imlifengfeng (http://www.imlifengfeng.com/blog)

iOS Quartz2D详解

2017/2/6 3 Comments 2,903 Views 6 Times

一、概述

Quartz2D的API是纯C语言的，它是一个二维绘图引擎，同时支持iOS和Mac系统。Quartz2D的API来自于Core Graphics框架，数据类型和函数基本都以CG作为前缀。通常，我们可以使用系统提供的控件去完成大部分UI，但是有些UI界面极其复杂、而且比较个性化，用普通的UI控件无法实现，这时可以利用Quartz 2D技术将控件内部的结构画出来，类似自定义控件。其实，iOS中大部分控件的内容都是通过Quartz2D画出来的，因此，Quartz2D在iOS开发中很重要的一个价值是：自定义view（自定义UI控件）。

Quartz 2D能完成的工作：

绘制图形：线条\三角形\矩形\圆\弧等；

绘制文字；

绘制\生成图片(图像)；

读取\生成PDF；

截图\裁剪图片;

自定义UI控件;

… …

二、图形上下文（Graphics Context）

图形上下文（Graphics Context）：是一个CGContextRef类型的数据。

图形上下文的作用:

（1）保存绘图信息、绘图状态

（2）决定绘制的输出目标（绘制到什么地方去？）

（输出目标可以是PDF文件、Bitmap或者显示器的窗口上）

相同的一套绘图序列，指定不同的Graphics Context，就可将相同的图像绘制到不同的目标上。

Quartz2D提供了以下几种类型的Graphics Context:

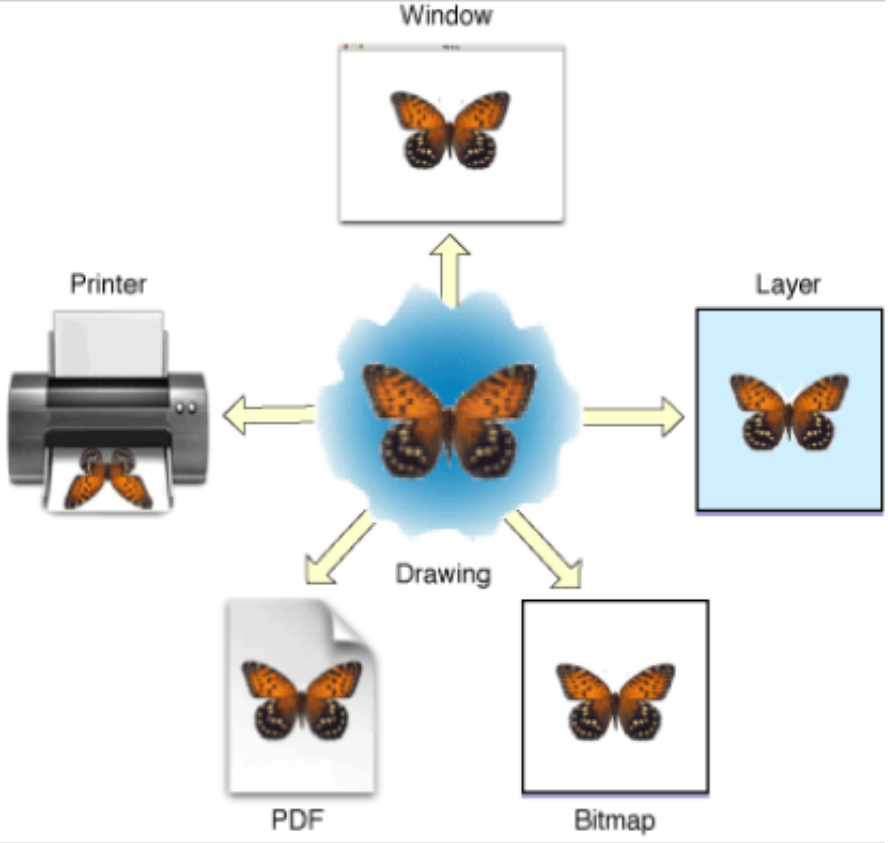
（1）Bitmap Graphics Context

（2）PDF Graphics Context

（3）Window Graphics Context

（4）Layer Graphics Context

（5）Printer Graphics Context



将当前的上下文copy一份,保存到栈顶(那个栈叫做”图形上下文栈”):

```
1 void CGContextSaveGState(CGContextRef c)
```

将栈顶的上下文出栈,替换掉当前的上下文（清空之前对于上下文设置）：

```
1 void CGContextRestoreGState(CGContextRef c)
```

三、使用Quartz2D自定义View

1、Quartz2D自定义view

如何利用Quartz2D自定义view？（自定义UI控件）如何利用Quartz2D绘制东西到view上？

首先，得有图形上下文，因为它能保存绘图信息，并且决定着绘制到什么地方去。

其次，那个图形上下文必须跟view相关联，才能将内容绘制到view上面。

自定义view的步骤：

（1）新建一个类，继承自UIView

（2）实现- (void)drawRect:(CGRect)rect方法，然后在这个方法中

（a）取得跟当前view相关联的图形上下文

```
CGContextRef ctx = UIGraphicsGetCurrentContext();
```

（b）绘制相应的图形内容

例如：画1/4圆（扇形）

```
CGContextMoveToPoint(ctx, 100, 100);

CGContextAddLineToPoint(ctx, 100, 150);

CGContextAddArc(ctx, 100, 100, 50, -M_PI_2, M_PI, 1);

CGContextClosePath(ctx);


[[UIColor redColor] set];
```

