Intermediate

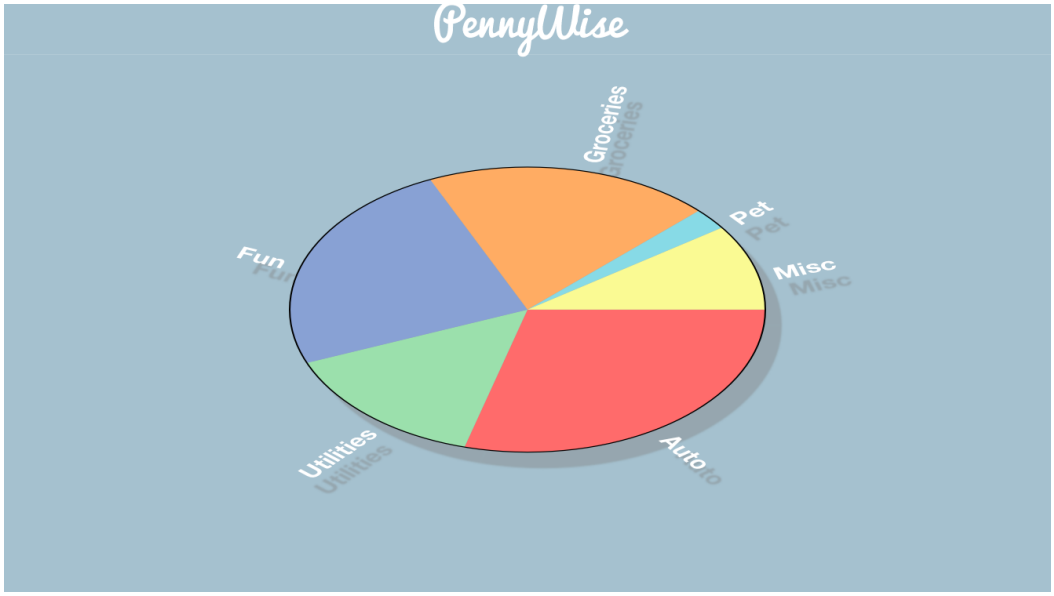
Core Graphics

Part 4: Drawing Text and Images

Core Graphics Hands-On Challenges

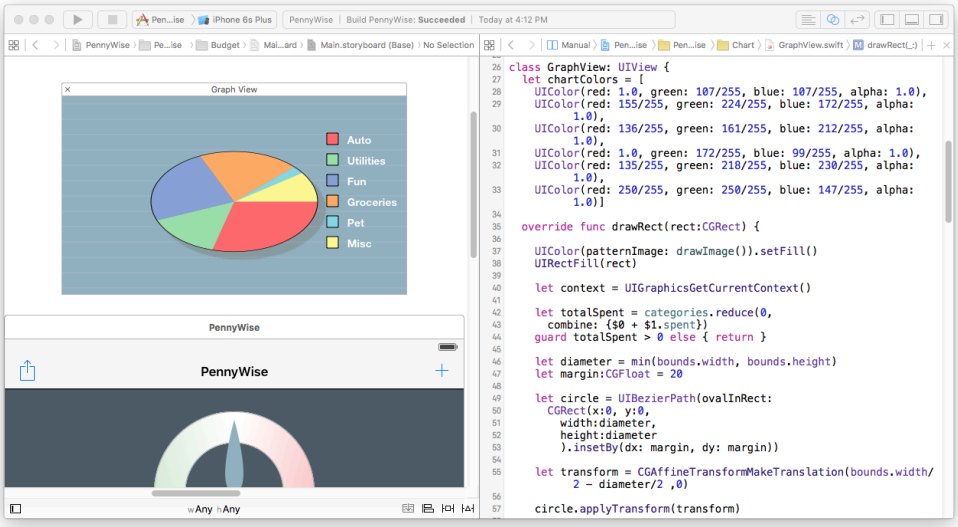
Challenge: Transforming Text

Instead of a text key for the pie chart, you're going to label it with text rotated around the chart. This will be a difficult challenge as there's going to be a lot of context rotation.



As you draw each slice you'll add some text to it.

Make sure you have Xcode set up so that you have `GraphView` showing in `Main.storyboard` and `GraphView.swift` in the Assistant Editor.



