Quartz2D 学习

本节目标：绘制出各种表图，和扇形图，曲线图等功能，类似支付宝统计曲线，并且封装成控件。为了增加趣味性和降低难度，会增加一个绘制卡通人物的任务。

> *题外话：笔者在第一次使用Quartz2D时，是为了制作一个类似于官方的日历控件，5年过去了（2015年8月28日），但过程之艰辛历历在目，代码虽然已经不知去向，但是代码思路还在脑海中。笔者想表达的是这样几点：1、时间久了能记住的不是代码中的函数或方法，而是思路 2、代码只有真正一行一行敲过才是自己的代码 3、敲代码是漫长而痛苦的过程，但是往往伴随着喜悦，看到旧代码你会想起你奋斗的日子中的酸甜 4、请妥善保存你自己的代码，不要跟笔者一样想回忆时找不到*

章节一 <Quartz 2D简介>

<Quartz 2D简介>

- **Quartz 2D 是一个二维图形绘制引擎，支持iOS环境和Mac OS X环境，也就是说该框架是跨平台的。**你用该框架写的代码可以应用在iOS和Mac OS两个平台上。OpenGL是一个覆盖平台更广的3D绘制引擎，Apple公司2014年推出了新的Metal平台也是支持OpenGL的。
- **Quartz 2D** API可以实现许多功能，如基于路径的绘图、透明度、阴影、颜色管理、反锯齿、PDF文档生成和PDF元数据访问等。
- **Quartz 2D** API是**Core Graphics**框架的一部分，因此其中的很多数据类型和方法都是以CG开头的。会经常见到Quartz 2D（Quartz）和Core Graphics两个术语交互使用

<Core Graphics简介>

- **Core Graphic**框架是一组**基于C的API**，可以用于一切绘图操作，这个框架的重要性，仅次于UIKit和Foundation
- 当使用UIKit创建按钮、标签或者其他UIView的子类时，UIKit会用Core Graphics将这些元素绘制在屏幕上。此外，UIEvent（UIKit中的事件处理类）也会使用Core Graphics，用来帮助确定触摸事件在屏幕上所处的位置
- **因为UIKit依赖于Core Graphics，所以当引入<UIKit/Uikit.h>时，Core Graphics框架会被自动引入**，即UIKit内部已经引入了Core Graphics框架的主头文件：<CoreGraphics/CoreGraphics.h>
- 为了让开发者不必触及底层的Core Graphics的C接口，UIKit内部封装了Core Graphics

的一些API，可以快速生成通用的界面元素。但是，有时候直接利用Core Graphics的C接口是很有必要和很有好处的，比如**本节课最终我们将会封装出一个控件，这个控件无法用UIKit提供的控件制作**。

<概念一Graphics Context（图形上下文）>

> *Graphics Context是图形上下文,可以将其理解为一块画布,我们可以在上面进行绘画操作,绘制完成后,将画布放到我们的view中显示即可,view看作是一个画框.*

现实世界中画布可以有很多种材料，Context也同样有很多种不同的类型:

- Bitmap Graphics Context
- PDF Graphics Context
- Window Graphics Context
- Layer Context
- Post Graphics Context

绘图上下文Context的语言类型是 **CGContextRef**

相关函数（API）

1. **管理图形上下文**
    - CGContextFlush 强制所有挂起的绘图操作在一个窗口上下文中立即被渲染到目标设备。（如同把画布放入画框中）
    - CGContextGetTypeID 返回Quartz图形上下文的类型标识符。
    - CGContextRelease 图形上下文的引用计数-1。
    - CGContextRetain 图形上下文的引用计数+1。
    - CGContextSynchronize 将一个窗口的图像上下文内容更新，即所有的绘图操作都会在下次同步到窗口上
2. **保存和恢复当前图形状态**
    - CGContextSaveGState 将当前图形状态的副本PUSH到图形状态栈中（每个图形上下文维护一个保存图形状态的堆栈。需要注意的是，注意,当前绘图环境的所有方面都是图形状态的元素），保存的图像状态如下:
        - line width （画笔的宽度）
        - fill color space （填充色空间）
        - stroke color space （画笔颜色）
        - alpha value （透明度）
        - font size （绘制文字的大小）
        - text drawing mode （文字绘制方式）
        - 等等 更多见博客
    - CGContextRestoreGState 将当前图形状态设置为最近一次保存的状态，即恢复

