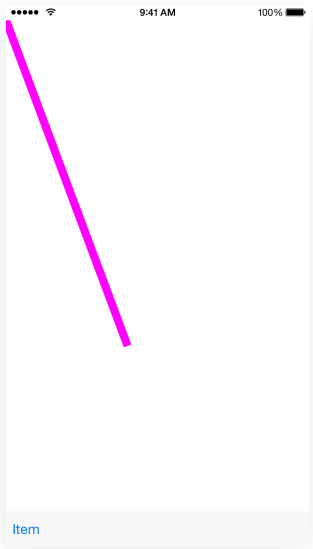
# FingerPainter
## Lesson 1

## Description

Add a custom view to the interface and override the `drawRect:` method to draw a straight line with Core Graphics.




## Learning Outcomes

- Relate interface components to the `UIView` class.
- Practice adding a new class to an Xcode project.
- Describe object-oriented inheritance and subclassing.
- Discover how interface components in Interface Builder can be bound to specific classes.
- Discover common Core Graphics functions and the `CGContextRef` data type.
- Replicate using Core Graphics functions to draw a line.

## Vocabulary

| | | |
|---|---|---|
| view | `UIView` | subclass |
| superclass | override | context |
| painter's model | point | path |
| stroke | Core Graphics | `CGContextRef` |

## Materials

- **FingerPainter Lesson 1** Xcode project

# Opening

How can we draw a single line on the screen?

# Agenda

- Explain how interface components are subclasses of `UIView`, and that overriding its `drawRect:` method can determine how the interface component is drawn on the screen.
- Discuss creating a custom `UIView` subclass to represent a custom view, in order to control exactly how the view is drawn.
- Add a new class (⌘N) called `CanvasView`.

```
class CanvasView {

}
```

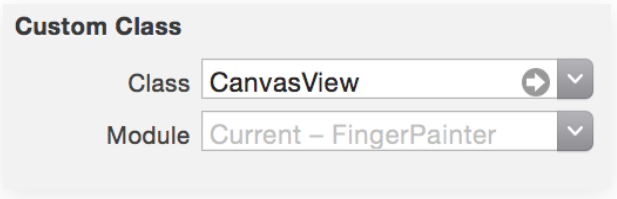
- Add an import statement for `UIKit` above the `CanvasView` class definition.

```
import UIKit
```

- Add a `UIView` superclass to the `CanvasView` class definition.

```
class CanvasView : UIView {
```

- Using Interface Builder and the Object Library (⌥⌘L), add a Toolbar to the bottom of the interface. Use the menu item *Editor* > *Resolve Auto Layout Issues* > *Add Missing Constraints* to add Auto Layout constraints for the toolbar.
- Using Interface Builder and the Object Library (⌥⌘L), add a new View object that occupies the remainder of the interface. Use the menu item *Editor* > *Resolve Auto Layout Issues* > *Add Missing Constraints* to add Auto Layout constraints for the view.
- With the new View object selected, open the Identity Inspector (⌥⌘3), and change the **Class** to `CanvasView`.




- Explain how the Canvas View in Interface Builder is now bound to the `CanvasView` class.
- Run the app (⌘R), and observe the toolbar and the white screen.

- Discuss the need to implement how the `CanvasView` is to be drawn.
- Explore the `UIView` class reference, drawing attention to the `drawRect:` method.
- Explain how, when iOS renders a view, it will call the `drawRect:` method of the view object.
- Update the `CanvasView` implementation with a custom `drawRect:` method.

```
override func drawRect(rect: CGRect) {
   let context = UIGraphicsGetCurrentContext()
   CGContextSetStrokeColorWithColor(context,
      UIColor.magentaColor().CGColor)
   CGContextSetLineWidth(context, 10)
   CGContextBeginPath(context)
   CGContextMoveToPoint(context, 0, 0)
   CGContextAddLineToPoint(context, 150, 400)
   CGContextStrokePath(context)
}
```

- Run the app (⌘R), and observe the diagonal magenta line appear.
- Discuss the `CG` function names in `drawRect:`.
- Explain that the Core Graphics framework is used to implement custom drawing with graphics primitives, such as lines.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `CGContextRef` (CGContext Reference) type definition.
- Discuss how Core Graphics tends to use C-style function calls rather than Swift methods.
- Explain how the *context* represents a drawing environment, and is used by other Core Graphics functions to create a drawing.
- Briefly discuss how Core Graphics uses a "painter's model" for drawing, perhaps summarizing how one might pick up a brush of a particular color and width, choose a starting and ending point, and then stroke the path between the two points.
- Explain that the `CGContextSetStrokeColorWithColor` function sets the active color for the drawing context.
- Explain that `CGContextSetLineWidth` function sets the active width of "the brush stroke" for a context.
- Explain that the `CGContextBeginPath` function informs Core Graphics that a path of points is about to be defined.
- Explain that the `CGContextMoveToPoint` function indicates the first point in the path.
- Explain that the `CGContextAddLineToPoint` function establishes a linear connection from the previous point to a new point.
- Explain that the `CGContextStrokePath` function then "paints the brush stroke" for the connected points in the path.

- Discuss how the obtaining of a drawing context, picking "a brush color and width," creating a path, then stroking that path, is a common line-drawing idiom for Core Graphics.

## Closing

What would you change to make the line color blue or red? What would you change to draw a diagonal line from the upper left to the lower right? Where is the origin on the canvas? In what direction to the x and y axes increase?

## Modifications And Extensions

- Modify `drawRect:` to fill the screen with diagonal lines.
- Investigate the `CGContextAddEllipseInRect` function, and fill the screen with multiple colored circles.

## Resources

Interface Builder Help: Adding an Object to Your Interface https://developer.apple.com/library/ios/recipes/xcode_help-IB_objects_media/Chapters/AddingObject.html

UIKIt User Interface Catalog: Toolbars https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/UIKitUICatalog/UIToolbar.html

Interface Builder Help: Creating a Custom View https://developer.apple.com/library/ios/recipes/xcode_help-IB_objects_media/Chapters/CreatingaLiveViewofaCustomObject.html

UIView Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIView_Class/index.html

Drawing and Printing Guide for iOS: iOS Drawing Concepts http://developer.apple.com/library/ios/documentation/2DDrawing/Conceptual/DrawingPrintingiOS/GraphicsDrawingOverview/GraphicsDrawingOverview.html

View Programming Guide https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewPG_iPhoneOS/CreatingViews/CreatingViews.html

Quartz 2D Programming Guide http://developer.apple.com/library/ios/documentation/GraphicsImaging/Conceptual/drawingwithquartz2d/Introduction/Introduction.html

CGContext Reference https://developer.apple.com/library/ios/documentation/GraphicsImaging/Reference/CGContext/index.html

UIColor Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIColor_Class/index.html

Start Developing iOS Apps Today: Finding Information https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/FindingInformation.html