（3）利用图形上下文将绘制的所有内容渲染显示到view上面

```
CGContextFillPath(ctx);
```

注：

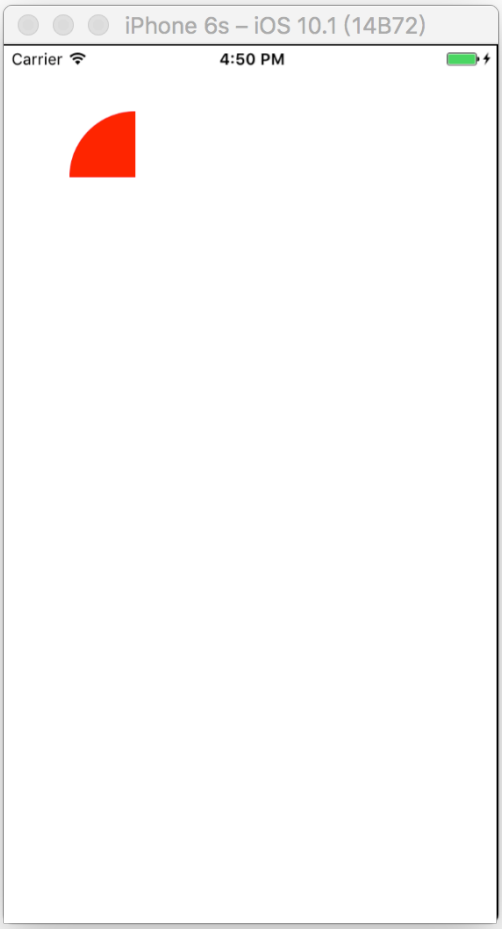
```
//M_PI的含义:π
//M_PI * 2的含义:2π

//M_PI_2的含义:π/2
//M_PI / 2的含义:π/2

// 画的图形路径
```

```objc
//bezierPathWithArcCenter:弧所在的圆心
//radius:圆的半径
//startAngle:开始角度,圆的最右侧为0度
//endAngle:截至角度,向下为正,向上为负.
//clockwise:时针的方向,yes:顺时针 no:逆时针
UIBezierPath *path = [UIBezierPath bezierPathWithArcCenter:self.center r
adius:radius startAngle:startA endAngle:endA clockwise:NO];
```

完整代码:

```objc
//
//  MyView.m
//  Quartz2DTest
//
//  Created by 李峰峰 on 2017/2/6.
//  Copyright © 2017年 李峰峰. All rights reserved.
//

#import "MyView.h"

@implementation MyView

- (instancetype)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        self.backgroundColor = [UIColor whiteColor];//设置背景为白色，为了
便于观察绘制后的效果
    }
    return self;
}


- (void)drawRect:(CGRect)rect {

    CGContextRef ctx = UIGraphicsGetCurrentContext();

    CGContextMoveToPoint(ctx, 100, 100);
    CGContextAddLineToPoint(ctx, 100, 150);
    CGContextAddArc(ctx, 100, 100, 50, -M_PI_2, M_PI, 1);
    CGContextClosePath(ctx);

    [[UIColor redColor] set];

    CGContextFillPath(ctx);

}

@end
```

运行效果:



2、核心方法drawRect:

- 为什么要实现drawRect:方法才能绘图到view上?
  - 因为在drawRect:方法中才能取得跟view相关联的图形上下文
- drawRect:方法在什么时候被调用?
  - 当view第一次显示到屏幕上时（被加到UIWindow上显示出来）
  - 调用view的setNeedsDisplay或者setNeedsDisplayInRect:时.
- 注意4点:
  - 手动调用drawRect:方法，不会自动创建跟View相关联的上下文。应该
    调用setNeedsDisplay方法,系统底层会自动调用drawRect，告诉系统重新绘制View.这样，系统底层会自动创建跟View相关联的上下文
  - setNeedsDisplay底层会调用drawRect,并不是立马调用的.只是设了

一个调用的标志.调用时刻是等下一次屏幕刷新时才去调用drawRect。屏幕每一秒刷新30-60秒次,所以1秒调用drawRect方法大概30-60次,速度非常快哦

- view内部有个layer（图层）属性，drawRect:方法中取得的是一个Layer Graphics Context，因此，绘制的东西其实是绘制到view的layer上去了
- View之所以能显示东西，完全是因为它内部的layer

3、Quartz2D绘图的代码步骤

第一步：获得图形上下文：

```
CGContextRef ctx = UIGraphicsGetCurrentContext();
```

第二步：拼接路径（下面代码是绘制一条线段）：

```
CGContextMoveToPoint(ctx, 10, 10);
CGContextAddLineToPoint(ctx, 100, 100);
```

第三步：绘制路径：

```
CGContextStrokePath(ctx); // CGContextFillPath(ctx);
```

四、Quartz2D重要函数

1、常用拼接路径函数

新建一个起点

```
void CGContextMoveToPoint(CGContextRef c, CGFloat x, CGFloat y)
```

添加新的线段到某个点

```
1 void CGContextAddLineToPoint(CGContextRef c, CGFloat x, CGFloat y)
```

