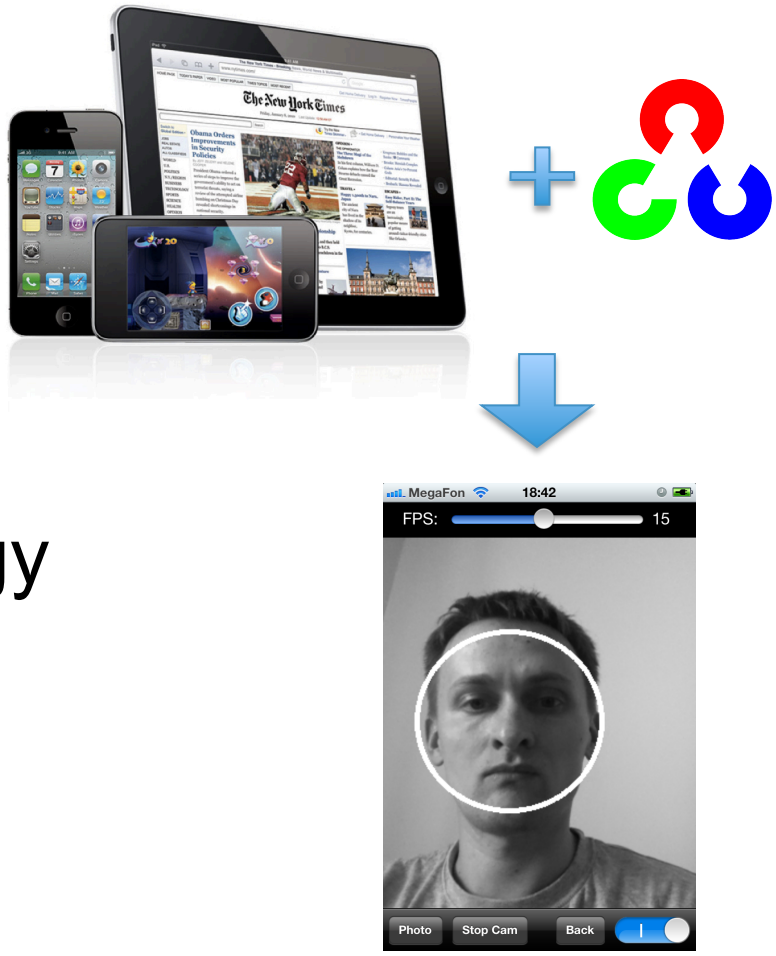
OpenCV for iOS

CVPR 2012

Getting your CV algorithm up and running on iOS

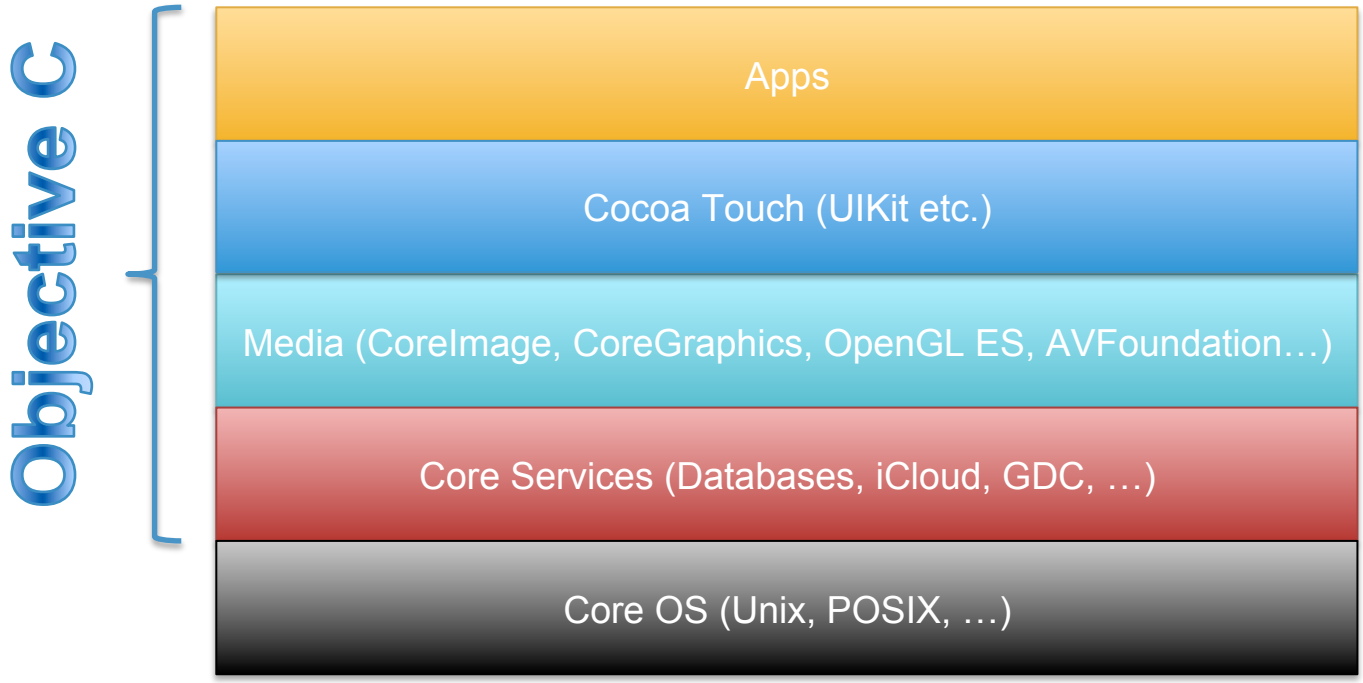
- It's fun
- It's trendy
- It can make you rich
- It's impressive way to show some technology



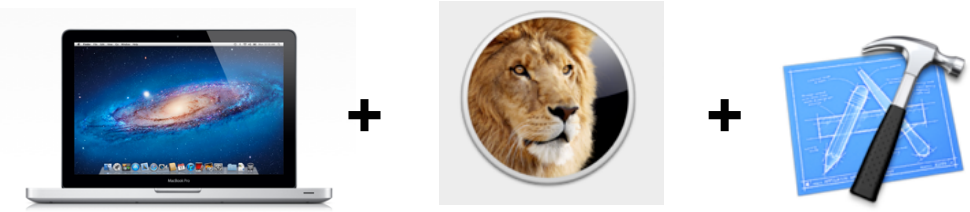
Plan for today

- Intro; iOS Simulator vs iOS Device.
- Objective-C
- Make iPhone say “Hello!”
- Make OpenCV say “Hello!” (on iPhone)
- Processing photos you shoot with iPhone.
- Processing live video
- Performance tips & tricks
- References

iOS Anatomy


Objective C
Apps
Cocoa Touch (UIKit etc.)
Media (CoreImage, CoreGraphics, OpenGL ES, AVFoundation…)
Core Services (Databases, iCloud, GDC, …)
Core OS (Unix, POSIX, …)

iOS Development

iOS Simulator: 

iOS Device: 

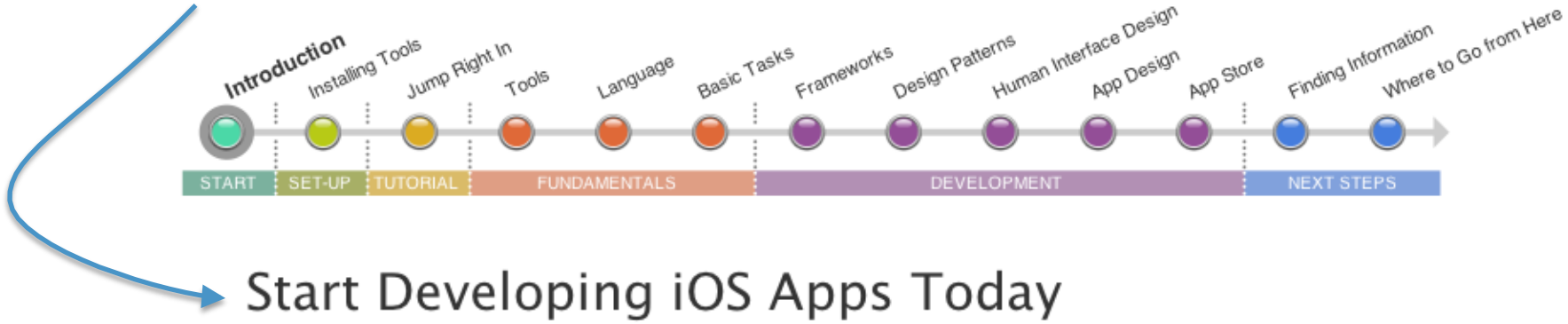
Device should be "provisioned" – a.k.a. registered to be used with the particular developer certificate

