

Chapter 13

UIGestureRecognizer and UINavigationController

- UITapGestureRecognizer Subclasses
- Detecting Taps
- Multiple Gesture Recognizers
- UINavigationController
- UILongPressGestureRecognizer
- UIPanGestureRecognizer

UIGestureRecognizer

- To detect a specific pattern of touches that make a gesture such as a pinch or a swipe we will use an instances of UIGestureRecognizer.
- A UITapGestureRecognizer intercepts touches that are on their way to being handled by a view.
- When it recognizes a particular gesture, it sends a message to an object that we choose.
- There are several types of gesture recognizers built into the SDK.
- We will use three of them to allow TouchTracker users to select, move, and delete lines.

We will cover tap, long press, and pan gestures

The concrete subclasses of `UIGestureRecognizer` are the following:

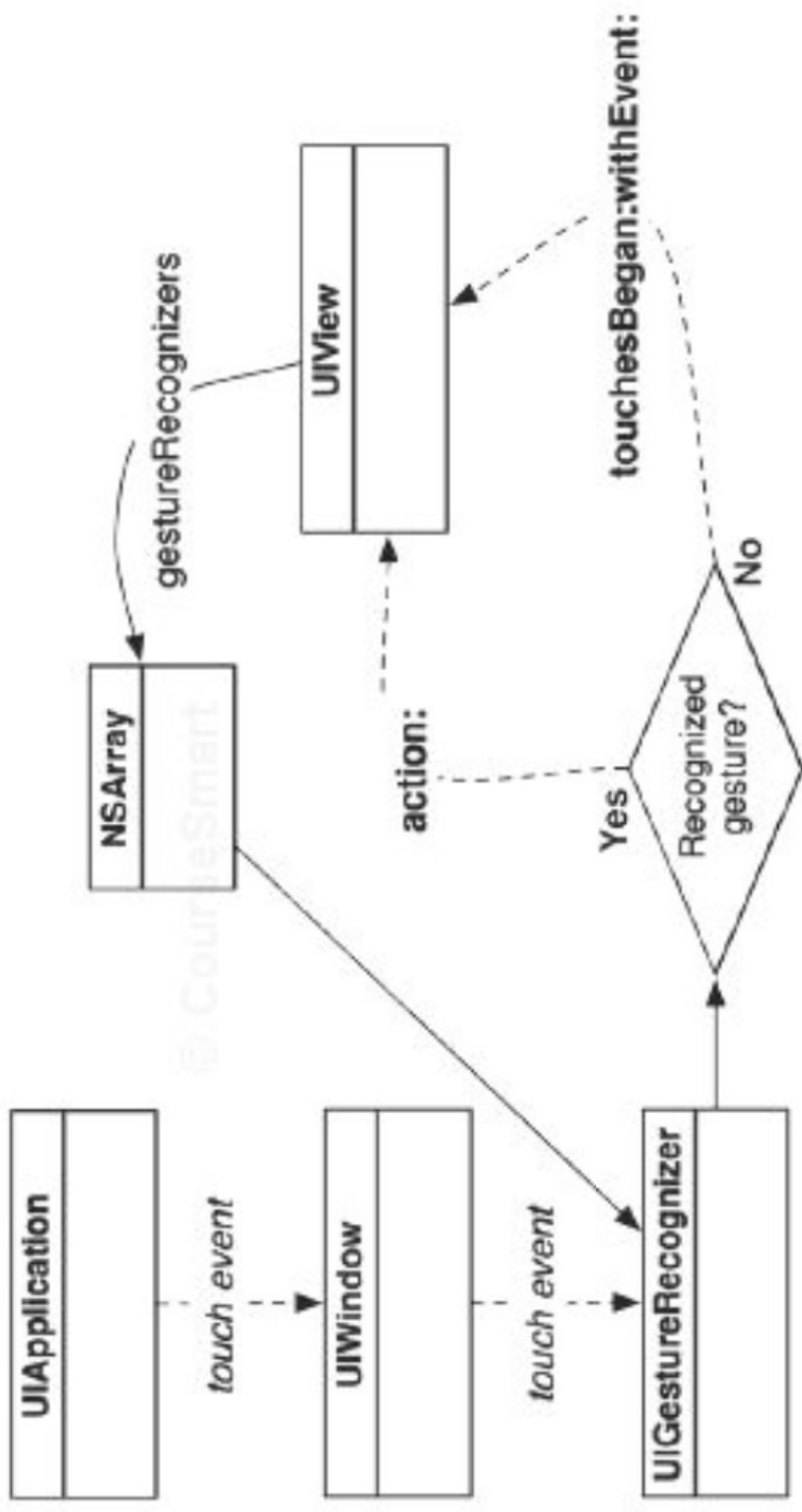
```
UITapGestureRecognizer  
UIPinchGestureRecognizer  
UIRotationGestureRecognizer  
UISwipeGestureRecognizer  
UIPanGestureRecognizer  
UIScreenEdgePanGestureRecognizer  
UILongPressGestureRecognizer
```

