize:3
0];
  281: #endif
  282:     [titleButton setColor:ccc3(159, 168, 176)];
  283:
  284:     CCControlButton *controlButton = [CCControlButton buttonWithLabel:titleButton
  285:                                                       backgroundSprite:backgroundButton];
  286:     [controlButton setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted]
;
  287:     [controlButton setTitleColor:ccWHITE forState:CCControlStateHighlighted];
  288:
  289:     controlButton.anchorPoint = ccp(0.5f, 1);
  290:     controlButton.position = ccp(100, 100);
  291:     [self addChild:controlButton z:1];
  292:
  293:     // Add the black background
  294:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
  295:     background.anchorPoint =  ccp(0, 0);
  296:     [background setContentSize:CGSizeMake(100, 75)];
  297:     [background setPosition: ccp(50, 50)];
  298:     [self addChild:background];
```

```
  299:
  300:     // Sets up event handlers
  301:     [controlButton addTarget:self action:@selector(eraserAction:) forControlEvents:CCControlEventTouch
Down];
  302: }
  303:
  304: - (void)eraserAction:(CCControlButton *)sender
  305: {
  306:     [self.delegate changeColorControl:ccWHITE];
  307:
  308: }
  309: - (void)initResetButton{
  310:
  311:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
  312:     // Add the button
  313:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
  314:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
  315:
  316: #ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
  317:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Touch Me!" fontName:@"HelveticaNeue-Bold"
fontSize:30];
  318: #elif __MAC_OS_X_VERSION_MAX_ALLOWED
  319:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Reset" fontName:@"Marker Felt" fontSize:30
];
  320: #endif
  321:     [titleButton setColor:ccc3(159, 168, 176)];
  322:
  323:     CCControlButton *controlButton = [CCControlButton buttonWithLabel:titleButton
  324:                                                       backgroundSprite:backgroundButton];
  325:     [controlButton setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted]
;
  326:     [controlButton setTitleColor:ccWHITE forState:CCControlStateHighlighted];
  327:
  328:     controlButton.anchorPoint = ccp(0.5f, 1);
  329:     controlButton.position = ccp(100, screenSize.height -100);
  330:     [self addChild:controlButton z:1];
  331:
  332:     // Add the black background
  333:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
  334:     [background setContentSize:CGSizeMake(100, 75)];
  335:     [background setPosition:ccp(100, screenSize.height -125)];
  336:     [self addChild:background];
  337:
  338:     // Sets up event handlers
  339:     [controlButton addTarget:self action:@selector(resetAction:) forControlEvents:CCControlEventTouchD
own];
  340: }
  341:
  342:
  343: - (void)resetAction:(CCControlButton*)sender{
  344:     [self.delegate clearDrawing];
  345: }
  346:
  347:
  348: - (CCControlButton *)standardButtonWithTitle:(NSString *)title
  349: {
  350:     /** Creates and return a button with a default background and title color. */
  351:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
  352:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
  353:
  354: #ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
  355:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:title fontName:@"HelveticaNeue-Bold" fontSiz
e:30];
  356: #elif __MAC_OS_X_VERSION_MAX_ALLOWED
  357:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:title fontName:@"Marker Felt" fontSize:30];
  358: #endif
  359:     [titleButton setColor:ccc3(159, 168, 176)];
  360:
  361:     CCControlButton *button = [CCControlButton buttonWithLabel:titleButton backgroundSprite:background
Button];
  362:     [button setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted];
  363:     [button setTitleColor:ccWHITE forState:CCControlStateHighlighted];
  364:
  365:     return button;
  366: }
  367:
  368:
  369: #pragma mark - ColorPicker
```

