Beginning

# Core Graphics

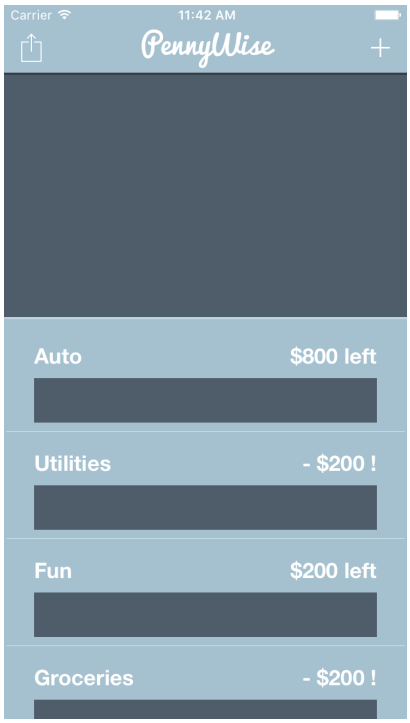
## Part 1: Getting Started

# Core Graphics Hands-On Challenges



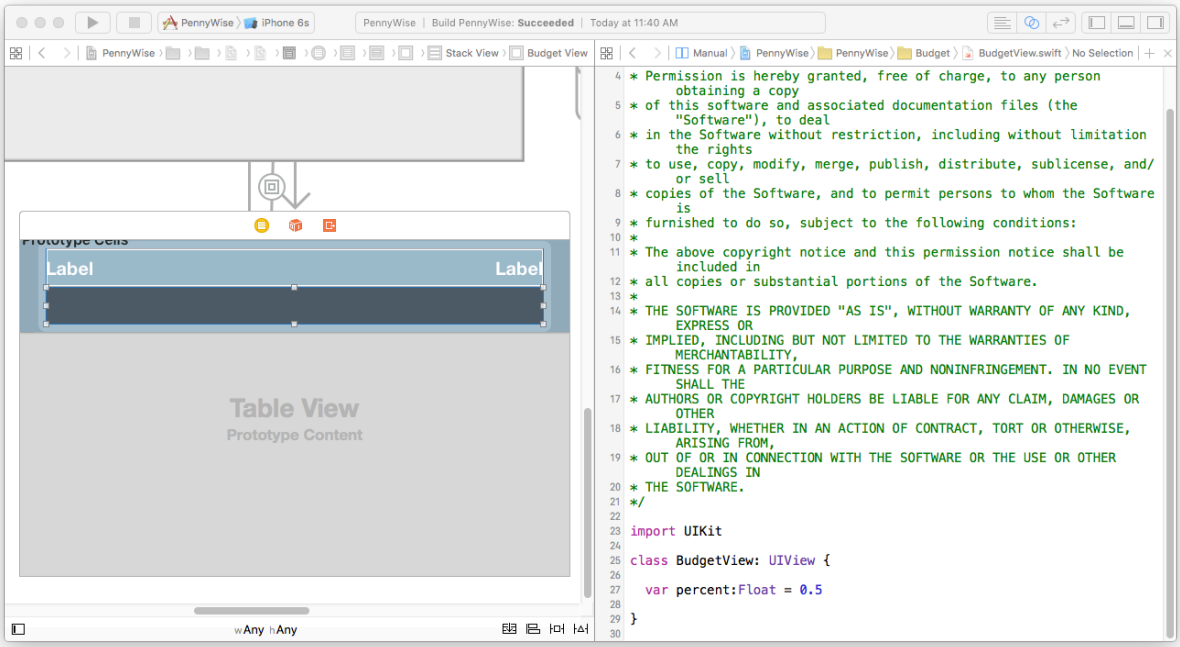
# Challenge: UIBezierPath Line

Currently there is no graphical indication of the amount spent in a category.




You're going to add a bar to each category and indicate with a color a warning when they've spent over 80% of the budget for that category, and indicate with another color when they've spent 100% of the budget.

Open up **Main.storyboard** and find the **Budget Table View Controller Scene**. Click on the dark strip which is **BudgetView** and open up the matching **BudgetView.swift** in the Assistant Editor.




To get BudgetView.swift in the Assistant Editor, you may have to click on the Jump Bar and choose Manual \ PennyWise \ PennyWise \ Budget \ BudgetView.swift.

