iPhone Application Development Lesson 9

By Dannis Mok

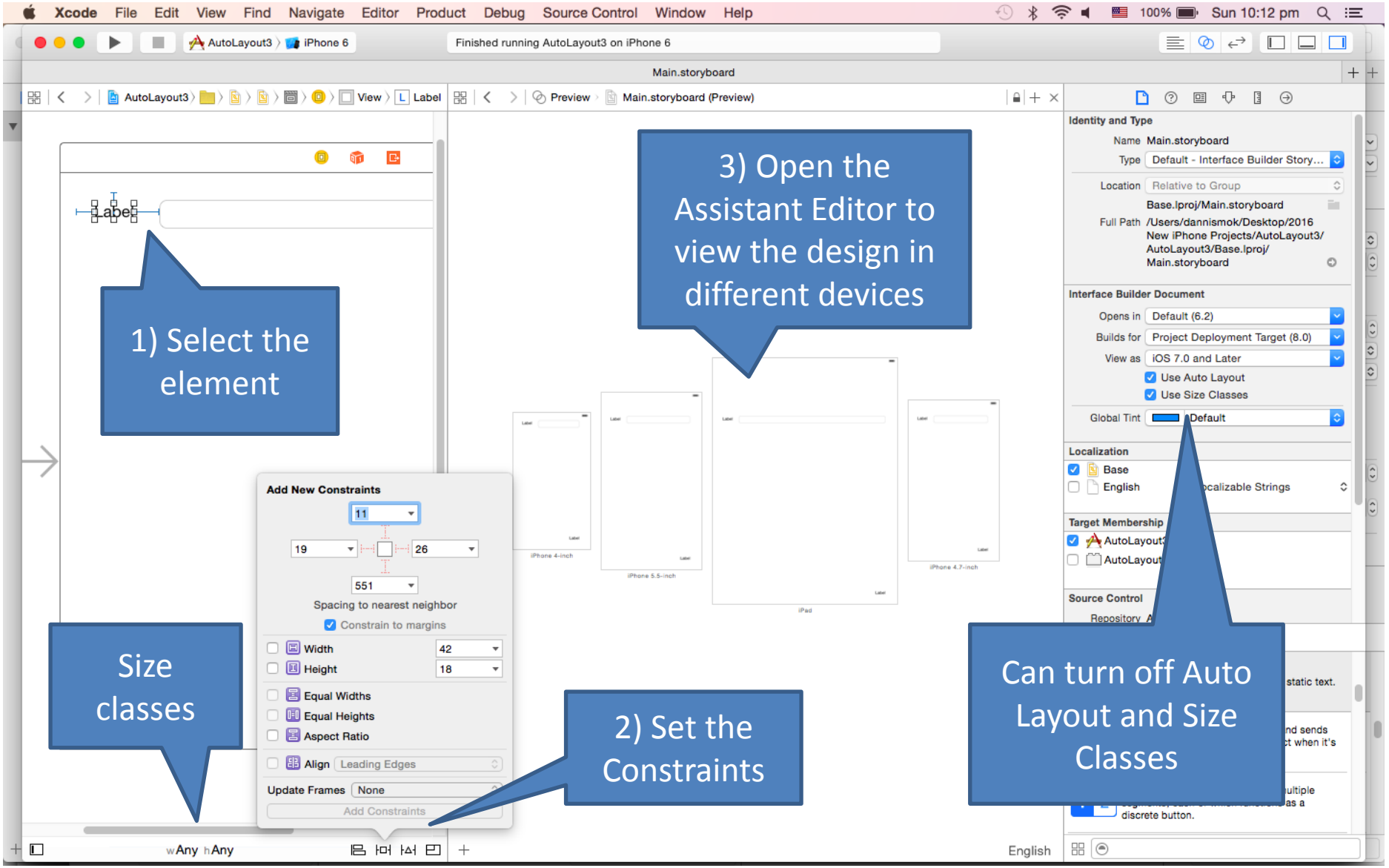
Auto Layout

- Auto Layout helps to add the constraints onto the UI elements and make them to adapt to different devices' sizes and resolutions.

- Size Classes

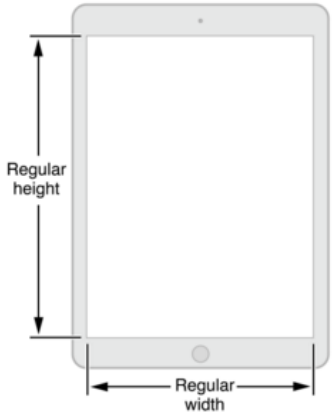
  iOS defines two size classes: regular and compact. The *regular* size class is associated with expansive space and the *compact* size class is associated with constrained space.

Auto Layout

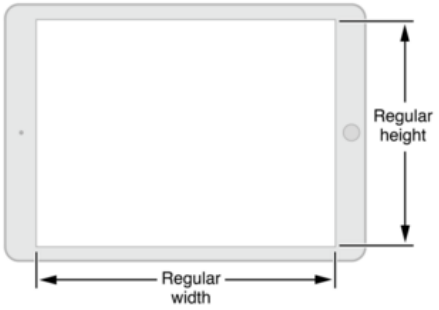

1) Select the element
3) Open the Assistant Editor to view the design in different devices
Size classes
2) Set the Constraints
Can turn off Auto Layout and Size Classes
Add New Constraints
Add Constraints
Identity and Type
Interface Builder Document
Localization
Target Membership
Source Control

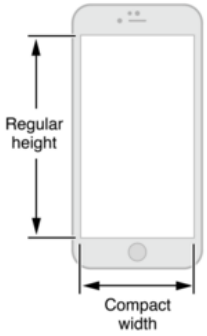
Size Classes

The size classes of iPad in portrait

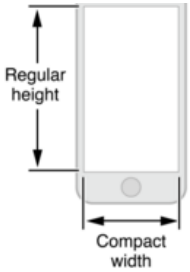


The size classes of iPad in landscape



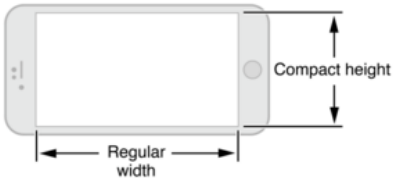


In portrait, iPhone 6 Plus uses the compact horizontal and regular vertical size classes.

