Developing for iOS with MonoTouch

by Brad Pillow
PillowSoft LLC

Credits: much borrowed from many other MonoTouch presentations on the web!

History

- First computer I used: Altair 8800 [1975]
- Was excited I had 4K of memory and could play audio over an AM radio [Basic]
- Went to Purdue and fell in love with C, Lisp and APL [anyone for programming in Stargate chevrons?]
- `life←{↑1 ω∨.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂ω`

HOW TO "READ" FM TUNER SPECIFICATIONS

Popular Electronics

WORLD'S LARGEST-SELLING ELECTRONICS MAGAZINE JANUARY 1975/75¢

PROJECT BREAKTHROUGH!

World's First Minicomputer Kit to Rival Commercial Models...

"ALTAIR 8800" SAVE OVER $1000

ALSO IN THIS ISSUE:

- An Under-$90 Scientific Calculator Project
- CCD's—TV Camera Tube Successor?
- Thyristor-Controlled Photoflashers

TEST REPORTS:

Technics 200 Speaker System
Pioneer RT-1011 Open-Reel Recorder
Tram Diamond-40 CB AM Transceiver
Edmund Scientific "Kirlian" Photo Kit
Hewlett-Packard 5381 Frequency Counter

History

- Went to work for Bell Labs, got to use this thing called "C with Classes" and CFront made by Bjarn Stroustrup [1983]
- Worked on a set-top box that had dynamic device independent graphics, animation, hyper-links, weather, news, chat, and a byte-code interpreter,... in [1983]
- Got addicted to C++, but checked out this Smalltalk like language called Objective-C
- I didn't like it then, still don't :)

History

- Started a local company called Truevision [1987]
- Built true-color image display and capture cards
- 5 seconds of Wikipedia fame: creator of the TGA image file format
- Was excited that I could raytrace 3 reflective spheres in 3 days on my x86 machine...now you can do it far better and faster than realtime on iOS devices

TRUEVISION

History

- Worked for Adobe for many years developing Adobe Premiere. Millions of lines of C++ code. Did a lot of Objective-C for the Mac version.
- A year ago, left Adobe to explore mobile...had enough with C++
- IPhone/iPad - so much power...but did I need to use that language I really don't like?
- Looked into alternatives - MonoTouch, Lua, Ruby, Gambit Scheme, PhoneGap (JavaScript)

Possibilities Abound

- MonoTouch - http://www.xamarin.com
- Ruby - http://www.rubymotion.com/
- Gambit-Scheme - http://www.iro.umontreal.ca/~gambit/
- Lua - http://www.coronalabs.com/
- PhoneGap - http://phonegap.com/

Hybrid Apps

- All of those languages allow creation of hybrid apps
- LightRoom from Adobe is a hybrid desktop Lua app. All upper level UI is in Lua, lower level computationally complex (imaging, video) is done in C++.
- Best iOS languages can call Obj-C or C++, useful for binding available toolkits

Why The History

- Mobile computing and GPU power is amazing!
- Tools are better than a quarter century ago, but haven't come as correspondingly far as our hardware
- MonoTouch a better alternative...perhaps?

Tooling

- Tooling is one of the most important factors in my choice of languages
- Good IDE
- Code completion
- Refactoring
- Unit Testing
- Performance Measurements
- Debugging

MonoTouch

- Ok IDE on Mac, better on PC
- Code completion - excellent...see ReSharper
- Refactoring - awesome, again from ReSharper
- Unit Testing - excellent
- Performance Measurements - on Mac use Instruments, on PC better to use VisualStudio
- Debugging - ok on Mac, excellent on PC

Always important to ask...what's it cost?

- Personal license - $399 (can probably get discount)
- Yearly renewal - $249
- Can test for free (code runs in simulator only, no deployment)

What is Mono?

- C# Compiler
- Runtime (CLR & DLR)
- Cross Platform

Cross Platform

Who’s Using It

- 500 developers download/day on average
- Enterprise: 3M, Medtronic, Target, TIBCO, etc.
- Consumer: AOL, iCircuit, Monster, Rdio, etc.
- Consultants / SIs: Accenture, ITR Mobility, etc.

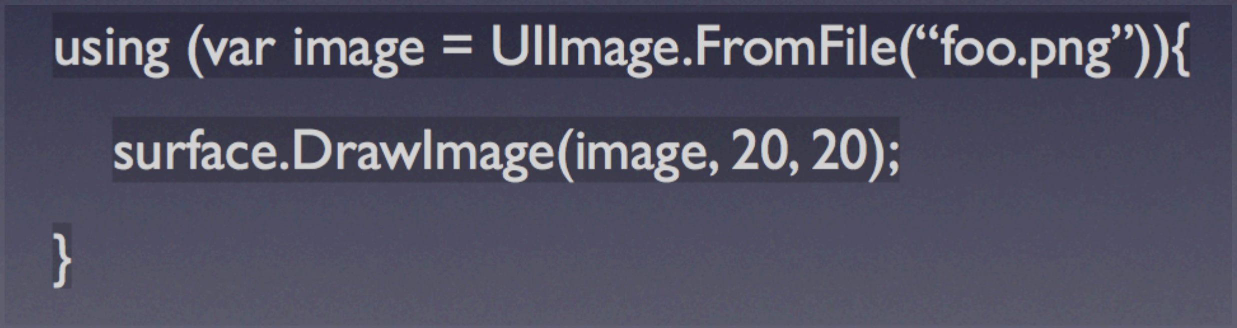
How do we use C# to develop mobile apps?

AOT Compilation

- Apple disallows Just-In-Time compilation (JIT)
- Cannot make writable memory executable – enforced by OS (except they break their own rules with Nitro)
- MonoTouch uses Ahead-of-Time Compilation (AOT)
  - Generates the native code that JIT would normally generate
- Links to runtime to create single ARM process capable native binary

Garbage Collection

- **Managed Code handles garbage collections**
- **Objective-C for iOS uses Retain Counts, this was a pain, now better with ARC**
- **MonoTouch handles garbage collection, all automatic**
- **Can view Objective-C object and C object allocation/deallocation uniformly, i.e. no dichotomy between creating UIKit objects and Core objects**
- **Need deterministic?**
  - Use when you need control.
  - Every object in MonoTouch implements IDisposable



```
using (var image = UIImage.FromFile("foo.png")){
    surface.DrawImage(image, 20, 20);
}
```

Strong Types

- Objective-C
  - Arrays are weakly typed
  - NSArray return values
- MonoTouch has strong types
  - UIView[] SubViews {get;}
  - vs
  - NSArray* subviews;
- Intellisense allows you to more easily explore the api

UI Development

- Tight integration between MD and IB
  - IB produces XIBs with MD parses
- Create UIView in MonoDev IDE
- Double click it to open in InterfaceBuilder (now XCode)
- Create UI as you normally would for XCode
- Outlets get mapped backed to a partial C# class file
- You reference these properties on the view in your main code