Set the class as `@IBDesignable` so that you can see it in your storyboard:

```
@IBDesignable
class BudgetView: UIView {
```

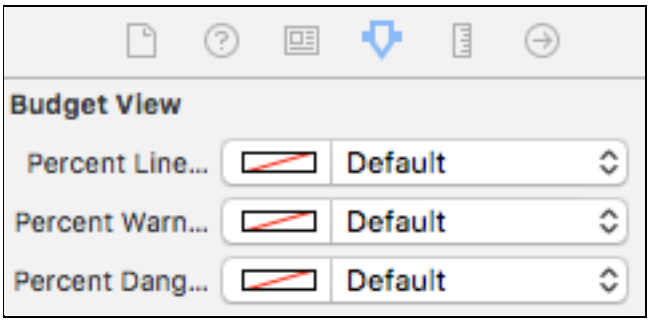
Just as we did in the demo for the number view, override `drawRect(_:)`:

```
    override func drawRect(rect: CGRect) {
    }
```

Create properties in BudgetView for the line. Create three different colors, as they will change as the spent percentage gets up to 100%. I've defined three colors in AppDelegate to match PennyWise's color scheme:

```
    @IBInspectable var percentLineColor:UIColor = appGreenColor
    @IBInspectable var percentWarningLineColor:UIColor =
                                                    appOrangeColor
    @IBInspectable var percentDangerLineColor:UIColor = appRedColor
```

Because they're inspectable properties, you can change them in the storyboard if you wish.




Create an inspectable property for the line width

```
    @IBInspectable var lineWidth:CGFloat = 13
```

In `drawRect(_:)`, create a Bézier path and set the line width.

```
    let path = UIBezierPath()
    path.lineWidth = lineWidth
```

Now set the drawing position to the start of the path:

```
    path.moveToPoint(CGPoint(x: 0, y: rect.height/2))
```

Calculate the end of the path, which is the width of the view's bounds multiplied by the percentage of the budget spent.

```
    let end = rect.width * CGFloat(percent)
     path.addLineToPoint(CGPoint(x: end, y: rect.height/2))
```

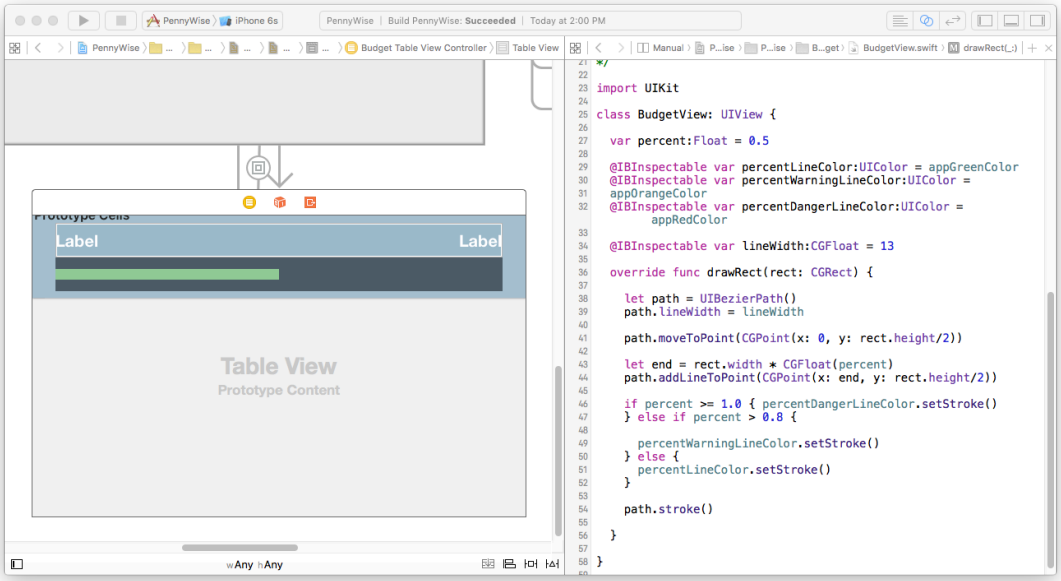
Set the stroke color depending on the amount spent:

```
      if percent >= 1.0 {
        percentDangerLineColor.setStroke()
      } else if percent > 0.8 {
        percentWarningLineColor.setStroke()
      } else {
        percentLineColor.setStroke()
      }
```