```
370:
371: - (void)colorpickerinit{
372:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
373:     CCNode *layer                       = [CCNode node];
374:     layer.position                      = ccp (screenSize.width -300 , screenSize.height / 2);
375:     [self addChild:layer z:1];
376:
377:     double layer_width = 0;
378:
379:     // Add the black background for the text
380:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
381:     [background setContentSize:CGSizeMake(150, 50)];
382:     [background setPosition:ccp(layer_width + background.contentSize.width / 2.0f, 0)];
383:     //[layer addChild:background];
384:
385:     layer_width += background.contentSize.width;
386:
387: #ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
388:     colorLabel = [CCLabelTTF labelWithString:@"#color" fontName:@"HelveticaNeue-Bold" fontSize:30];
389: #elif __MAC_OS_X_VERSION_MAX_ALLOWED
390:     colorLabel = [CCLabelTTF labelWithString:@"#color" fontName:@"Marker Felt" fontSize:30];
391: #endif
392:     colorLabel.position = background.position;
393:     //[layer addChild:colorLabel];
394:
395:     // Create the colour picker
396:     CCControlColourPicker *colourPicker = [CCControlColourPicker colourPickerWithHueFile:@"hueBackgrou
nd.png"
397:                                                                                 tintBackgroundFile:@"tintBackgro
und.png"
398:                                                                                    tintOverlayFile:@"tintOverlay
.png"
399:                                                                                         pickerFile:@"picker.png"
400:                                                                                          arrowFile:@"arrow.png"]
;
401:     colourPicker.color                  = ccc3(37, 46, 252);
402:     colourPicker.position               = ccp (layer_width + colourPicker.contentSize.width / 2, 0);
403:     colourPicker.arrowDirection         = CCControlColourPickerArrowDirectionLeft;
404:
405:     // Add it to the layer
406:     [layer addChild:colourPicker];
407:
408: #if NS_BLOCKS_AVAILABLE
409:     // Add block for value changed event
410:     [colourPicker setBlock:^(id sender, CCControlEvent event)
411:      {
412:          [self colourValueChanged:sender];
413:      }
414:           forControlEvents:CCControlEventValueChanged];
415: #else
416:     // Add the target-action pair
417:     [colourPicker addTarget:self action:@selector(colourValueChanged:) forControlEvents:CCControlEvent
ValueChanged];
418: #endif
419:
420:     layer_width += colourPicker.contentSize.width;
421:
422:     // Set the layer size
423:     layer.contentSize                   = CGSizeMake(layer_width, 0);
424:     layer.anchorPoint                   = ccp (0.5f, 0.5f);
425:
426:     // Update the color text
427:     [self colourValueChanged:colourPicker];
428: }
429:
430: - (void)colourValueChanged:(CCControlColourPicker *)sender
431: {
432:     colorLabel.string   = [NSString stringWithFormat:@"#%02X%02X%02X",sender.color.r, sender.color.g,
sender.color.b];
433:
434:     [self.delegate changeColorControl:sender.color];
435: }
436:
437:
438: #pragma mark - Window Controls
439:
440: - (void)expandPanel{
441:
442:
443: }
```

```
  444:
  445: - (void)collapsePanel{
  446:
  447: }
  448:
  449: - (void)initEraserSwitch{
  450:
  451:
  452:
  453:     CCNode *layer               = [CCNode node];
  454:     //layer.position             = ccp (screenSize.width / 2, screenSize.height / 2);
  455:     layer.position = ccp(100, 100);
  456:     [self addChild:layer z:1];
  457:
  458:     double layer_width = 0;
  459:
  460:     // Add the black background for the text
  461:     CCScale9Sprite *background  = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
  462:     background.contentSize      = CGSizeMake(80, 50);
  463:     background.position         = ccp(layer_width + background.contentSize.width / 2.0f, 0);
  464:     [layer addChild:background];
  465:
  466:     layer_width += background.contentSize.width;
  467:
  468: #ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
  469:     self.displayValueLabel      = [CCLabelTTF labelWithString:@"on" fontName:@"HelveticaNeue-Bold" fon
tSize:30];
  470: #elif __MAC_OS_X_VERSION_MAX_ALLOWED
  471:     self.displayValueLabel      = [CCLabelTTF labelWithString:@"Eraser" fontName:@"Marker Felt" fontSi
ze:30];
  472: #endif
  473:     displayValueLabel.position  = background.position;
  474:     [layer addChild:displayValueLabel];
  475:
  476:     // Create the switch
  477:     self.switchControl          = [self makeControlSwitch];
  478:     switchControl.position      = ccp (layer_width + 10 + switchControl.contentSize.width / 2, 0);
  479:     switchControl.on            = NO;
  480:     [layer addChild:switchControl];
  481:
  482:     [switchControl addTarget:self action:@selector(switchValueChanged:) forControlEvents:CCControlEven
tValueChanged];
  483:
  484:     // Set the layer size
  485:     layer.contentSize           = CGSizeMake(layer_width, 0);
  486:     layer.anchorPoint           = ccp (0.5f, 0.5f);
  487:
  488:
  489: }
  490:
  491:
  492: - (CCControlSwitch *)makeControlSwitch
  493: {
  494:     return [CCControlSwitch switchWithMaskSprite:[CCSprite spriteWithFile:@"switch-mask.png"]
  495:                                        onSprite:[CCSprite spriteWithFile:@"switch-on.png"]
  496:                                       offSprite:[CCSprite spriteWithFile:@"switch-off.png"]
  497:                                     thumbSprite:[CCSprite spriteWithFile:@"switch-thumb.png"]
  498:                                         onLabel:[CCLabelTTF labelWithString:@"On" fontName:@"Arial-Bo
ldMT" fontSize:16]
  499:                                        offLabel:[CCLabelTTF labelWithString:@"Off" fontName:@"Arial-B
oldMT" fontSize:16]];
  500: }
  501:
  502:
  503: - (void)switchValueChanged:(CCControlSwitch *)sender
  504: {
  505:     if ([sender isOn])
  506:     {
  507:         displayValueLabel.string    = @"Eraser";
  508:
  509:         [self.delegate eraserMode:true];
  510:     } else
  511:     {
  512:         displayValueLabel.string    = @"Eraser";
  513:         [self.delegate eraserMode:false];
  514:     }
  515: }
  516:
  517:
  518: #pragma mark- Brush Selection Delegate
```

