UI Design

Mobile Application Development in iOS

School of EECS

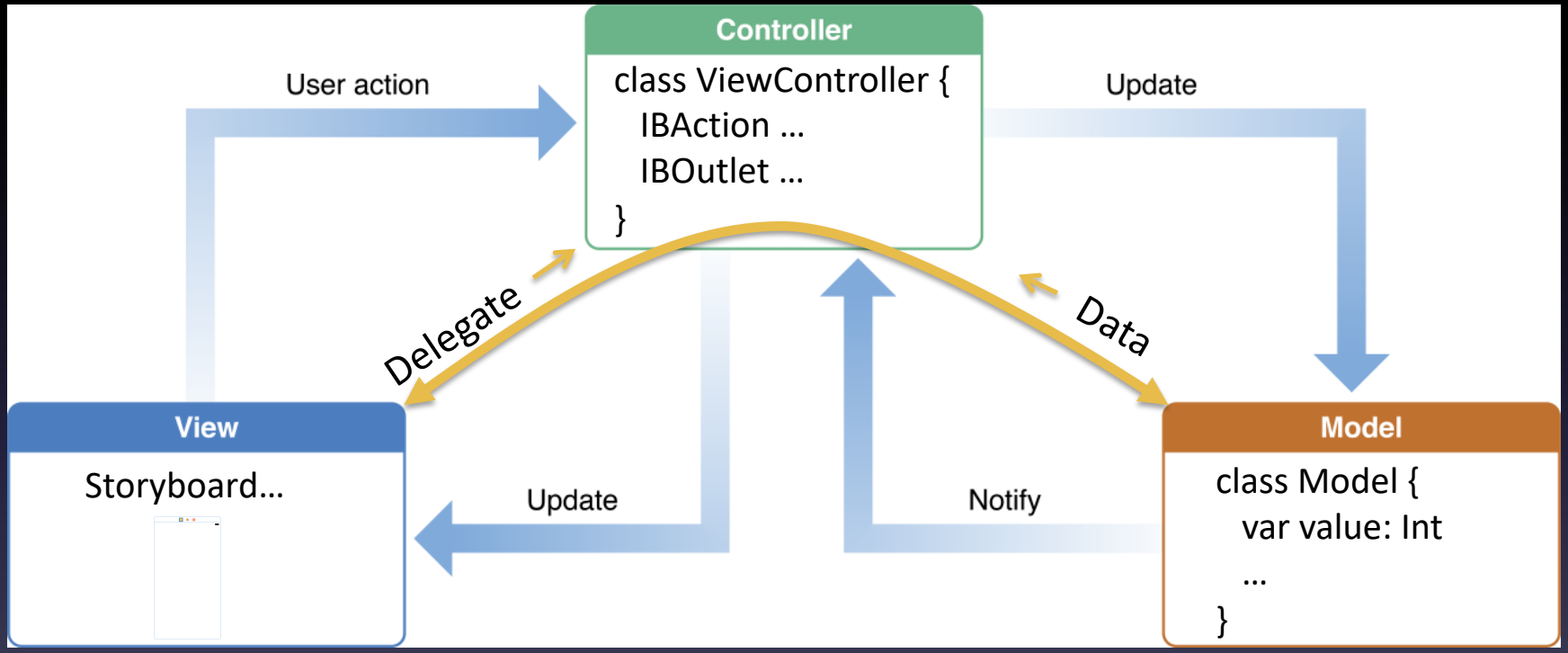
Washington State University

Instructor: Larry Holder

Outline

- Model-View-Controller (MVC) design
- Timers
- Multi-threading

Model-View-Controller (MVC)



General Scenario

- Want to modify view when some event happens within model
- E.g., if counter == 0, then display message