UIGestureRecognizer Subclasses

- UITapGestureRecognizer is an abstract class
- You do not instantiate it itself.
- There are seven subclasses of UITapGestureRecognizer
- Each one is responsible for recognizing a particular gesture.
- To use an instance of a UITapGestureRecognizer subclass
 - We give it a target- action pair
 - Attach it to a view.
- Whenever the gesture recognizer recognizes its gesture on the view, it will send the action message to its target.
- All UITapGestureRecognizer action messages have the same form:
 - - (void) action:(UITapGestureRecognizer *) gestureRecognizer;

Recognizing Gestures

- When recognizing a gesture, the gesture recognizer intercepts the touches destined for the view.
- A view with gesture recognizers may not receive the typical UIResponder messages like `touchesBegan:withEvent:`.



Detecting Taps with UITapGestureRecognizer

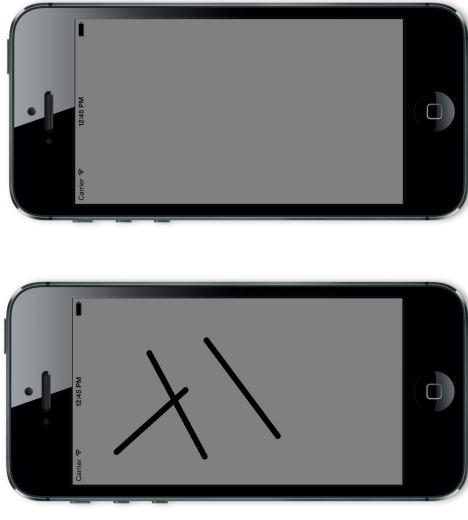
- Modify ~~INIDrawView~~ so that it can intercept gestures and act upon them.

```
21 - (instancetype) initWithFrame:(CGRect) r
22 {
23     self = [super initWithFrame: r];
24     if (self) {
25         self.linesInProgress = [[NSMutableDictionary alloc] init]; //Multiple lines being drawn
26         self.finishedLines = [[NSMutableArray alloc] init];
27         self.backgroundColor = [UIColor grayColor];
28         self.multipleTouchEnabled = YES;
29
30         UITapGestureRecognizer *doubleTapRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector(doubleTap:)];
31         doubleTapRecognizer.numberOfTapsRequired = 2;
32         [self addGestureRecognizer: doubleTapRecognizer];
33     }
34     return self;
35 }
36
```

```
- (void) doubleTap:(UITapGestureRecognizer *) gr
{
    NSLog(@" Recognized Double Tap");
    [self.linesInProgress removeAllObjects];
    [self.finishedLines removeAllObjects];
    [self setNeedsDisplay];
}
```

Warning
because the
method doubleTap:
is not implemented
yet.

The argument to the action method for a
gesture recognizer is the instance of
UITapGestureRecognizer that sent the message.



```
2014-06-24 12:45:07.904 TouchTracker[1408:60b] touchesEnded:withEvent:
2014-06-24 12:45:21.483 TouchTracker[1408:60b] touchesBegan:withEvent:
2014-06-24 12:45:21.932 TouchTracker[1408:60b] Recognized Double Tap
2014-06-24 12:45:21.933 TouchTracker[1408:60b] touchesCancelled:withEvent:
```

Delaying touchesBegan:....