```
519:
520: - (void)initBrushSelectionButton{
521:
522:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
523:     // Add the button
524:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
525:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
526:
527: #ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
528:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Touch Me!" fontName:@"HelveticaNeue-Bold"
fontSize:30];
529: #elif __MAC_OS_X_VERSION_MAX_ALLOWED
530:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:@"Brushes" fontName:@"Marker Felt" fontSize:
30];
531: #endif
532:     [titleButton setColor:ccc3(159, 168, 176)];
533:
534:     CCControlButton *controlButton = [CCControlButton buttonWithLabel:titleButton
535:                                                       backgroundSprite:backgroundButton];
536:     [controlButton setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted]
;
537:     [controlButton setTitleColor:ccWHITE forState:CCControlStateHighlighted];
538:
539:     controlButton.anchorPoint = ccp(0.5f, 1);
540:     controlButton.position = ccp(screenSize.width -100, screenSize.height -100);
541:     [self addChild:controlButton z:1];
542:
543:     // Add the black background
544:     CCScale9Sprite *background = [CCScale9Sprite spriteWithFile:@"buttonBackground.png"];
545:     [background setContentSize:CGSizeMake(100, 75)];
546:     [background setPosition:ccp(screenSize.width -100, screenSize.height -125)];
547:     [self addChild:background];
548:
549:     // Sets up event handlers
550:     [controlButton addTarget:self action:@selector(brushButtonAction:) forControlEvents:CCControlEvent
TouchDown];
551: }
552:
553:
554: - (void)brushButtonAction:(CCControlButton*)sender{
555:     if (self.brushSelection.layerHidden) {
556:         [self showBrushSelectionPanel];
557:     }else{
558:         [self hideBrushSelectionPanel];
559:     }
560:
561: }
562:
563:
564: - (void)showBrushSelectionPanel{
565:
566:     self.brushSelection.layerHidden = false;
567:     //[sprite runAction: [CCMoveBy actionWithDuration:2 position:ccp(50,10)]];
568:     [self.brushSelection runAction:[CCMoveTo actionWithDuration:2 position:ccp(0,0)]];
569: }
570:
571: - (void)hideBrushSelectionPanel{
572:     // Get window size
573:     CGSize screenSize = [[CCDirector sharedDirector] winSize];
574:
575:     self.brushSelection.layerHidden = true;
576:     [self.brushSelection runAction:[CCMoveTo actionWithDuration:2 position:ccp(-screenSize.width,0)]];
577:
578: }
579:
580: - (void)hidePanel{
581:
582:     [self hideBrushSelectionPanel];
583: }
584:
585:
586: - (void)brushSelected:(NSString *)brushname{
587:     [self.delegate changeBrushControl:brushname];
588: }
589:
590:
591: @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:
   83:  Sets up the screens to be track intersecting vectors from pointables.
   84:  */
   85:
   86: - (void)onConnect:(NSNotification *)notification;
   87: {
   88:     NSLog(@"Leap: Connected");
   89:
   90:
   91:     //NSArray* screens = controller.calibratedScreens;
   92:     NSArray* screens = controller.locatedScreens;
   93:
   94:
   95:     if ([screens count] > 0){
   96:         leapScreen = [screens objectAtIndex:0];
   97:         NSLog(@"Screens: %0.0ld", (unsigned long)[screens count]);
   98:
   99:
  100:     }else{
  101:         NSLog(@"No Screens");
  102:     }
  103:
  104:     NSLog(@"running");
  105:
  106: }
  107: /**
  108:  LeapMotion SDK Delegate Callback
  109:  Disconnect
  110:  Notifies the application that the LeapMotion has been disconnected and hold or release any processes
in regard to the LeapMotion
  111:  */
  112:
  113: - (void)onDisconnect:(NSNotification *)notification{
  114:     NSLog(@"Leap: Disconnected");
  115: }
  116:
  117: /**
  118:  LeapMotion SDK Delegate Callback
  119:  Exits
  120:  Releases memory and sets object instances to nil (null)
  121:  */
  122: - (void)onExit:(NSNotification *)notification{
  123:     NSLog(@"Leap: Exited");
  124: }
  125: /**
  126:  LeapMotion SDK Delegate Callback
  127:  OnFrame Event notifies the application that an incoming frame has been processed and the data can be
used to control the application
  128:  */
  129:
  130: - (void)onFrame:(NSNotification *)notification{
  131:     ///NSLog(@"OnFrame");
  132:     LeapController *aController = (LeapController *)[notification object];
  133:     // Get the most recent frame and report some basic information
  134:     LeapFrame *frame = [aController frame:0];
  135:
  136:     //Try and find the same one as last time.
  137:     if ([[frame pointables] count] != 0) {
  138:         NSArray* leapPointables = [frame pointables];
  139:
  140:         LeapPointable* tool;
  141:         if (lastTag != -1){
  142:             for (LeapPointable* pointable in leapPointables){
  143:
  144:                 if (lastTag == pointable.id){
  145:
  146:                     tool = pointable;
  147:                     lastTag = pointable.id;
  148:                     break;
  149:                 }
  150:
  151:             }
  152:
  153:             //Find a new point able
  154:             if (tool == nil){
  155:
  156:                 tool = [self pointableClosestToScreen:leapPointables];
```