Actions

- Objects emit broadcast messages to receivers
- You can do this C#
- MonoDevelop takes care of the details for you
- Creates partial methods for you extend

MonoTouch Events

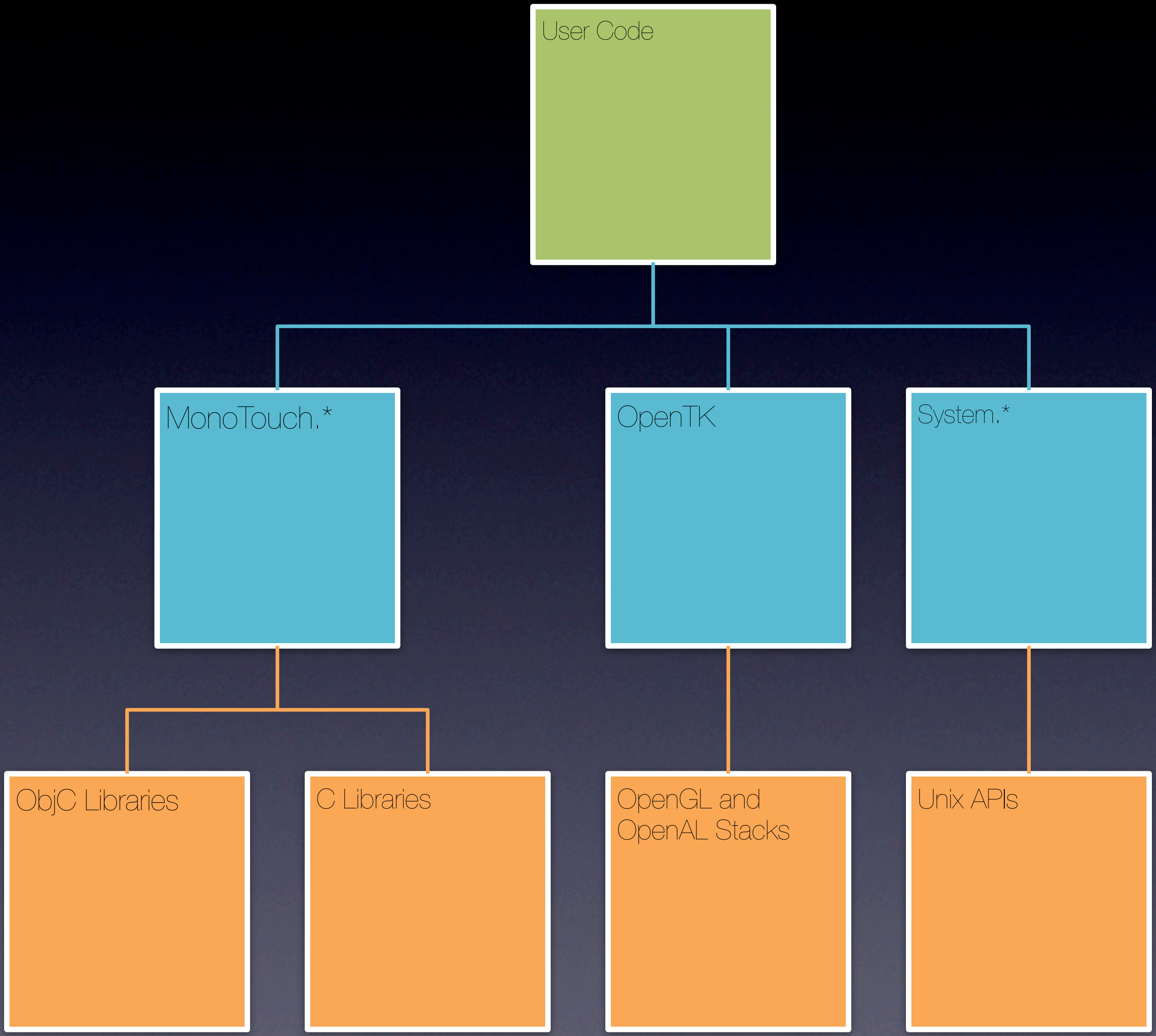
- Supports Objective-C pattern (including blocks)
  - webView.Delegate = new MyWebViewDelegate();
- C# style as well (what I use)
  - webView.PageLoaded += delegate { HideSpinningBusyWheel();}

The Bindings

- MonoTouch namespace MonoTouch.Foo namespace
- Maps to CocoaTouch's Foo Framework
- 1:1 Mapping of classes.
- MonoTouch.UIKit.UILabel
- CocoaTouch's UIKit framework, UILabel class

Native API Access - iOS

- 1,700 C# classes
  - 1:1 mapping to native
  - Objective-C libraries
  - CoreFoundation
  - iOS C libraries
- Projected into C#
  - Strongly typed
  - C# Events/Properties
  - Surface Lambdas
  - LINQ
  - Generics
  - Lambdas
  - Anonymous methods
  - Named arguments



MONOTOUCH APIS

.NET APIs

- mscorlib
- System
- System.Core (LINQ)
- System.Data
- Mono.Data.Sqlite
- System.ServiceModel
  - WCF
- System.Json
- System.Web.Services
- System.Xml
- System.Xml.Linq

MonoTouch

- AddressBook/ AddressBookUI
- AudioToolbox/ AVFoundation
- CoreAnimation
- Coregraphics
- CoreLocation
- GameKit
- MediaPlayer
- MessageUI
- StoreKit
- SystemConfiguration
- UIKit