添加一个矩形

```
1 void CGContextAddRect(CGContextRef c, CGRect rect)
```

添加一个椭圆

```
1 void CGContextAddEllipseInRect(CGContextRef context, CGRect rect)
```

添加一个圆弧

```
1 void CGContextAddArc(CGContextRef c, CGFloat x, CGFloat y,
2
3 CGFloat radius, CGFloat startAngle, CGFloat endAngle, int clockwise)
```

2、常用绘制路径函数

一般以CGContextDraw、CGContextStroke、CGContextFill开头的函数，都是用来绘制路径的。

Mode参数决定绘制的模式

```
1 void CGContextDrawPath(CGContextRef c, CGPathDrawingMode mode)
```

绘制空心路径

```
1 void CGContextStrokePath(CGContextRef c)
```

绘制实心路径

```
1 void CGContextFillPath(CGContextRef c)
```

3、矩阵操作函数

利用矩阵操作，能让绘制到上下文中的所有路径一起发生变化。

缩放：

```
1 void CGContextScaleCTM(CGContextRef c, CGFloat sx, CGFloat sy)
```

旋转：

```
1 void CGContextRotateCTM(CGContextRef c, CGFloat angle)
```

平移：

```
1 void CGContextTranslateCTM(CGContextRef c, CGFloat tx, CGFloat ty)
```

4、其他常用函数

设置线段宽度

```
CGContextSetLineWidth(ctx, 10);
```

设置线段头尾部的样式

```
CGContextSetLineCap(ctx, kCGLineCapRound);
```

设置线段转折点的样式

```
CGContextSetLineJoin(ctx, kCGLineJoinRound);
```

设置颜色

```
CGContextSetRGBStrokeColor(ctx, 1, 0, 0, 1);
```

五、Quartz2D的内存管理

关于Quartz2D内存管理，有以下原则：

（1）使用含有“Create”或“Copy”的函数创建的对象，使用完后必须释放，否则将导致内存泄露。

（2）使用不含有“Create”或“Copy”的函数获取的对象，则不需要释放

（3）如果retain了一个对象，不再使用时，需要将其release掉。

（4）可以使用Quartz 2D的函数来指定retain和release一个对象。例如，如果创建了一个CGColorSpace对象，则使用函数CGColorSpaceRetain和CGColorSpaceRelease来retain和release对象。

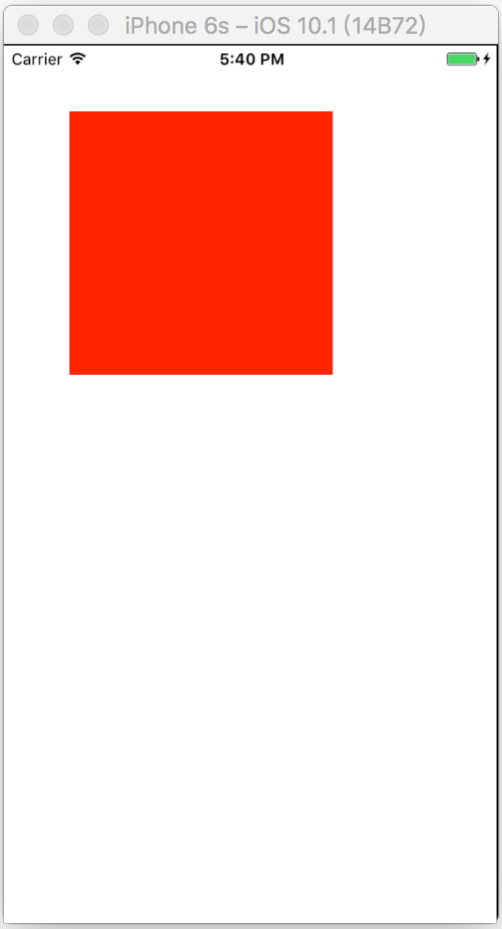
（5）也可以使用Core Foundation的CFRetain和CFRelease。注意不能传递NULL值给这些函数。

六、Quartz2D使用案例

1、画矩形、正方形

```
- (void)drawRect:(CGRect)rect {

    //1.获取上下文
    CGContextRef ctx = UIGraphicsGetCurrentContext();
    //2.描述路径
    UIBezierPath *path = [UIBezierPath bezierPathWithRect:CGRectMake(50, 50, 200, 200)];
    //3.把路径添加到上下文
    CGContextAddPath(ctx, path.CGPath);

    [[UIColor redColor] set];// 路径的颜色

    //4.把上下文的内容渲染到View的layer.
    //CGContextStrokePath(ctx);// 描边路径
    CGContextFillPath(ctx);// 填充路径

}
```

运行效果:



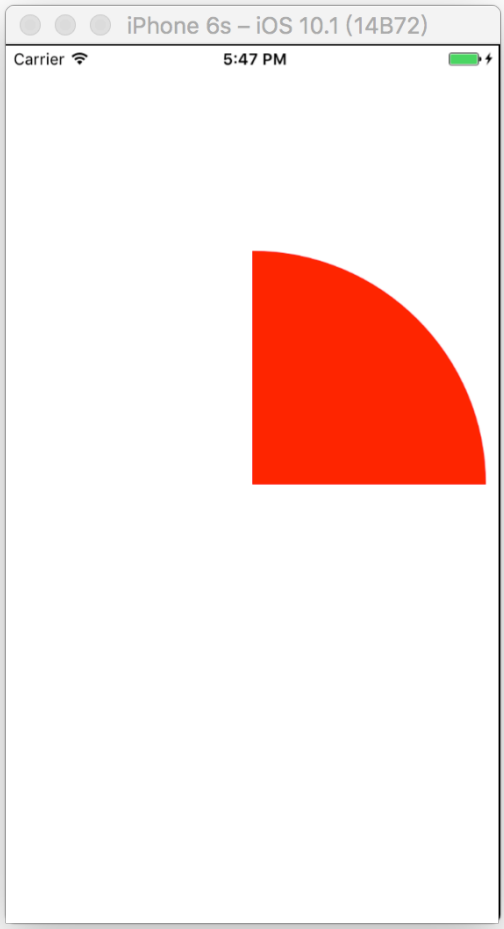
2、画扇形

除上面“二、使用Quartz2D自定义View”中的方法外，也可以使用OC中自带画图方法实现，如下：

```
- (void)drawRect:(CGRect)rect {

    CGPoint center = CGPointMake(rect.size.width * 0.5, rect.size.height * 0.5);
    CGFloat radius = rect.size.width * 0.5 - 10;
    CGFloat startA = 0;
    CGFloat endA = -M_PI_2;
    // 画弧的路径
    UIBezierPath *path = [UIBezierPath bezierPathWithArcCenter:center radius:radius startAngle:startA endAngle:endA clockwise:NO];
    // 添加一根线到圆心
    [path addLineToPoint:center];
    // 闭合路径
    [path closePath];
    // 路径颜色
```

```
14      [[UIColor redColor] set];
15      // 填充路径
16      [path fill];
17      // 描边路径
18      //[path stroke];
19
20  }
```

运行效果:



注:

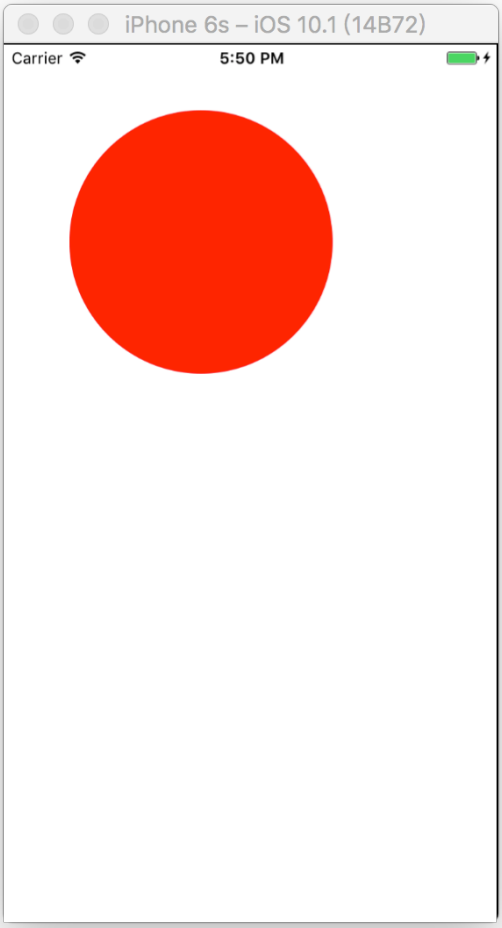
判断一个点是否在一个矩形框内

```
1  CGRectContainsPoint (rect,point) ;//判断point这个点是否在rect这个矩形框内
```

3、画圆形

```objc
- (void)drawRect:(CGRect)rect {

    //1.获取上下文
    CGContextRef ctx = UIGraphicsGetCurrentContext();
    //2.描述路径
    // cornerRadius:圆角半径。矩形的宽高都为200，如果圆角为100，那么两个角之间弧线上任意一点到矩形中心的距离都为100,所以为圆形
    UIBezierPath *path = [UIBezierPath bezierPathWithRoundedRect:CGRectMake(50, 50, 200, 200) cornerRadius:100];
    //3.把路径添加到上下文
    CGContextAddPath(ctx, path.CGPath);

    [[UIColor redColor] set];// 路径的颜色

    //4.把上下文的内容渲染到View的layer.
    // CGContextStrokePath(ctx);// 描边路径
    CGContextFillPath(ctx);// 填充路径

}
```

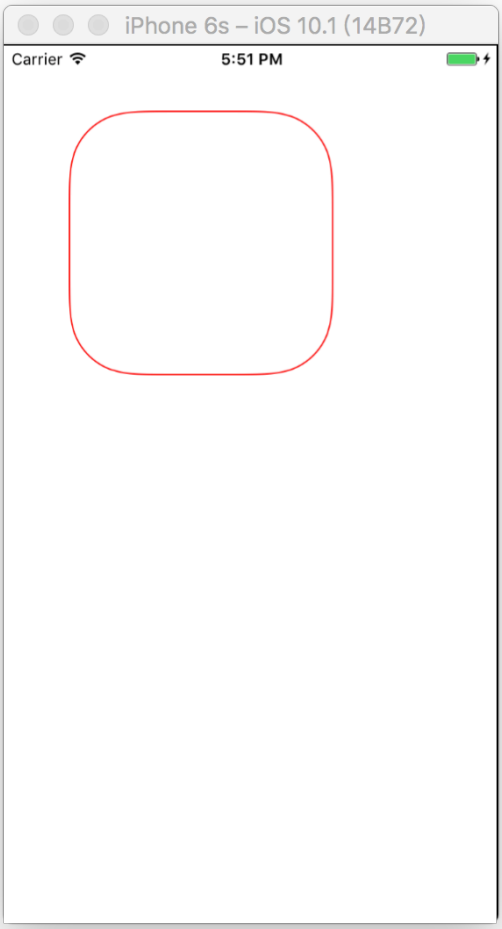
运行效果:



4、画圆角矩形

```
- (void)drawRect:(CGRect)rect {

    //1.获取上下文
    CGContextRef ctx = UIGraphicsGetCurrentContext();
    //2.描述路径
    // cornerRadius:圆角半径。
    UIBezierPath *path = [UIBezierPath bezierPathWithRoundedRect:CGRectMake(50, 50, 200, 200) cornerRadius:50];
    //3.把路径添加到上下文
    CGContextAddPath(ctx, path.CGPath);

    [[UIColor redColor] set];// 路径的颜色

    //4.把上下文的内容渲染到View的layer.
    CGContextStrokePath(ctx);// 描边路径
    //CGContextFillPath(ctx);// 填充路径

}
```

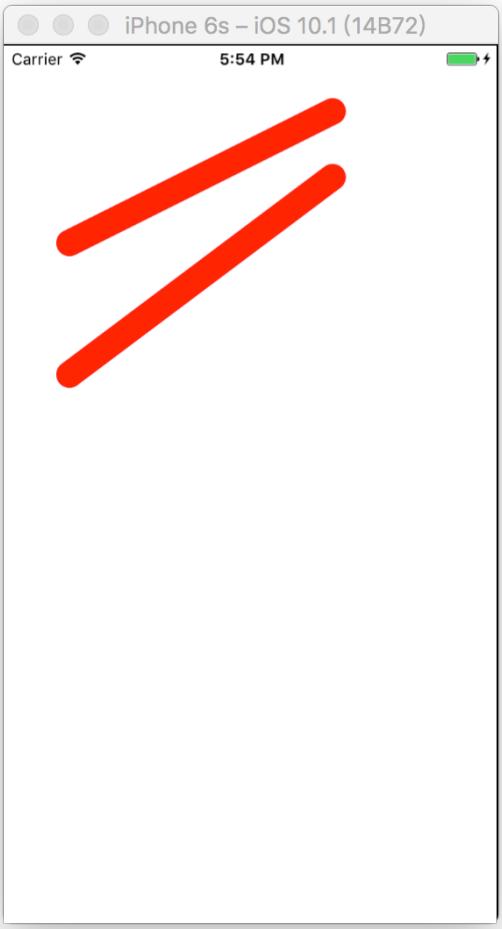
运行效果:



5、画直线

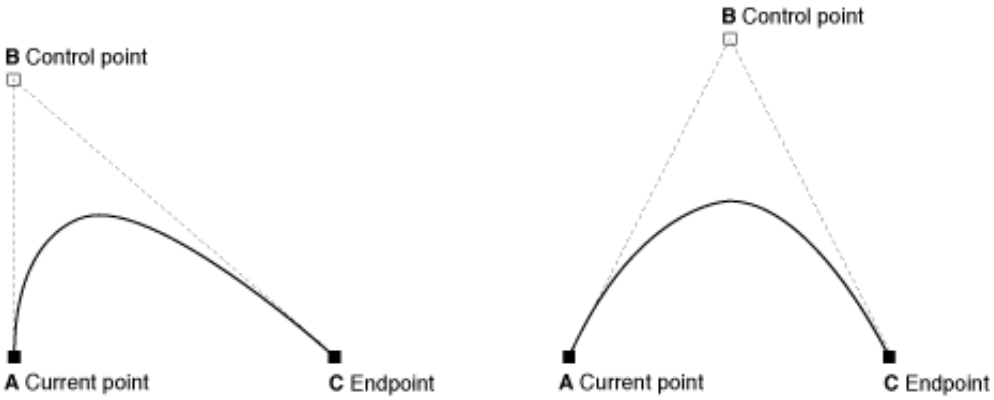
```objc
- (void)drawRect:(CGRect)rect {

    //1.获取跟View相关联的上下文(uigraphics开头)
    CGContextRef ctx = UIGraphicsGetCurrentContext();

    //2.描述路径
    //一条路径可以绘制多条线 路径:path      路径绘制多条线:path使用了两次（2次的起点到终点），都是将线添加到某个点
    UIBezierPath *path = [UIBezierPath bezierPath];
    //设置起点
    [path moveToPoint:CGPointMake(50, 150)];
    //添加一根线Line到某个点
    [path addLineToPoint:CGPointMake(250, 50)];

    //画第二根线
    [path moveToPoint:CGPointMake(50, 250)];
    [path addLineToPoint:CGPointMake(250, 100)];

    //设置线宽
    CGContextSetLineWidth(ctx, 20);
    //设置线的连接样式
    CGContextSetLineJoin(ctx, kCGLineJoinBevel);
    //设置线的顶角样式
    CGContextSetLineCap(ctx, kCGLineCapRound);// 圆角线条
    //设置线条颜色
    [[UIColor redColor] set];

    //3.把路径添加到上下文
    CGContextAddPath(ctx, path.CGPath);
    //4.把上下文当中绘制的内容渲染到跟View关联的layer
    CGContextStrokePath(ctx);

}
```