```
  157:                 lastTag = tool.id;
  158:
  159:             }
  160:
  161:
  162:         }else{
  163:             //Find a new pointable
  164:             tool = [self pointableClosestToScreen:leapPointables];
  165:             lastTag = tool.id;
  166:
  167:         }
  168:
  169:         //Get the screen
  170:         LeapVector* normalized = [leapScreen intersect:tool normalize:YES clampRatio:2.0];
  171:
  172:         if ([leapScreen isValid]){
  173:             double x = normalized.x * [leapScreen widthPixels];
  174:             double y = normalized.y * [leapScreen heightPixels];
  175:
  176:             CGPoint pointer = CGPointMake(x, y);
  177:
  178:             //Convert to Local coordinates from Screen Coordinates
  179:             CCDirector* director = [CCDirector sharedDirector];
  180:             NSPoint var = [director.view.window convertScreenToBase:pointer];
  181:
  182:             //Logging
  183:             //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);
  184:             //SimplePoint* simplePoint = [[SimplePoint alloc] initWithPosition:var withZ:tool.tipPosit
ion.z];
  185:             //[[NSNotificationCenter defaultCenter] postNotificationName:@"CoordHUDUpdate" object:simp
lePoint];
  186:
  187:             if (gameSettings.depthOpacityMode){
  188:
  189:                 float opacity = [self opacityPercentage:tool.tipPosition.z];
  190:
  191:                 //Update the controls
  192:                 [controlsLayer updateOpacitySlider:opacity];
  193:
  194:             }
  195:
  196:             if (inputMode == kDepthMode){
  197:
  198:                 if (tool.tipPosition.z > 0){
  199:                     painting = FALSE;
  200:
  201:
  202:                 }else{
  203:                     painting = TRUE;
  204:
  205:                 }
  206:             }
  207:
  208:             //Update the HUD View
  209:             [self.hudLayer toolMoved:var toolID:[NSString stringWithFormat:@"%0.0d",tool.id]];
  210:             if (painting){
  211:                 [self movedToolTexture:var tool:tool];
  212:             }else{
  213:                // NSLog(@"Not Painting");
  214:             }
  215:
  216:         }else{
  217:             NSLog(@"Leap Screen is invalid");
  218:         }
  219:
  220:     }else{
  221:
  222:
  223:         NSLog(@"No frame");
  224:         //Remove the marker from the HUD view
  225:         if (currentPointable != nil) {
  226:
  227:             [self endLineDrawingTexture:currentPoint tool:currentPointable];
  228:             [self.hudLayer endTrackingTool];
  229:         }
  230:
  231:         lastTag = -1;
  232:
  233:         framesSinceLastFound ++;
```

```
234:         if (framesSinceLastFound > kMaxFrames){
235:
236:             framesSinceLastFound = 0;
237:         }
238:
239:     }
240: }
241:
242: #pragma mark - TextureScene
243:
244: /** Moves the tool on the screen when painting */
245: - (void)movedToolTexture:(CGPoint)point tool:(LeapPointable*)pointable{
246:
247:     if (currentPointable != nil){
248:
249:         [self moveLineDrawingTexture:point tool:pointable];
250:         currentPointable = pointable;
251:     }else{
252:         [self beginLineDrawingTexture:point tool:pointable];
253:         currentPointable = pointable;
254:     }
255: }
256:
257: /** Begin drawing to the canvas
258:  @param point is the current coordinate the LeapPointable is interescting with the screen
259:  @param pointable is a reference to the pointable currently drawing
260:  */
261:
262: - (void)beginLineDrawingTexture:(CGPoint)point tool:(LeapPointable*)pointable{
263:
264:     [self.textureScene beginDraw:point withZ:pointable.tipPosition.z];
265:     currentPoint = point;
266: }
267: /** Update drawing with a moved image on the canvas
268:   @param point is the current coordinate the LeapPointable is interescting with the screen
269:   @param pointable is a reference to the pointable currently drawing
270:  */
271: - (void)moveLineDrawingTexture:(CGPoint)point tool:(LeapPointable*)pointable{
272:
273:     [self.textureScene updateDraw:point withZ:pointable.tipPosition.z];
274:     currentPoint = point;
275:
276: }
277:
278: /** End the drawing
279:   @param point is the current coordinate the LeapPointable is interescting with the screen
280:   @param pointable is a reference to the pointable currently drawing
281:  */
282: - (void)endLineDrawingTexture:(CGPoint)point tool:(LeapPointable*)pointable{
283:     [self.textureScene endDraw:point];
284:     currentPointable = nil;
285: }
286:
287: #pragma mark - Keyboard Events
288:
289: /** Change Input Mode */
290: - (void)changeMode:(InputMode)mode{
291:     //NSLog(@"Changemode");
292:     inputMode = mode;
293:     gameSettings.inputMode = mode;
294: }
295:
296:
297: /** Change Paiting state
298:  @param paintState changes the painting sate
299:  */
300: - (void)painting:(BOOL)paintingState{
301:     painting = paintingState;
302:     gameSettings.painting = paintingState;
303: }
304:
305:
306: #pragma mark - ControlsDelegate
307:
308: /** Change the color of the brush
309:  Updates the HUD Layer and the Texture Layer
310:   @param color is the color to be changed
311:  */
312:
313: - (void)changeColorControl:(ccColor3B)color{
```

