Intermediate

Core Graphics

Part 5: PDF Printing

Core Graphics Hands-On Challenges

Copyright © 2016 Razeware LLC.

All rights reserved. No part of this book or corresponding materials (such as text, images, or source code) may be reproduced or distributed by any means without prior written per- mission of the copyright owner.

This book and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

All trademarks and registered trademarks appearing in this book are the property of their respective owners.

Challenge: PDF Printing

You'll create a second page in the PDF to print out the pie chart.

In **PDFViewController.swift**, first create a method to render the pie chart in the same way as we did the budget.

```
func renderPieChart() {
}
```

In `createPDF()`, after calling `renderCategories()`, start a new page with:

```
UIGraphicsBeginPDFPage()
```

Color the background and call your new method:

```
renderBackground()
renderPieChart()
```

In `renderPieChart()`, draw a heading at the top of the page:

```
let attributes = [NSFontAttributeName:
                      UIFont.boldSystemFontOfSize(26),
             NSForegroundColorAttributeName: UIColor.whiteColor()]
let title = "Pie Chart of Expenses"
title.drawAtPoint(CGPoint(x: 32, y: 40),
                  withAttributes: attributes)
```

Here you just set the text attributes of title and drew the title with those attributes.

You'll now use an instance of GraphView to draw the graph into the context.

GraphView has been designed to take the full size of the screen, so, at the end of `renderPieChart()`, instantiate **GraphView** with the size of the device:

```
  let graphView = GraphView(frame:UIScreen.mainScreen().bounds)
```

The view needs to be in landscape, so swap the sizes around if it isn't:

```
if graphView.frame.width < graphView.frame.height {
  graphView.frame.size.width =
                  UIScreen.mainScreen().bounds.height
  graphView.frame.size.height =
                  UIScreen.mainScreen().bounds.width
}
```

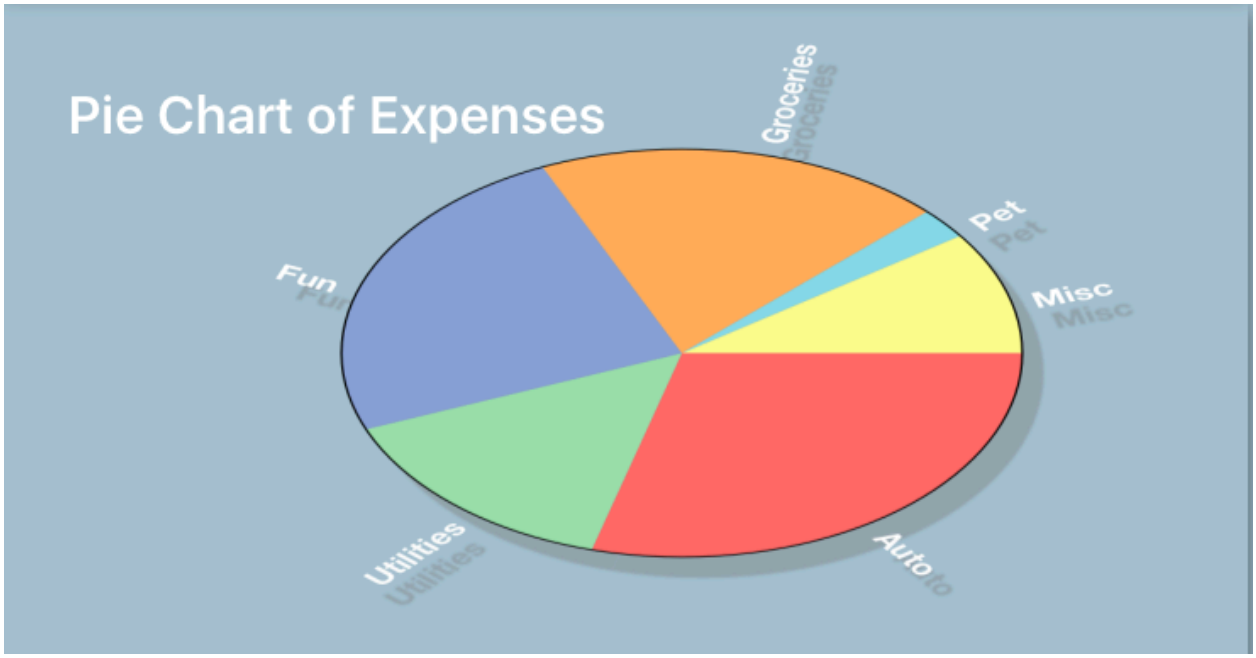
Make the graph view's background color transparent so that the PDF background color shows through.

```
graphView.backgroundColor = UIColor.clearColor()
```

Render the graph view into the current context:

```
guard let context = UIGraphicsGetCurrentContext() else { return }
graphView.layer.renderInContext(context)
```

Build and run the application. Tap on the share icon at the top left of the screen.