运行效果:



6、画曲线

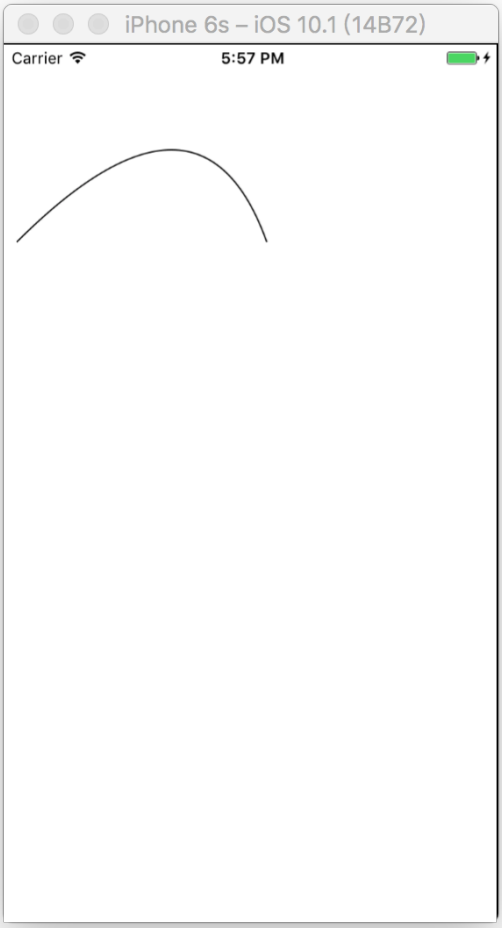
本塞尔曲线原理:




```
- (void)drawRect:(CGRect)rect {

    //1.获取跟View相关联的上下文.】
```

```
4       CGContextRef ctx = UIGraphicsGetCurrentContext();
5       //2.描述路径
6       UIBezierPath *path = [UIBezierPath  bezierPath];
7       //画曲线,设置起点.还有一个控制点(用来控制曲线的方向跟弯曲程度)
8       //设置起点
9       [path moveToPoint:CGPointMake(10, 150)];
10      //添加一要曲线到某个点
11      [path addQuadCurveToPoint:CGPointMake(200, 150) controlPoint:CGPoint
    Make(150, 10)];
12      //3.把路径添加到上下文当中
13      CGContextAddPath(ctx, path.CGPath);
14      //4.把上下文的内容渲染View上
15      CGContextStrokePath(ctx);
16
17  }
```

运行效果:

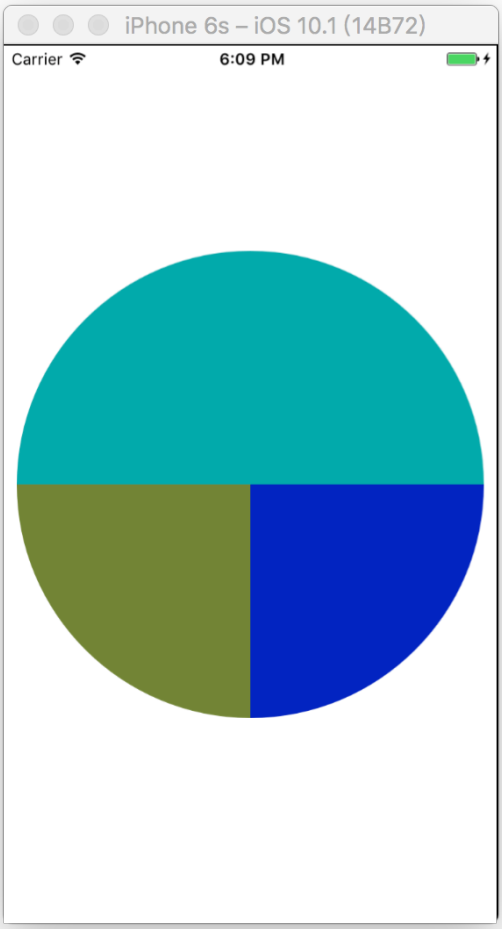


7、画饼图

方法1:

```objc
- (void)drawRect:(CGRect)rect {

    NSArray *dataArray =  @[@25,@25,@50];
    // 画弧
    CGPoint center = CGPointMake(rect.size.width * 0.5, rect.size.height * 0.5);
    // 半径
    CGFloat radius = rect.size.width * 0.5 - 10;

    CGFloat startA = 0;
    CGFloat angle = 0;
    CGFloat endA = 0;

    for (NSNumber *num in dataArray) {
        startA = endA;
        // 遍历出第一个对象25,angle =25/100 *2π, 即angle = π/2, 所以为1/4圆,
        angle = num.intValue / 100.0 * M_PI * 2;
        // 截至角度= 开始的角度+ 遍历出的对象所占整个圆形的角度
        endA = startA + angle;
        // 顺势针画贝塞尔曲线
        UIBezierPath *path = [UIBezierPath bezierPathWithArcCenter:center radius:radius startAngle:startA endAngle:endA clockwise:YES];
        // 设置随机颜色
        [[self randomColor] set];
        // 添加一根线到圆心
        [path addLineToPoint:center];
        // 填充路径
        [path fill];
        // 描边路径
//      [path stroke];
    }

}


/**
 生成随机颜色

 @return UIColor
 */
-(UIColor *)randomColor{

    CGFloat redLevel    = rand() / (float) RAND_MAX;
    CGFloat greenLevel  = rand() / (float) RAND_MAX;
    CGFloat blueLevel   = rand() / (float) RAND_MAX;

    return [UIColor colorWithRed: redLevel green: greenLevel blue: blueLevel alpha: 1.0];
}
```