Environment setup tutorials at docs.opencv.org.

	iOS Simulator	iOS Device
Cost	Free*	$99 per year
Prerequisities	OSX 10.7 + Xcode 4.3.x or later	… + iOS-5 capable device
Target CPU	x86 (SSE, no Neon)	ARM (Neon, no SSE)
Speed	fast	slower
RAM	lot's of	limited
OpenGL	+	+
Photo Library, Still Photo processing	+	+
Camera	-	+
Accelerometer	-	+
Debugging, Logging	+	+
Performance evaluation	-	+

Getting started

- Register at developer.apple.com (free)
- Mac + Xcode is enough to start: download myriads of samples from developer.apple.com
- Enroll yourself to iOS development program (annual fee)
- Create your certificate, put to your computer(s)
- Provision your device(s)
- Google for and read this



iOS Language: Objective C

- Objective C = Smalltalk + C
  - Apple's take: Objective C++ (can use classes, STL etc.)
- Since iOS 5 includes ARC (automatic reference counting)
- Dynamic – methods are called "by name", not by indices in vtable
- Has very rich class library with dedicated root object NSObject
- .h extension for headers, .m & .mm extension for implementation

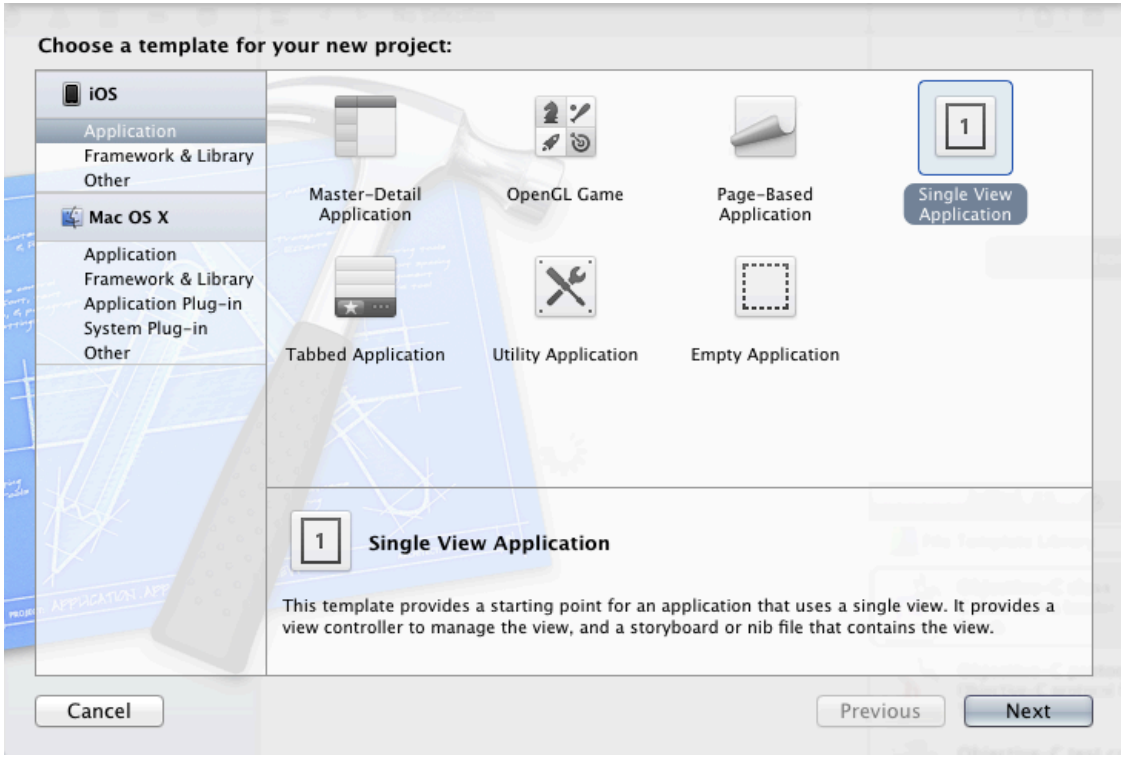
Hello world in Objective-C

```
#import <Foundation/Foundation.h>
int main(int argc, char *argv[])
{
  //printf("\n\nHello, world!\n");
  @autoreleasepool {
  NSLog([[NSString stringWithUTF8String:"\n\nHello"]
        stringByAppendingString:@", world!"]);
  return 0; }
}
```

- It's a superset of C
- #import == #include with automatic "guards".
- [recipient messageWithArg1:A andArg2:B…];
- Since OSX 10.7 & iOS 5 memory is automatically managed

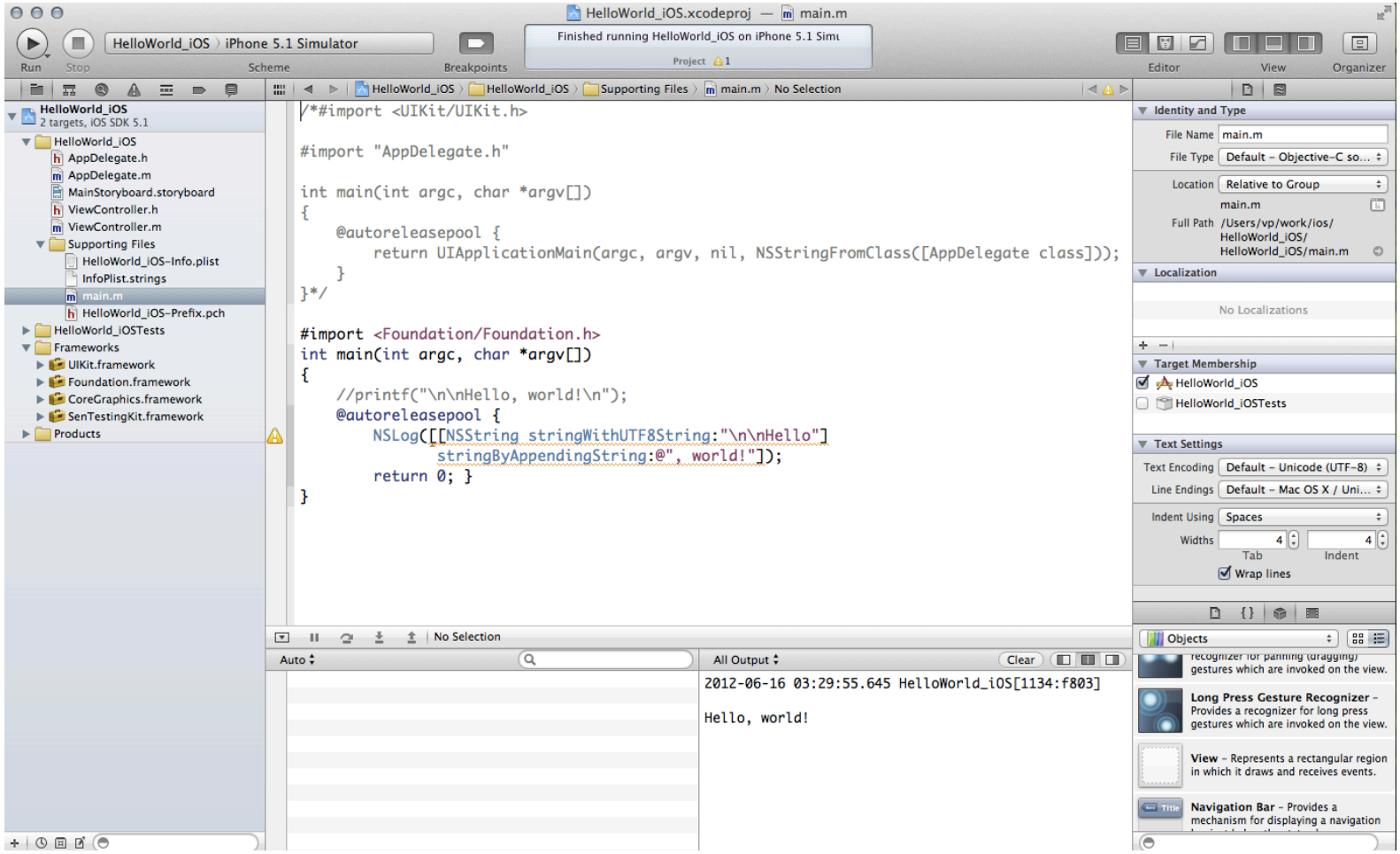
Hello world in iOS (1/16)