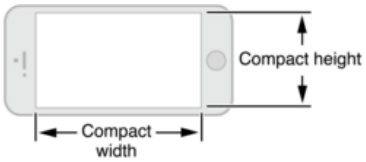


In portrait, iPhone 6, iPhone 5, and iPhone 4s use the compact horizontal and regular vertical size classes.

In landscape, iPhone 6 Plus uses the regular horizontal and compact vertical size classes.



In landscape, these devices use the compact size class in both the horizontal and vertical dimensions.

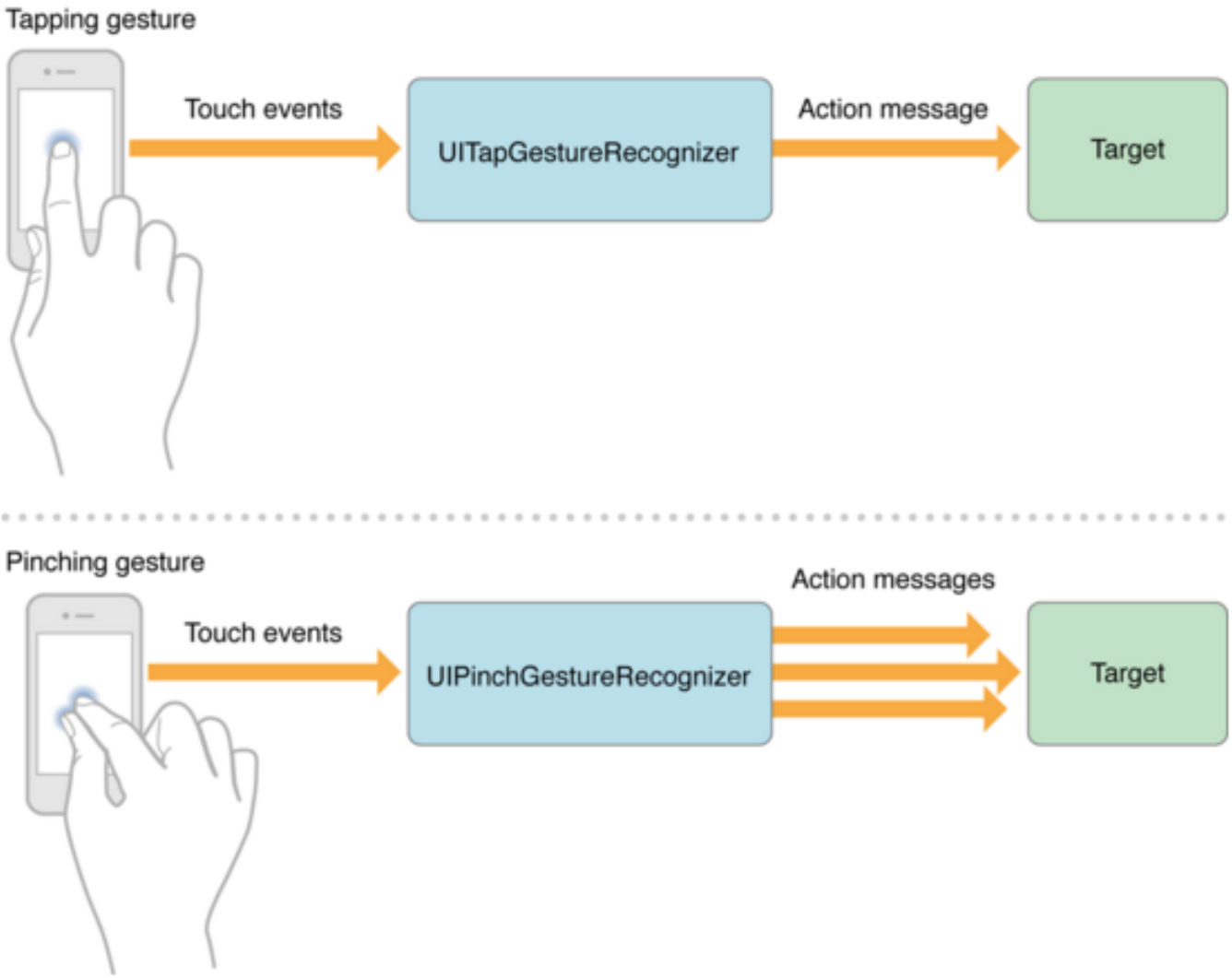


Gesture Processing

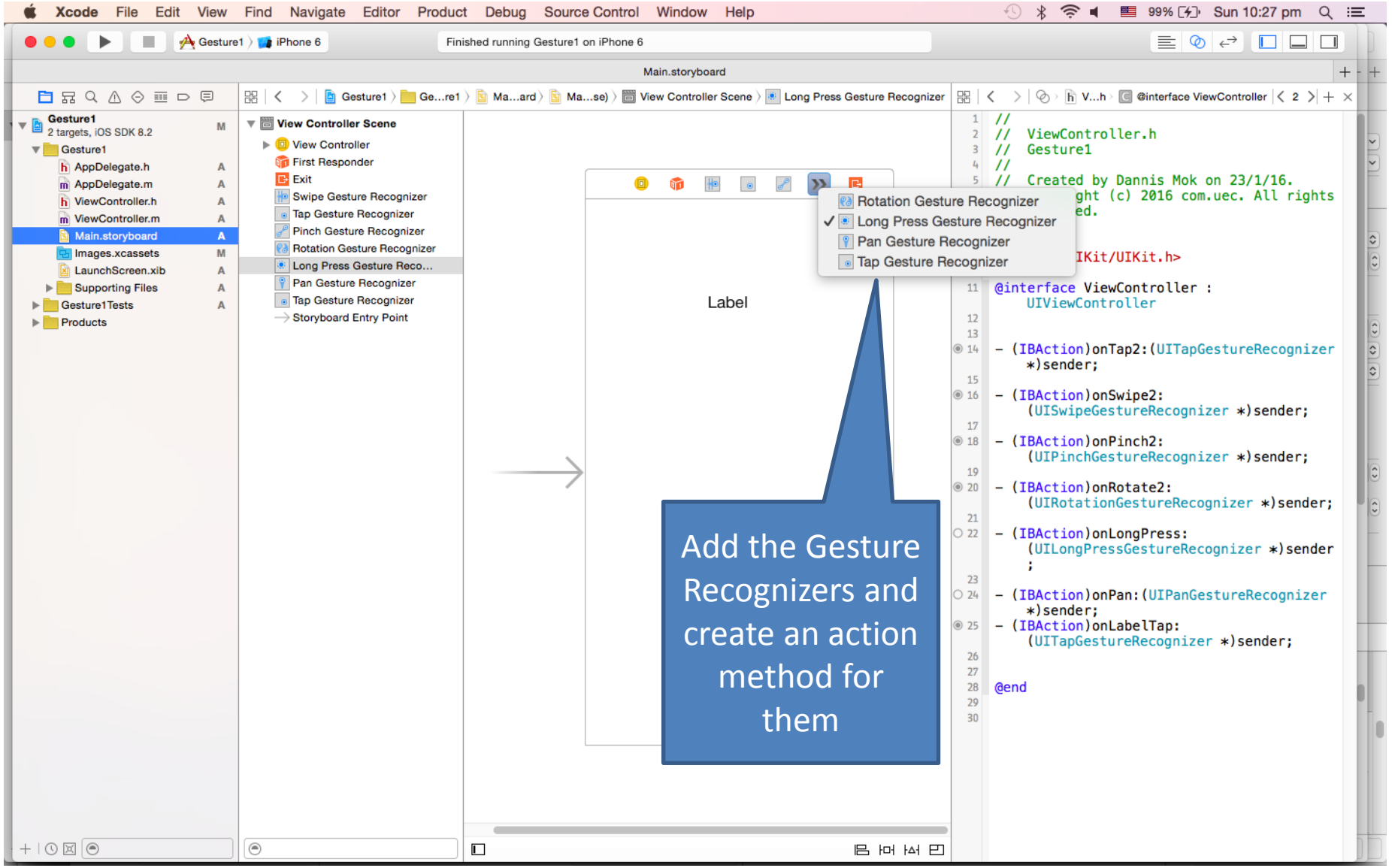
The UIKit framework provides predefined gesture recognizers that detect common gestures

Gesture	UIKit class
Tapping (any number of taps)	`UITapGestureRecognizer`
Pinching in and out (for zooming a view)	`UIPinchGestureRecognizer`
Panning or dragging	`UIPanGestureRecognizer`
Swiping (in any direction)	`UISwipeGestureRecognizer`
Rotating (fingers moving in opposite directions)	`UIRotationGestureRecognizer`
Long press (also known as “touch and hold”)	`UILongPressGestureRecognizer`