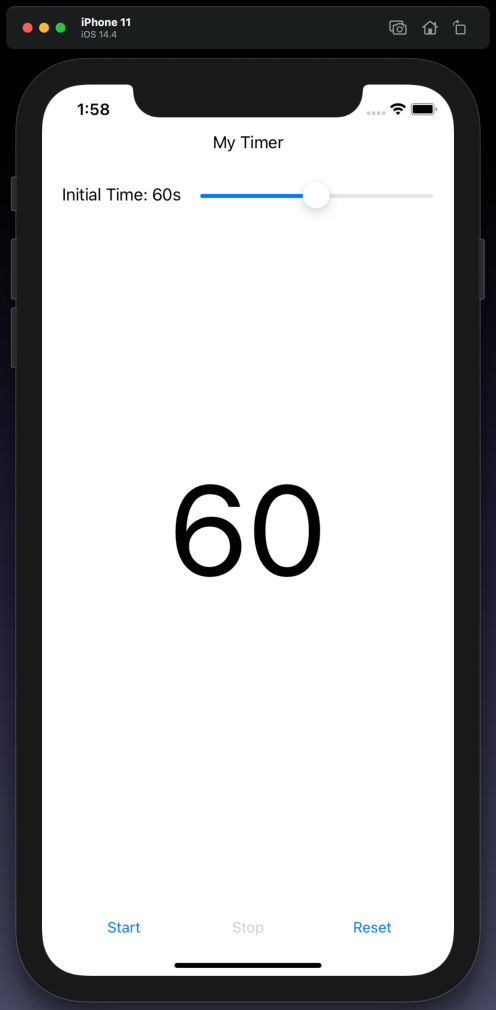
Bad Solution

```
class BadModel {
    var label: UILabel!
    var counter: Int = 10

    func decrement() {
        counter = counter – 1
        if counter == 0 {
            label.text = "Time's up!"
        }
    }
}

class MyViewController: UIViewController {
    @IBOutlet weak var myLabel: UILabel!
    var badModel = BadModel()

    badModel.label = myLabel
    badModel.decrement()
}
```

Good (MVC) Solution

```
protocol MyDelegate {
    func timesUp()
}

class GoodModel {
    var delegate: MyDelegate?
    var counter: Int = 10

    func decrement() {
        counter = counter – 1
        if counter == 0 {
            delegate?.timesUp()
        }
    }
}

class MyViewController: UIViewController, MyDelegate {
    @IBOutlet weak var myLabel: UILabel!
    var goodModel = GoodModel()

    goodModel.delegate = self

    func timesUp() {
        myLabel.text = "Time's up!"
          // What if we want to make other changes?
    }
}
```

MyTimer Example

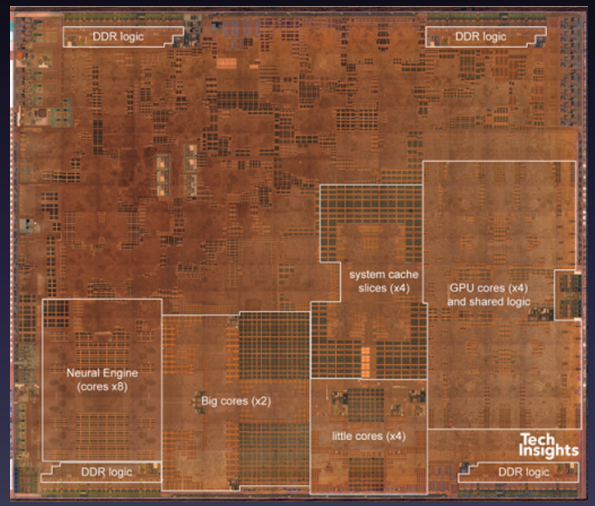


Model: MyTimer Class

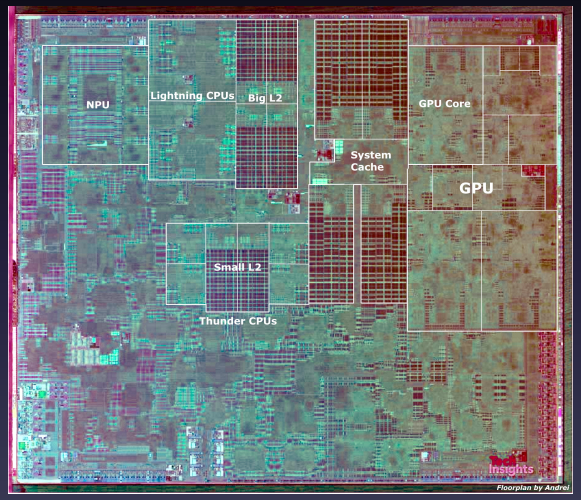
```
protocol MyTimerDelegate {
    func timeChanged (time: Int)
    func timesUp ()
}

class MyTimer {
    var delegate: MyTimerDelegate?
    var initialTime: Int = 60
    var currentTime: Int = 60

    func setInitialTime(_ initTime: Int) {
        initialTime = initTime
    }
    func start() {      // todo
    }
    func stop() {       // todo
    }
    func reset() {      // todo
    }
}
```

SideBar: Multi-Threading

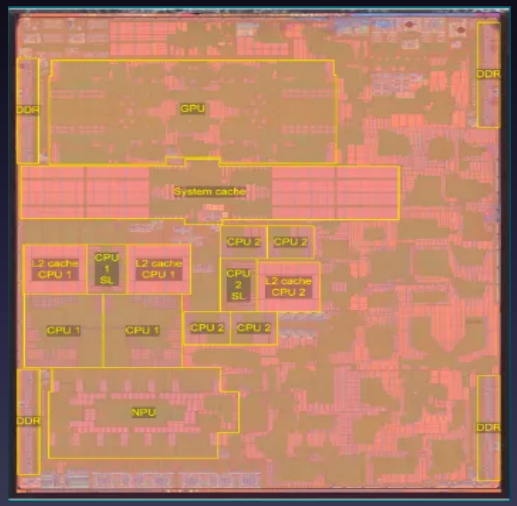
- Running multiple tasks concurrently
- Tasks assigned to threads
- Threads assigned to cores



