Chapter 7
Delegation and Text Input

- Protocols and Delegations
- Using the Debugger
- Build Phases
- Objective-C supplement (NSDictionary)

Text Fields

- We have already seen one way to display text on user interfaces using a UILabel.
- Now we will take a look at another way to display text using a UITextField.
- An instance of UITextField allows the user to modify the text, much like a username or password field on a website.

```
- ( void) loadView
{
    // Create a view
    CGRect frame = [UIScreen mainScreen].bounds;
    INIHypnosisView *backgroundView = [[INIHypnosisView alloc] initWithFrame: frame];

    CGRect textFieldRect = CGRectMake( 40, 70, 240, 30);
    UITextField *textField = [[UITextField alloc] initWithFrame: textFieldRect];
    // Setting the border style on the text field will allow us to see it more easily
    textField.borderStyle = UITextBorderStyleRoundedRect;

    [backgroundView addSubview: textField];

    // Set it as "the" view of this view controller
    self.view = backgroundView;
}
```

UIResponder

- UIResponder is an abstract class in the UIKit framework.
- It is the superclass of three classes that we have already encountered:

  UIView,

  UIViewController, and

  UIApplication
- UIResponder defines methods for handling events:

  touch events,

  motion events (like a shake), and

  remote control events (like pausing or playing).
- Subclasses override these methods to customize how they respond to events.

firstResponders

- The UIWindow has a pointer called firstResponder which indicates who should respond to the other types of events.

  When you select a text field the window moves its firstResponder pointer to that text field.

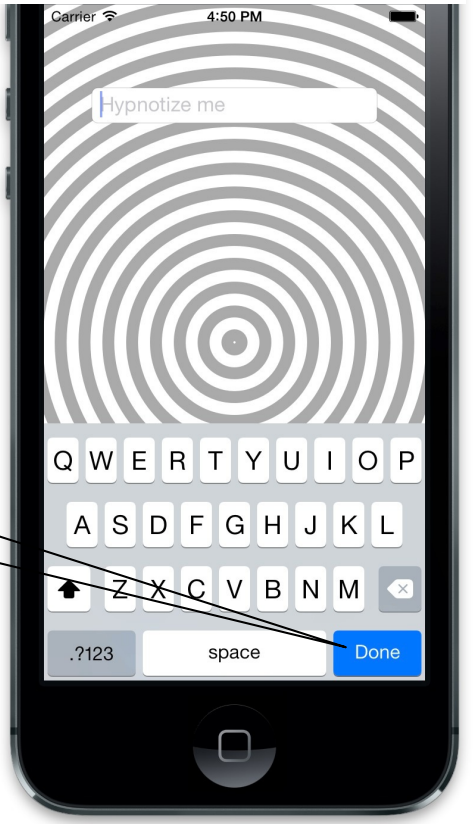
  Motion and remote control events are sent to the first responder.
- When a text field or a text view becomes firstResponder, it shows its keyboard.
- When it loses first responder status, it hides its keyboard.
- If you want a view to become first responder, you send it the message becomeFirstResponder and the keyboard appears.
- When you want to hide the keyboard, you send it the message resignFirstResponder.
- Most views refuse to become first responder; they do not want to steal focus from the currently selected text field or text view. An instance of UISlider, for example, handles touch events but will never accept first responder status.

More Text Field Traits

```
34
35      CGRect textFieldRect = CGRectMake( 40, 70, 240, 30);
36      UITextField *textField = [[UITextField alloc] initWithFrame: textFieldRect];
37      // Setting the border style on the text field will allow us to see it more easily
38      textField.borderStyle = UITextBorderStyleRoundedRect;
39      textField.placeholder = @"Hypnotize me";
40      // textField.returnKeyType = UIReturnKeyDone;
41
```


```
34
35      CGRect textFieldRect = CGRectMake( 40, 70, 240, 30);
36      UITextField *textField = [[UITextField alloc] initWithFrame: textFieldRect];
37      // Setting the border style on the text field will allow us to see it more easily
38      textField.borderStyle = UITextBorderStyleRoundedRect;
39      textField.placeholder = @"Hypnotize me";
40      textField.returnKeyType = UIReturnKeyDone;
41
```

Selecting Appropriate Keyboard



Delegation and Delegates

- Recall: Target- Action pattern is one form of callbacks that is used by UIKit: When a button is tapped, it sends its action message to its target.
- Another form of callbacks Apple uses is the delegation pattern.
- In our example we introduce the text field to one of our objects (in this case BNRHypnosisViewController): " This is your delegate, when anything interesting happens in your life, send a message to it." The text field keeps a pointer to its delegate. Many of the messages it sends to its delegates are informative: " OK, I am done editing!".
- Here are few examples:

```
- ( void) textFieldDidEndEditing:( UITextField *) textField;
- ( void) textFieldDidBeginEditing:( UITextField *) textField;
```

- Notice that it always sends itself as the first argument to the delegate method.
- Some of the messages it sends to its delegate are queries: " I am about to end editing and hide the keyboard. OK?" such as:

```
- ( BOOL) textFieldShouldEndEditing:( UITextField *) textField;
```

- The class designated as a delegate will be responsible for implementing all required methods and any optional methods needed by the delegate.