Create new “Single View Application”

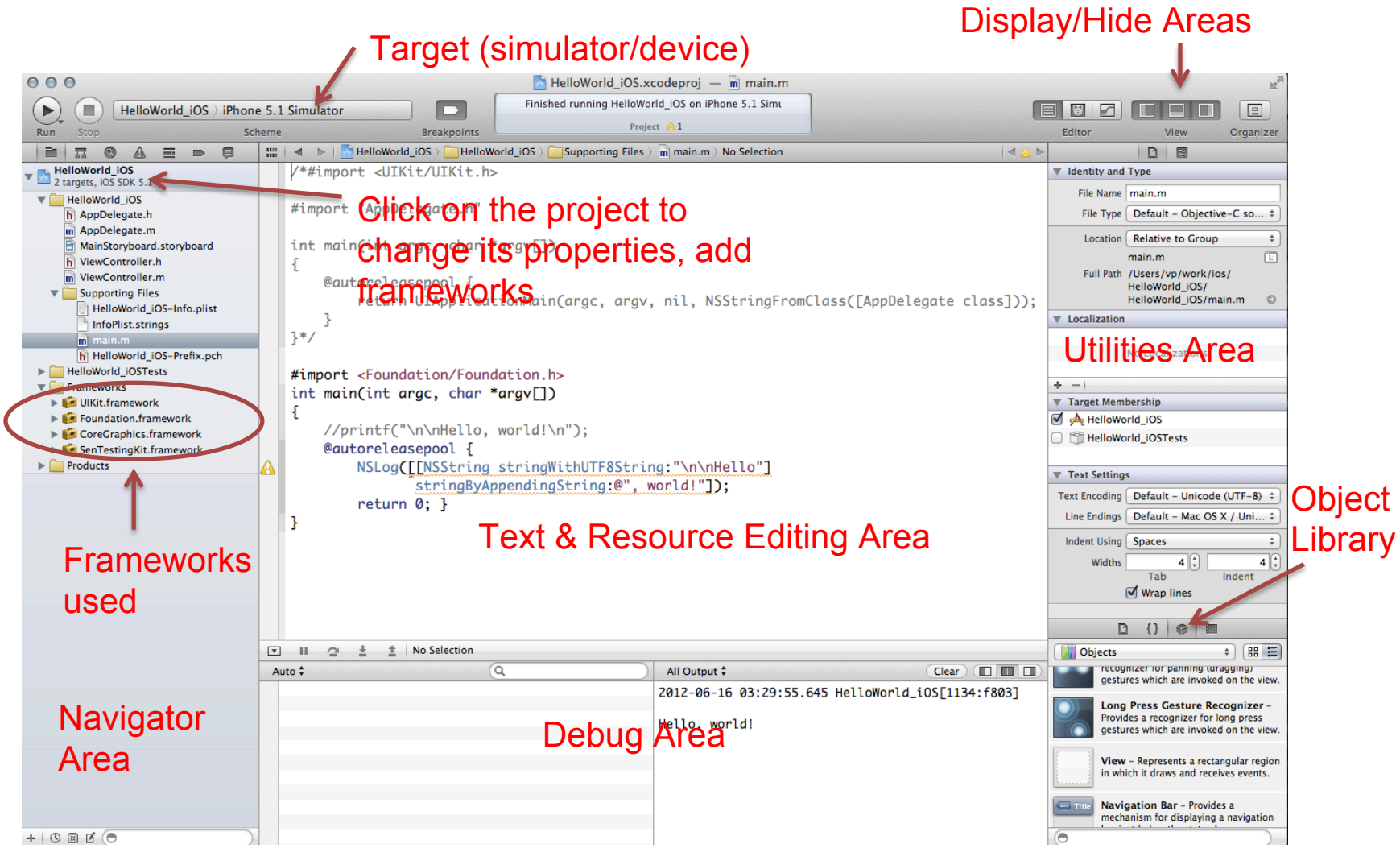


Hello world in iOS (2/16)

Open main.m, comment off original code and put in the “hello world” code shown before. Run



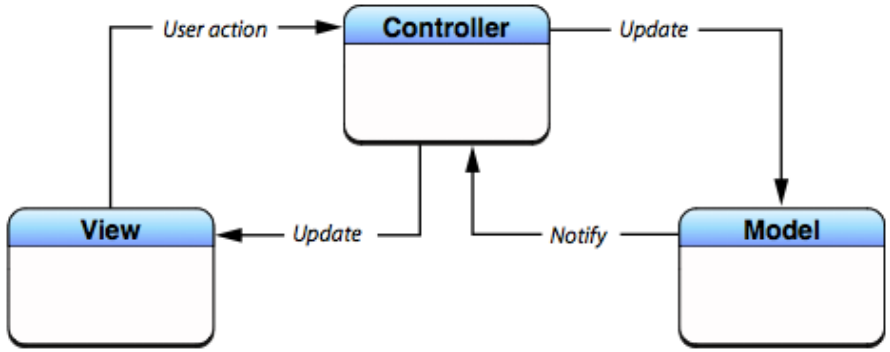
Xcode 4: iOS IDE



Cocoa(Touch) key concept: MVC

Model-View-Controller

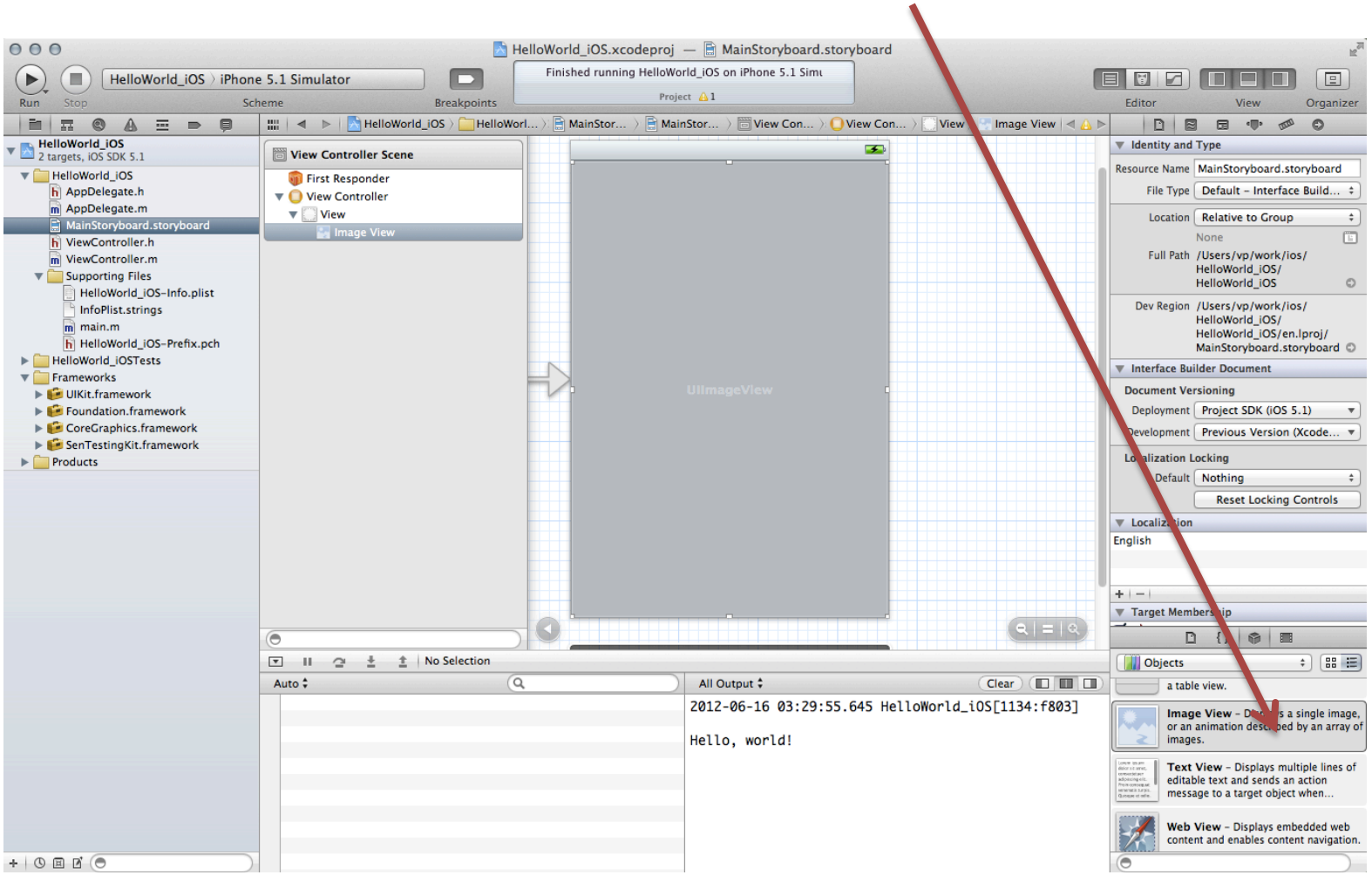
- For simple demos Model is a part of Controller class.