Third Party

- OpenTK
  - OpenGL
  - OpenAL
- Sqlite-CS
- XnaTouch
- CocosNet
- ServiceStack

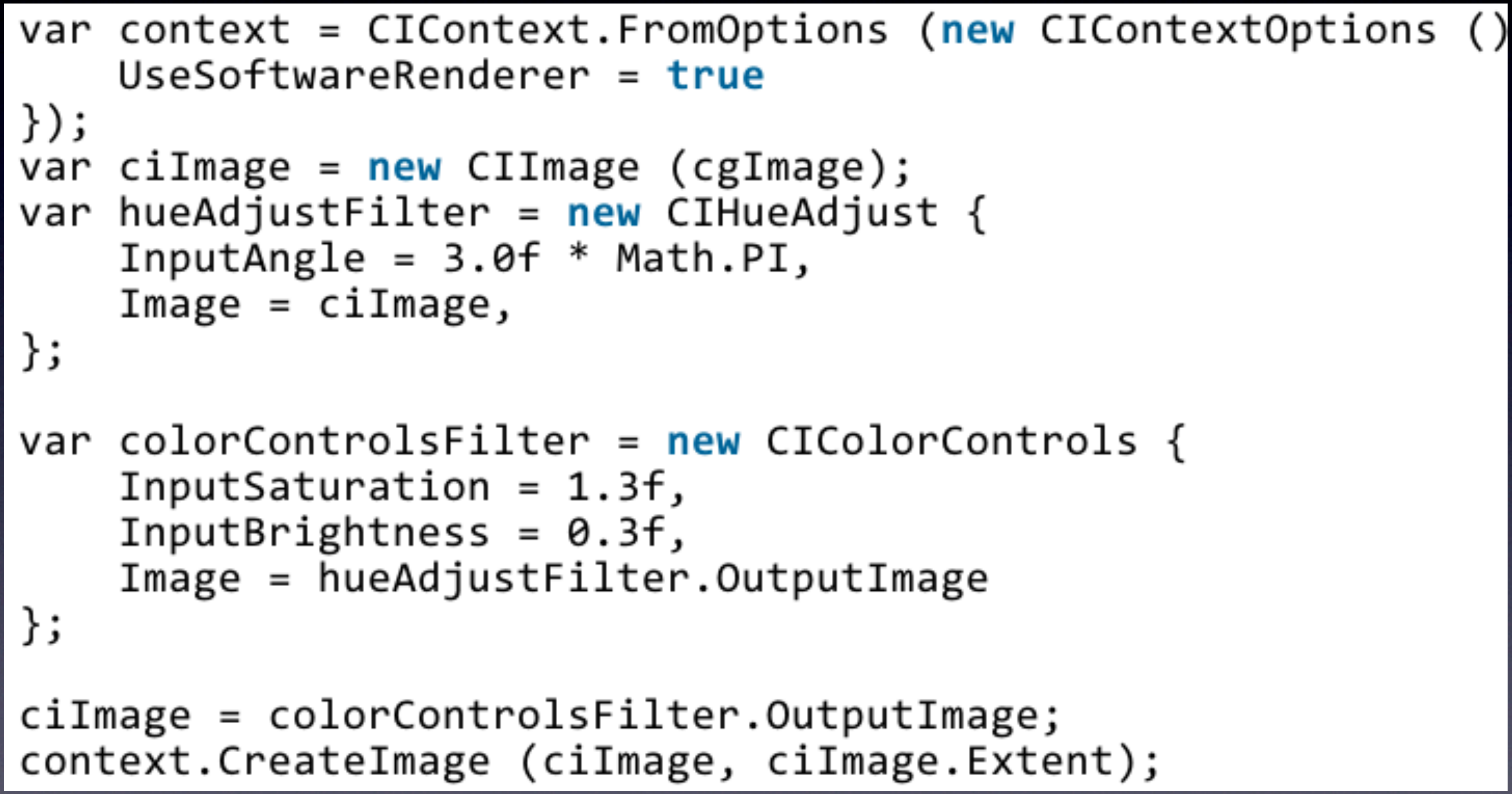
Less Code!

Objective C

```
CIContext *context =
     [CIContext contextWithOptions:
        [NSDictionary dictionaryWithObject:[NSNumber numberWithBool:YES]
                  forKey:kCIContextUseSoftwareRenderer]];
CIImage *ciImage = [CIImage initWithCGImage:cgImage];

CIFilter *hueAdjustFilter = [CIFilter filterWithName:@"CIHueAdjust"];
CIFilter *colorControlsFilter = [CIFilter filterWithName:@"CIColorControls"];

[hueAdjustFilter setValue:[NSNumber numberWithDouble:3.0 * M_PI] forKey:@"inputAngle"];

[colorControlsFilter setDefaults];
[colorControlsFilter setValue:[NSNumber numberWithDouble:1.3] forKey:@"inputSaturation"];
[colorControlsFilter setValue:[NSNumber numberWithDouble:0.3] forKey:@"inputBrightness"];

[hueAdjustFilter setValue:ciImage forKey:@"inputImage"];
[colorControlsFilter setValue:[hueAdjustFilter valueForKey:@"outputImage"] forKey:@"inputImage"];
ciImage = [colorControlsFilter valueForKey:@"outputImage"];

[context [createCGImage: ciImage fromExtent:[ciImage extent]]];
```

Less Code - MonoTouch

C#



```
var context = CIContext.FromOptions (new CIContextOptions ()
    UseSoftwareRenderer = true
});
var ciImage = new CIImage (cgImage);
var hueAdjustFilter = new CIHueAdjust {
    InputAngle = 3.0f * Math.PI,
    Image = ciImage,
};

var colorControlsFilter = new CIColorControls {
    InputSaturation = 1.3f,
    InputBrightness = 0.3f,
    Image = hueAdjustFilter.OutputImage
};

ciImage = colorControlsFilter.OutputImage;
context.CreateImage (ciImage, ciImage.Extent);
```

Debugger

- MonoTouch debugger leverages Mono’s new Soft-Debugger
- Supports the Simulator
- Supports the Device
  - Even over wifi