Gesture Processing



Gesture Processing



Gesture Processing

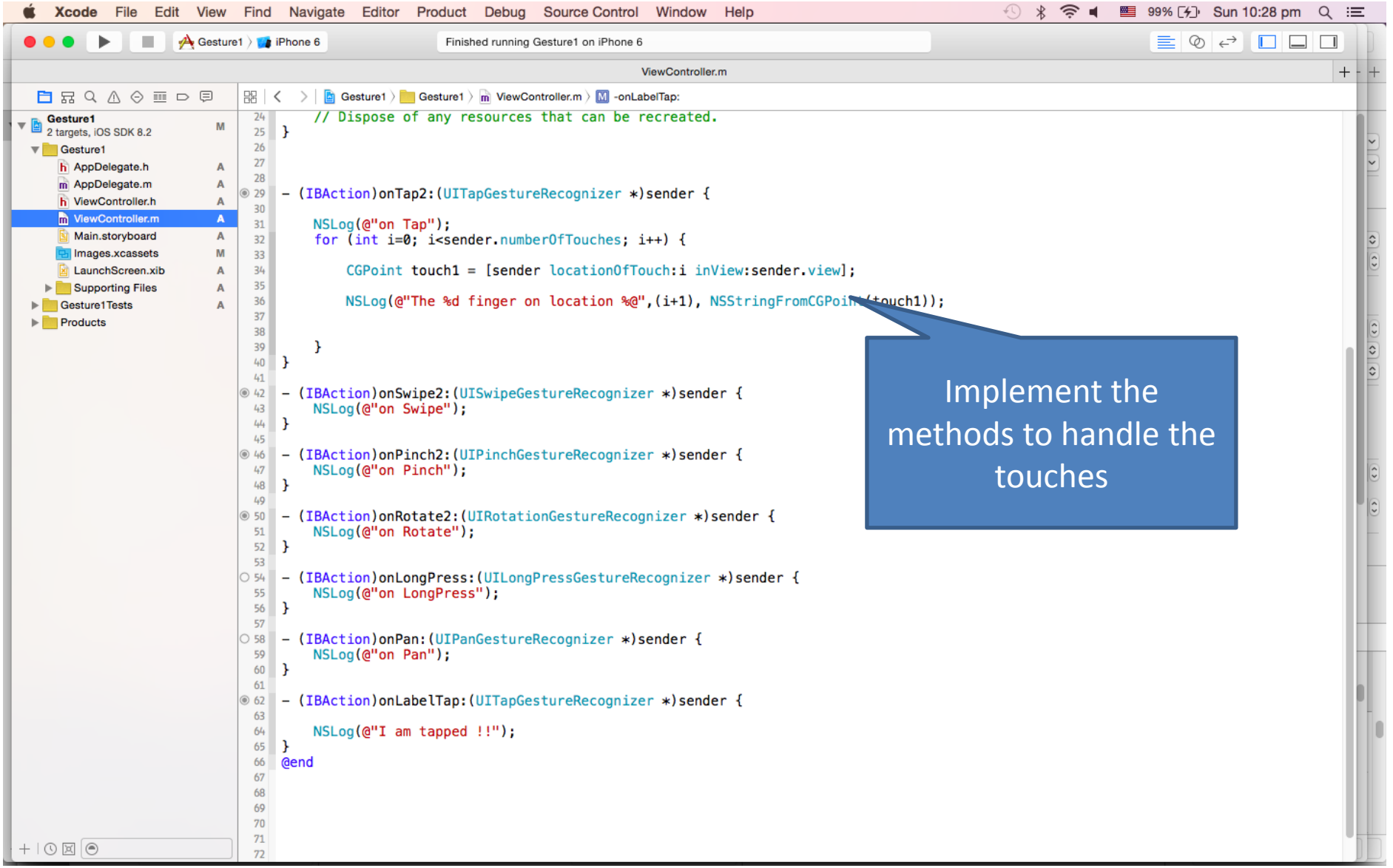