You can find GraphView.swift in the jump bar of the right hand pane – **Manual / PennyWise / PennyWise / Chart / GraphView.swift**.

Remove the pattern drawing from `drawRect(_:)`. Remove this code at the top of the method:

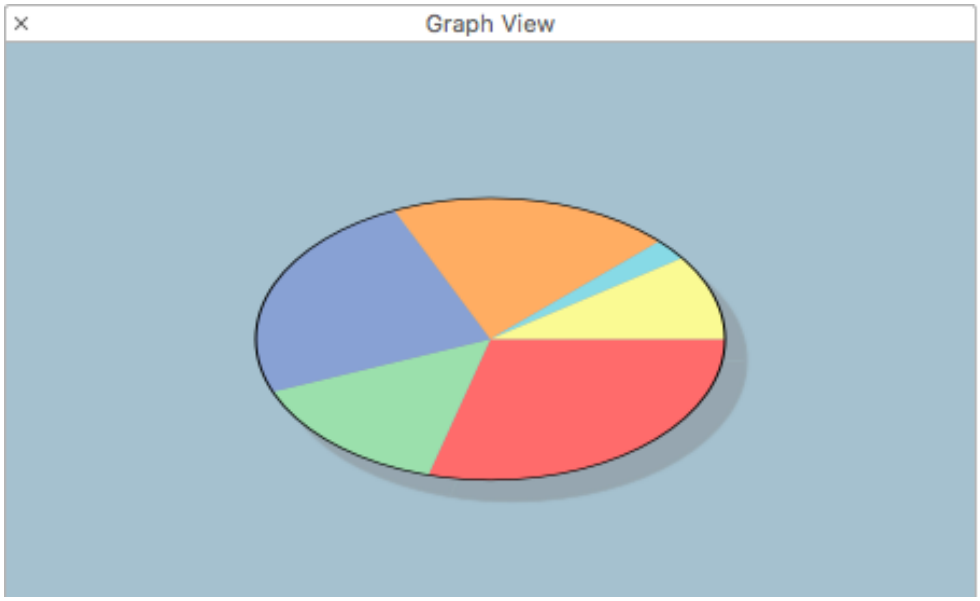
```
    UIColor(patternImage: drawImage()).setFill()
    UIRectFill(rect)
```

At the end of `drawRect(_:)`, remove:

```
    drawKey(workingCategories, rect: rect)
```

This will stop the text key from the demo being drawn, leaving the pie chart clear for your text drawing.

GraphView should now look like this in the storyboard:



Create a new method in **GraphView.swift** with the category name and pie's center point and radius as its parameters:

```
    func drawText(name: String, centerPoint:CGPoint,
                                        radius:CGFloat) {
    }
```

Call this method inside the loop in `drawRect(_:)` after filling the slice with `slice.fill()` and before translating the context with `CGContextTranslateCTM(context, centerPoint.x, centerPoint.y)`:

```
drawText(workingCategories[index].name,
                centerPoint: centerPoint,
                radius:radius)
```

In `drawText(_:centerPoint:radius:)`, get a reference to the current context.

```
let context = UIGraphicsGetCurrentContext()
```

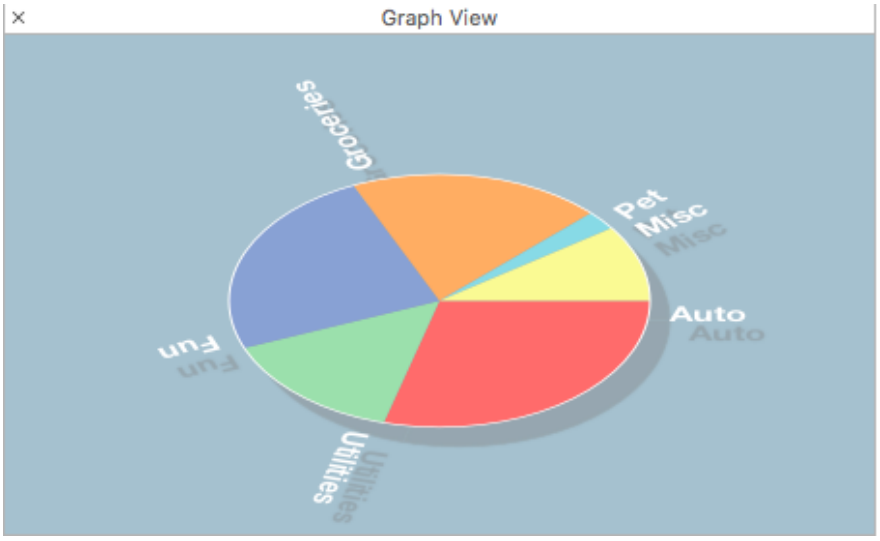
Set up the attributes for the text:

```
let font = UIFont(name: "HelveticaNeue-Bold", size: 18)!
let attributes = [NSFontAttributeName: font,
      NSForegroundColorAttributeName: UIColor.whiteColor()]
```