状态。

3. **设置图形状态参数（修改画笔属性）**
    - CGContextSetFlatness 设置弯曲的路径中的图形上下文的准确性。
    - CGContextSetInterpolationQuality 设置图形上下文的插值质量水平。
    - CGContextSetLineCap 图形环境中的画线的端点的样式设置。
    - CGContextSetLineDash 设置图形上下文中的虚线的模式。
    - CGContextSetLineJoin 设置图像上下文中的接接线的样式。
    - **CGContextSetLineWidth 设置图像上下文中的线的宽度。**
    - CGContextSetMiterLimit 设置图像上下文中的连接线的斜接限制。
    - CGContextSetPatternPhase 设置一个上下文的段落模式。
    - CGContextSetFillPattern 在指定的图形上下文设置的填充图案模式。
    - CGContextSetRenderingIntent 在当前图形状态设置渲染意向。
    - CGContextSetShouldAntialias 设置图形上下文的抗锯齿开启或关闭。
    - CGContextSetStrokePattern 在指定的图形上下文设置描边图案。
4. **构建路径**

> *These functions are used to define the geometry of the current path. For more information on how paths are defined, see CGPath Reference. 翻译：这些函数是用来定义路径的几何形状。对于如何定义的路径的更多信息，请参阅CGPath参考。*

- **CGContextMoveToPoint 画笔移动到某点**
- **CGContextAddLines 绘制线条**
- CGContextAddLineToPoint
- **CGContextAddArc 绘制弧线 根据中心点和半径，顺逆时针方向，0为顺时针**

```
CGContextAddArc(context, 100, 100, 30, 0, M_PI, 1);
```

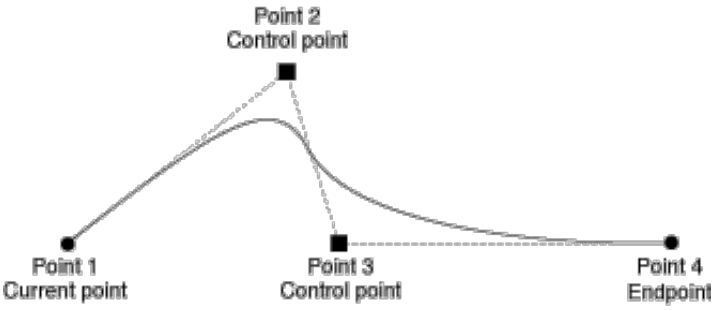
- **CGContextAddArcToPoint 从当前点A(150,50)绘制半径为N(50)的弧线到点C(130,150)，B(100,80)点为顶点** [链接详解]

代码示例

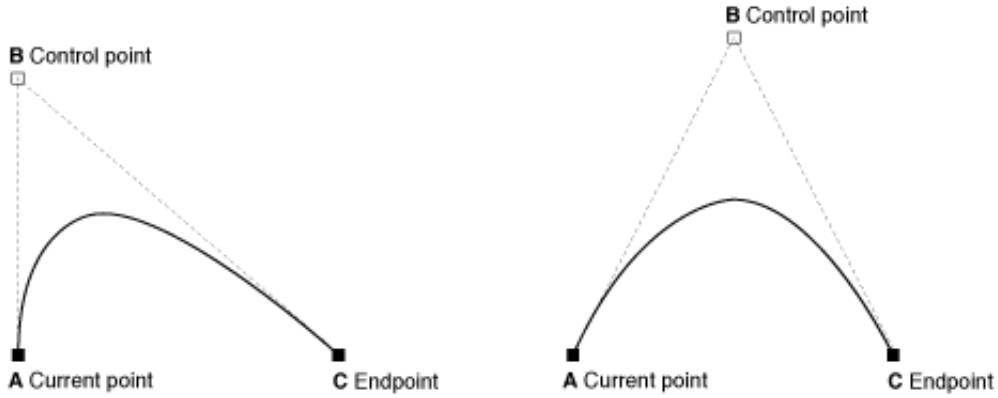
```
CGContextRef context=UIGraphicsGetCurrentContext();
CGContextSetRGBStrokeColor(context,1,0,0,1);
CGContextMoveToPoint(context,150,50);
CGContextAddLineToPoint(context,100,80);
CGContextAddLineToPoint(context,130,150);

CGContextMoveToPoint(context,150,50);//圆弧的起始点
CGContextAddArcToPoint(context,100,80,130,150,50);
CGContextStrokePath(context);
```