Assigning delegate to our text field

```
37      // Setting the border style on the text field will allow us to see it more easily
38      textField.borderStyle = UITextBorderStyleRoundedRect;
39      textField.placeholder = @"Hypnotize me";
40      textField.returnKeyType = UIReturnKeyDone;
41
42      textField.delegate = self;        Assigning to 'id<UITextFieldDelegate>' from incompatible type 'INIHypnosisViewController *const __strong'
43
44      [backgroundView addSubview: textField];
```

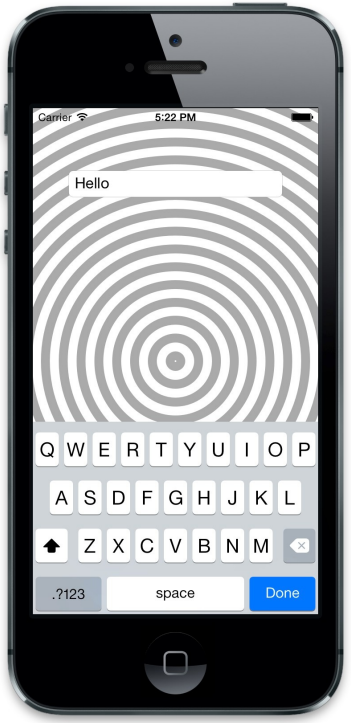
```
}

- (BOOL) textFieldShouldReturn:( UITextField *) textField
{
    NSLog(@"%@", textField. text);
    return YES;
}
```

Will be resolved once we introduce protocols

HypnoNerd

```
2014-06-02 17:20:57.666 HypnoNerd[5470:60b]  INIHypnosisVi
2014-06-02 17:21:07.614 HypnoNerd[5470:60b] Hello
2014-06-02 17:21:10.505 HypnoNerd[5470:60b] Hello
2014-06-02 17:21:11.326 HypnoNerd[5470:60b] Hello
2014-06-02 17:21:12.350 HypnoNerd[5470:60b] Hello
```



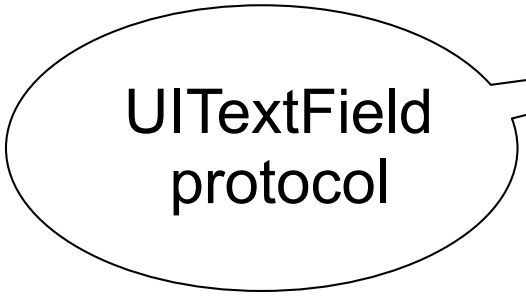
Delegates

In Summary:

- Delegation is an approach to callbacks in which a callback function implemented by the delegate is responsible for responding to an event generated by the delegating class.
- The delegating class assigns the responsibility of implementing certain tasks to other classes, namely it's delegates.
- The delegate class needs and chooses to implement the tasks listed in the delegating class.
- The delegate needs to and has to take the responsibility of implementing the tasks it needs.
- The delegate need to implement the methods that correspond to the events it wants to hear

Protocols

- Think of protocol as an interface definition between two classes.
- For every object that can have a delegate, there is a corresponding protocol that declares the messages that the object can send its delegate.
- The delegate implements methods from the protocol for events it is interested in.
- In Java or C#, you would use the word “ interface” instead of “ protocol”.




```
@protocol UITextFieldDelegate <NSObject>

@optional

- (BOOL)textFieldShouldBeginEditing:(UITextField *)textField;
- (void)textFieldDidBeginEditing:(UITextField *)textField;
- (BOOL)textFieldShouldEndEditing:(UITextField *)textField;
- (void)textFieldDidEndEditing:(UITextField *)textField;
- (BOOL)textField:(UITextField *)textField
        shouldChangeCharactersInRange:(NSRange)range
                    replacementString:(NSString *)string;
- (BOOL)textFieldShouldClear:(UITextField *)textField;
- (BOOL)textFieldShouldReturn:(UITextField *)textField;

@end
```

More on Protocols

- Protocol is not a class; it is simply a list of method declarations.
- You cannot create instances of a protocol, it cannot have instance variables.
- Methods are not implemented anywhere in the protocol. Instead, implementation is left to each class that conforms to the protocol.
- Methods declared in a protocol can be required or optional.
- By default, protocol methods are required.
- If a protocol has optional methods, these are preceded by the directive @ optional.
- Before sending an optional message, the object first asks its delegate if it is okay to send that message by sending the message respondsToSelector:.
- Every object implements respondsToSelector: which checks at runtime whether an object implements a given method.
- If a method in a protocol is required, then the message will be sent without checking first. This means that if the delegate does not implement that method, an unrecognized selector exception will be thrown, and the application will crash.