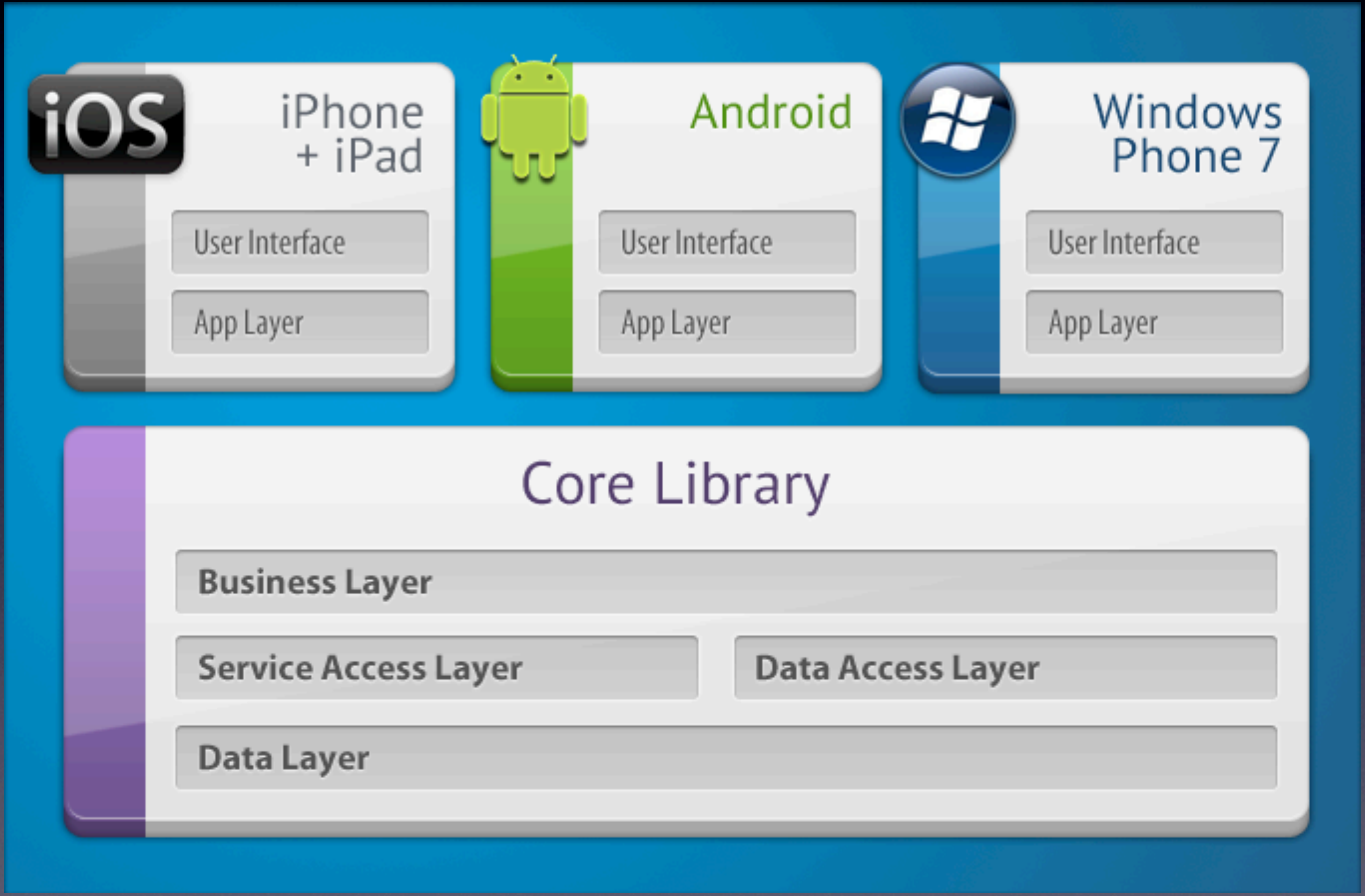
Debugger Features

- Breakpoints
- Catch-points
- Inspection
- Watches
- Immediate / Expression Evaluator
- Call Stack
- Stepping

Debugger Caveats

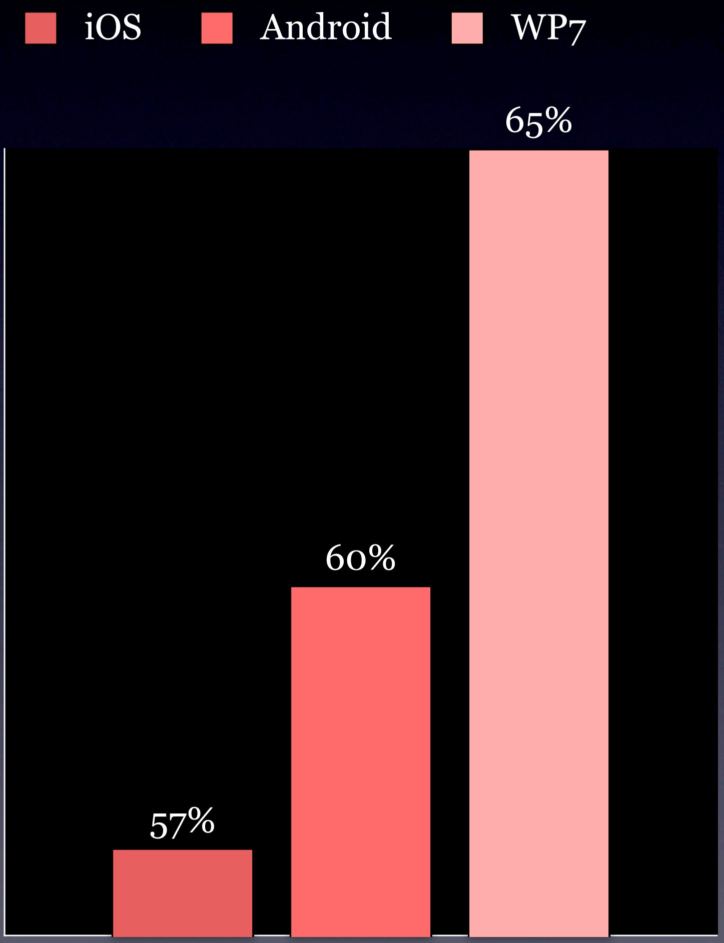
- Debug binaries on devices are very large
- Cannot debug Main or FinishedLaunching on device
  - You'll get the iOS timeout abort
- Consumes more memory runtime
- Performance hit