- **CGContextAddCurveToPoint 绘制3次贝塞尔（2个Control点）** 链接详解



- **CGContextAddQuadCurveToPoint 绘制2次贝塞尔曲线（1个Control点）** 链接详解



- **CGContextAddPath 添加路径**

- CGContextCopyPath
- **CGContextAddRect 绘制矩形**
- CGContextAddRects 添加多个矩形
- CGContextBeginPath 开始路径
- CGContextClosePath 结束路径
- CGContextAddEllipseInRect 在矩形内绘制椭圆

5. **绘画路径**

> *These functions are used to stroke along or fill in the current path.*
> *翻译：这些功能是用于边缘或填充在当前路径。*

- CGContextClearRect 清除矩形区域内绘制过的图形
- CGContextDrawPath 绘制路径
- CGContextEOFillPath 绘制路径（填充封闭区域，使用奇偶规则）
- CGContextFillPath 绘制路径（填充封闭区域，使用非零绕数规则）
- CGContextFillRect 绘制矩形（填充矩形）
- CGContextFillRects 绘制矩形（多个，填充矩形）
- CGContextFillEllipseInRect 绘制椭圆（填充椭圆）
- CGContextStrokePath 绘制线条
- CGContextStrokeRect 绘制矩形 （不填充，仅边框）
- CGContextStrokeRectWithWidth 绘制矩形 （不填充，仅边框，指定线条宽度）
- CGContextReplacePathWithStrokedPath
- CGContextStrokeEllipseInRect 绘制椭圆（填充椭圆）
- CGContextStrokeLineSegments 一些直线

6. **设置颜色，色彩空间和阴影值** （这里不再过多解释）

- CGContextSetAlpha
- CGContextSetCMYKFillColor
- CGContextSetFillColor

- CGContextSetCMYKStrokeColor
- CGContextSetFillColorSpace
- CGContextSetFillColorWithColor
- CGContextSetGrayFillColor
- CGContextSetGrayStrokeColor
- CGContextSetRGBFillColor
- CGContextSetRGBStrokeColor
- CGContextSetShadow
- CGContextSetShadowWithColor
- CGContextSetStrokeColor
- CGContextSetStrokeColorSpace
- CGContextSetStrokeColorWithColor

7. **转换用户空间**

绘制文字或图片时，默认是文字或文字的倒影，需要根据x轴进行矩阵变换，也就是说下面的函数在绘制文字和图片时会经常用到。相关链接

> *These functions allow you to examine and change the current transformation matrix (CTM) in a graphics context.*
> *翻译：这些功能允许你检查和更改图形上下文的当前转换矩阵（CTM）。*

- CGContextConcatCTM
- CGContextGetCTM
- CGContextRotateCTM
- CGContextScaleCTM
- CGContextTranslateCTM

代码示例

```
  UIImage *image=[UIImage imageNamed:@"a.png"];

  CGContextRef context = UIGraphicsGetCurrentContext();

  CGContextTranslateCTM(context, 0,  image.size.height);
  CGContextScaleCTM(context, 1.0, -1.0);

  CGContextDrawImage(context, CGRectMake(0.0, 0.0,
image.size.width, image.size.height), image.CGImage);
```

更多函数可查阅苹果开发者官方文档

Sample Code下载