- Gesture recognizers work by inspecting touch events to determine if their particular gesture occurred.
- Before a gesture is recognized, all UIResponder messages will be delivered to a view as normal.
- Since a tap gesture recognizer is recognized when a touch begins and ends within a small area in a small amount of time, the UITapGestureRecognizer cannot claim the touch is a tap just yet and touchesBegan: withEvent: is sent to the view.
- When the tap is finally recognized, the gesture recognizer claims the touch involved in the tap for itself and no more UIResponder messages will be sent to the view for that particular touch.
- In order to communicate this touch take-over to the view, touchesCancelled: withEvent: is sent to the view and the NSSet of touches contains that UITouch instance.
- To prevent this red dot from appearing temporarily, you can tell a UITapGestureRecognizer to delay the sending of touchesBegan: withEvent: to its view if it is still possible for the gesture to be recognized.

```
UITapGestureRecognizer *doubleTapRecognizer = [[UITapGestureRecognizer  
doubleTapRecognizer.numberOfTapsRequired = 2;  
doubleTapRecognizer.delaysTouchesBegan = YES;  
[self addGestureRecognizer: doubleTapRecognizer];  
}
```


Multiple Gesture Recognizers

- We need to add another gesture recognizer that allows the user to select a line.
- We will use this to enable us to delete the selected line.
- We will install another UITapGestureRecognizer on the INIDrawView that only requires one tap.
- In situations where you have multiple gesture recognizers, it is not uncommon to have a gesture recognizer fire when you really want another gesture recognizer to handle the work.
- In these cases, you set up dependencies between recognizers that say, “Just wait a moment before you fire, because this gesture might be mine!”
- In initWithFrame:, make it so the UITapGestureRecognizer must wait for the doubleTapRecognizer to fail before it can assume that a single tap is not just the first of a double tap.

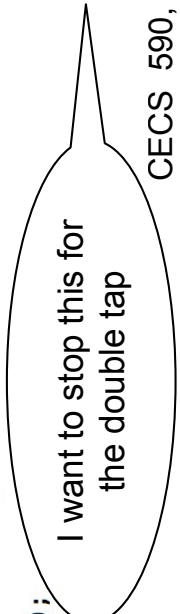
```
5 UITapGestureRecognizer *tapRecognizer = [[UITapGestureRecognizer alloc]
6 tapRecognizer.delaysTouchesBegan = YES;
7 [tapRecognizer requireGestureRecognizerToFail: doubleTapRecognizer];
8 [self addGestureRecognizer: tapRecognizer];
9 }

UITapGestureRecognizer *tapRecognizer = [[UITapGestureRecognizer alloc] initWithTarget: self action:@selector(tap:)]];
tapRecognizer.delaysTouchesBegan = YES;
[self addGestureRecognizer: tapRecognizer];

return self;

- (void) tap:(UITapGestureRecognizer *) gr
{
    NSLog(@" Recognized tap");
}

2014-06-24 13:33:07.534 TouchTracker[2027:60b] Recognized tap
2014-06-24 13:33:08.625 TouchTracker[2027:60b] Recognized tap
2014-06-24 13:33:10.122 TouchTracker[2027:60b] Recognized tap
2014-06-24 13:33:14.089 TouchTracker[2027:60b] touchesBegan:withEvent:
2014-06-24 13:33:14.089 TouchTracker[2027:60b] touchesMoved:withEvent:
...
2014-06-24 13:33:16.151 TouchTracker[2027:60b] touchesEnded:withEvent:
2014-06-24 13:33:17.433 TouchTracker[2027:60b] Recognized tap
2014-06-24 13:33:17.816 TouchTracker[2027:60b] Recognized Double Tap
```