Photo Taking

Photo Taking

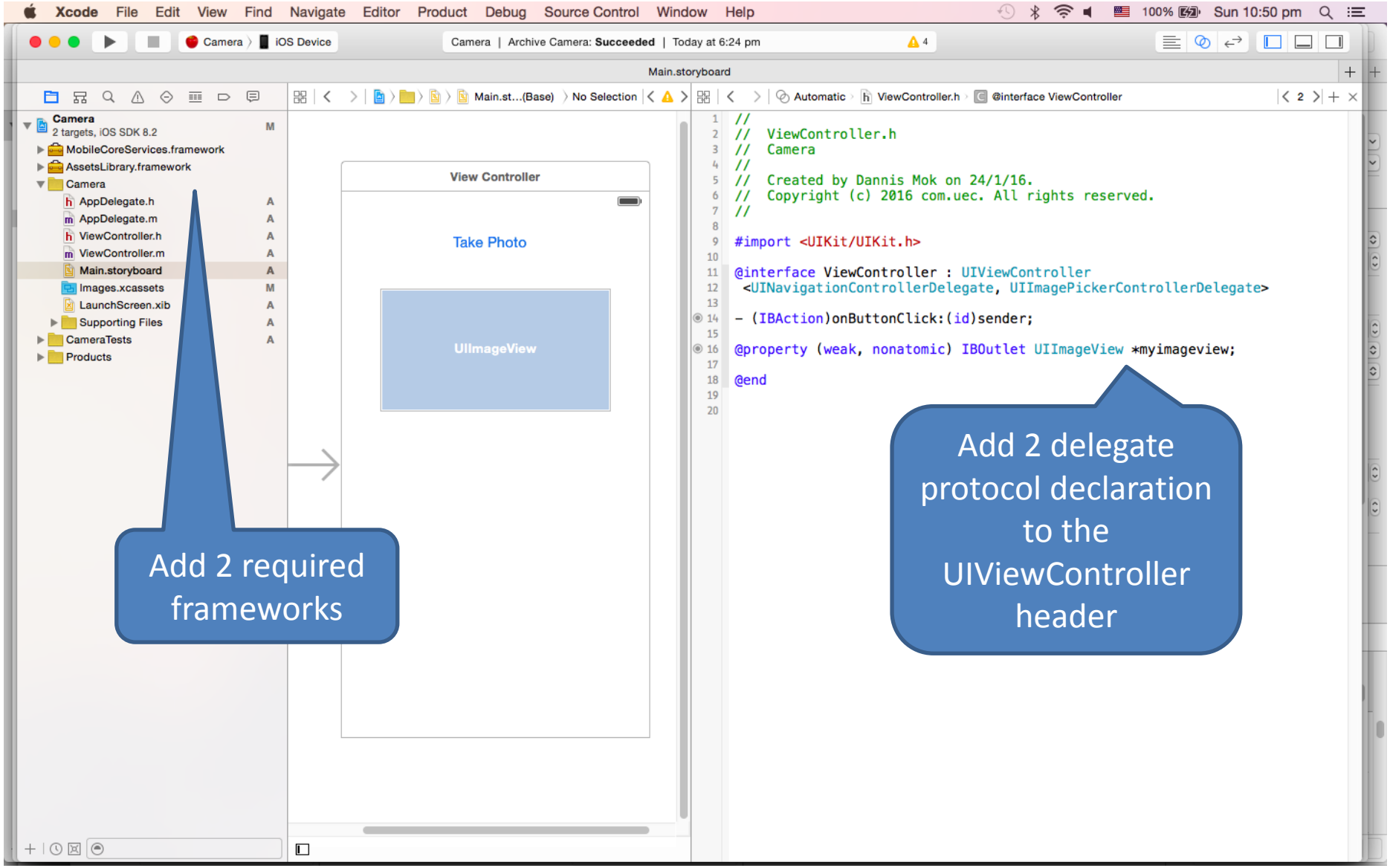