```
314:
315:     [self.hudLayer changeColor:color];
316:     [self.textureScene changeColor:color];
317:
318: }
319: /** Change the thickness of the brush
320:  Updates the HUD Layer and the Texture Layer
321:  @param value is the thinkness(width) value
322:  */
323: - (void)changeThicknessControl:(float)value{
324:
325:     [self.hudLayer changeScale:value];
326:     [self.textureScene changeScale:value];
327: }
328: /** Change the brush type
329:  Updates the HUD Layer and the Texture Layer
330:  @param brushname is the name of the brush to be changed
331:  */
332: - (void)changeBrushControl:(NSString *)brushname{
333:
334:     [self.hudLayer changeBrush:brushname];
335:     [self.textureScene changeBrush:brushname];
336: }
337: /** Change the opacity of the brush
338:  Updates the HUD Layer and the Texture Layer
339:  @param value is the opacity value
340:  */
341: - (void)changeOpacityControl:(float)value{
342:     [self.textureScene changeOpacity:value];
343: }
344:
345: /** Clears the drawing */
346: - (void)clearDrawing{
347:
348:     [self.textureScene clearDrawing];
349:
350:     //**Turns off eraser mode if it is on
351:     if (gameSettings.eraserMode){
352:         gameSettings.eraserMode = false;
353:
354:         //Update texture mode and update Controls layer
355:     }
356: }
357: /** Update the eraser mode
358:  Updates the HUD Layer and the Texture Layer
359:  @param mode is the current state of the eraser
360:  */
361: - (void)eraserMode:(bool)mode{
362:
363:     [self.hudLayer erasingMode:mode];
364:     [self.textureScene erasingMode:mode];
365:
366: }
367:
368: /** Return the Opacity value based on Z position */
369: - (float)opacityPercentage:(float)value{
370:     //NSLog(@"value %0.0f", value);
371:     if (value < kOpMinRange){
372:         return kOpMax;
373:     }else if(value > kOpMaxRange){
374:         return kOpMin;
375:     }else {
376:
377:         float percentage = [self findPecentageDifference:kOpMaxRange withMin:kOpMinRange withValue:val
ue];
378:         //NSLog(@"percentage %0.0f", percentage);
379:
380:         percentage = 100 - percentage;
381:
382:         return percentage;
383:
384:     }
385:
386: }
387:
388:
389: /** Find the percentage between two numbers */
390: - (float)findPecentageDifference:(float)max withMin:(float)min withValue:(float)value{
391:     return (value - min)/(max - min)*100;
392: }
```

