

Intermediate

Core Graphics

Part 1: Gradients

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 permission of the copyright owner.

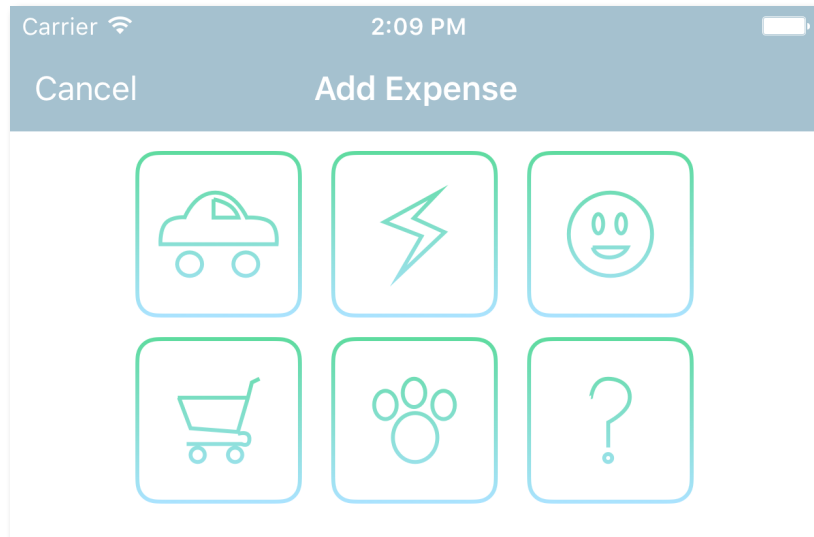
This book and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright 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: Blending Modes

You'll now change the Category expense icons to be outlined with a gradient rather than filled.

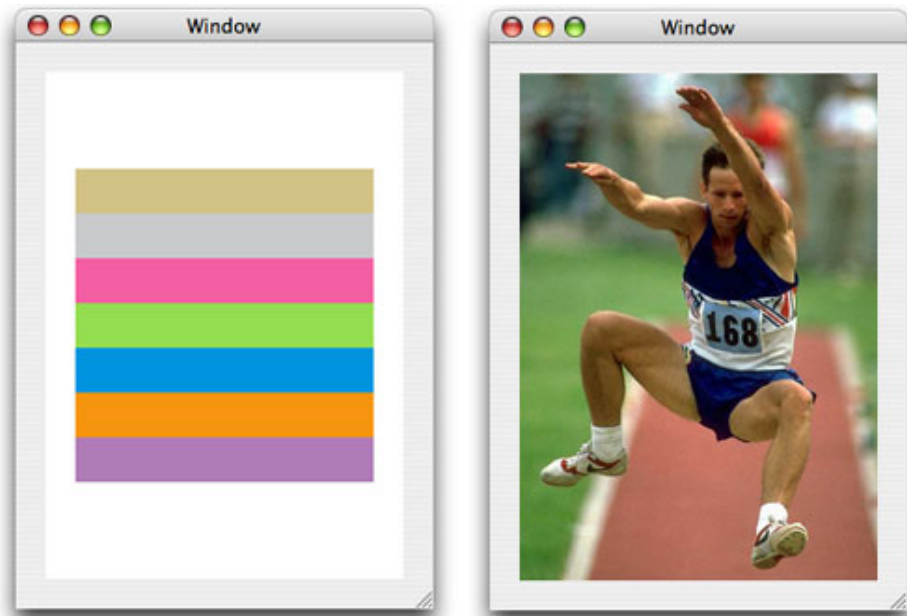


You can do this using blending modes. If you've used Photoshop, you've probably come across blending modes. It's beyond the scope of this tutorial to go into them all, but you can achieve great effects by blending layers.

Blending modes determine how layer colors interact with each other. When you place an image or a colored shape on top of another image or colored shape, the default is that the second image overwrites the first one. That's the **Normal** blending mode.

Multiply blending mode multiplies each pixel color from the top image with the pixel from the bottom image resulting in a darker color.





For example, if you place the previous left image with a blending mode of Multiply over the image on the right, then this is the result



Apple has a great visual list of the most common blending modes in its Quartz 2d Programming Guide at <http://apple.co/1Qu1z32>



The blending mode that you will be using will process the alpha channel rather than the color.

You'll stroke the icon's path as usual, but then set the context's blending mode to `SourceIn`. You'll then do a gradient fill.

The result of this is that only the pixels that have a color (i.e. the ones that were stroked) will be filled with the gradient.

First open **CategoryCell.swift** in the Challenge Starter app.

Instead of filling the cell with the gradient before stroking the icon path, you're going to draw the gradient at the very end of the method after stroking the path.

Remove

```
edge.addClip()
```

as we don't need that any longer.

Replace that line with stroking the edge instead:

```
edge.stroke()
```

Cut the `endPoint` and gradient drawing code:

```
let endPoint = CGPoint(x: 0, y: rect.height)

drawGradient(cellGradientStart,
  endColor: cellGradientEnd,
  startPoint: .zero,
  endPoint: endPoint)
```

Shortly you'll paste this in at the end of the method.



At the end of `drawRect(_:)`, add a conditional -

```
if !selected {  
    let context = UIGraphicsGetCurrentContext()  
    CGContextSetBlendMode(context, .SourceIn)
```

This gets a reference to the current context and sets the blend mode to `SourceIn`.

Paste in those cut gradient lines here and end the conditional:

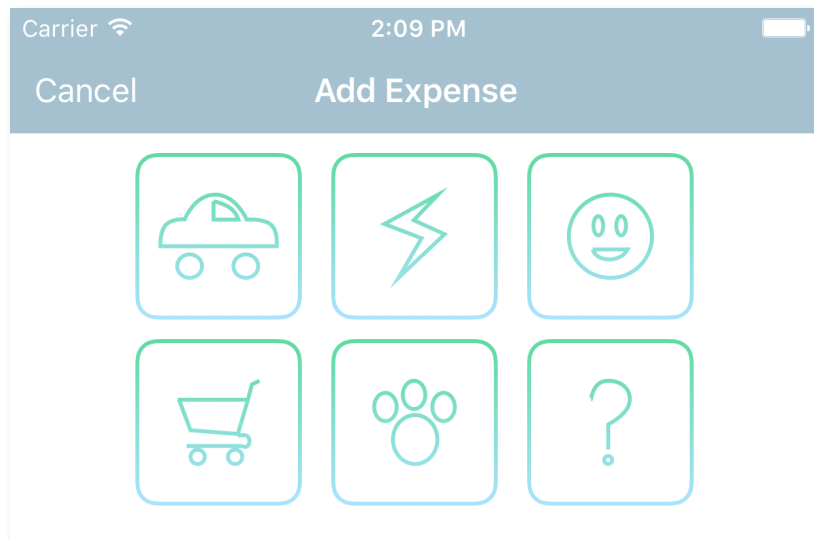
```
        let endPoint = CGPoint(x: 0, y: rect.height)  
        drawGradient(cellGradientStart ,  
                    endColor: cellGradientEnd,  
                    startPoint: .zero,  
                    endPoint: endPoint)  
    }
```

Here you're drawing the gradient.

If the context did not have a blending mode set, the gradient would overwrite the path, but because of the blending mode, only the non-alpha pixels from the icon path are picked out.



Run the application and see that the icons are drawn with a gradient outline (as long as they are not selected).



You can make really cool effects with blending modes - I encourage you to have a look at the Core Graphics Developer guide and experiment.

But beware that you don't do it too much, because blending layers is a hit on GPU processing.

