



ITM103 iOS Application Development

Topic 4: Debugging & Testing Apps

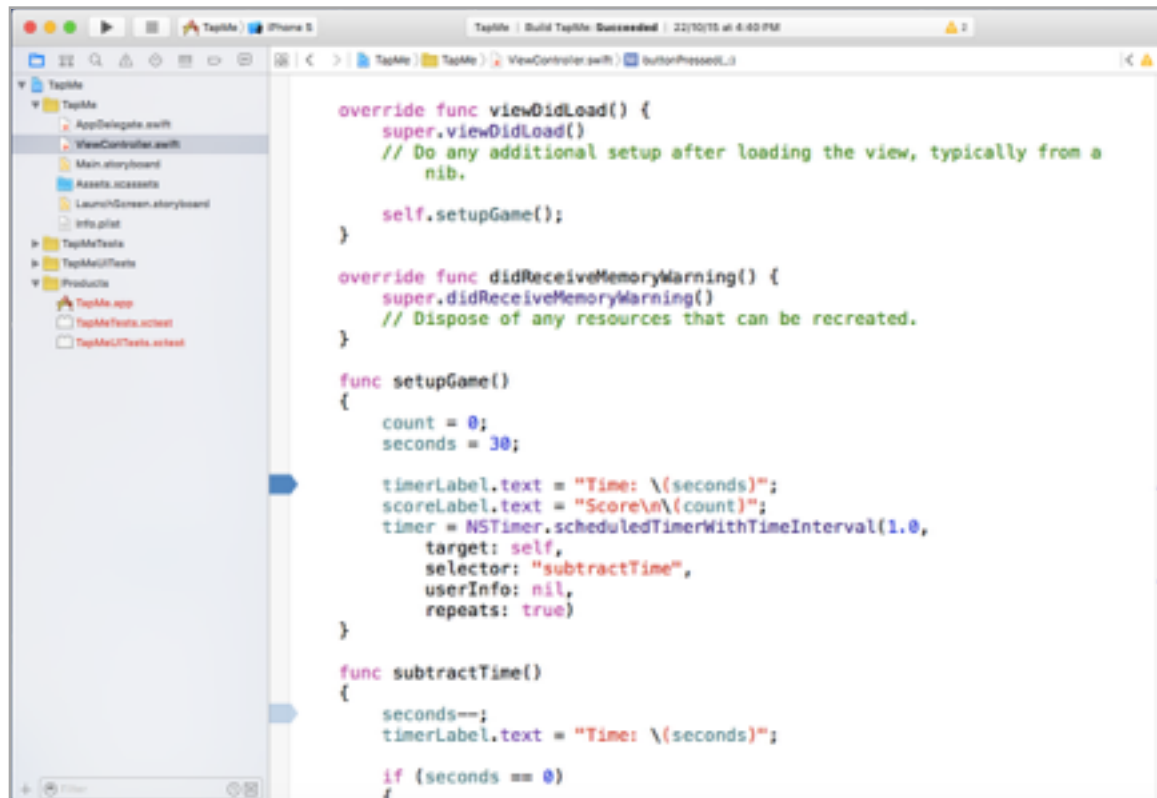


Objectives

- By the end of the lesson, you will be able to:
 - know how to debug your application
 - use Xcode's visual debugger to step through the execution of apps, monitor the current state of the application and examine the variable values

Using Breakpoints

- Breakpoints tell the debugger where to pause.
- By setting a breakpoint on a specific line of code, allows you to execute the subsequent line by line.



Using Breakpoints

The screenshot shows the Xcode IDE with a Swift file open. The code defines three methods: `viewDidLoad()`, `didReceiveMemoryWarning()`, and `setupGame()`. A breakpoint is set on the line `timerLabel.text = "Time: \(seconds)";` in the `setupGame()` method. The call stack on the left shows the sequence of calls: `0 ViewController.setupGame() -> ()`, `1 ViewController.viewDidLoad() -> ()`, `2 UIApplicationMain() -> ()`, `3 main() -> ()`, and `4 start() -> ()`. The Watch window at the bottom shows the current state of the program, with `self` set to `(TapMe.ViewController: 0x7f78a1f0)`.