- View is created visually as a **Storyboard** element
- For each view component (image view, button, slider …) we add corresponding **IBOutlet**-marked property in the controller class
- For each action we add **IBAction**-marked method in the controller

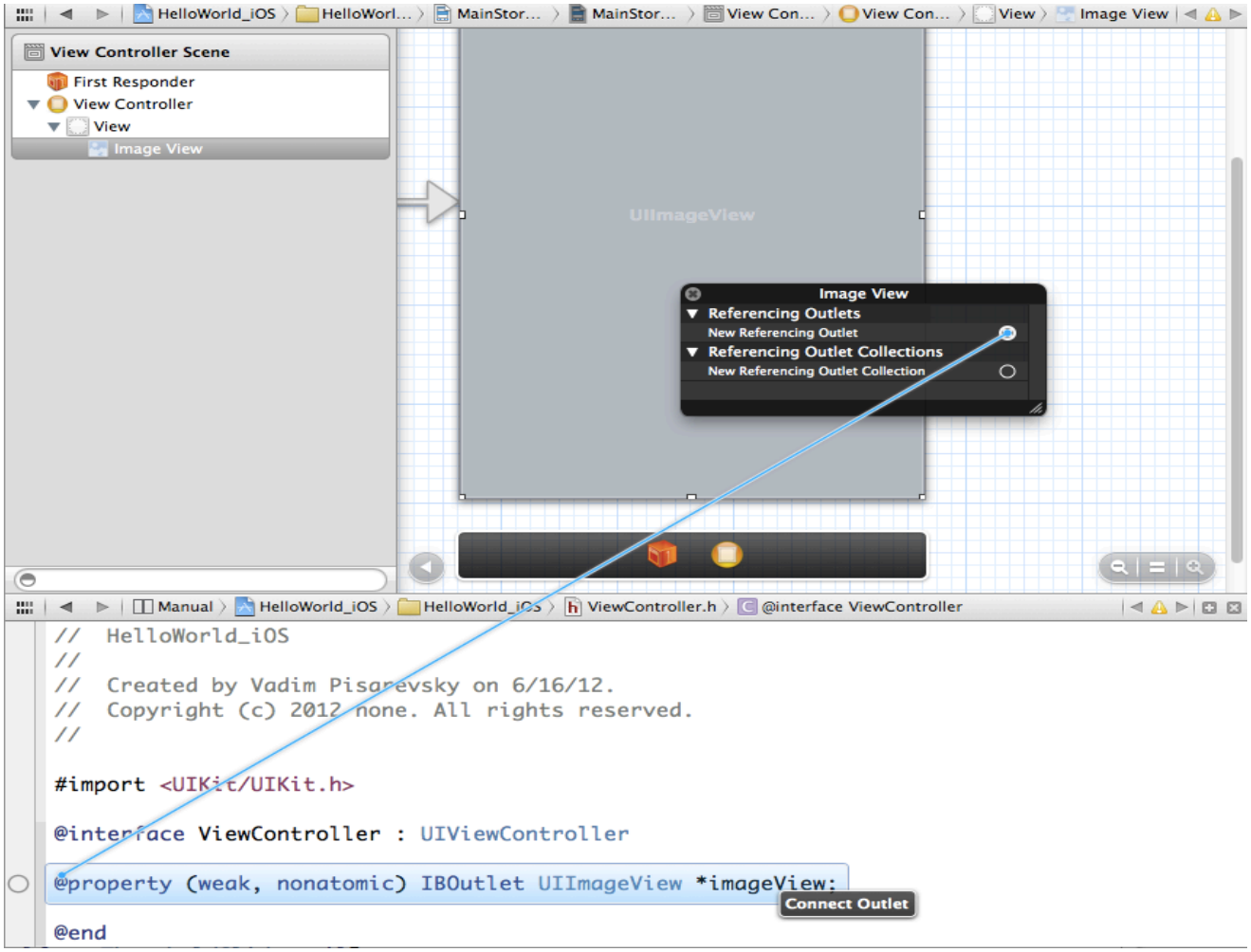
Hello world in iOS (3/16)

Restore main.m. Open the main storyboard, add image view



Hello world in iOS (4/16)

Open Assistant Editor (in Xcode View menu). Open ViewController.h, add imageView Outlet property, connect it with the view in StoryBoard.



Class declaration in Objective-C

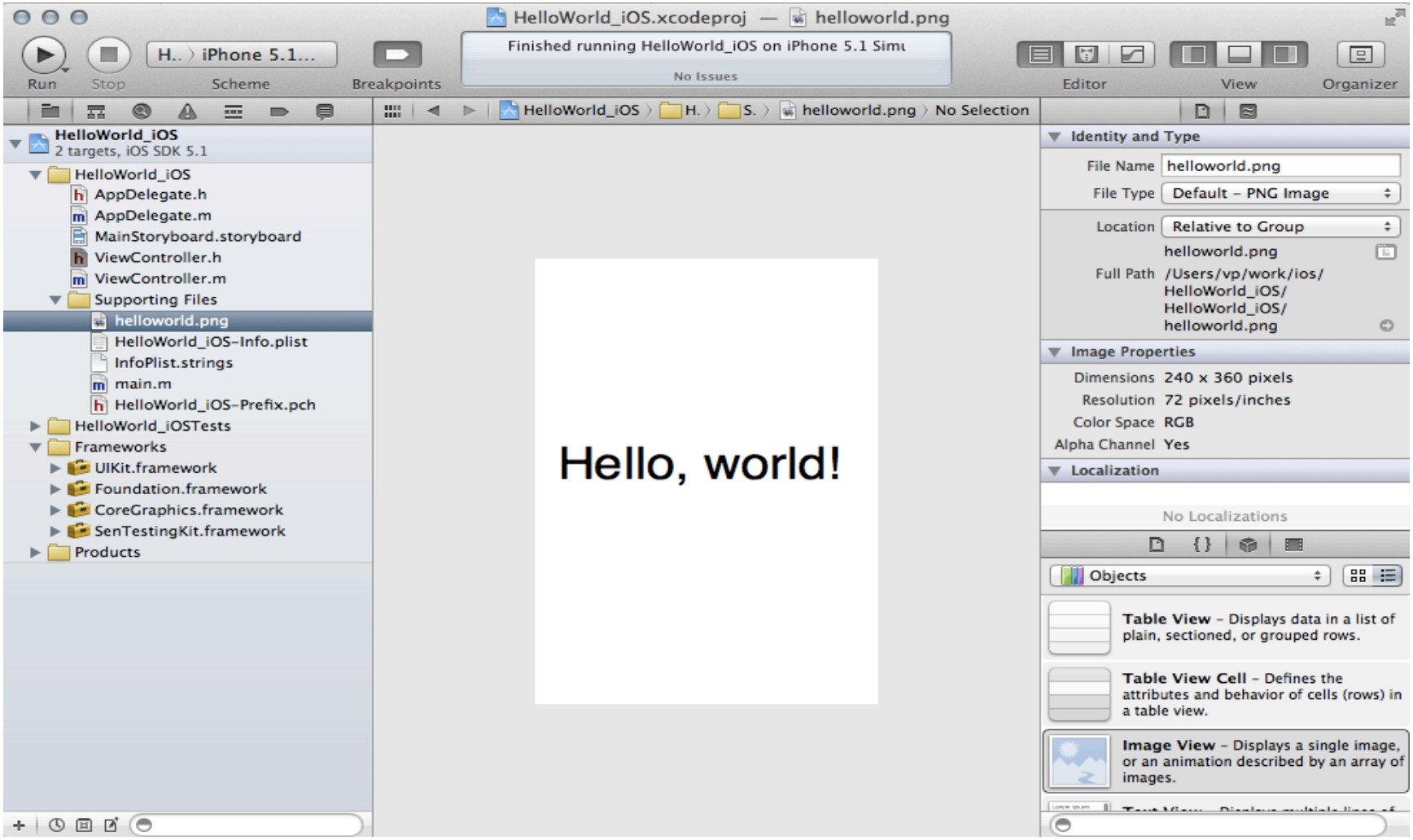
```
#import <UIKit/UIKit.h>
@interface ViewController : UIViewController {
    UIImage* image;
}
@property (weak, nonatomic) IBOutlet UIImageView *imageView;
@property (weak, nonatomic) IBOutlet UIBarButtonItem *loadButton;
-(IBAction)effectButtonPressed:(id)sender;
@end
```