运行效果:



方法二:

```
1  - (void)drawRect:(CGRect)rect {
2
3      CGPoint center = CGPointMake(self.bounds.size.width * 0.5, self.boun
   ds.size.height * .5);
4      CGFloat radius = self.bounds.size.width * 0.5 - 10;
5      CGFloat startA = 0;
6      CGFloat endA = 25 / 100.0 * M_PI * 2;
7      UIBezierPath *path = [UIBezierPath bezierPathWithArcCenter:center ra
   dius:radius startAngle:startA endAngle:endA clockwise:YES];
8      [[UIColor redColor] set];
9      //添加一根线到圆心
10     [path addLineToPoint:center];
11     [path fill];
12
13     //第二个扇形
14     startA = endA;
15     CGFloat angle = 25 / 100.0 * M_PI * 2;
16     endA = startA + angle;
17     UIBezierPath *path2 = [UIBezierPath bezierPathWithArcCenter:center r
```

```
adius:radius startAngle:startA endAngle:endA clockwise:YES];
    [[UIColor greenColor] set];
    //添加一根线到圆心
    [path2 addLineToPoint:center];
    [path2 fill];

    startA = endA;
    angle = 50 / 100.0 * M_PI * 2;
    endA = startA + angle;
    UIBezierPath *path3 = [UIBezierPath bezierPathWithArcCenter:center r
adius:radius startAngle:startA endAngle:endA clockwise:YES];
    [[UIColor blueColor] set];
    //添加一根线到圆心
    [path3 addLineToPoint:center];
    [path3 fill];

}


/**
 生成随机颜色

 @return UIColor
 */
-(UIColor *)randomColor{

    CGFloat redLevel    = rand() / (float) RAND_MAX;
    CGFloat greenLevel  = rand() / (float) RAND_MAX;
    CGFloat blueLevel   = rand() / (float) RAND_MAX;

    return [UIColor colorWithRed: redLevel green: greenLevel blue: blueL
evel alpha: 1.0];
}
```

注:

如果想实现点击以下变换颜色可以加上如下代码:

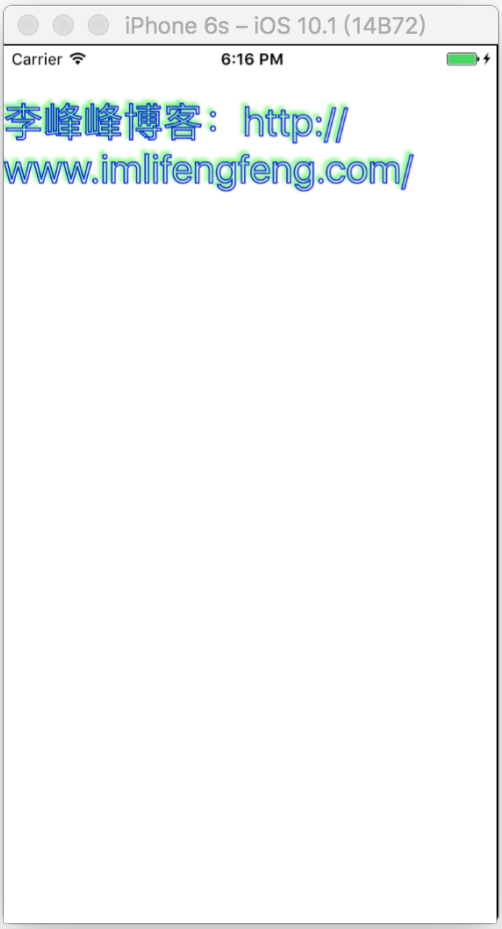
```
- (void)touchesBegan:(NSSet<UITouch *> *)touches withEvent:(UIEvent *)eve
nt {

    //重绘
    [self setNeedsDisplay];

}
```

8、绘制文字

```
- (void)drawRect:(CGRect)rect {

```

```
3       NSString *str = @"李峰峰博客: http://www.imlifengfeng.com/";
4
5       NSMutableDictionary *dict = [NSMutableDictionary dictionary];
6       //设置字体
7       dict[NSFontAttributeName] = [UIFont systemFontOfSize:30];
8       //设置颜色
9       dict[NSForegroundColorAttributeName] = [UIColor redColor];
10      //设置描边
11      dict[NSStrokeColorAttributeName] = [UIColor blueColor];
12      dict[NSStrokeWidthAttributeName] = @3;
13      //设置阴影
14      NSShadow *shadow = [[NSShadow alloc] init];
15      shadow.shadowColor = [UIColor greenColor];
16      shadow.shadowOffset = CGSizeMake(-2, -2);
17      shadow.shadowBlurRadius = 3;
18      dict[NSShadowAttributeName] = shadow;
19
20      //设置文字的属性
21      //drawAtPoint不会自动换行
22      //[str drawAtPoint:CGPointMake(0, 0) withAttributes:dict];
23      //drawInRect会自动换行
24      [str drawInRect:self.bounds withAttributes:dict];
25
26  }
```