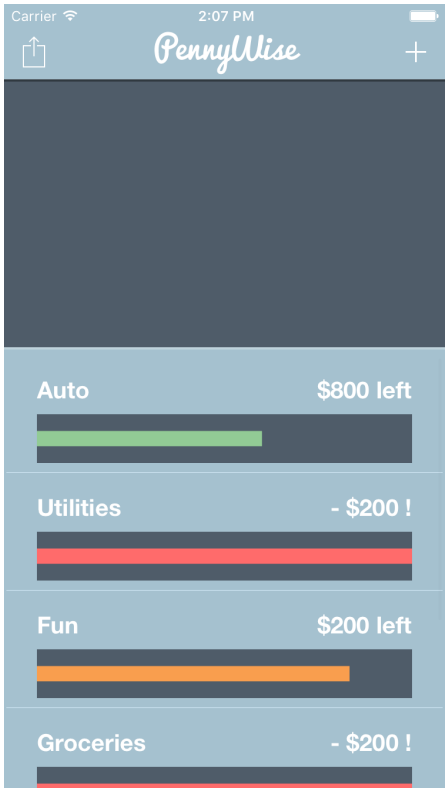
Finally stroke the path:

```
      path.stroke()
```

In the storyboard you should have a line that runs half the way across, matching the value of percent.



Build and run, and you should have differently colored bars showing how much of the budget for each category has been spent.




Tap on **Auto** and have a **$500** car service. Tap **Done** to add the expense.

You should see Auto's bar turn from green to orange.


Tap on **Auto** again and put in **$100** of fuel. Tap **Done** to add the expense.

What's happened to the bar?


This is where `setNeedsDisplay()` comes in. The bar size needs to be refreshed when the percent is changed. `setNeedsDisplay()` indicates to the layout engine that the view needs to be refreshed and `drawRect(_:)` will be done during the refresh.

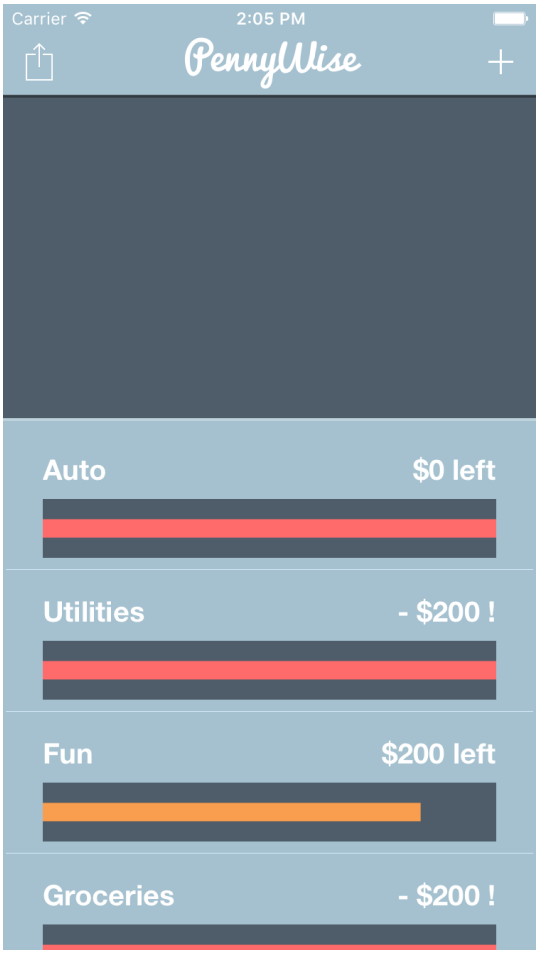
In **BudgetView**, change:

```
    var percent:Float = 0.5
```

to

```
    var percent:Float = 0.5 {
      didSet {
        setNeedsDisplay()
      }
    }
```

Build and run the app and repeat what you did above - put in a $500 Auto expense followed by a $100 Auto expenses - you should see the bar gradually climb to the right. Eventually you run out of budget and the bar turns red.


Carrier
2:05 PM
PennyWise
Auto
$0 left
Utilities
- $200 !
Fun
$200 left
Groceries
- $200 !