- @interface … @end delimits class declaration
- there must be one and only one base class (UIViewController in our sample)
- data members are declared inside {}.
- @property declares a property, IBOutlet qualifier denotes outlets.
- Instance methods are declared as

  – (<returntype>) <methodnamepart1>: (<arg1type>)arg1 [<methodnamepath2>:…];
- No need to declare overloaded methods, only the newly added
- Actions have IBAction return type

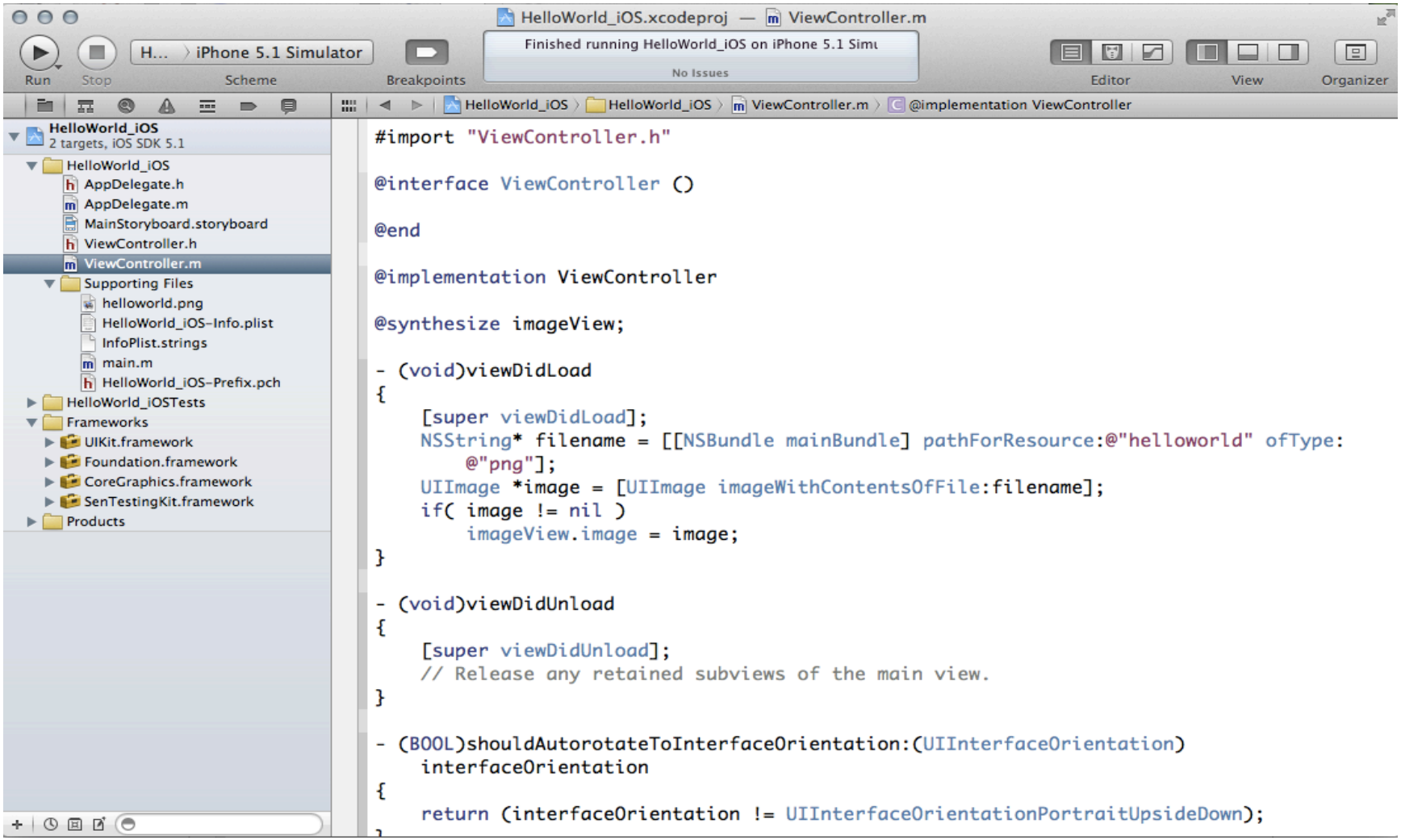
Hello world in iOS (5/16)

Create image (e.g. white rect with text “Hello, world!”), add it to the project



Hello world in iOS (6/16)

Open ViewController.mm; synthesize imageView property; add code to load and display image on startup.



Class implementation in Objective-C

```
#import “ViewController.h”
@implementation ViewController
@synthesize imageView;
- (void)viewDidLoad {
    [super viewDidLoad]; // call the superclass method
    NSString* filename = [[NSBundle mainBundle]
        pathForResource:@"helloworld" ofType:@"png"];
    UIImage *image = [UIImage imageWithContentsOfFile:filename];
    if( image != nil ) imageView.image = image;
}
…
@end
```

- @implementation … @end delimits class implementation
- @synthesize automatically generates **correct** code for reading and writing properties.
- “- viewDidLoad” is the place to put additional initialization. Xcode generates stub for it.
- Some methods, including viewDidLoad, require to call the original superclass method.
- Local variables are automatically destructed if they are not needed anymore
- To show the image, just assign it to the image property of UIImageViewer.

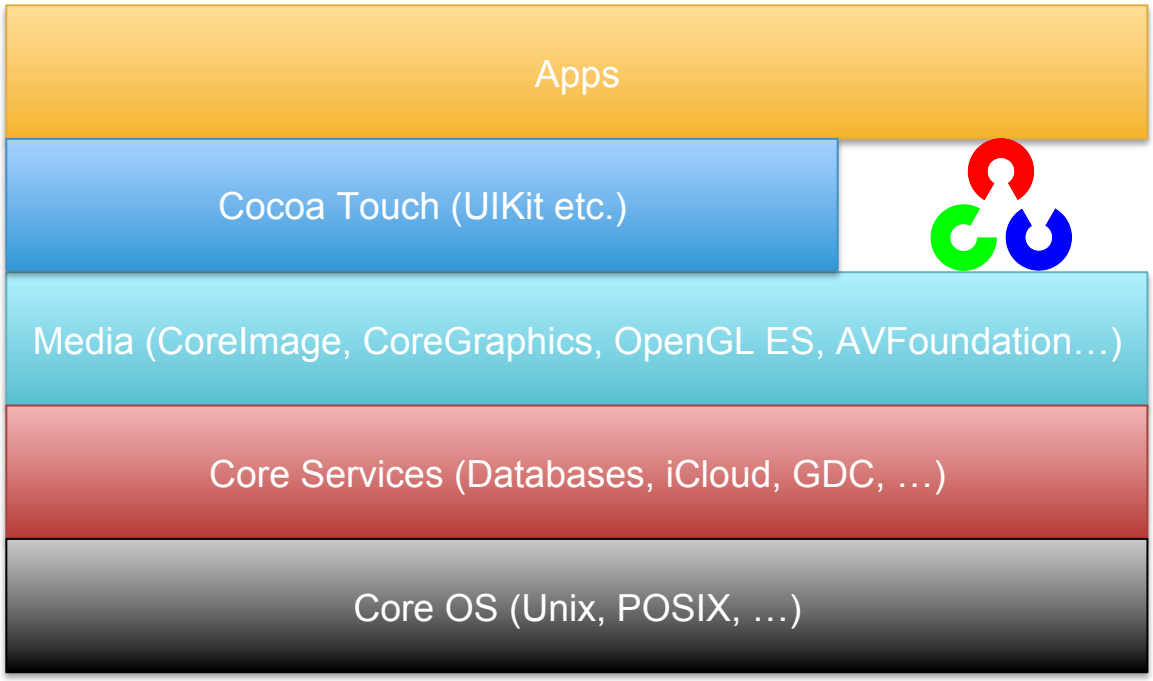
Finally, run it

Let’s now add OpenCV!