Going Cross Platform

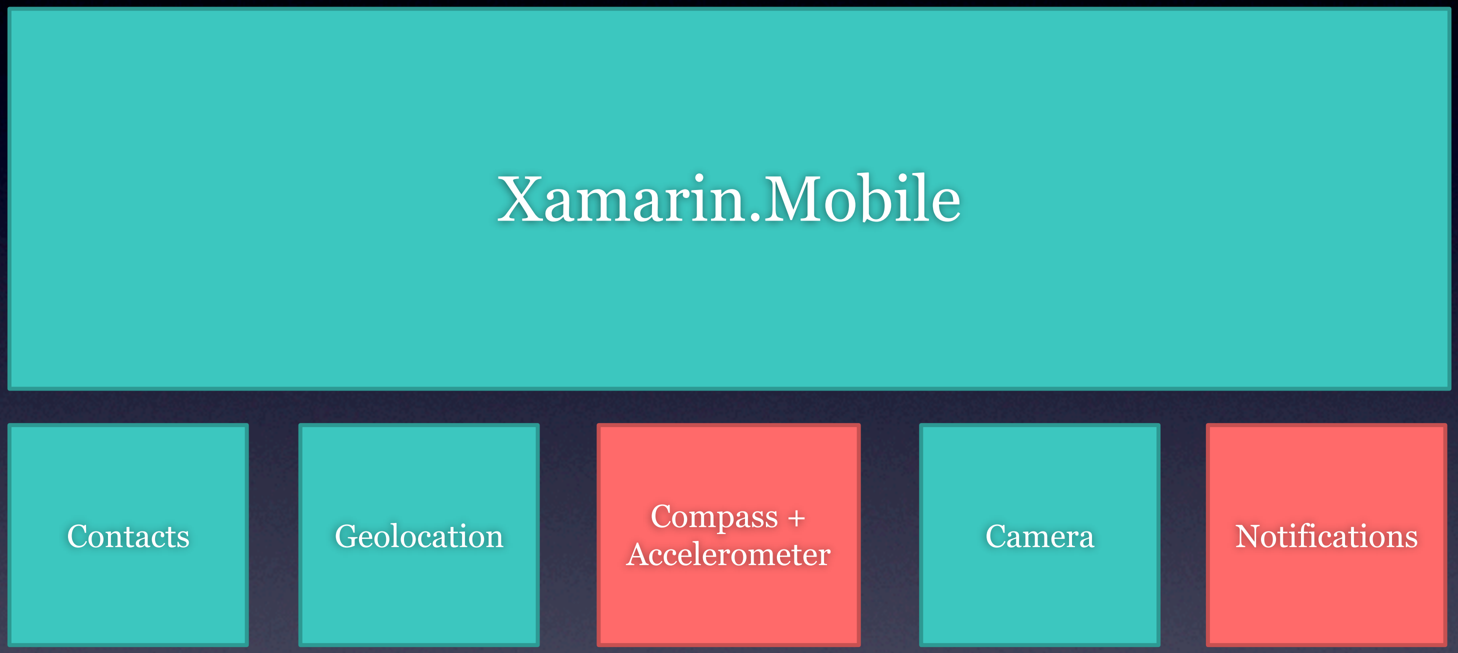


CODE REUSE FOR MWC APP

- 100% reuse of Core Library (1635 LOC)
- iPhone + iPad (2476 LOC)
- Android (1095 LOC)
- WP7 (896 LOC)



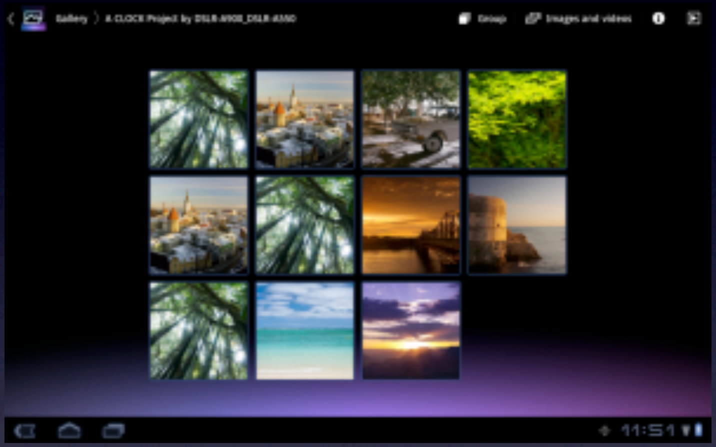
Frameworks For Reuse

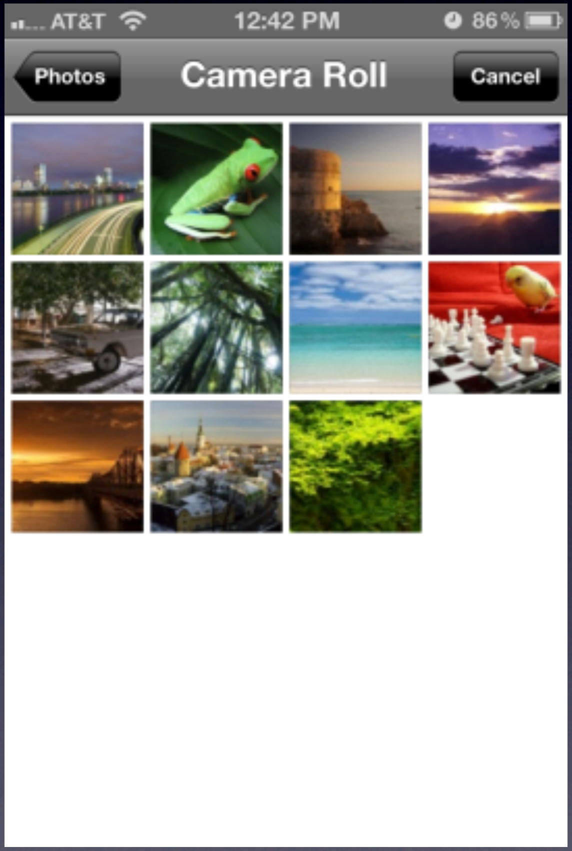


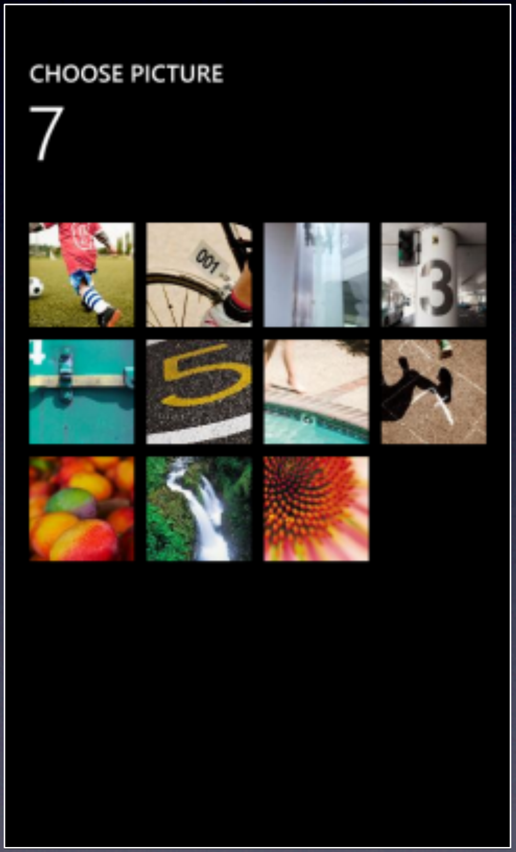
Code Reuse Example

```
ImageView image = FindViewById<ImageView> (Resource.Id.image);

var picker = new MediaPicker (this);
picker.PickPhotoAsync()
      .ContinueWith (t => {
     if (t.IsCanceled || t.IsFaulted) // user canceled or error
                 return;

     Bitmap b = BitmapFactory.DecodeFile (t.Result.Path);
     RunOnUiThread (() => image.SetImageBitmap (b));
});
```

Code Reuse Look



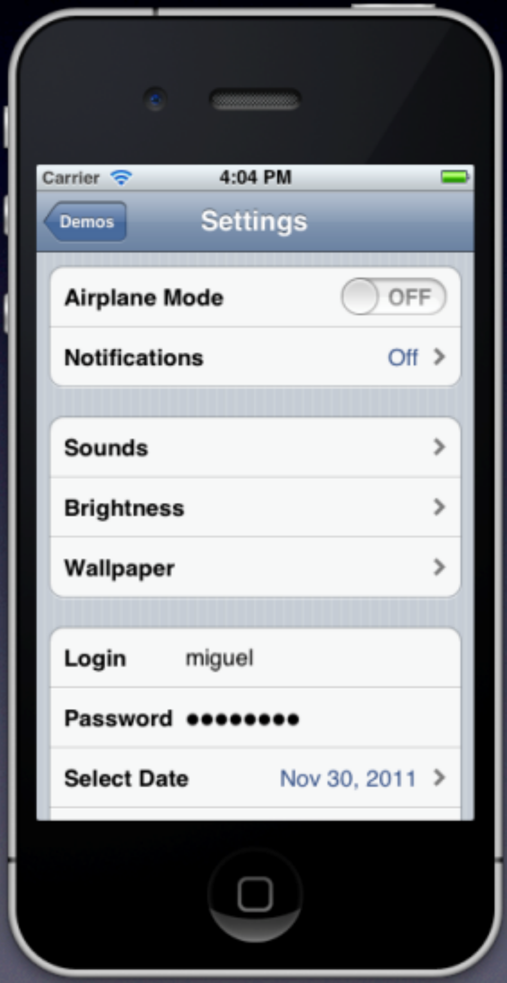




Speedy UI Development

- MonoTouch.Dialog
- Encapsulates use of UITableView
- Simple table UI elements
- Hides away UITableViewSource, etc.