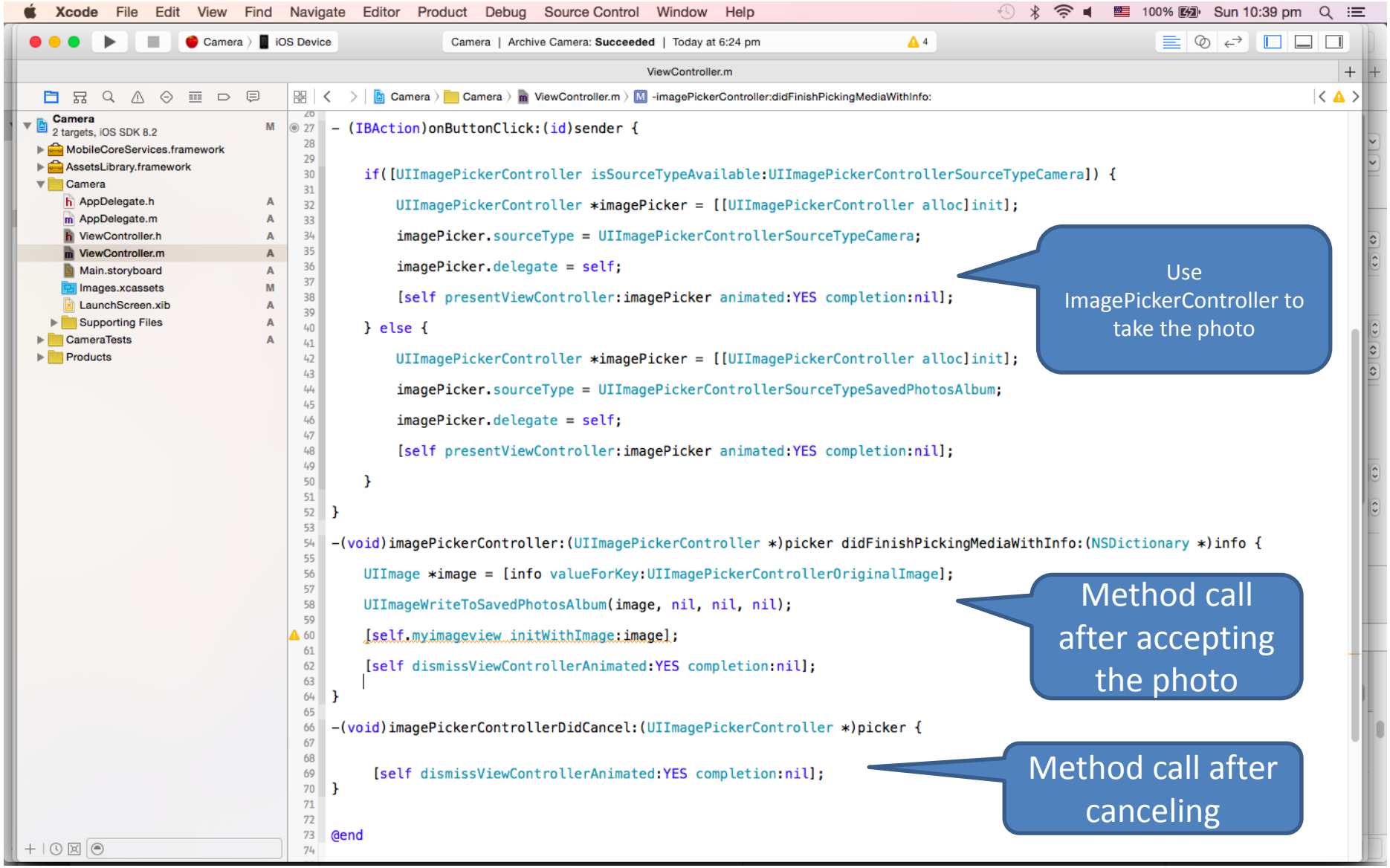
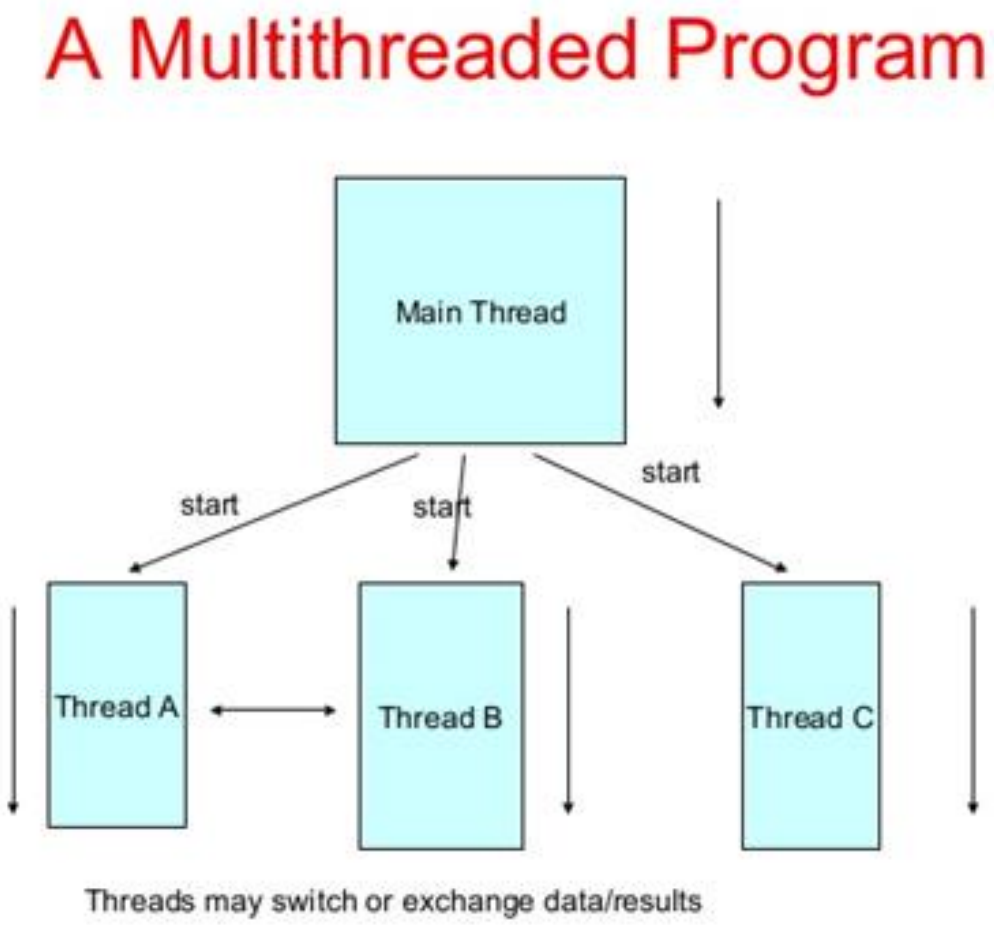