OpenCV for iOS

- Started with GSoC 2011 project. Sort of “alpha stage” now.
- Thanks to C++ ⇔ Objective C interoperability native OpenCV for iOS “just works”.
- opencv2.framework for simpler integration
- No ARM/iOS-specific acceleration yet ☹ (planned for OpenCV 2.5)
- No camera/video support within Highgui – use iOS facilities.

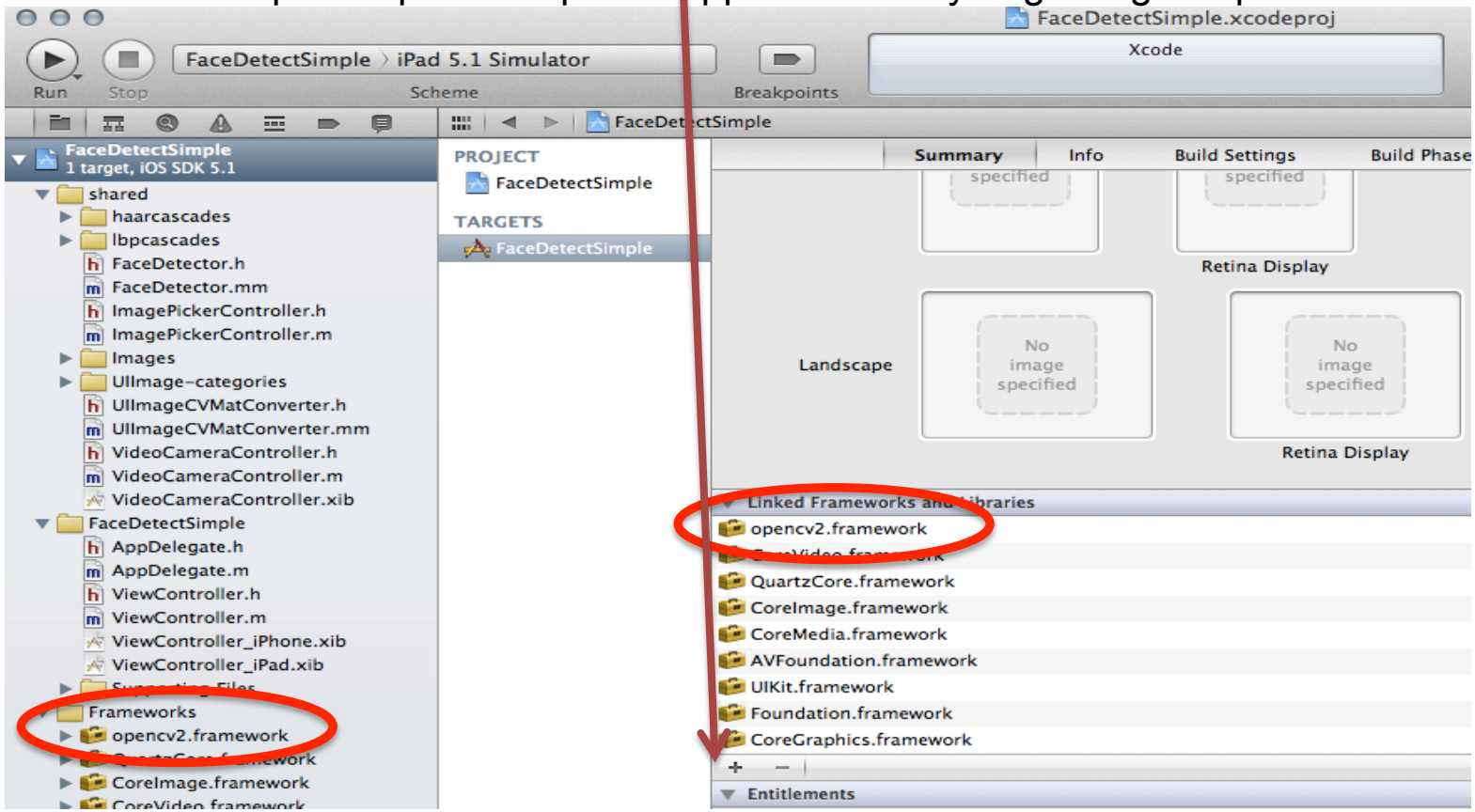
OpenCV+iOS Anatomy



OpenCV on iOS is just another framework

Hello world in iOS (7/16)

1. Build framework from OpenCV source:
   http://docs.opencv.org/svn/tutorials
2. Add framework to your project
3. Change project language to Objective C++.
4. Add #import <opencv2/opencv.hpp> to the very beginning of *.pch file.



Converting images: UIImage⇔cv::Mat

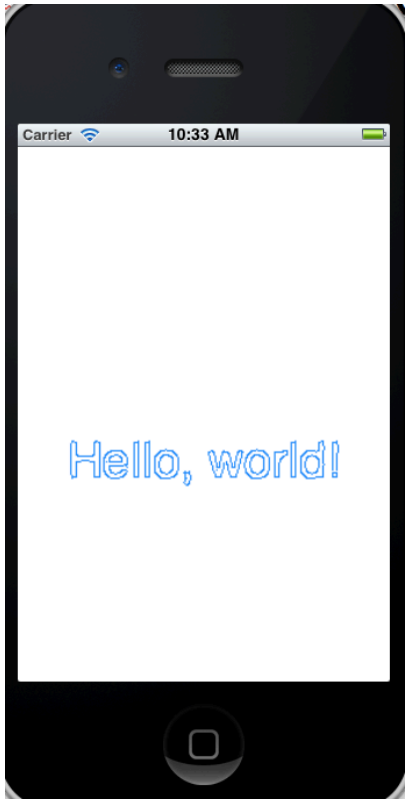
```
static UIImage* MatToUIImage(const cv::Mat& m) {
    CV_Assert(m.depth() == CV_8U);
    NSData *data = [NSData dataWithBytes:m.data length:m.elemSize()*m.total()];
    CGColorSpaceRef colorSpace = m.channels() == 1 ?
        CGColorSpaceCreateDeviceGray() : CGColorSpaceCreateDeviceRGB();
    CGDataProviderRef provider = CGDataProviderCreateWithCFData((__bridge CFDataRef)data);
    // Creating CGImage from cv::Mat
    CGImageRef imageRef = CGImageCreate(m.cols, m.cols, m.elemSize1()*8, m.elemSize()*8,
        m.step[0], colorSpace, kCGImageAlphaNoneSkipLast|kCGBitmapByteOrderDefault,
        provider, NULL, false, kCGRenderingIntentDefault);
    UIImage *finalImage = [UIImage imageWithCGImage:imageRef];
    CGImageRelease(imageRef); CGDataProviderRelease(provider);
    CGColorSpaceRelease(colorSpace); return finalImage;
}

static void UIImageToMat(const UIImage* image, cv::Mat& m) {
    CGColorSpaceRef colorSpace = CGImageGetColorSpace(image.CGImage);
    CGFloat cols = image.size.width, rows = image.size.height;
    m.create(rows, cols, CV_8UC4); // 8 bits per component, 4 channels
    CGContextRef contextRef = CGBitmapContextCreate(m.data, m.cols, m.rows, 8,
        m.step[0], colorSpace, kCGImageAlphaNoneSkipLast | kCGBitmapByteOrderDefault);
    CGContextDrawImage(contextRef, CGRectMake(0, 0, cols, rows), image.CGImage);
    CGContextRelease(contextRef); CGColorSpaceRelease(colorSpace);
}
```

These are not super-fast functions; minimize such conversions

Hello world in iOS (8/16)

Open ViewController.mm; insert the conversion functions after import statement, modify viewDidLoad as shown

```
…
if( image != nil ) {
    cv::Mat m, gray;
    UIImageToMat(image, m);
    cv::cvtColor(m, gray, CV_RGBA2GRAY);
    cv::GaussianBlur(gray, gray, cv::Size(5, 5), 1.2, 1.2);
    cv::Canny(gray, gray, 0, 50);
    m = cv::Scalar::all(255);
    m.setTo(cv::Scalar(0, 128, 255, 255), gray);
    imageView.contentMode = UIViewContentModeScaleAspectFit;
    imageView.image = MatToUIImage(m);
}
}
```