Simplifies UI like this



Carrier 4:04 PM

Demos — Settings

Airplane Mode OFF

Notifications Off

Sounds

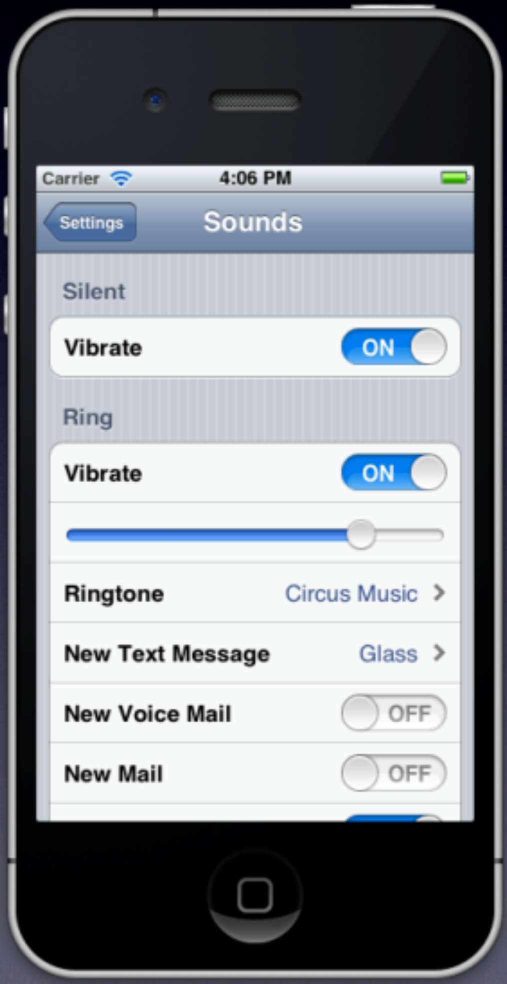
Brightness

Wallpaper

Login miguel

Password ●●●●●●●●

Select Date Nov 30, 2011



Carrier 4:06 PM

Settings — Sounds

Silent

Vibrate ON

Ring

Vibrate ON

Ringtone Circus Music

New Text Message Glass

New Voice Mail OFF

New Mail OFF

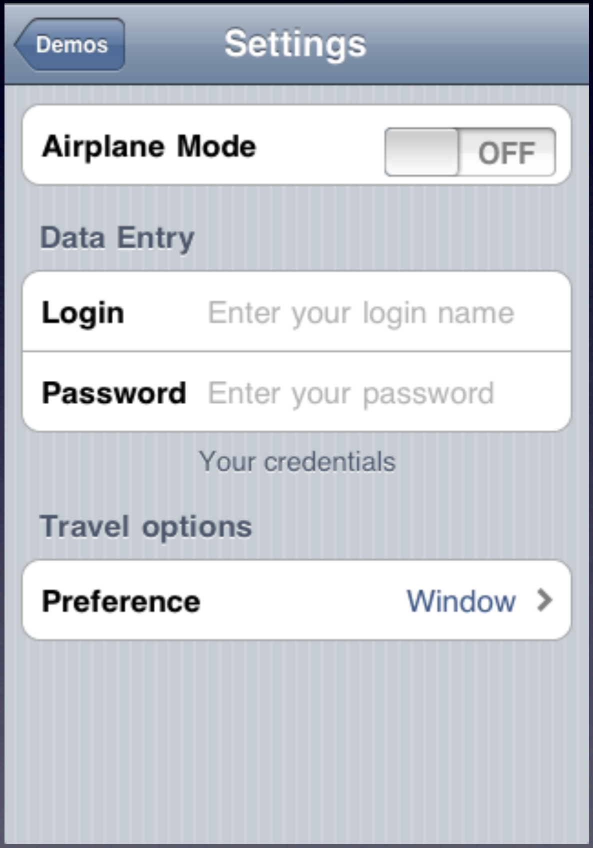
API

- MonoTouch.Dialog is a framework that brings declarative UI programming to iOS

```
return new RootElement ("Settings") {
    new Section (){
        new BooleanElement ("Airplane Mode", false),
        new RootElement ("Notifications", 0, 0) {
            new Section (null,
                 "Turn off Notifications...n" +
                     "Alerts and ....")
            {
                new BooleanElement ("Notifications",
false)
            }
        }},
```

Second API

- using reflection and attributes


Demos Settings

Airplane Mode OFF

Data Entry

Login Enter your login name

Password Enter your password

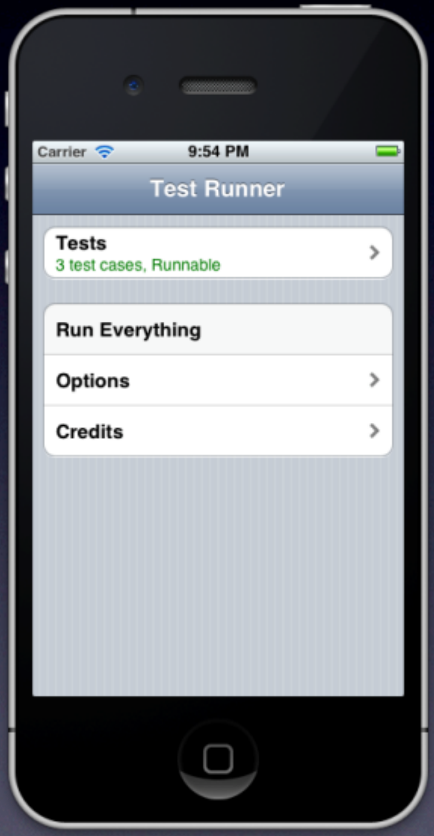
Your credentials

Travel options

Preference Window

```
class AccountInfo {
    [Section]
    public bool AirplaneMode;

    [Section ("Data Entry", "Your credentials")]

    [Entry ("Enter your login name")]
    public string Login;

    [Caption ("Password"),
    Password ("Enter your password")]
    public string passwd;

    [Section ("Travel options")]
    public SeatPreference preference;
}
```