Photo Taking



Multi-Threading



Multi-Threading

```
-(void)run1:(NSTimer *)timer {
    static int n = 0;
    if(n == 10) {
        [timer invalidate];
    }
    NSLog(@"%d",n++);
}
-(void)thread1 {
    for(int i=0;i<5;i++) {
        NSLog(@"%d",i);
    }
}

-(void)thread2 {
    for(int i=100;i<110;i++) {
        NSLog(@"%d",i);
    }
}
```

NSTimer has a built-in thread. It will be scheduled to run this method and display the n value till it reaches 10

This thread method will print 0 to 5 to the log file

This thread method will print 100 to 110 to the log file

Multi-Threading

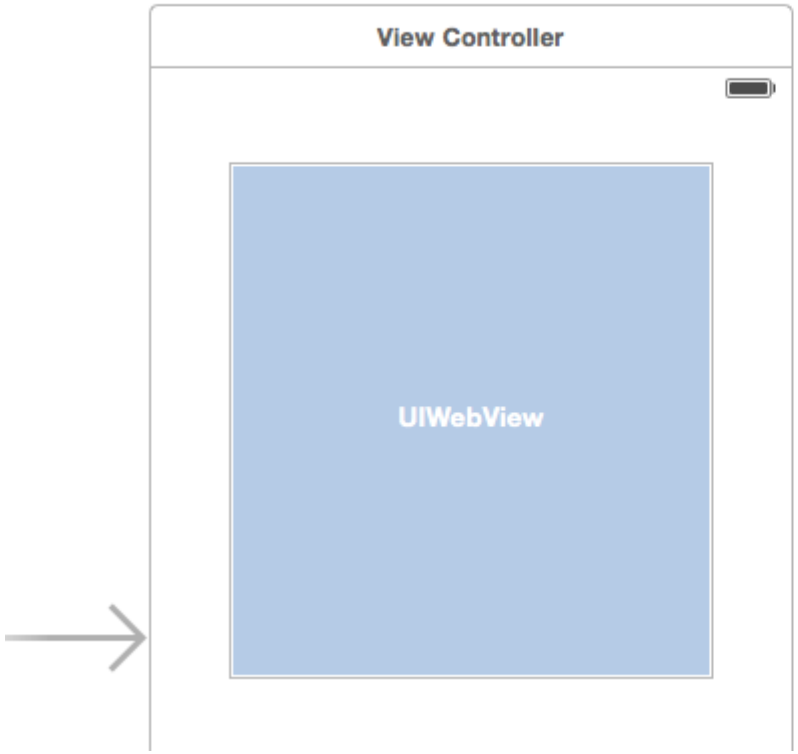
```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    NSLog(@"A");

    [NSThread sleepForTimeInterval:3.0];

    NSLog(@"B");

    [NSTimer scheduledTimerWithTimeInterval:1.0 target:self selector:@selector(run1:) userInfo:
        nil repeats:YES];


    NSThread *t1 = [[NSThread alloc]initWithTarget:self selector:@selector(thread1) object:nil];

    NSThread *t2 = [[NSThread alloc]initWithTarget:self selector:@selector(thread2) object:nil];

    [t1 start];

    [t2 start];

}
```

1) Make the main thread sleep for 3 seconds

2) Use NSTimer to run the method

3) Create 2 threads to run the 2 different methods

Web View




```
1  //
2  //  ViewController.h
3  //  Network1
4  //
5  //  Created by Dannis Mok on 24/1/16.
6  //  Copyright (c) 2016 com.uec. All rights
       reserved.
7  //
8
9  #import <UIKit/UIKit.h>
10
11 @interface ViewController : UIViewController
12
13
14 @property (weak, nonatomic) IBOutlet
       UIWebView *myWebView;
15
16
17
18 @end
19
20
```

Web View

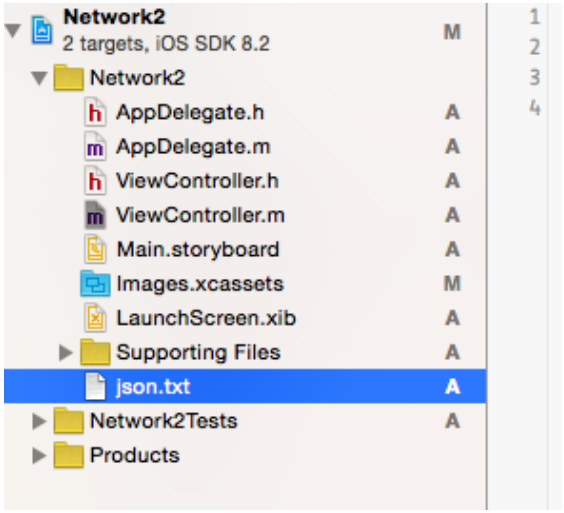
```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Print the HTML code in the WebView

    NSString *html = @"<h1>iPhone Programming</h1><h2>from Apple</h2><img src='apple.jpg' />";

    [myWebView loadHTMLString:html baseURL:nil];

    // Print the file content in the WebView

    NSURL *url = [NSURL URLWithString:@"http://localhost:8888/hello.html"];

    NSURLRequest *request = [NSURLRequest requestWithURL:url];

    [myWebView loadRequest:request];

    // Create an Image from the local file and then display it

    UIImageView *newimage = [UIImageView new];

    [newimage setImage:[UIImage imageNamed:@"apple.jpg"]];

    CGPoint point = CGPointMake(0,0);

    [newimage setCenter:point];

    [self.view addSubview:newimage];

}
```