A12: 6 CPU cores, 4 GPU cores, 8 NPU cores, iPhone XS



A13: 6 CPU cores, 4 GPU cores, 8 NPU cores, iPhone 11



A14: 6 CPU cores, 4 GPU cores, 16 NPU cores, iPhone 12

Multi-Threading in iOS

- Manipulate “queues”, not threads
- Grand Central Dispatch (GCD)
  - iOS mechanism for sending tasks to different queues
  - You can create your own queues, and assign tasks to them
  - Threads take tasks from queues and run them on a core
- Main queue used for UI and user interaction
  - Don’t put intensive tasks on Main queue!
- Also, UI changes must occur on Main queue

```
DispatchQueue.global().async {
    // Do something in background
    // Not affecting UI
}
```

```
DispatchQueue.main.async {
    // Do something easy
    // Or affecting UI
}
```

Timers

- Timer class
  - Wait specified time interval, then call function
  - Can repeat
- Start immediately

```
Timer.scheduledTimer(withTimeInterval interval: TimeInterval,
    repeats: Bool, block: @escaping (Timer) -> Void) -> Timer
```

- Stop `Timer.invalidate()`

Timers

- Create

```
init(timeInterval interval: TimeInterval,
    repeats: Bool, block: @escaping (Timer) -> Void) -> Timer
```

- Create to fire later

```
init(fire: Date, timeInterval interval: TimeInterval,
    repeats: Bool, block: @escaping (Timer) -> Void) -> Timer
```

- Add to run loop

```
RunLoop.current.add(timer, forMode: .common)
```

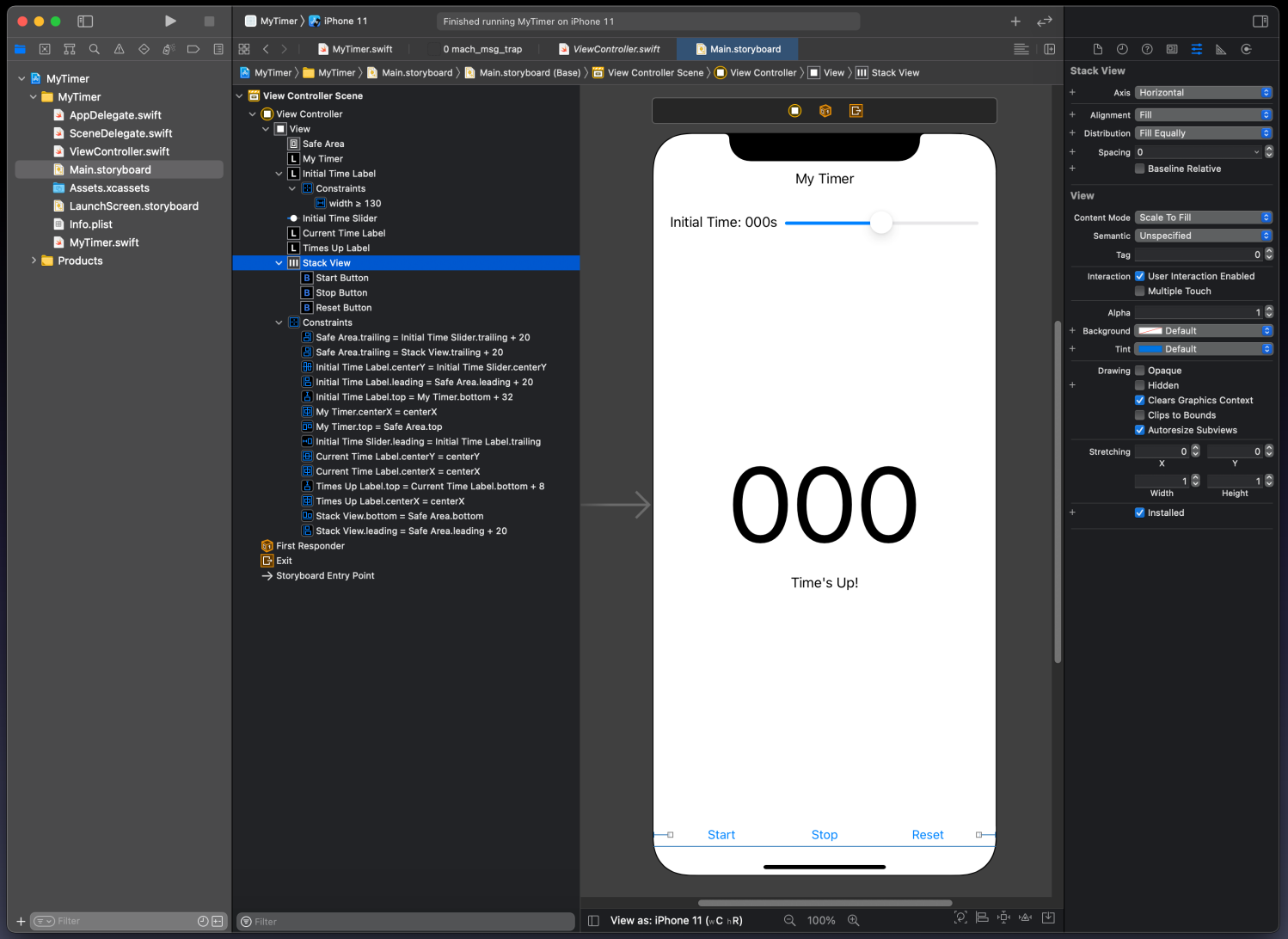
Model: MyTimer Class

```
var timer: Timer?

func start() {
    // Start immediately
    timer = Timer.scheduledTimer(withTimeInterval: 1,
        repeats: true, block: handleTick)
}

func handleTick (timer: Timer) {
    // todo
}

func stop() {
    timer?.invalidate()
}

func reset() {
    // todo
}
```