Unit Testing Apps



- NUnit Lite Runner on device/ simulator
- Same NUnit syntax you may already know from C# land
- Possible to run the same NUnit tests on desktop, iOS and WP7 device

Performance Analysis

Improved memory management to find, diagnose and fix memory leaks

- New MonoTouch memory profiler
  - Track managed objects' memory usage growth
  - Track which objects are still referenced
  - Track where objects are being referenced from
- Generational garbage collector
- Improves garbage collection diagnostics

- More info: New MonoTouch memory profiler and Generational garbage collector

How I Develop

- Create project on Mac
- Submit to local git repo, shared by git-o-lite
- Pull from repo on PC
- Use VisualStudio 2012 with VSMonoTouch plugin
- Enjoy ReSharper: the best in refactoring, unit testing code clean up, etc.
- Do lots of unit testing
- Commit back to Mac for UI testing
- Lather, rinse, repeat

Limitations

- Generics and JIT issues
- I use generics ALOT. Combine this with lots of interfaces and you have problems.
- Issue is that compiler does't know to instantiate generic till execution reaches it
- No worries for class objects as code is all the same (ala type erasure in Java)
- Struct objects need to be unique code instantiation...compiler may note and you'll get a dreaded "JIT exception error"

Limitations

- In practice it may not cause issues
- For me, it does... too many JIT errors
- If you aren’t a big user of layering of interfaces and generics, likely little to no issue

Pros

- Fast to develop
- Wealth of C# libraries to draw on
- Debugging is much nicer with ability to "see inside" objects more clearly
- Xamarin is very quick at responding to tech issues
- Updated releases for new versions of iOS typically available in release from the day after Apple releases, betas much earlier

Summary

C# for iOS

- Makes iOS easily accessible for .NET developers.
- Thin layer on top of CocoaTouch – same native look & feel
- MonoTouch.Dialog for easier UI creation

Cross Platform

- Standard .NET libraries for tasks such as:
  - File Access
  - Database Access
  - Web Service Access
  - Business Logic
- Xamarin.Mobile provides same API for common phone functionality

Moral of the story

- If you like Objective-C, just ignore this
- If your projects are small and only iOS, ignore this
- If you are building cross-platform mobile apps, don;t like Objective-C or are a fan of #, then you should definitely take a look.

Resources

Xamarin
http://www.xamarin.com

Xamarin Docs
http://docs.xamarin.com/ios
http://docs.xamarin.com/android

Where To Find Me

Web: http://www.pillowsoft.com
Email bpillow@pillowsoft.com
GitHub: https://github.com/pillowsoft
LinkedIn: http://www.linkedin.com/pub/brad-pillow/0/7a/810

Questions?

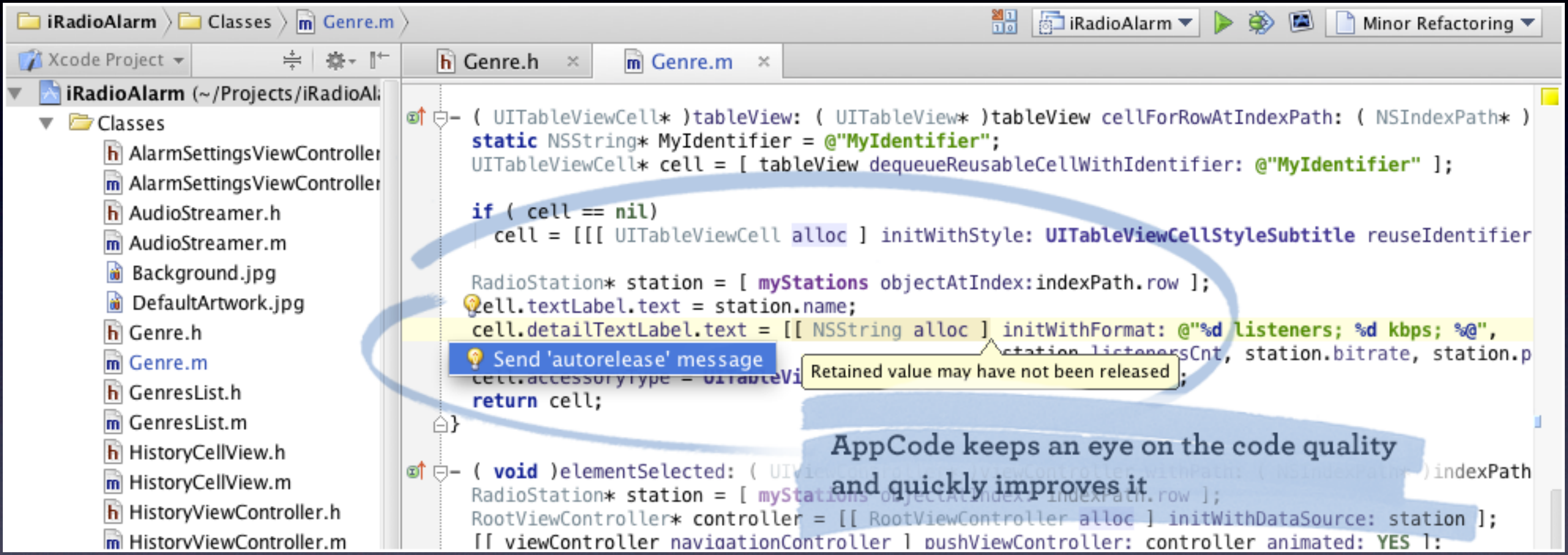
- Presentation will be posted to GitHub
- Search on my GitHub Favorites by C# to find interesting samples

Bonus Items

- PaintCode - http://www.paintcodeapp.com/
- ReSharpers cousin :AppCode
- PhoneGap

AppCode

http://www.jetbrains.com/objc/



```
- ( UITableViewCell* )tableView: ( UITableView* )tableView cellForRowAtIndexPath: ( NSIndexPath* )
    static NSString* MyIdentifier = @"MyIdentifier";
    UITableViewCell* cell = [ tableView dequeueReusableCellWithIdentifier: @"MyIdentifier" ];

    if ( cell == nil)
      cell = [[[ UITableViewCell alloc ] initWithStyle: UITableViewCellStyleSubtitle reuseIdentifier

    RadioStation* station = [ myStations objectAtIndex:indexPath.row ];
    cell.textLabel.text = station.name;
    cell.detailTextLabel.text = [[ NSString alloc ] initWithFormat: @"%d listeners; %d kbps; %@",
                                                     station.listenersCnt, station.bitrate, station.p
    cell.accessoryType = UITableVi                                          ;
    return cell;
}

- ( void )elementSelected: ( UIViewController* )viewController withPath: ( NSIndexPath* )indexPath
    RadioStation* station = [ myStations objectAtIndex: indexPath.row ];
    RootViewController* controller = [[ RootViewController alloc ] initWithDataSource: station ];
    [[ viewController navigationController ] pushViewController: controller animated: YES ];
```