I want to stop this for the double tap

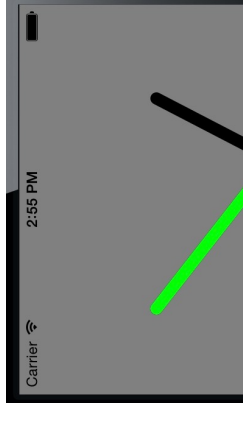
Selecting And Deleting Lines

- First, add a weak property to hold onto a selected line to the class extension in INIDrawView.m
- Since the property is weak, the finishedLines array will hold the strong reference to the line and selectedLine will be set to nil if the line is removed from finishedLines by clearing the screen.
- In drawRect:, add some code to the bottom of the method to draw the selected line in green.
- Implement lineAtPoint: in INIDrawView.m to get an INILine close to the given point.

```

57 - (INILine *) lineAtPoint:(CGPoint) p
58 {
59     // Find a line close to p
60     for (INILine * l in self.finishedLines) {
61         CGPoint start = l.begin;
62         CGPoint end = l.end;
63         // Check a few points on the line
64         for (float t = 0.0; t <= 1.0; t += 0.05) {
65             float x = start.x + t * (end.x - start.x);
66             float y = start.y + t * (end.y - start.y);
67             // If the tapped point is within 20 points, let's return this line
68             if (hypot(x - p.x, y - p.y) < 20.0) {
69                 return l;
70             }
71         }
72     }
73     // If nothing is close enough to the tapped point, then we did not select a line
74     return nil;
75 }
76
77

```



UIMenuController

- Now we will make it so that when the user selects a line, a menu appears right where the user tapped that offers the option to delete that line.
- There is a built-in class for providing this sort of menu called UIMenuController
- A menu controller has a list of UIMenuItem objects and is presented in an existing view.
- Each item has a title and an action which is the message that it sends the first responder of the window.
- The first responder must have all actions corresponding to the messages sent by the menu items implemented otherwise the menu will not appear.
- Not every view accepts to become first responder. Thus, we must check with it first

Creating the UIMenuController

```

44 - (void) tap:(UITapGestureRecognizer *) gr
45 {
46     NSLog(@" Recognized tap");
47     CGPoint point = [gr locationInView: self];
48     self.selectedLine = [self lineAtPoint: point];
49
50     if (self.selectedLine) {
51         // Make ourselves the target of menu item action messages
52         [self becomeFirstResponder];
53         // Grab the menu controller
54         UIMenuController *menu = [UIMenuController sharedMenuController];
55         // Create a new "Delete" UIMenuItem
56         UIMenuItem *deleteItem = [[UIMenuItem alloc] initWithTitle:@"Delete" action:@selector(deleteLine:)];
57         menu.menuItems = @[deleteItem];
58         // Tell the menu where it should come from and show it
59         [menu setTargetRect: CGRectMake(point.x, point.y, 2, 2) inView: self];
60         [menu setMenuVisible: YES animated: YES];
61     } else {
62         // Hide the menu if no line is selected
63         [[UIMenuController sharedMenuController] setMenuVisible: NO animated: YES];
64     }
65
66     [self setNeedsDisplay];
67 }

```

```

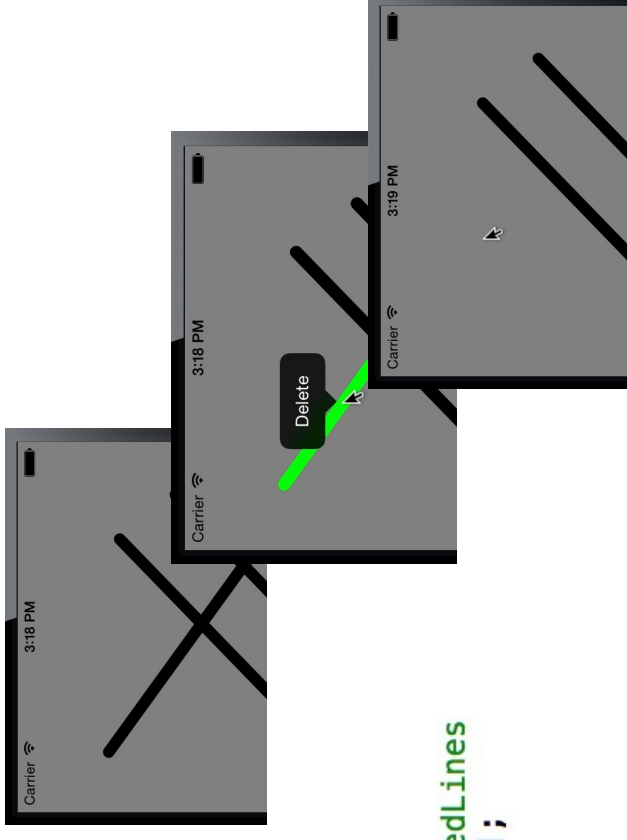
35 - (BOOL) canBecomeFirstResponder
36 {
37     return YES;
38 }
39
40