```
393:
394:
395: /**
396:  Using all the pointables, gets the closest one to the screen
397:  @param pointables is an array of pointables currently observered by the LeapMotion
398:  @return pointable that is closest by the screen
399:  */
400: - (LeapPointable*)pointableClosestToScreen:(NSArray*)pointables{
401:
402:     LeapPointable* closestPointable;
403:     for (LeapPointable*pointable in pointables){
404:
405:         //Check for the first iteration that the closest is not equal to nil
406:         if (closestPointable != nil){
407:
408:             if (closestPointable.tipPosition.z > pointable.tipPosition.z){
409:                 closestPointable = pointable;
410:             }
411:
412:         }else{
413:             closestPointable = pointable;
414:         }
415:
416:     }
417:
418:     return closestPointable;
419: }
420:
421: /**
422:  Find the closest LeapPointable to the current last vector
423:  @param leapVector is the position of the last pointbale
424:  @param pointables is an array of pointables currently observered by the LeapMotion
425:  @return LeapPointable closest to a leapVector
426:  */
427: - (LeapPointable*)pointableClosestToVector:(LeapVector*)leapVector withPointables:(NSArray*)pointables
{
428:
429:
430:     LeapPointable* closestPointable;
431:     //Check to make sure there is atleast one object in the array
432:
433:     //if the array is empty, throw an exception
434:     if ([pointables count] == 0){
435:         NSLog(@"Cannot pass item 0 array");
436:         return nil;
437:     }
438:     //If there is only one object in the array, return it
439:     else if ([pointables count] == 1){
440:         return [pointables objectAtIndex:0];
441:     }else{
442:
443:         //Get the distance for the first point
444:         float minDistance = 0;
445:         closestPointable = [pointables objectAtIndex:0];
446:         minDistance = [leapVector distanceTo:closestPointable.tipPosition];
447:
448:
449:         for (int i = 1; i < [pointables count]; i++){
450:
451:             LeapPointable* point = [pointables objectAtIndex:i];
452:             float distance = [leapVector distanceTo:point.tipPosition];
453:             if ( distance < minDistance){
454:                 minDistance = distance;
455:                 closestPointable = point;
456:             }
457:         }
458:     }
459:
460:     return closestPointable;
461: }
462:
463:
464: @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:  */
23: @interface GameScene : CCScene  {
24:
25: }
26: /**
27:  Scene initializes each object and assigns interlinking pointers and delegates to each class
28:  @return scene for CCDirector to begin running
29:  */
30: +(CCScene *) scene;
31:
32: @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:
22:     //setup delegates
23:     hudLayer.delegate = scene;
24:     controlsLayer.delegate = scene;
25:
26:
27:     SketchRenderTextureScene* textureScene = [SketchRenderTextureScene node];
28:     [scene addChild:backgroundLayer z:0];
29:     [scene addChild:controlsLayer z:3];
30:         // add layer as a child to scene
31:         [scene addChild:hudLayer z:5];
32:
33:
34:     [scene addChild:textureScene z:2];
35:
36:
37:     scene.hudLayer = hudLayer;
38:     scene.backgroundLayer = backgroundLayer;
39:     scene.controlsLayer = controlsLayer;
40:
41:     scene.textureScene = textureScene;
42:
43:         // return the scene
44:         return scene;
45: }
46: @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:
   12: #define kVelMax 1000
   13: #define kVelMin 0
   14:
   15: #define kOpMinRange -80
   16: #define kOpMaxRange 120
   17:
   18: #define kOpMin 0
   19: #define kOpMax 100
   20:
   21: #define kNormalizedVelMax 15
   22: #define kNormalizedVelMin 0
   23:
   24: #define kMaxFrames 1000
   25:
   26: extern int const BLOCK_SIZE;
   27:
   28:
   29: typedef enum {
   30:     kPressKeyMode,
   31:     kDepthMode,
   32: } InputMode;
   33:
   34: /**
   35:  GameSettings is a globally shared class instance which tracks all the game settings.
   36:
   37:  This class can be accessed by any object in the game.
   38:  */
   39: @interface GameSettings : NSObject{
   40:
   41:
   42: }
   43: @property (nonatomic,readwrite) BOOL depthOpacityMode;      /**< depthOpacityMode controls use of z ax
is control of opacity */
   44: @property (nonatomic,readwrite) BOOL painting;      /**< painting indicates wether or not the applicat
ion is painting at that moment*/
   45: @property (nonatomic,readwrite) BOOL eraserMode;            /**< eraserMode controls erasing on drawin
g canvas */
   46: @property (nonatomic,readwrite) InputMode inputMode;       /**< inputMode controller input mode for l
eapmotion */
   47: /** Singleton
   48:  Intiailizes and Returns a shared instance of the class
   49:  @return sharedInstance of the class.
   50:  */
   51: + (GameSettings *)sharedInstance;
   52:
   53: @end
```

```
//
//  GameSettings.m
//  LeapPuzz
//
//  Created by cj on 4/16/13.
//
//

#import "GameSettings.h"

//Constants
int const  BLOCK_SIZE = 128;

@implementation GameSettings
@synthesize depthOpacityMode;
@synthesize eraserMode;
@synthesize inputMode;
@synthesize painting;

/** Singleton SharedInstance
 Intiailizes and Returns a shared instance of the class
 */
+ (GameSettings *)sharedInstance
{
    static GameSettings *sharedInstance;

    @synchronized(self)
    {
        if (!sharedInstance)
            sharedInstance = [[GameSettings alloc] init];
        return sharedInstance;
    }
}

/**
 Initialize the class and sets the default values
 */
- (id)init
{
    if (self = [super init]) {
        // Init Defaults
        self.depthOpacityMode = false;
        self.painting = false;
    }
    return self;
}
@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:
   41:     InputMode inputMode;        /**< inputMode is the current mode of input*/
   42:
   43:     ccColor3B lastColor;        /**< lastColor is the lastColor to be selected */
   44:     ccColor3B previousColor;    /**< previousColor is the color before the lastcolor to be selected */
   45:     NSString* lastBrush;        /**< lastBrush is last brush to be selected */
   46:     float lastScale;            /**< lastScale is last scale to be selected */
   47:
   48:
   49:
   50:     CCSprite* paintingIndicator; /**< paintingIndicator shows the state at which the object is current
ly paintg */
   51:     BOOL eraseMode;             /**< eraseMode determines weather the pointable is painting or erasing
 */
   52:
   53:
   54:     GameSettings* gameSettings; /**< gameSettings singleton to global seetings*/
   55:
   56:
   57: }
   58:
   59: @property (nonatomic, weak) id <HUDDelegate> delegate; /**< colorLabel displays name of color in hash
value */
   60: @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 */
   61:
   62:
   63: /**
   64:  ToolMoved updates the last known tracked position of the tool.
   65:  @param point is the coordinate location on the screen in which pointable interesects
   66:  @param toolid is LeapSDK provided tool id of the tool moving
   67:  */
   68: - (void)toolMoved:(CGPoint)point toolID:(NSString*)toolid;
   69: /**
   70:  StartTrackingTool begins the process of tracking a tool starting with a new path
   71:  @param point is the coordinate location on the screen in which pointable interesects
   72:  @param toolid is LeapSDK provided tool id of the tool moving
   73:  */
   74: - (void)startTrackingTool:(CGPoint)point toolID:(NSString*)toolid;
   75: /**
   76:  MoveTrackingTool updates the position and path of a tool.
```