Conforming to a Protocol

- the class must explicitly state that it conforms to a protocol
- This is done either in the class header file or the class extension: the protocols that a class conforms to are added to a comma-delimited list inside angled brackets in the interface declaration.

```
#import <Foundation/Foundation.h>

@interface INIHypnosisViewController : UIViewController < UITextFieldDelegate>

@end
```

My class conforms to UITextField Delegate protocol

```
9       textField.placeholder = @"Hypnotize me";
0       textField.returnKeyType = UIReturnKeyDone;
1
2       textField.delegate = self;
3
4       [backgroundView addSubview: textField];
```

Warning is gone

Another Delegate Example

```
23
24  - (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
25  {
26      self = [super initWithNibName: nibNameOrNil bundle:nibBundleOrNil ];
27      if (self)
28      {
29          locationManager = [[CLLocationManager alloc] init];
30          [locationManager setDelegate: self];
31          [locationManager setDesiredAccuracy: kCLLocationAccuracyBest];
32          [locationManager startUpdatingLocation];
33      }
34      return self;
35  }
36
```

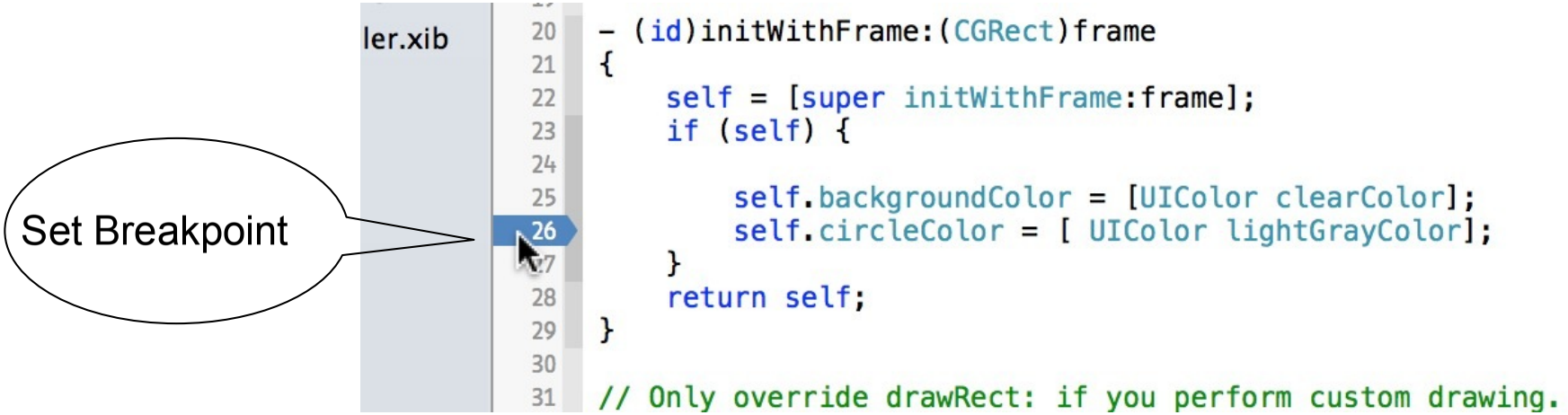
WhereamiViewController is the delegate for locationManager and is responsible for implementing the needed methods.