Use the Web View to display HTML

Use the Web View to show web page content

Create and Image and show it dynamically

JSON & Network




```
1 [{"id":"1","name":"Peter","phone":"99434939","email":"flower1.jpeg"},
2  {"id":"2","name":"Mary","phone":"29349349","email":"flower2.jpeg"},
3  {"id":"3","name":"Sam","phone":"9239432","email":"flower3.jpeg"},
4  {"id":"4","name":"David","phone":"39493943","email":"flower4.jpeg"}]
```

Can create a local JSON format file

JSON & Network

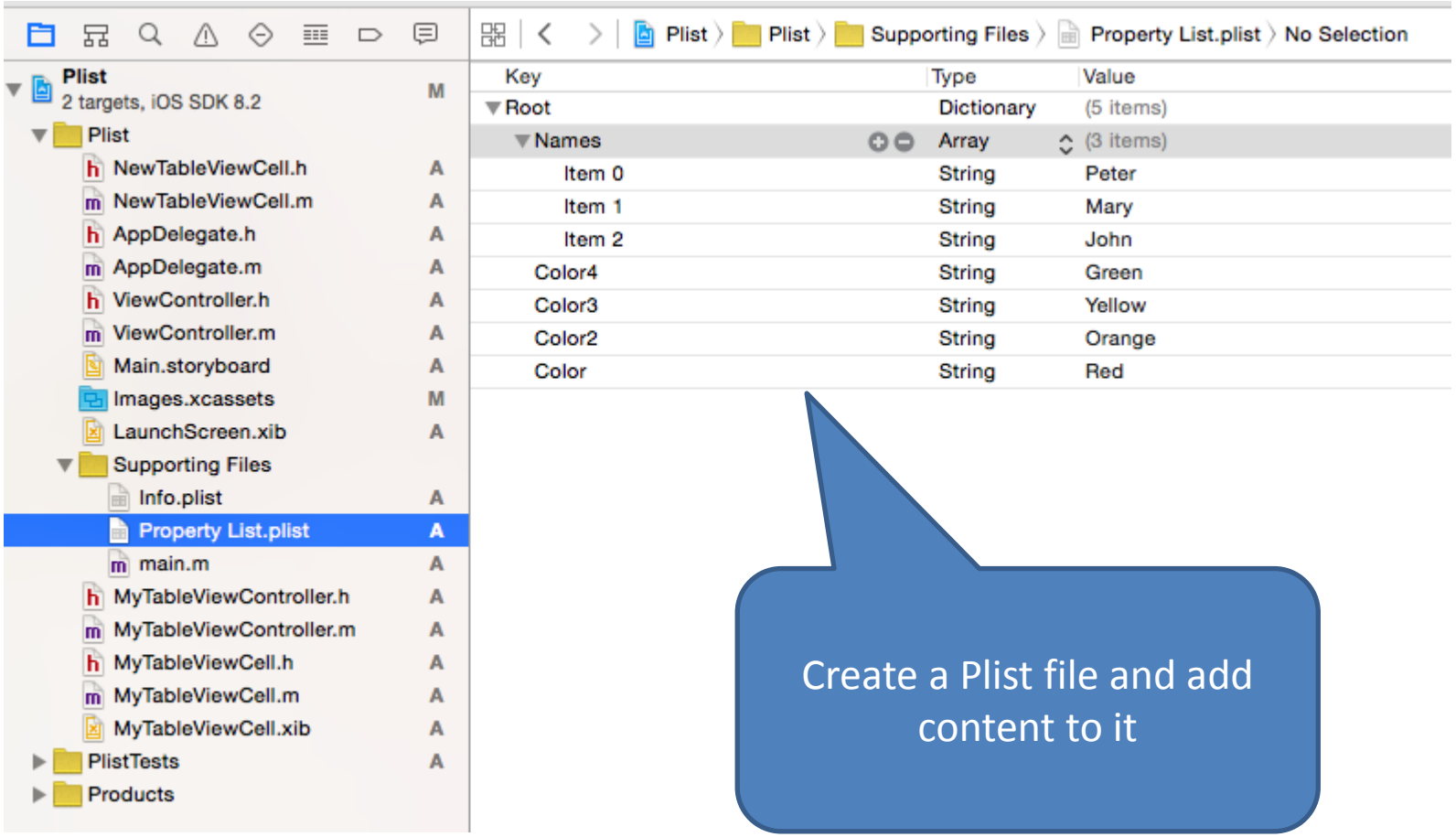
```
- (void)viewDidLoad {
    [super viewDidLoad];

    // Read data from JSON and then read the image from the network

    NSString *path = [[NSBundle mainBundle]pathForResource:@"json" ofType:@"txt"];

    NSData *data = [NSData dataWithContentsOfFile:path];

    NSArray *jsonarray = [NSJSONSerialization JSONObjectWithData:data options:
        NSJSONReadingMutableContainers error:nil];

    for(NSDictionary *p in jsonarray) {

        NSString *sid = [p objectForKey:@"id"];
        NSString *name = [p objectForKey:@"name"];
        NSString *tel = [p objectForKey:@"phone"];
        NSString *photo = [p objectForKey:@"email"];

        NSLog(@"%@ %@ %@ %@",sid,name,tel,photo);

        NSMutableString *path = [NSMutableString stringWithString:@"http://localhost:8888/"];

        path = [[path stringByAppendingString:photo]mutableCopy];

        NSLog(@"%@",path);

        NSURL *url = [NSURL URLWithString:path];

        NSURLRequest *request = [NSURLRequest requestWithURL:url];

        [NSURLConnection sendAsynchronousRequest:request queue:[NSOperationQueue mainQueue]
            completionHandler:^(NSURLResponse *response, NSData *data, NSError *connectionError)
            {
            UIImage *image = [UIImage imageWithData:data];

            NSLog(@"%@, %@",response,  image);
        }];

    }
```