```
 77:  @param point is the coordinate location on the screen in which pointable interesects
 78:  @param toolid is LeapSDK provided tool id of the tool moving
 79:  */
 80: - (void)moveTrackingTool:(CGPoint)point toolID:(NSString*)toolid;
 81: /**
 82:  EndTracking tool singles the end of the tool being tracked.
 83:  The tool may be lost or no longer drawing
 84:  */
 85: - (void)endTrackingTool;
 86:
 87:
 88: - (void)changeColor:(ccColor3B)color;
 89: - (void)changeBrush:(NSString*)brushname;
 90: - (void)changeScale:(float)size;
 91: - (void)erasingMode:(BOOL)mode;
 92:
 93: @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:
   55: #pragma CCControlButtonTest_HelloVariableSize Private Methods
   56:
   57: - (CCControlButton *)standardButtonWithTitle:(NSString *)title
   58: {
   59:     /** Creates and return a button with a default background and title color. */
   60:     CCScale9Sprite *backgroundButton = [CCScale9Sprite spriteWithFile:@"button.png"];
   61:     CCScale9Sprite *backgroundHighlightedButton = [CCScale9Sprite spriteWithFile:@"buttonHighlighted.p
ng"];
   62:     CCLabelTTF *titleButton = [CCLabelTTF labelWithString:title fontName:@"Marker Felt" fontSize:30];
   63:     [titleButton setColor:ccc3(159, 168, 176)];
   64:     CCControlButton *button = [CCControlButton buttonWithLabel:titleButton backgroundSprite:background
Button];
   65:     [button setBackgroundSprite:backgroundHighlightedButton forState:CCControlStateHighlighted];
   66:     [button setTitleColor:ccWHITE forState:CCControlStateHighlighted];
   67:
   68:     return button;
   69: }
   70:
   71:
   72: @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
```

```
 1: //
 2: //  LPPoint.h
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 5/8/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import <Foundation/Foundation.h>
10:
11:
12:
13: /**
14:  LPLinePoint is a plotted point for drawing onto the canvas
15:  */
16: @interface LPLinePoint : NSObject{
17:
18:
19: }
20:
21: @property (nonatomic, readwrite) float x; /**< x coordinate */
22: @property (nonatomic, readwrite) float y; /**< y coordinate */
23: @property (nonatomic, readwrite) float width; /**< width of the point */
24:
25: /**
26:  * Init constructor with existing point to create with no width
27:  * @param p an point (x,y)
28:  * @return object instance
29:  */
30: - (id)initWithPosition:(CGPoint)p;
31: /**
32:  * Init constructor with x and y values with no width
33:  * @param xVal coordinate value
34:  * @param yVal coordinate value
35:  * @return object instance
36:  */
37: - (id)initWithX:(float)xVal withY:(float)yVal;
38: /**
39:  * Init constructor with existing point with width
40:  * @param p a point (x,y)
41:  * @param wVal width of the point
42:  * @return object instance
43:  */
44: - (id)initWithPosition:(CGPoint)p withWidth:(float)wVal;
45: /**
46:  * Init constructor with x and y values with width
47:  * @param xVal coordinate value
48:  * @param yVal coordinate value
49:  * @param wVal width of the point
50:  * @return object instance
51:  */
52: - (id)initWithX:(float)xVal withY:(float)yVal withWidth:(float)wVal;
53:
54: /**
55:  * Returns point based on x and y
56:  * @return CGPoint
57:  */
58: - (CGPoint)point;
59: @end
```

