

2011 Esri Developer Summit

Palm Springs, CA

Introduction to ArcGIS API for iOS

Divesh Goyal

Eric Ito



Agenda

- Introduction
- Getting Started
- Objective-C basics
- Common design patterns
- Key Concepts
- Q&A

... Remember to turn in your surveys.

ArcGIS - A Complete Geographic Information System



... For Authoring, Serving & Using Geographic Knowledge

ArcGIS Web & Mobile APIs



ArcGIS API for iOS

- **Build native applications using Objective-C**
 - iPhone 3GS, iPhone 4, iPod Touch, iPad
 - iOS 3.1.2 and up



Web or Native applications?

- ESRI supports **both**
- **Advantages of native applications**
 - Tighter integration with other native apps
 - Access to resources
 - Contacts, calendar events, photos
 - Marketing/Hosting/Reporting via AppStore
- **Disadvantages**
 - Dedicated effort to write and maintain

Before you begin..

- You need an Intel-based Mac running OSX 10.6 (Snow Leopard)



- Join iOS Developer Program
 - Standard : AppStore distribution
 - Enterprise : In-House distribution

- Download iOS SDK (4.2.x) & Xcode IDE (3.2.x)



- ArcGIS API for iOS
 - v1.0 : Resource center
 - v1.8 in public beta : Beta Community Portal

Objective-C

- **Its C**
 - **Semicolons and curly braces**
 - **Pointers (uh-oh)**
- **But not your average C**
 - **colons and square brackets too**
 - **A different syntax**
- **Thankfully, Cocoa Touch frameworks provide elegant APIs**

Objective-C basics

- **Class = Interface + Implementation**

- **MyController.h**

```
@interface MyController: UIViewController {  
    // private variables here  
}  
// method declarations here  
@end
```

- **MyController.m**

```
@implementation MyController  
// method implementations here  
@end
```

Objective-C basics

Contd.

- **Protocol**

- **Declaring a Protocol**

```
@protocol UIApplicationDelegate
    @required
    // method definitions here
    @optional
    // method definitions here
@end
```

- **Adopting a protocol**

```
@interface MyDelegate: NSObject <UIApplicationDelegate> {
}
@end
```

Objective-C basics

Contd.

- Invoking methods = passing messages to objects

[object message]

foo.alloc();

[foo alloc];

foo.alloc().init();

[[foo alloc] init];

point.setCenter(c);

[point setCenter:c];

point.set(x,y);

[point setX:x andY:y];

Objective-C Basics

Contd.

- **Messages are read like English**
 - `presentViewController:`
 - `writeToFile:`
 - `applyEditsWithFeaturesToAdd:toUpdate:toDelete:`
- **Can get verbose**
 - `gestureRecognizer:shouldRecognizeSimultaneouslyWith
GestureRecognizer:`

Objective-C basics

Contd.

- **Garbage collection is for kids, real developers manage their own memory**

- **You own an object if you**

- **alloc**

```
MyObject* foo = [MyObject alloc];
```

- **retain**

```
[foo retain];
```

- **copy**

```
[foo copy];
```

- **If you own an object, you're responsible for releasing it**

```
[foo release];
```

Objective-C basics

Contd.

- Properties make memory management easier
- Syntactic sugar – dot notation

```
@interface MyController: UIViewController {  
    MyObject* _foo;  
}  
@property (nonatomic, retain) MyObject* foo  
  
@end
```