Draw the attributed string outside the pie.

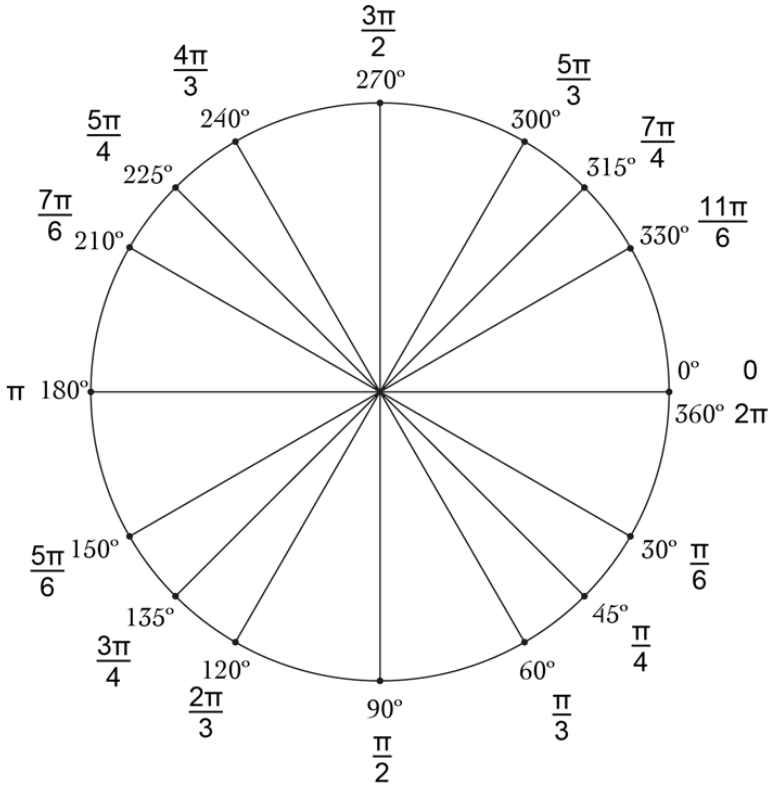
```
name.drawAtPoint(CGPoint(x:centerPoint.x + radius + 10,
                         y:centerPoint.y),
                 withAttributes:attributes)
```

In the storyboard, that's looking pretty good, except for some of the text is being drawn upside down when the context is rotated.



Here's where it'll get a bit complicated. You'll need to rotate the text, but only if it appears on the left hand side.

This is the unit circle marked with radians and degrees.



I find it helpful to look at the unit circle when working out rotations. The angle in radians straight down is π/2. The angle at the top is 3π/2.

If the angle of rotation is between π/2 and 3π/2 then you need to flip the text.

Add the following code **before**:

```
name.drawAtPoint(CGPoint(x:centerPoint.x + radius + 10,
                         y:centerPoint.y),
                 withAttributes:attributes)
```

Get the current transform from the context:

```
  let transform = CGContextGetCTM(context)
```

Extract the rotation of the context from this transform variable. This is a really useful formula and piece of code to remember.

```
    let radians = atan2(transform.b, transform.a)
```

Remember, if the angle of rotation is between π/2 and 3π/2 then you need to flip the text, so replace this line:

```
name.drawAtPoint(CGPoint(x:centerPoint.x + radius + 10,
                         y:centerPoint.y),
                 withAttributes:attributes)
```

with this conditional:

```
    if abs(radians) > π / 2 && abs(radians) < 3 / 2 * π {
    } else {
      name.drawAtPoint(CGPoint(x:centerPoint.x + radius + 10,
                               y:centerPoint.y),
                       withAttributes:attributes)
    }
```

You'll flip the text by rotating the context half the way around.