```

```

- (void) deleteLine:(id) sender
{
    // Remove the selected line from the list of finishedLines
    [self.finishedLines removeObject: self.selectedLine];
    // Redraw everything
    [self setNeedsDisplay];
}

```



UILongPressGestureRecognizer

- In instantiate and add a UILongPressGestureRecognizer in initWithFrame: of INIDrawView.
- By default, a touch must be held 0.5 seconds to become a long press, we can change the minimumPressDuration of the gesture recognizer if you like.
- The long press gesture occurs over time and is defined by three separate events:

The user touches a view, the long press recognizer notices a possible long press but must wait to see whether the touch is held long enough to become a long press gesture (UIGestureRecognizerStatePossible)

Once the user holds the touch long enough, the long press is recognized and the gesture has begun (UIGestureRecognizerStateBegan)

When the user removes the finger, the gesture has ended (UIGestureRecognizerStateEnded).

When a gesture recognizer transitions to any state other than the possible state, it sends its action message to its target.

Implementing UILongPressGestureRecognizer

```

1 // ...
2
3 UITapGestureRecognizer *tapRecognizer = [[UITapGestureRecognizer alloc] initWithTarget: self action:@ selector(tap:)];
4 tapRecognizer.delaysTouchesBegan = YES;
5 [tapRecognizer requireGestureRecognizerToFail: doubleTapRecognizer];
6 [self addGestureRecognizer: tapRecognizer];
7
8 UILongPressGestureRecognizer *pressRecognizer = [[UILongPressGestureRecognizer alloc] initWithTarget: self action:@selector(longPress:)];
9 [self addGestureRecognizer: pressRecognizer];
10
11 return self;
12 }

```

```

13 - (void) longPress:(UIGestureRecognizer *) gr
14 {
15     NSLog(@"long press with state %ld", gr.state);
16     if (gr.state == UIGestureRecognizerStateBegan) {
17         CGPoint point = [gr locationInView: self];
18         self.selectedLine = [self lineAtPoint: point];

```

```

19         if (self.selectedLine) {
20             [self.linesInProgress removeAllObjects];
21         }

```

```

22     } else if (gr.state == UIGestureRecognizerStateEnded)
23     {
24         self.selectedLine = nil;
25         [self setNeedsDisplay];
26     }

```

```

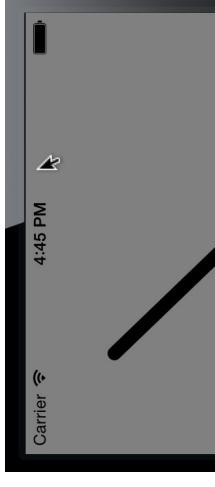
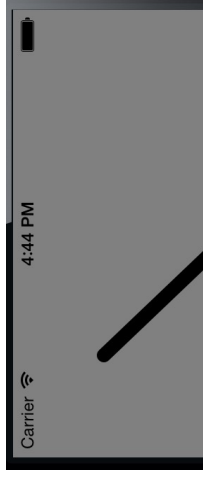
typedef NS_ENUM(NSUInteger, UIGestureRecognizerState) {
    UIGestureRecognizerStatePossible, // the recognize
    UIGestureRecognizerStateBegan, // the recognize
    UIGestureRecognizerStateChanged, // the recognize
    UIGestureRecognizerStateEnded, // the recognize
    UIGestureRecognizerStateCancelled, // the recognize
    UIGestureRecognizerStateFailed, // the recognize
    // Discrete Gestures – gesture recognizers that recognize
    UIGestureRecognizerStateRecognized = UIGestureRecognizerStateFailed
};

```

```

2014-06-24 16:45:05.630 TouchTracker[2987:60b] long press with state 1
2014-06-24 16:45:15.930 TouchTracker[2987:60b] long press with state 3

```



Moving The Selected Line With UIPanGestureRecognizer

- Once a line is selected during a long press, we want to be able to move that line around the screen by dragging it with a finger
- We need a gesture recognizer for a finger moving around the screen.
- This gesture is called panning, and its gesture recognizer subclass is `UIPanGestureRecognizer`.
- A gesture recognizer does not share the touches it intercepts
- Once a gesture recognizer recognized its gesture, it “eats” that touch, and no other recognizer gets a chance to handle it.
- In our case, this is bad:
 - the entire pan gesture you want to recognize happens within a long press gesture.
 - We need the long press recognizer and the pan recognizer to be able to recognize their gestures simultaneously.