Read the file content into the NSData object

Read the data and the use the image path to download the image

Use asynchronous thread to download the image

Plist file

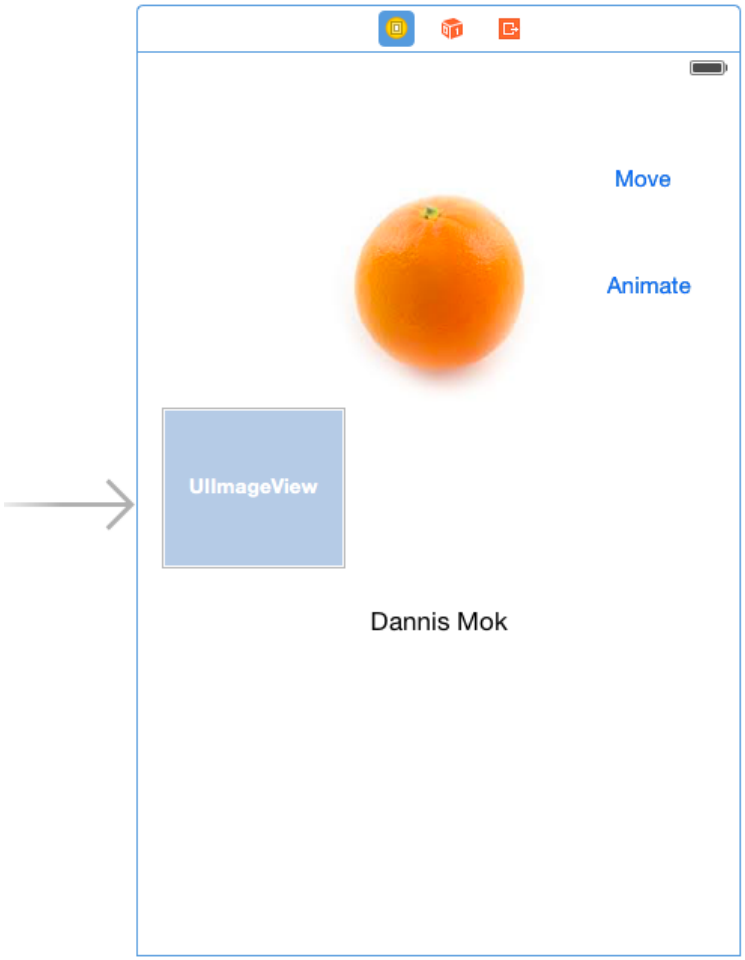

Plist
2 targets, iOS SDK 8.2
Plist
NewTableViewCell.h
NewTableViewCell.m
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Images.xcassets
LaunchScreen.xib
Supporting Files
Info.plist
Property List.plist
main.m
MyTableViewController.h
MyTableViewController.m
MyTableViewCell.h
MyTableViewCell.m
MyTableViewCell.xib
PlistTests
Products
Plist
Plist
Supporting Files
Property List.plist
No Selection
Key
Type
Value
Root
Dictionary
(5 items)
Names
Array
(3 items)
Item 0
String
Peter
Item 1
String
Mary
Item 2
String
John
Color4
String
Green
Color3
String
Yellow
Color2
String
Orange
Color
String
Red
Create a Plist file and add content to it

Plist file

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    NSString *plistpath = [[NSBundle mainBundle]pathForResource:@"Property List" ofType:@"plist"
        ];

    NSMutableDictionary *plist = [NSMutableDictionary dictionaryWithContentsOfFile:plistpath];

    NSString *color = [plist objectForKey:@"Color"];

    NSLog(@"The color is %@",color);

}
```

How to read the content of the Plist file and then print it out

UIView Properties




```
//
//  ViewController.h
//  UIViewTest
//
//  Created by Dannis Mok on 20/1/16.
//  Copyright (c) 2016 com.uec. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController


@property (weak, nonatomic) IBOutlet UIImageView *myOrange;


@property (weak, nonatomic) IBOutlet UILabel *myName;

- (IBAction)onClick:(id)sender;
@property (weak, nonatomic) IBOutlet UIImageView *myOrange2
    ;
- (IBAction)onClick2:(id)sender;

@end
```

Can use the UIView animation properties

UIView Properties

```
- (IBAction)onClick:(id)sender {

    float x1 = myName.center.x + 10;
    float y1 = myName.center.y - 10;


    CGPoint newPoint = CGPointMake(x1,y1);

    myName.center = newPoint;

    NSLog(@"%f",myName.center.x);
    NSLog(@"%f",myName.center.y);


    [UIView beginAnimations:nil context:nil];
    [UIView setAnimationRepeatCount:2];
    [UIView setAnimationDuration:2];


    myOrange.bounds = CGRectMake(0, 0, 100, 100);

    myOrange.backgroundColor = [UIColor redColor];
    myName.backgroundColor = [UIColor blueColor];
    myName.textColor = [UIColor yellowColor];

    myOrange.alpha = 0.5;
    myOrange.transform = CGAffineTransformMakeRotation(30);
    myName.transform = CGAffineTransformMakeScale(1.5, 1.5);
    myOrange.transform = CGAffineTransformMakeTranslation(20, 0);

    [UIView commitAnimations];



    UIImage *orange = [UIImage imageNamed:@"orange.jpg"];

    UIImageView *newImageView = [[UIImageView alloc]initWithImage:orange];

    newImageView.frame = CGRectMake(0, 0, 100, 100);
    newImageView.center = CGPointMake(50,50);

}
```

Move the UI element by moving its center

Wrap the property changes in the animation block