Inside the if part of the conditional, first save the context and then rotate it by π radians:

```
  CGContextSaveGState(context)
  CGContextRotateCTM(context, π)
```

Now, still inside that conditional, draw the text string string with all points negative:

```
  name.drawAtPoint(CGPoint(x:-centerPoint.x - radius + 10,
                           y:-centerPoint.y),
                   withAttributes:attributes)
```

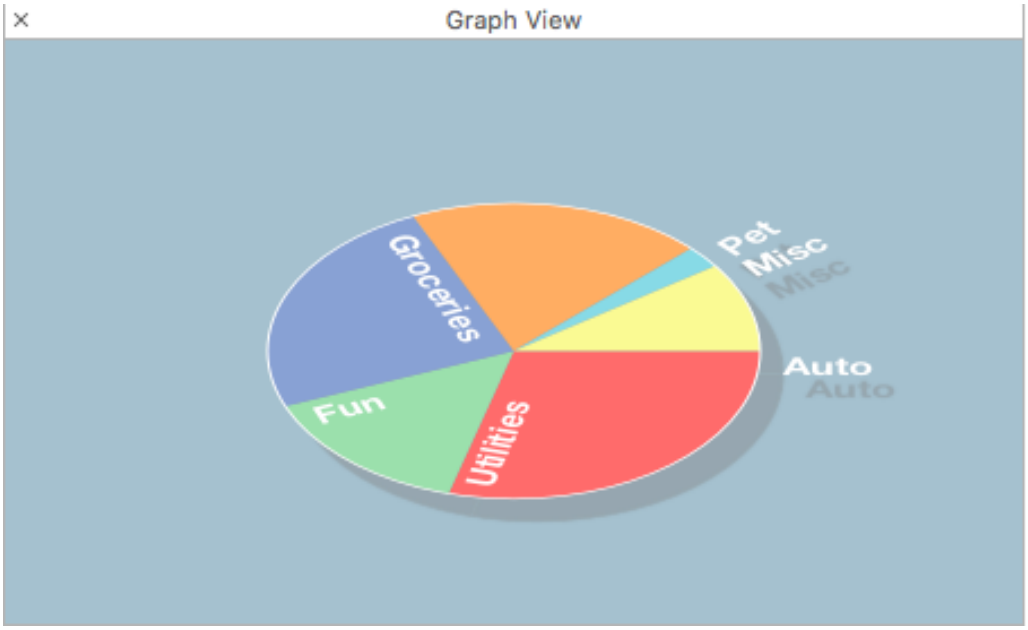
And restore the context state to before the rotation.

```
  CGContextRestoreGState(context)
```

The conditional should read:

```
if abs(radians) > π / 2 && abs(radians) < 3 / 2 * π {
  CGContextSaveGState(context)
  CGContextRotateCTM(context, π)
  name.drawAtPoint(CGPoint(x:-centerPoint.x - radius + 10,
                           y:-centerPoint.y),
                   withAttributes:attributes)
  CGContextRestoreGState(context)
} else {
  name.drawAtPoint(CGPoint(x:centerPoint.x + radius + 10,
                           y:centerPoint.y),
                   withAttributes:attributes)
}
```

The text in the storyboard is now the right way up, but it's inside the pie:



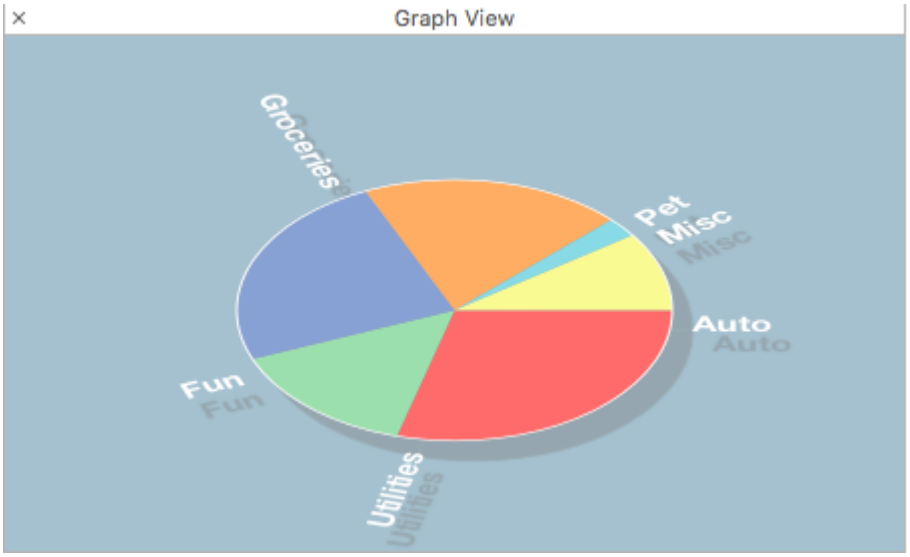
You'll now have to find out the how big the text string is to draw it outside.