UIPanGestureRecognizer and Simultaneous Recognizers

Recognition
neous Gestur...
requirements

UIGestureRecognizerDelegate

Inherits from: None

Conforms to: NSObject

Framework: UIKit in iOS 3.2 and later. [More related items...](#)

Next

Overview

Delegates of a gesture recognizer—that is, an instance of `UIGestureRecognizer`—adopt the `UIGestureRecognizerDelegate` protocol to fine-tune an app's gesture-recognition behavior. The delegates receive messages from a gesture recognizer, and their responses to these messages enable them to affect the operation of the gesture recognizer or to specify a relationship between it and another gesture recognizer, such as allowing simultaneous recognition or setting up a failure requirement.

Tasks

Regulating Gesture Recognition

- `gestureRecognizerShouldBegin:`
- `gestureRecognizer:shouldReceiveTouch:`

Controlling Simultaneous Gesture Recognition

- `gestureRecognizer:shouldRecognizeSimultaneouslyWithGestureRecognizer:`

Setting Up Failure Requirements

- `gestureRecognizer:shouldRequireFailureOfGestureRecognizer:`
- `gestureRecognizer:shouldBeRequiredToFailByGestureRecognizer:`

@interface INIDrawView() <UIGestureRecognizerDelegate>

@property (nonatomic, strong) UIPanGestureRecognizer *moveRecognizer;

@property (nonatomic, strong) NSMutableDictionary *linesInProgress;

@property (nonatomic, strong) NSMutableArray *finishedLines;

@property (nonatomic, weak) INILine * selectedLine;

@end

Set INIDrawView to conform to UIPGestureRecognizerDelegate

This will allow us to implement

- (BOOL) gestureRecognizer:
shouldRecognizeSimultaneouslyWithGestureRecognizer:

We are declaring the
moveRecognizer as a property and
not a local variable in initWithFrame

6/24/14

CECS 590, I. Imam

14

Implementing UIPanGestureRecognizer

```

[self addGestureRecognizer: pressRecognizer];

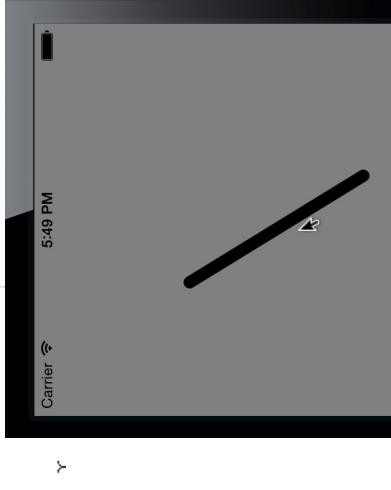
self.moveRecognizer = [[UIPanGestureRecognizer alloc] initWithTarget: self action:@selector(moveLine:)];
self.moveRecognizer.delegate = self;
self.moveRecognizer.cancelsTouchesInView = NO;
[self addGestureRecognizer: self.moveRecognizer];

```

```

- (BOOL) gestureRecognizer:(UIGestureRecognizer *) gestureRecognizer shouldRecognizeSimultaneouslyWithGestureRecognizer:(UIGestureRecognizer *) other
{
    if (gestureRecognizer == self.moveRecognizer) {
        return YES;
    }
    return NO;
}

```



```

- (void) moveLine:(UIPanGestureRecognizer *) gr
{
    // If we have not selected a line, we do not do anything here
    if (!self.selectedLine) {
        return;
    }
    // When the pan recognizer changes its position...
    if (gr.state == UIGestureRecognizerStateChanged) {
        // How far has the pan moved?
        CGPoint translation = [gr translationInView: self];
        // Add the translation to the current beginning and end points of the line
        CGPoint begin = self.selectedLine.begin;
        CGPoint end = self.selectedLine.end;
        begin.x += translation.x;
        begin.y += translation.y;
        end.x += translation.x;
        end.y += translation.y;
        // Set the new beginning and end points of the line
        self.selectedLine.begin = begin;
        self.selectedLine.end = end;
        // Redraw the screen
        [self setNeedsDisplay];
        [gr setTranslation: CGPointZero inView: self];
    }
}

```