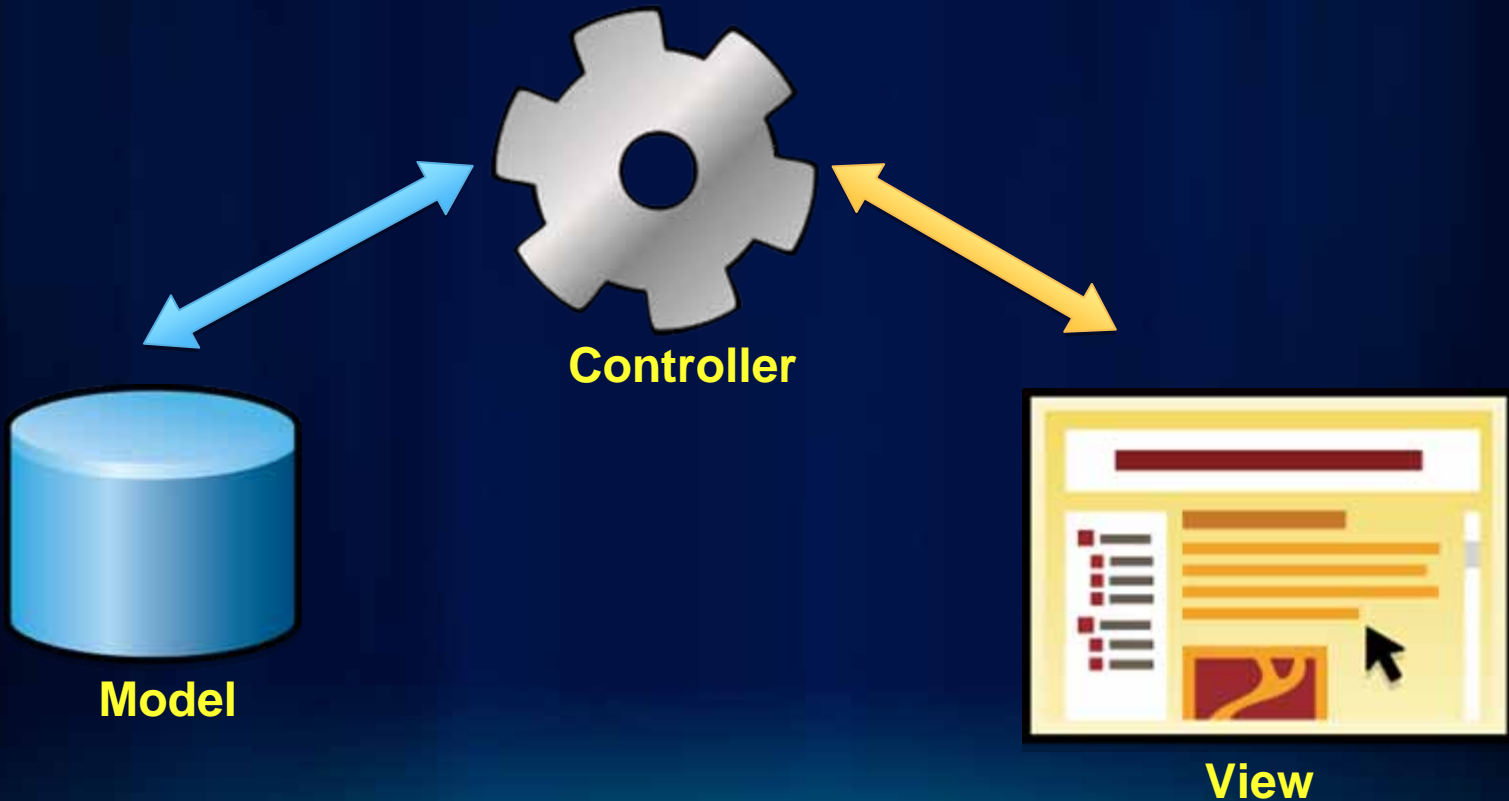
```
@implementation MyController  
  
@synthesize foo= _foo;  
  
@end
```

```
myController.foo = bar; //bar automatically retained  
myController.foo = nil; //bar automatically released
```

Monitor memory footprint with *Instruments*

Common design patterns

- Model – View – Controller



Common Design Patterns

Contd.

- Delegation



windowShouldClose:





ArcGIS API

```
function onMapLoad() {  
    var map = new esri.Map("map", {  
        basemap: "topographic",  
        layers: [esri.layers.ArcGISDynamicMapServiceLayer(  
            "http://services.esri.com/arcgis/rest/services/ESRI_Logo/MapServer")  
        )],  
        map.addLayer(new esri.layers.ArcGISDynamicMapServiceLayer(  
            "http://services.esri.com/arcgis/rest/services/ESRI_Logo/MapServer")  
        )  
    });  
}
```

```
function getDriverInfo() {  
    var features = results;  
    for (var f=0; f<features.length; f++) {  
        var feature = features[f];  
        if (feature.attributes["Driver"] == "ESRI") {  
            console.log("ESRI Driver: " + feature.attributes["Driver"]);  
        }  
    }  
}
```

```
do {  
    Symbol = new esri.Symbols.SimpleLineSymbol(  
        "solid", "#FF0000", 2);  
    feature.setSymbol(Symbol);  
} while (feature.attributes["Driver"] == "ESRI");  
}
```

```
new dojo.Color("0, 0, 0, 0.5");  
feature.setSymbol(  
    new esri.Symbols.SimpleLineSymbol(  
        "solid", "#FF0000", 2)  
    )  
);  
}
```

```
new dojo.Color("0, 0, 0, 0.5");  
feature.setSymbol(  
    new esri.Symbols.SimpleLineSymbol(  
        "solid", "#FF0000", 2)  
    )  
);  
}
```

```
new dojo.Color("0, 0, 0, 0.5");  
feature.setSymbol(  
    new esri.Symbols.SimpleLineSymbol(  
        "solid", "#FF0000", 2)  
    )  
);  
}
```

```
new dojo.Color("0, 0, 0, 0.5");  
feature.setSymbol(  
    new esri.Symbols.SimpleLineSymbol(  
        "solid", "#FF0000", 2)  
    )  
);  
}
```

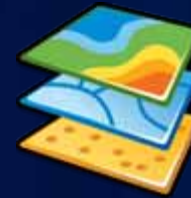
What you can do with the API

- Display maps
- Perform analysis
- Visualize results
- Collect data
 - **Thu 8:30 AM & 1:30 PM**



Displaying a Map

- **UI Component : AGSMapView**
 - Responds to gestures
 - Displays GPS location
- **Mashup individual layers**
 - ArcGIS Server Tiled, Dynamic, Image
 - Bing, Open Street Map
- **Open ArcGIS.com web maps**



Respond to Map events through Delegates

Map Delegates

- **Layer Delegate**
 - Map loaded, failed to load
 - Layer loaded, failed to load
- **Touch Delegate**
 - Tap, Tap and Hold
 - Move Tap and Hold
- **Callout Delegate**
 - Did Show Callout, Should Show Callout
 - Did Click Accessory Button

Responding to Map Touch events

1. Adopt the Delegate protocol

```
@interface MyController: UIViewController <AGSMapViewTouchDelegate> {  
}
```

2. Implement the protocol methods

```
@implementation MyController  
  
- (void) mapView:(AGSMapView*) mapView  
  didClickAtPoint:(CGPoint) screen  
    mapPoint:(AGSPoint*) mappoint  
    graphics:(NSDictionary*) graphics {  
    //handle touch event  
}
```

3. Set Delegate

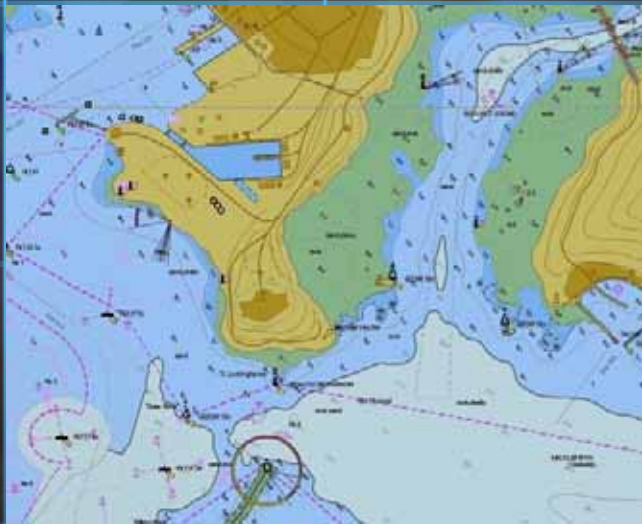
```
self.mapView.touchDelegate = self;
```

DeKalb County Board

Fulton County Dept. of Health and Wellness/District 3, Unit 2, G

DEMO

Display a map



Performing Analysis

- **Query, Find, Identify Task**

- Search for features in the map



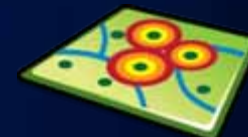
- **Geoprocessing Task**

- Spatial analysis using GP tools and models



- **Locator**

- Geocode and reverse geocode addresses



- **GeometryService Task**

- Geometric operations such as cut, union, buffer, etc.