Inside the first part of the conditional before you save the context, get the size of the string:

```
    let textWidth = name.sizeWithAttributes(attributes)
```

Add the width to the x value. Replace the drawing of the name text in the if part of the conditional with:

```
    name.drawAtPoint(CGPoint(x:-centerPoint.x - radius -
                                    textWidth.width - 10,
                             y:-centerPoint.y),
                     withAttributes:attributes)
```



The names are now drawn outside the pie, but they are being drawn at the edge of the slice, not in the center of the slice.

To be able to center this text, `drawText(_:centerPoint:radius:)` will need to know the angle of each slice.

Change the method header to:

```
func drawText(name: String,
              centerPoint:CGPoint,
              radius:CGFloat,
              angle:CGFloat)
```

and change the method call in `drawRect(_:)`:

```
drawText(workingCategories[index].name,
         centerPoint: centerPoint,
         radius:radius,
         angle:angle)
```

You'll need rotate the context by half the angle before drawing the text:

In `drawText(_:centerPoint:radius:angle:)`, after the context variable declaration `let context = UIGraphicsGetCurrentContext()`, save the context:

```
  CGContextSaveGState(context)
```

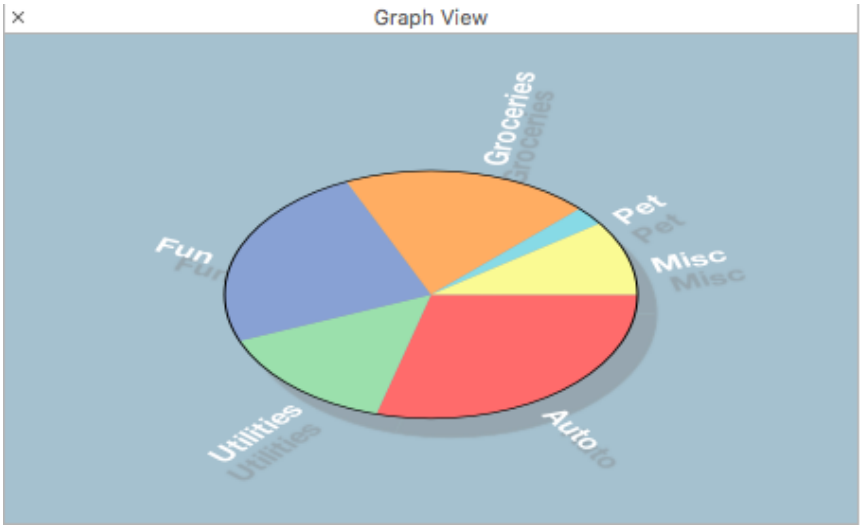
Translate the context to the center so that the rotation takes place correctly, then rotate the context by half the slice's angle, and translate the context back again:

```
  CGContextTranslateCTM(context, centerPoint.x, centerPoint.y)
  CGContextRotateCTM(context, angle/2)
  CGContextTranslateCTM(context, -centerPoint.x, -centerPoint.y)
```

At the end of the method remember to balance the saved state by restoring the state to what it was at the beginning of the method:

```
  CGContextRestoreGState(context)
```

The text in the storyboard is now being drawn around the edge of the pie, and centered in the slice.



Build and run the app and admire your 3d pie chart.

If you got all the way to the end of this challenge, well done! It's the most difficult challenge of the series. But as well as drawing text, it should have given you more confidence at rotating and translating the context.

You can do anything with Core Graphics!