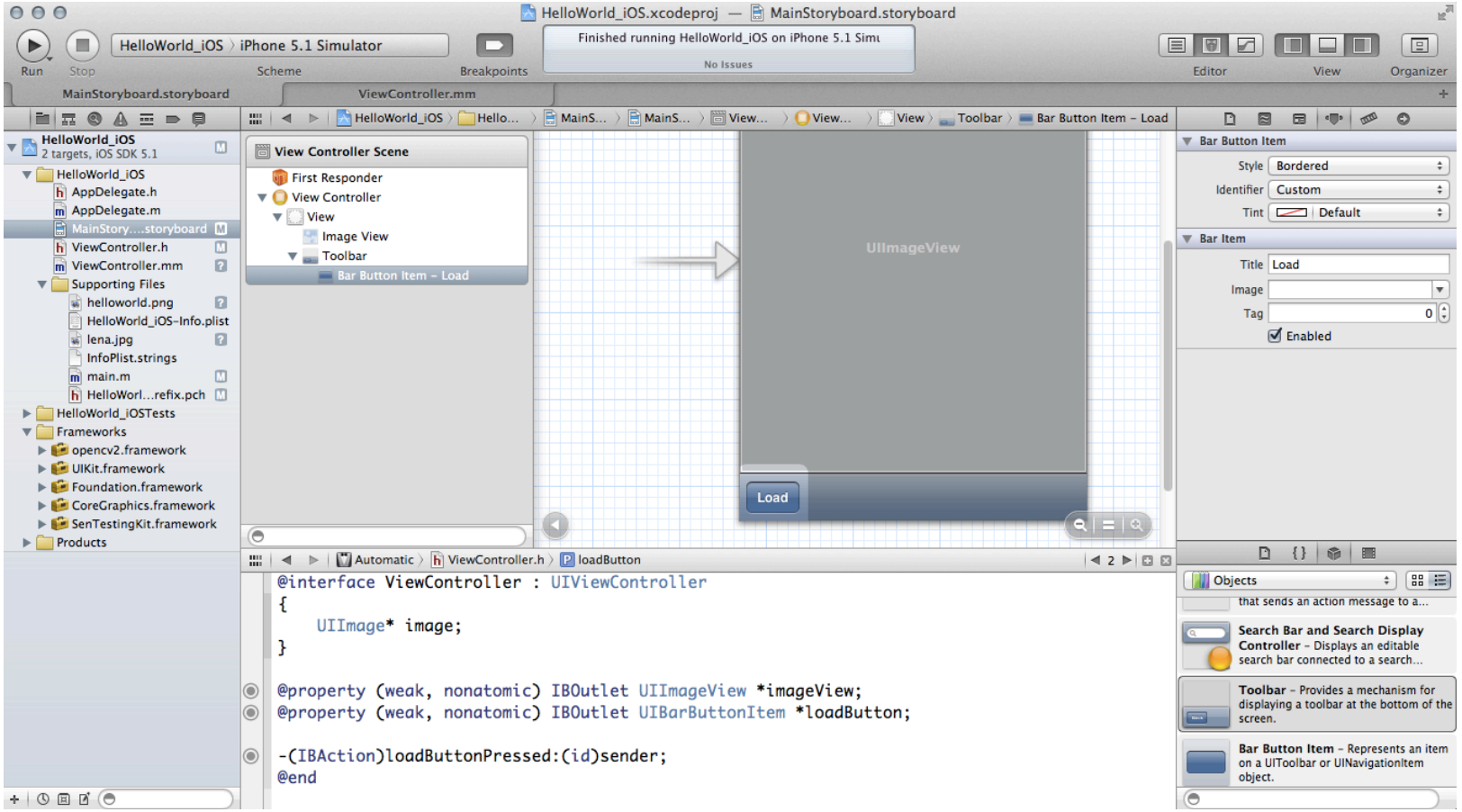
Capturing/Loading Still Images

Processing fixed image from app resources is not practical. How to load an arbitrary image?

1. Modify StoryBoard, add toolbar & “load” button. Add action for “button tapped” event.
2. Inside the callback open the photo library and let user select photo from there.
3. process the loaded photo and show the result.

Hello world in iOS (9/16)

Open StoryBoard, add toolbar (it will already contain a button), rename it, add the outlet and action for the button, connect them to button.



Hello world in iOS (10/16)

Open ViewController.mm; add the following code for loadButtonPressed action (did you forget to add another @synthesize for button outlet?)

```
- (IBAction)loadButtonPressed:(id)sender {
    UIImagePickerController* picker = [[UIImagePickerController alloc] init];
    picker.delegate = self;
    if (![UIImagePickerController isSourceTypeAvailable:
          UIImagePickerControllerSourceTypePhotoLibrary])
        return;
    picker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
    [self presentModalViewController:picker animated:YES];
}
```

If you try to build it, you will compile error.

“picker.delegate = self;” is Cocoa way of customizing behavior of standard classes. ViewController needs to conform to certain interfaces.

Hello world in iOS (11/16)

Fix ViewController interface

```
@interface ViewController : UIViewController
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>
{
    UIImage* image;
}
```

- Angle brackets are used in Objective-C to specify supported protocols (a.k.a. interfaces in Java)
- Instead of creating a class derived from UIImagePickerController, we set a delegate class, but the delegate must conform to the 2 protocols specified above. See http://en.wikipedia.org/wiki/Delegation_pattern for details
- We need to add 2 methods to implement the protocols

Hello world in iOS (12/16)

Fix ViewController implementation

```
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {
    [picker dismissModalViewControllerAnimated:YES];
    UIImage *temp = [info objectForKey:@"UIImagePickerControllerOriginalImage"];
    imageView.image = processWithOpenCV(temp); }

- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker {
    [picker dismissModalViewControllerAnimated:YES]; }
```

- As in the case of overridden methods, there is no need to declare the protocol methods – just implement them.
- processWithOpenCV() is our function in which we copied the processing part of didLoad method.
- The delegate can be any class, not necessarily the ViewController. So it's possible to create black-box component for faster prototyping.

Let’s run it!

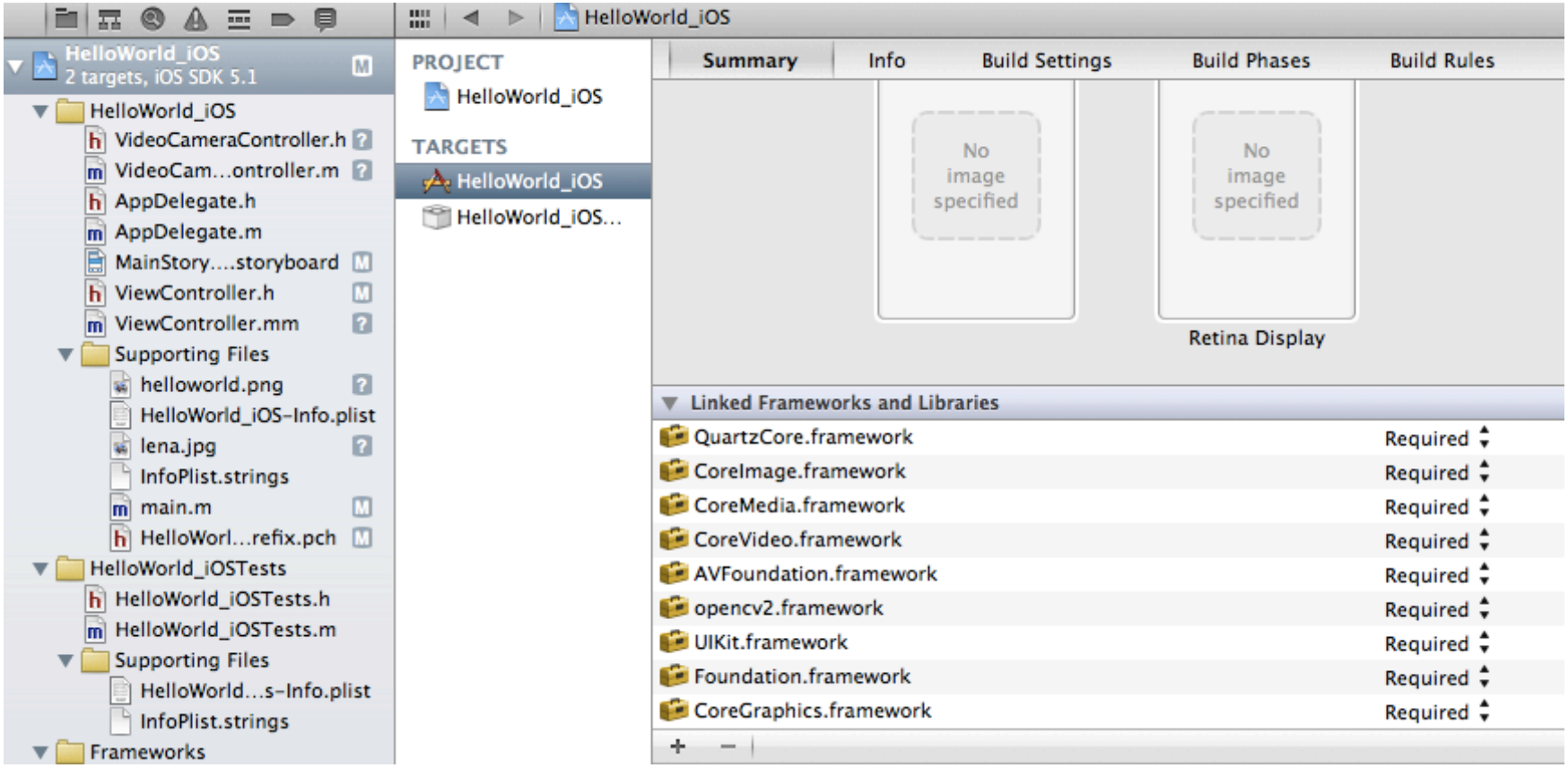
Tip: If you run it for the first time, the photo library will be empty. You can drag image to the simulator, which will display it in Safari. Press mouse/ trackpad for 1-2sec until the menu arrives. Save the image to library.