ViewController

```
class ViewController: UIViewController, MyTimerDelegate {
    var myTimer = MyTimer()

    // todo: interface outlets and actions

    func timeChanged(time: Int) { // todo
    }

    func timesUp() {   // todo
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        myTimer.delegate = self
        myTimer.setInitialTime(initialTime)
        // todo: further initializations
    }
}
```

Storyboard


MyTimer
iPhone 11
Finished running MyTimer on iPhone 11
MyTimer.swift
0 mach_msg_trap
ViewController.swift
Main.storyboard
AppDelegate.swift
SceneDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
MyTimer.swift
Products
View Controller Scene
View Controller
View
Safe Area
My Timer
Initial Time Label
Constraints
width ≥ 130
Initial Time Slider
Current Time Label
Times Up Label
Stack View
Start Button
Stop Button
Reset Button
Constraints
Safe Area.trailing = Initial Time Slider.trailing + 20
Safe Area.trailing = Stack View.trailing + 20
Initial Time Label.centerY = Initial Time Slider.centerY
Initial Time Label.leading = Safe Area.leading + 20
Initial Time Label.top = My Timer.bottom + 32
My Timer.centerX = centerX
My Timer.top = Safe Area.top
Initial Time Slider.leading = Initial Time Label.trailing
Current Time Label.centerY = centerY
Current Time Label.centerX = centerX
Times Up Label.top = Current Time Label.bottom + 8
Times Up Label.centerX = centerX
Stack View.bottom = Safe Area.bottom
Stack View.leading = Safe Area.leading + 20
First Responder
Exit
Storyboard Entry Point
My Timer
Initial Time: 000s
000
Time's Up!
Start
Stop
Reset
Stack View
Axis
Horizontal
Alignment
Fill
Distribution
Fill Equally
Spacing
Baseline Relative
View
Content Mode
Scale To Fill
Semantic
Unspecified
Tag
Interaction
User Interaction Enabled
Multiple Touch
Alpha
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clips to Bounds
Autoresize Subviews
Stretching
X
Y
Width
Height
Installed
View as: iPhone 11
100%

ViewController

```
// Interface outlets

@IBOutlet weak var initialTimeLabel: UILabel!
@IBOutlet weak var initialTimeSlider: UISlider!
@IBOutlet weak var currentTimeLabel: UILabel!
@IBOutlet weak var timesUpLabel: UILabel!
@IBOutlet weak var startButton: UIButton!
@IBOutlet weak var stopButton: UIButton!
@IBOutlet weak var resetButton: UIButton!
```

ViewController

```
// Interface actions

@IBAction func initialTimeSliderValueChanged(_ sender: UISlider) {
    // todo
}

@IBAction func startTapped(_ sender: UIButton) {
    startButton.isEnabled = false
    stopButton.isEnabled = true
    resetButton.isEnabled = false
    myTimer.start()
}

@IBAction func stopTapped(_ sender: UIButton) {
    // todo
}

@IBAction func resetTapped(_ sender: UIButton) {
    // todo
}
```

Resources

- Grand Central Dispatch (GCD)
  - developer.apple.com/documentation/dispatch
- Timer
  - developer.apple.com/documentation/foundation/timer