运行效果:



9、加水印

```objc
//
//  ViewController.m
//  Quartz2DTest
//
//  Created by 李峰峰 on 2017/2/6.
//  Copyright © 2017年 李峰峰. All rights reserved.
//

#import "ViewController.h"
#import "MyView.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];

    UIImageView *myImageView = [[UIImageView alloc]initWithFrame:CGRectMake(0, 0, [UIScreen mainScreen].bounds.size.width, [UIScreen mainScreen].bounds.size.height)];
    [self.view addSubview:myImageView];


    //生成一张图片
    //0.加载图片
    UIImage *oriImage = [UIImage imageNamed:@"test"];
    //1.创建位图上下文(size:开启多大的上下文,就会生成多大的图片)
    UIGraphicsBeginImageContext(oriImage.size);
    //2.把图片绘制到上下文当中
    [oriImage drawAtPoint:CGPointZero];
    //3.绘制水印(虽说UILabel可以快速实现这种效果，但是我们也可以绘制出来)
    NSString *str = @"李峰峰博客";


    NSMutableDictionary *dict = [NSMutableDictionary dictionary];
    dict[NSFontAttributeName] = [UIFont systemFontOfSize:20];
    dict[NSForegroundColorAttributeName] = [UIColor redColor];

    [str drawAtPoint:CGPointZero withAttributes:dict];
    //4.从上下文当中生成一张图片
    UIImage *newImage = UIGraphicsGetImageFromCurrentImageContext();
    //5.关闭位图上下文
    UIGraphicsEndImageContext();


    myImageView.image = newImage;

}

@end
```

运行效果:

10、屏幕截图

```
-(void)touchesBegan:(NSSet<UITouch *> *)touches withEvent:(UIEvent *)event {

    //生成图片
    //1.开启一个位图上下文
    UIGraphicsBeginImageContext(self.view.bounds.size);
    //2.把View的内容绘制到上下文当中
    CGContextRef ctx =  UIGraphicsGetCurrentContext();
    //UIView内容想要绘制到上下文当中，必须使用渲染的方式
    [self.view.layer renderInContext:ctx];
    //3.从上下文当中生成一张图片
    UIImage *newImage = UIGraphicsGetImageFromCurrentImageContext();
    //4.关闭上下文
    UIGraphicsEndImageContext();
    //把图片转成二进制流
    //NSData *data = UIImageJPEGRepresentation(newImage, 1);
```

```
16      NSData *data = UIImagePNGRepresentation(newImage);
17
18      [data writeToFile:@"/Users/lifengfeng/Downloads/imlifengfeng.jpg" at
    omically:YES];
19
20  }
```

运行效果:

截图而已，就跟普通截图一样，自己试。

以上就是关于Quartz2D一些常用的案例，Quartz2D还可以实现更多效果，具体的根据需求去实现。

原创文章，转载请注明： *转载自李峰峰博客 (http://www.imlifengfeng.com/blog/)*

本文链接地址: *iOS Quartz2D详解 (http://www.imlifengfeng.com/blog/?p=514)*

IOS (HTTP://WWW.IMLIFENGFENG.COM/BLOG/?TAG=IOS)

QUARTZ2D (HTTP://WWW.IMLIFENGFENG.COM/BLOG/?TAG=QUARTZ2D)

< 上一篇 (http://www.imlifengfeng.com/blog/?p=512)

下一篇 > (http://www.imlifengfeng.com/blog/?p=528)

Pacino

2017年2月8日 上午8:57 (http://www.imlifengfeng.com/blog/?p=514#comment-124)

博主你的博客在微信打不开，去申诉一下吧。 😂

回复 (http://www.imlifengfeng.com/blog/?p=514&replytocom=124#respond)

imlifengfeng

2017年2月8日 上午9:18 (http://www.imlifengfeng.com/blog/?p=514#comment-126)

谢谢反馈，可能是微信误报，已经申诉

回复 (http://www.imlifengfeng.com/blog/?p=514&replytocom=126#respond)

linwei

2017年2月8日 下午2:52 (http://www.imlifengfeng.com/blog/?p=514#comment-127)

直接使用UIGraphicsBeginImageContext的话，容易像素化模糊，UIGraphicsBeginImageContextWithOptions(orilmage.size, NO, [UIScreen mainScreen].scale)这样可能更好些

回复 (http://www.imlifengfeng.com/blog/?p=514&replytocom=127#respond)

发表评论

电子邮件地址不会被公开。 必填项已用*标注

昵称 *

邮箱 *

网站

亲，评论前先来个小计算吧： – 3 = 6

发表评论

(HTTP://WEIBO.COM/DEVFENG/HOME)

(HTTPS://TWITTER.COM/IMLIFENGFENG)

(HTTPS://GITHUB.COM/IMLIFENGFENG)

(HTTP://WWW.IMLIFENGFENG.COM/BLOG/?FEED=ATOM)