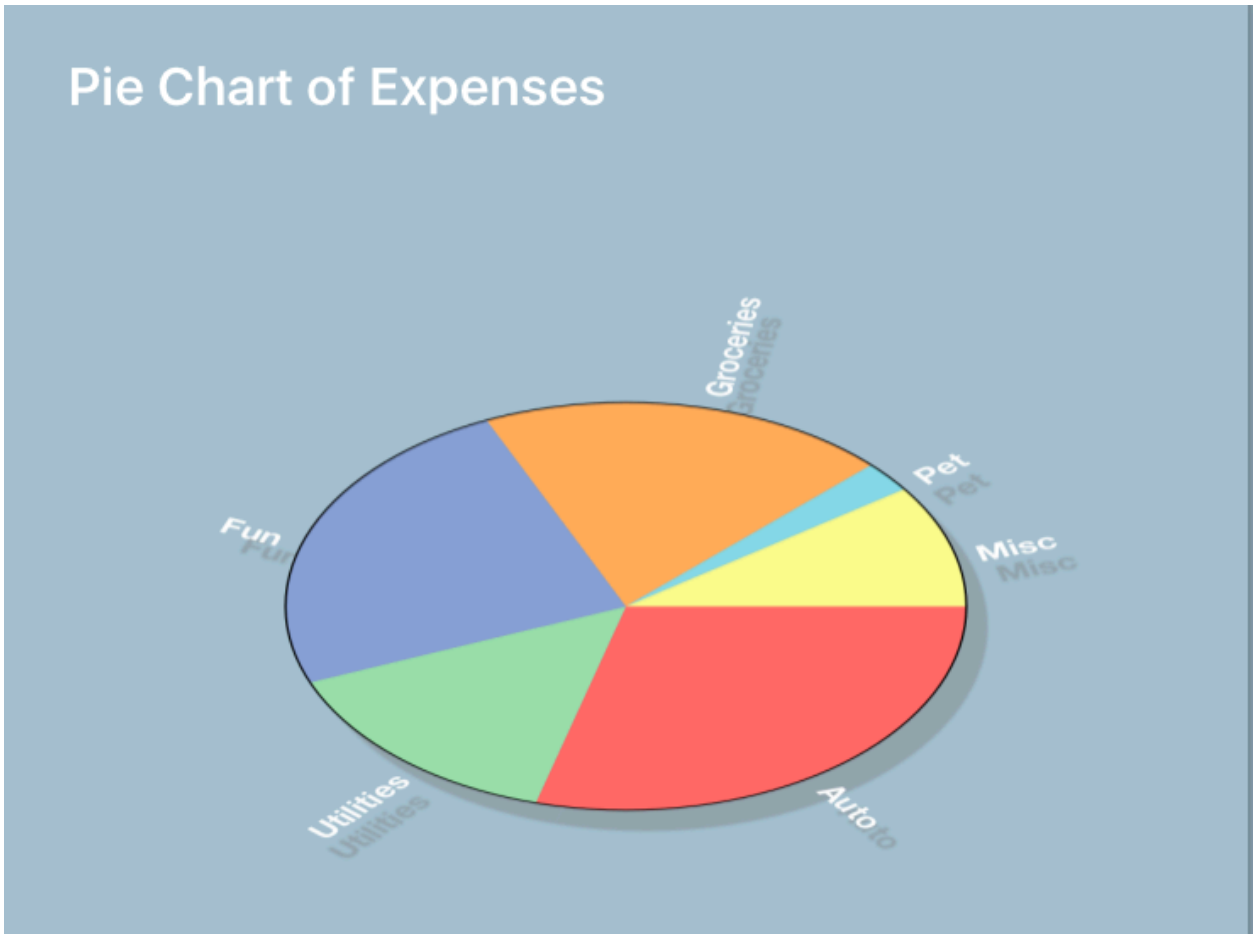
The chart's a bit too high up the page, so in `renderPieChart()` before rendering the layer, translate the context half way down:

```
CGContextSaveGState(context)
CGContextTranslateCTM(context,
     (612 - graphView.frame.width) / 2,
     (792 - graphView.frame.height) / 3)
```

After rendering the layer, restore the context state:

```
CGContextRestoreGState(context)
```

Build and run the application to see your final PDF.



Uber Haxx0r Challenge: Page Numbers

As an extra challenge print the page number at the bottom of each page of the PDF.

You'll find the solution in the Challenge Final project.

Congratulations! You've made it all the way through the series. Thanks for staying with it. I hope you enjoyed the series and I can't wait to see what you draw with your Core Graphic skills!