```
 1: //
 2: //  LPPoint.m
 3: //  LeapPaint
 4: //
 5: //  Created by cj on 5/8/13.
 6: //  Copyright (c) 2013 cjdesch. All rights reserved.
 7: //
 8:
 9: #import "LPLinePoint.h"
10:
11: @implementation LPLinePoint
12: @synthesize x;
13: @synthesize y;
14: @synthesize width;
15:
16: /** init 2d point with CGPoint */
17: - (id)initWithPosition:(CGPoint)p{
18:     if (self = [super init]) {
19:
20:         self.x = p.x;
21:         self.y = p.y;
22:         self.width = 0.0f;
23:
24:
25:     }
26:     return self;
27: }
28:
29: /** Init Point with 2 separate values */
30: - (id)initWithX:(float)xVal withY:(float)yVal{
31:
32:     if (self = [super init]) {
33:
34:         self.x = xVal;
35:         self.y = yVal;
36:         self.width = 0.0f;
37:
38:
39:     }
40:     return self;
41: }
42:
43:
44: /** Init  point with CGPoint and width Value */
45: - (id)initWithPosition:(CGPoint)p withWidth:(float)wVal{
46:     if (self = [super init]) {
47:
48:         self.x = p.x;
49:         self.y = p.y;
50:         self.width = wVal;
51:
52:
53:     }
54:     return self;
55: }
56:
57:
58: /** Init Point with x and y values with width*/
59: - (id)initWithX:(float)xVal withY:(float)yVal withWidth:(float)wVal{
60:
61:     if (self = [super init]) {
62:
63:         self.x = xVal;
64:         self.y = yVal;
65:         self.width = wVal;
66:
67:
68:     }
69:     return self;
70: }
71:
72: /** Return the CGPoint type from the object */
73: - (CGPoint)point{
74:     return CGPointMake(self.x, self.y);
75: }
76:
77:
78:
79: @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:
   16: @interface LPTool : CCSprite
   17:
   18: @property (nonatomic, strong) NSString* toolID; /**< toolID is the ID number assigned by the LeapMotio
n SDK */
   19: @property (nonatomic, readwrite) BOOL updated; /**< updated is if the sprite has been updated in that
frame.*/
   20:
   21:
   22:
   23:
   24: @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:
13:
14: @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:
13: /**
14:  * 2D or 3D space coordinate for temporarily maniulapting points
15:  *
16:  */
17:
18: @interface SimplePoint : NSObject {
19:
20: }
21:
22: @property (nonatomic, readwrite) float x; /**< x coordinate */
23: @property (nonatomic, readwrite) float y; /**< y coordinate */
24: @property (nonatomic, readwrite) float z; /**< z coordinate */
25: @property (nonatomic, readwrite) BOOL is3d; /**< is3d is 2d or 3d point type */
26:
27:
28: /**
29:  * Init constructor with existing point to create a 2d Point
30:  * @param p an point (x,y)
31:  * @return object instance
32:  */
33: - (id)initWithPosition:(CGPoint)p;
34: /**
35:  * Init constructor with x and y values to create a 2d point
36:  * @param xVal coordinate value
37:  * @param yVal coordinate value
38:  * @return object instance
39:  */
40: - (id)initWithX:(float)xVal withY:(float)yVal;
41: /**
42:  * Init constructor with existing point to create a 3d Point
43:  * @param p a point (x,y)
44:  * @param zVal coordinateValue
45:  * @return object instance
46:  */
47: - (id)initWithPosition:(CGPoint)p withZ:(float)zVal;
48: /**
49:  * Init constructor with x, y and z values to create 3D point
50:  * @param xVal coordinate value
51:  * @param yVal coordinate value
52:  * @param zval coordinate value
53:  * @return object instance
54:  */
55: - (id)initWithX:(float)xVal withY:(float)yVal withZ:(float)zVal;
56:
57: /**
58:  * Returns point based on x and y
59:  * @return CGPoint
60:  */
61: - (CGPoint)point;
62: @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: @synthesize x,y,z;
14: @synthesize is3d;
15:
16:
17: /** init 2d point with CGPoint */
18: - (id)initWithPosition:(CGPoint)p{
19:     if (self = [super init]) {
20:
21:         self.x = p.x;
22:         self.y = p.y;
23:         self.z = 0.0f;
24:         self.is3d = false;
25:
26:     }
27:     return self;
28: }
29:
30: /** Init 2d Point with 2 separate values */
31: - (id)initWithX:(float)xVal withY:(float)yVal{
32:
33:     if (self = [super init]) {
34:
35:         self.x = xVal;
36:         self.y = yVal;
37:         self.z = 0.0f;
38:         self.is3d = false;
39:
40:     }
41:     return self;
42: }
43:
44:
45: /** Init 3d point with CGPoint and z Value */
46: - (id)initWithPosition:(CGPoint)p withZ:(float)zVal{
47:     if (self = [super init]) {
48:
49:         self.x = p.x;
50:         self.y = p.y;
51:         self.z = zVal;
52:         self.is3d = true;
53:
54:     }
55:     return self;
56: }
57:
58:
59: /** Init 3d Point with 3 separate values */
60: - (id)initWithX:(float)xVal withY:(float)yVal withZ:(float)zVal{
61:
62:     if (self = [super init]) {
63:
64:         self.x = xVal;
65:         self.y = yVal;
66:         self.z = zVal;
67:         self.is3d = true;
68:
69:     }
70:     return self;
71: }
72:
73: /** Return the CGPoint type from the object */
74: - (CGPoint)point{
75:     return CGPointMake(self.x, self.y);
76: }
77:
78:
79: @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:
16: @interface SimplePointObject : NSObject {
17:
18:
19: }
20:
21: @property (nonatomic, readwrite) CGPoint point; /**< point is the X and Y coordinates */
22:
23: - (id)initWithPosition:(CGPoint)p;
24: - (id)initWithX:(float)x withY:(float)y;
25:
26: @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:
18:     ccColor3B lastColor;
19:     ccColor3B previousColor;
20:     NSString* lastBrush;
21:     float lastScale;
22:     bool eraseMode;
23:
24:
25:
26: }
27:
28: @property (nonatomic,readwrite) float opacity;
29:
30: - (void)beginDraw:(CGPoint)point withZ:(float)z;
31: - (void)updateDraw:(CGPoint)point withZ:(float)z;
32: - (void)endDraw:(CGPoint)point;
33:
34: - (void)changeColor:(ccColor3B)color;
35: - (void)changeBrush:(NSString*)brushname;
36: - (void)changeScale:(float)size;
37: - (void)changeOpacity:(float)o;
38: - (void)erasingMode:(BOOL)mode;
39:
40: - (void)clearDrawing;
41: @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
```