New at v1.8

- **Geometry Engine**
 - native, high performance engine for performing geometric operations on the device
- **Routing Task**
 - Solve point-to-point and multipoint routes
 - Driving directions
 - Barriers, Time Windows, Best Sequence



Common Pattern for using Tasks

1. Adopt the Task Delegate protocol

```
@interface MyController: UIViewController <AGSLocatorDelegate> {  
}
```

2. Implement the protocol methods

```
- (void)locator:(AGSLocator*)locator  
    operation:(NSOperation*)op  
didFindLocationsForAddress:(NSArray*)candidates {  
    //todo  
}  
  
- (void)locator:(AGSLocator*)locator  
    operation:(NSOperation*)op  
didFailLocationsForAddress:(NSError*)error {  
    //todo  
}
```

Common Pattern for using Tasks

3. Instantiate the task

```
self.locator =  
[AGSLocator locatorWithURL:[NSURL URLWithString:kGeoLocatorURL]];
```

4. Set Delegate

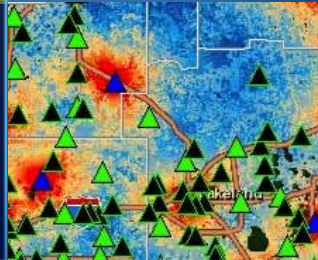
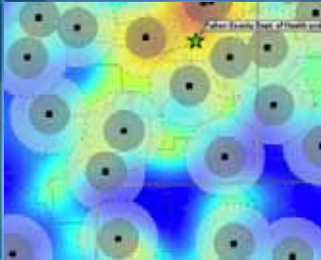
```
self.locator.delegate = self;
```

5. Perform operation

```
NSOperation* op =  
[self.locator locationsForAddress:addresses returnFields:outFields];
```

Fulton County Dept. of Health and Wellness/District 3, Unit 3, G...

DeKalb County Board



DEMO

Using Tasks

Visualizing Results

- **Graphics**
 - **Geometry**
 - **Attribute**
 - **Symbol**

- **Symbols**
 - **Picture, Marker, Line, Fill, and Composite**
 - **Text (new at v1.8)**

Visualizing Results

Contd.

```
//create the symbol
```

```
AGSPictureMarkerSymbol *marker =
```

```
[AGSPictureMarkerSymbol pictureMarkerSymbolWithImageNamed:@"BluePushpin.png"];
```

```
//create the graphic
```

```
AGSGraphic *graphic = [AGSGraphic graphicWithGeometry:point
```

```
                    symbol:marker
```

```
                    attributes:dictionary
```

```
//add the graphic to the graphics layer
```

```
[self.graphicsLayer addGraphic:graphic];
```

Visualizing Results

Contd.

- **Renderers**
 - **Simple**
 - **Unique Value , Class Breaks**
 - **Temporal**

Visualizing Results

Contd.

```
//renderer for places
```

```
AGSUniqueValueRenderer *placeRend= [[AGSUniqueValueRenderer alloc] init];  
placeRend.field1 = @"TYPE";
```

```
//unique value for city
```

```
AGSSimpleMarkerSymbol *citySymbol= [AGSSimpleMarkerSymbol simpleMarkerSymbol];  
citySymbol.style = AGSSimpleMarkerSymbolStyleDiamond;  
citySymbol.outline.color = [UIColor blueColor];  
AGSUniqueValue* city = [AGSUniqueValue uniqueValueWithValue: @"city"  
                        symbol:citySymbol];  
[placeRend.uniqueValues addObject:city];
```

```
//unique value for town
```

```
AGSSimpleMarkerSymbol *townSymbol = [AGSSimpleMarkerSymbol simpleMarkerSymbol];  
townSymbol.style = AGSSimpleMarkerSymbolStyleCross;  
townSymbol.outline.width = 3.0;  
AGSUniqueValue* town = [AGSUniqueValue uniqueValueWithValue: @"town"  
                        symbol:townSymbol];  
[placeRend.uniqueValues addObject:town];
```