Adding Camera input

- Camera is not supported by Simulator ☹
- Setting up camera, retrieving frames, displaying them, handling rotations etc. takes a lot of code.
- We'll re-use component created by our GSoC student Eduard: VideoCameraController.
- It's also built using delegate pattern. The delegate will be our good old ViewController.
- Camera grabs video in YUV4:2:0 format. We'll only process and display luminance plane.

Hello world in iOS (13/16)

Copy VideoCameraController.h/.m to the project directory and add them to project. Add frameworks QuartzCore, CoreImage, CoreMedia, CoreVideo.



Hello world in iOS (14/16)

Update ViewController interface part

```
#import <UIKit/UIKit.h>
#import "VideoCameraController.h"

@interface ViewController : UIViewController <UIImagePickerControllerDelegate,
UINavigationControllerDelegate, VideoCameraControllerDelegate>
{
    UIImage* image;
    VideoCameraController* videoCamera;
}
…
```

- #import can be used with your own files and save you from writing extra #ifndef #define #endif.
- Protocols and delegates are not exclusive things in standard Cocoa components; well-designed components may introduce custom protocols and use delegates. See VideoCameraController.h/.m for details.
- ARC works with pointers to your own classes as well (e.g. with VideoCamera*)

Hello world in iOS (15/16)

Setup camera in ViewController didLoad method

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    videoCamera = [[VideoCameraController alloc] init];
    videoCamera.defaultAVCaptureSessionPreset = AVCaptureSessionPreset352x288;
    videoCamera.defaultFPS = 15;
    videoCamera.delegate = self;
    [videoCamera start];
}
```

- VideoCamera class is super-easy to use. Leave the default resolution, fps etc. or customize them as needed.
- Using lowest possible resolution and reasonable framerate can save a lot of power and make apps more responsive.

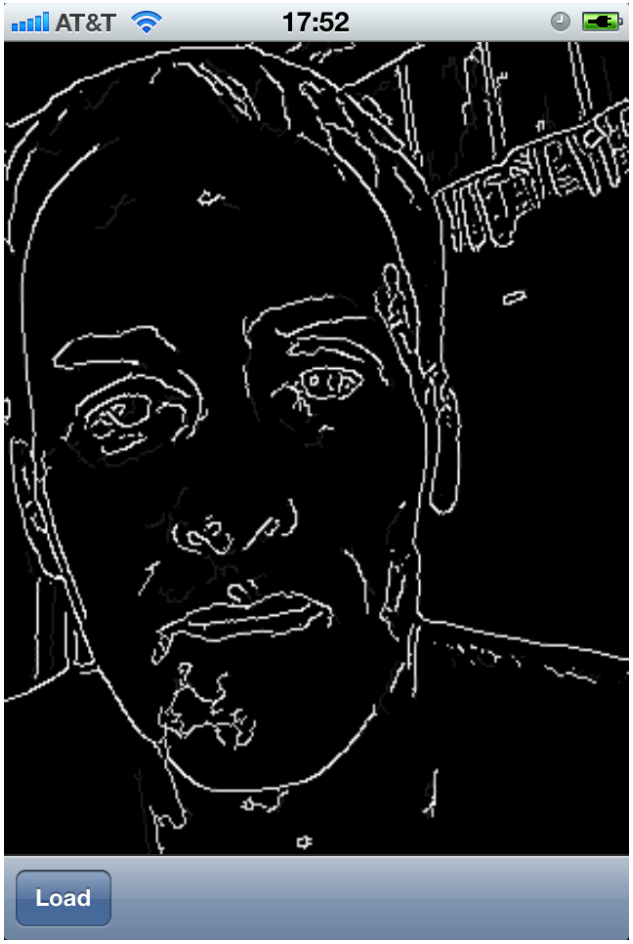
Hello world in iOS (16/16)

Implement VideoCameraControllerDelegate protocol

```
- (void)processImage:(const vImage_Buffer)imagebuf withRenderContext:
(CGContextRef)contextOverlay {
    cv::Mat gray((int)imagebuf.height, (int)imagebuf.width,
                 CV_8U, imagebuf.data, imagebuf.rowBytes);
    cv::GaussianBlur(gray, gray, cv::Size(5, 5), 1.2, 1.2);
    cv::Canny(gray, gray, 0, 30);
}
- (void)videoCameraViewController:
(VideoCameraController*)videoCameraViewController capturedImage:(UIImage
*)result { [self.imageView setImage:result];}


- (void)videoCameraViewControllerDone:
(VideoCameraController*)videoCameraViewController {}
- (BOOL)allowMultipleImages { return YES; }
- (BOOL)allowPreviewLayer { return NO; }
- (UIView*)getPreviewView { return imageView; }
```

- processImage:withRenderContext: should process image in-place.
- videoCameraViewController:capturedImage shows the result
- getPreviewView should return UIImageView where the live view is shown

The final app result



Tip: To make a screenshot of an app, press home button (at the bottom center) and without releasing it press exit button (right side of the top edg) for a short time.

Bonus track: Face Detection

Look at FaceDetectSimple app on the USB key and try to add the same functionality to our HelloWorld sample.

Hints:

- Add lbpcascade_frontalface.xml as a resource and load it in didLoad.
- You may need to transpose input image before running CascadeClassifier, because currently orientation is not handled properly. Then transpose the found face coordinates back to visualize results correctly.

General iOS performance tips

- Do expensive initialization once (e.g. in NSViewController's didLoad method)
- Setup your camera preview resolution and FPS to the lowest possible values
- Use Accelerate.framework
- Use OpenGL and CoreGraphics for rendering, CoreImage for image processing
- Use Grand Dispatch Central (implements data-level and task-level parallelism)
- Profile your apps using Xcode profiler
- Adv: Use Neon intrinsics & OpenGL ES shaders

Performance and power efficiency go together.

Using Accelerate Framework

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

...
cv::Mat src, dst;
...

dst.create(cvRound(src.rows*0.75), cvRound(src.cols*0.75), src.type());

vImage_Buffer srcbuf = { src.data, src.rows, src.cols, src.step };
vImage_Buffer dstbuf = { dst.data, dst.rows, dst.cols, dst.step };

vImageScale_Planar8( &srcbuf, &dstbuf, NULL, kvImageNoFlags );
```

- Formats: {ARGB, RGBA, Grayscale} x {byte, float}
- Functionality: geometrical transforms, non-separable linear filters, morphology, depth conversions, histogram equalization, LUT
- Likely, Neon-based implementation (no GPU?)
- For even better results use CoreImage and OpenGL ES
- Accelerate also includes veclib, vDSP, BLAS.

Resources

- http://developer.apple.com (including excellent introductory guides to Objective-C, Cocoa and iOS programming)
- http://docs.opencv.org/trunk/ (see iOS tutorial – to be greatly extended; we have dedicated GSoC 2012 project on OpenCV+iOS)
- http://code.opencv.org/svn/gsoc2012/ios/trunk - our GSoC iOS project repository where more and more interesting stuff is being added