Motion Effects and Parallax

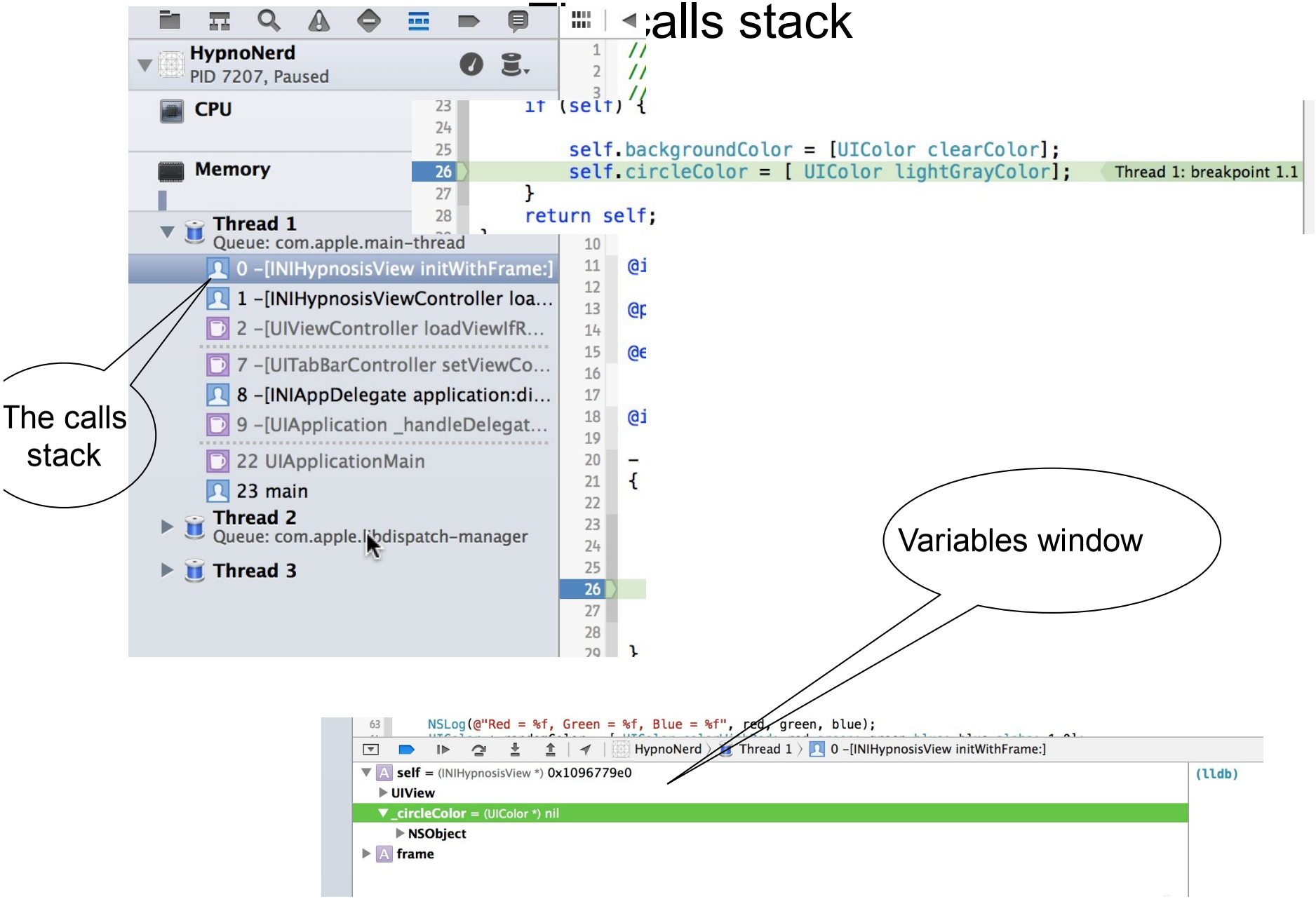
- Parallax is the difference in the apparent position of an object based on the angle along which an object is viewed
- Distant object appear to move slower than nearer objects when observed from a moving vehicle.
- Apple introduced Parallax effect in the way we view icons and objects on the screen based on the tilt angle with which we look at an iOS device.
- To see the effect of adding this effect to our messages displayed on our screen, we need to have the application running on an actual device

Using the Debugger

- When an application is launched from Xcode, the debugger is attached to it.
- The debugger monitors the current state of the application such as the method being currently executing and the values of the variables that are accessible from that method.
- Using the debugger can help you understand what an application is actually doing, which, in turn, helps you find and fix bugs.



calls stack


HypnoNerd
PID 7207, Paused
CPU
Memory
Thread 1
Queue: com.apple.main-thread
0 -[INIHypnosisView initWithFrame:]
1 -[INIHypnosisViewController loa...
2 -[UIViewController loadViewIfR...
7 -[UITabBarController setViewCo...
8 -[INIAppDelegate application:di...
9 -[UIApplication _handleDelegat...
22 UIApplicationMain
23 main
Thread 2
Queue: com.apple.libdispatch-manager
Thread 3
The calls stack
if (self) {
self.backgroundColor = [UIColor clearColor];
self.circleColor = [UIColor lightGrayColor];
Thread 1: breakpoint 1.1
}
return self;
Variables window
NSLog(@"Red = %f, Green = %f, Blue = %f", red, green, blue);
HypnoNerd 〉 Thread 1 〉 0 -[INIHypnosisView initWithFrame:]
self = (INIHypnosisView *) 0x1096779e0
(lldb)
UIView
_circleColor = (UIColor *) nil
NSObject
frame

Stepping through methods