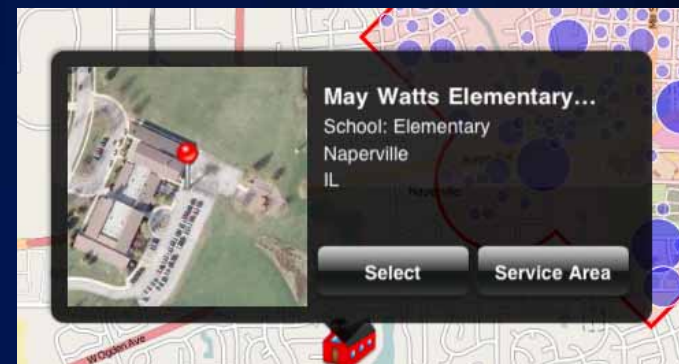
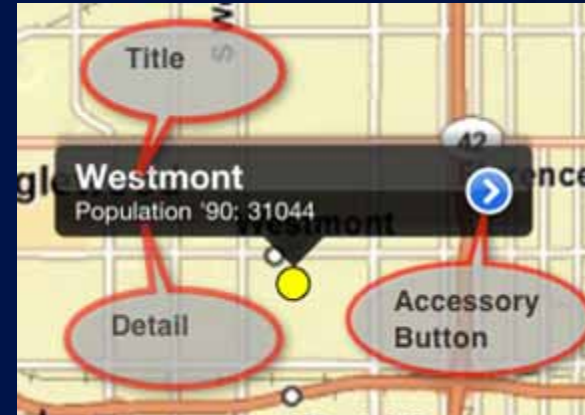
```
//assign the renderer
```

```
self.graphicsLayer.renderer = placeRend;
```

Visualizing Results

Contd.

- **Callout**
 - Displayed automatically when user taps on a graphic
- **Content**
 - Title & Detail
 - Image
 - Accessory button
 - Custom UI View



Specifying Content for the Callout

1. Adopt the Delegate Protocol

```
@interface MyController: UIViewController <AGSInfoTemplateDelegate> {  
}
```

2. Implement the protocol methods

```
@implementation MyController  
  
- (NSString *) titleForGraphic:(AGSGraphic*)graphic  
                        screenPoint:(CGPoint)screen  
                        mapPoint:(AGSPoint*)map {  
    //todo  
}  
  
- (NSString *) detailForGraphic:(AGSGraphic*)graphic  
                        screenPoint:(CGPoint)screen  
                        mapPoint:(AGSPoint*)map {  
    //todo  
}
```

3. Set the delegate on the graphic

```
AGSGraphic *graphic = ...  
graphic.infoTemplateDelegate = self;
```

Responding to the Callout's Accessory button

1. Adopt the Delegate Protocol

```
@interface MyController: UIViewController <AGSMMapViewCalloutDelegate> {  
}
```

2. Implement the protocol methods

```
@implementation MyController  
  
- (void) mapView:(AGSMMapView*) mapView  
  didClickCalloutAccessoryButtonForGraphic:(AGSGraphic*) graphic {  
    //todo  
}
```

3. Set the delegate

```
self.mapView.calloutDelegate = self;
```

DeKalb County Board

Fulton County Dept. of Health and Wellness/District 3, Unit 3, G

DEMO

Visualizing Results



More Resources

- **iOS Resource Center**
 - **Conceptual help, API Reference**
 - **Blog, Forums**
- **Samples on ArcGIS.com**
 - **ArcGIS for iOS Developer Samples group**
- **API v1.8 Public Beta**
 - **betacommunity.esri.com**
- **Web Course : Getting Started with the ArcGIS API for iOS**
 - **training.esri.com**



Related Sessions

- **Choosing a Mobile Deployment Platform**
 - Today, 2:45 PM
- **Meet the Teams**
 - Today, 6 PM
- **Advanced Development with ArcGIS API for iOS**
 - Thu, 8:30 AM, Repeat 1:30 PM

Thank You

Help make this session better...

... Turn in your surveys.



esri