Call stack

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a
    nib.

    self.setupGame();
}

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}

func setupGame()
{
    count = 0;
    seconds = 30;

    timerLabel.text = "Time: \(seconds)";
    scoreLabel.text = "Score\n\(count)";
    timer = NSTimer.scheduledTimerWithTimeInterval(1.0,
        target: self,
        selector: "subtractTime",
        userInfo: nil,
        repeats: true)
}
```

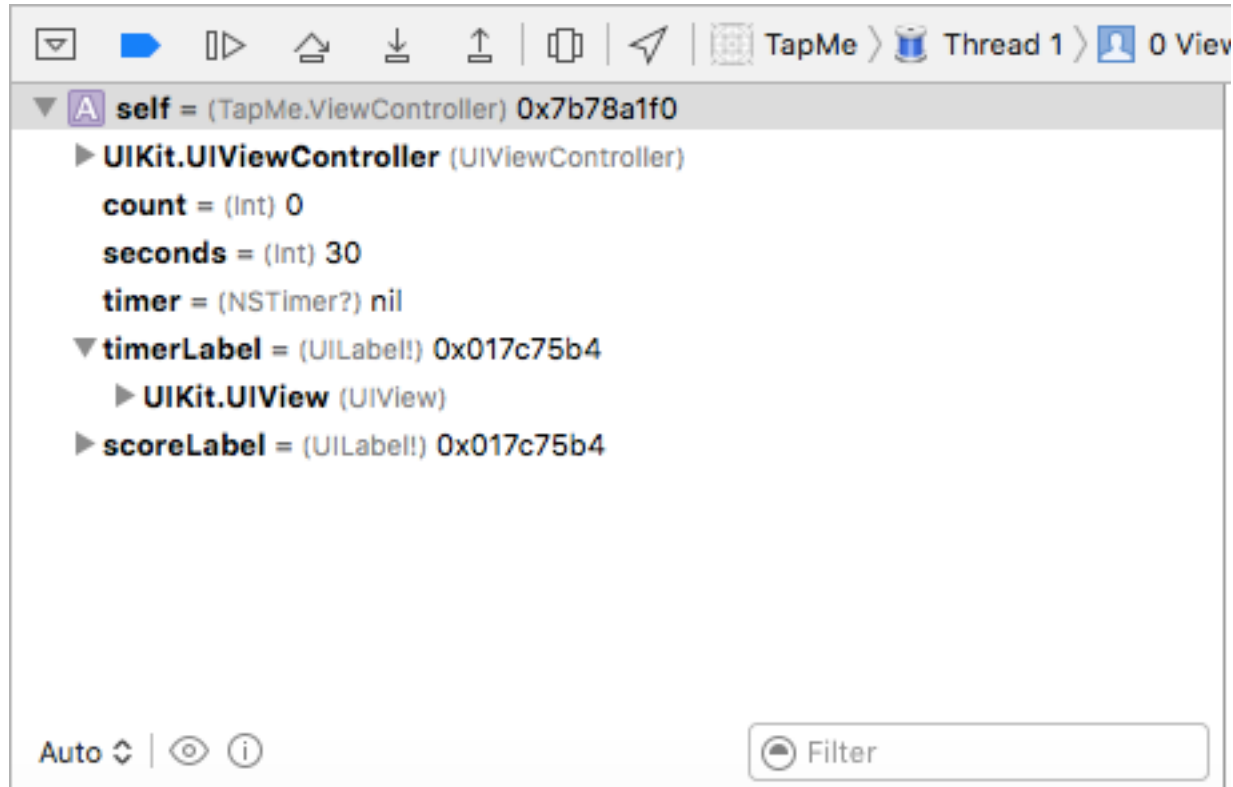
Current program position


Thread 1: breakpoint 1.1

Watch

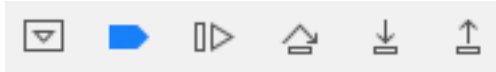
`self = (TapMe.ViewController: 0x7f78a1f0)`





Stepping through codes



Note: If you cannot see the variables view, find the  control at the right corner of the console. Click on the center button to see both the console and the variables view.

Stepping through codes



Button	Description
<i>Pause/Continue</i> 	<i>Pauses the application running in the debugger. When the application is in the pause state, this button will be “Continue”. Continue the application and resumes</i>
<i>Step Over</i> 	<i>Continues to execute the next line of code. If the next line is a function call, it executes the function and proceeds to the next line.</i>
<i>Step Into</i> 	<i>Continues to execute the next line of code. If the next line is a function call, it will go into the codes in the function call.</i>
<i>Step Out</i> 	<i>Processes until the current function exits and stops in the function that called it.</i>