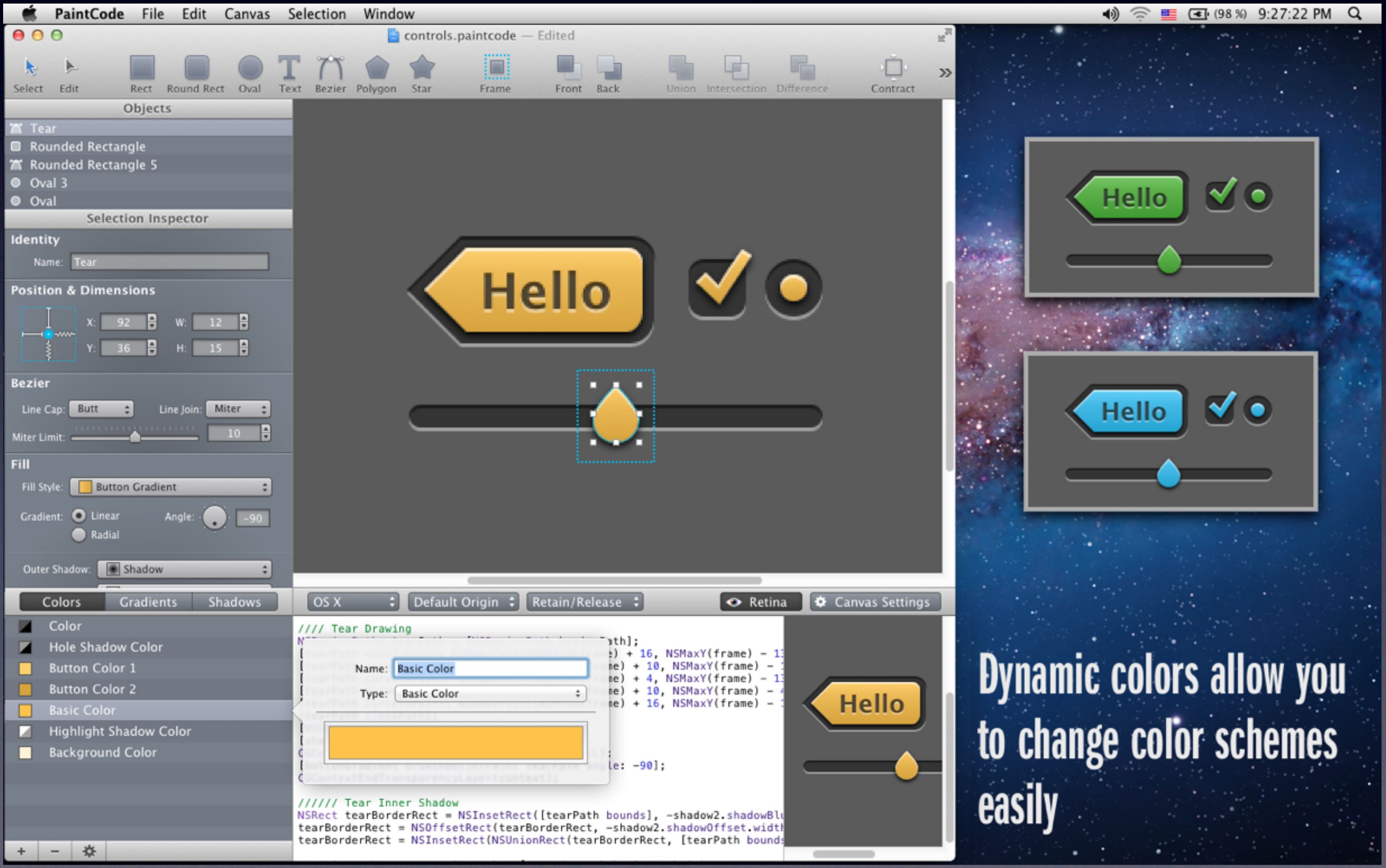
Retained value may have not been released

- Built using Java
- Has all the ReSharper tech applied to Obj-C...
- Personal License: $99

PaintCode

- http://www.paintcodeapp.com/

$99


PaintCode File Edit Canvas Selection Window
9:27:22 PM
controls.paintcode — Edited
Select Edit Rect Round Rect Oval Text Bezier Polygon Star Frame Front Back Union Intersection Difference Contract
Objects
Tear
Rounded Rectangle
Rounded Rectangle 5
Oval 3
Oval
Selection Inspector
Colors Gradients Shadows
Color
Hole Shadow Color
Button Color 1
Button Color 2
Basic Color
Highlight Shadow Color
Background Color
Hello
Canvas Settings
Dynamic colors allow you to change color schemes easily

Identity
Name: Tear

Position & Dimensions
X: 92 W: 12
Y: 36 H: 15

Bezier
Line Cap: Butt Line Join: Miter
Miter Limit: 10

Fill
Fill Style: Button Gradient
Gradient: [✓ Linear] Radial Angle: -90
Outer Shadow: Shadow

OS X | Default Origin | Retain/Release | Retina

Name: Basic Color
Type: Basic Color

```
//// Tear Drawing
N                                                        th];
[                                         ) + 16, NSMaxY(frame) - 13
[                                         e) + 10, NSMaxY(frame) - 1
[                                         e) + 4, NSMaxY(frame) - 13
[                                         e) + 10, NSMaxY(frame) - 4
[                                         e) + 16, NSMaxY(frame) - 1

[
[
C
[                                         le: -90];
C

////// Tear Inner Shadow
NSRect tearBorderRect = NSInsetRect([tearPath bounds], -shadow2.shadowBlu
tearBorderRect = NSOffsetRect(tearBorderRect, -shadow2.shadowOffset.width
tearBorderRect = NSInsetRect(NSUnionRect(tearBorderRect, [tearPath bounds
```

PhoneGap

Free! www.phonegap.com

- Uses shell app and web view for javascript handling
- Provides bridge to/fro Obj-C
  - Use javascript, typescript, dart, coffeescript, etc. to make cross platform mobile apps
    - Performance issues
      - Native calls
      - Direct Canvas

Credits

Content for this borrowed from:

- Easily create iOS user interfaces with MonoTouch.Dialog
- C# on the iPhone with MonoTouch - Chris Hardy
- Introduction to MonoTouch - Mike Bluestein
- Easier development of iOS applications using C# - Jonas Follesø
- MonoTouch and Mono for Android - Chris Hardy
- An Introduction to MonoTouch 5.2 Mobile App Development - Xamarin