一、view的触摸事件

```
/** 目标
 * 1.了解iOS中的三种事件类型 触摸事件、加速计事件、远程事件
 * 2.掌握触摸事件的4个方法
touchBegin/touchesMoved/touchesEnded/touchesEnded/touchesCancelled
 */
```

"什么是响应者对象？"

» 继承了UIResponder的对象就是响应者对象，只有响应者对象才能够接收并处理事件

» UIApplication,UIViewController,UIView都继承了UIResponder,所以他们都能够接收事件处理

"响应者对象有哪些事件？" PPT

"响应者对象中触摸事件"

1.触摸事件的4个方法touchBegin/touchesMoved/touchesEnded/touchesEnded/touchesCancelled

2.触摸方法的touches对象存放着UITouch【触摸】对象

3.理解触摸对象touch.view[触摸的view]/touch.tapaccount[触摸的次数]/locationInView[触摸的位置]/previousLocationInView[上一次位置]

二、涂鸦(Quartz2D方法)

```
/**
 * 掌握触摸方法与绘图的结合使用
 */
```

"途鸦1-使用Quartz2d"

步骤
A.绘图功能
1.自定一个View【PaintView】
2.添加一个类型为组的pointsOfAllLine属性，用于存储所有笔划的点
3.在touchBegin方法初始化一个数组pointsOfALine,添加到pointsOfAllLine中
*pointsOfAllLine用于存储 "所有笔划" 的点
*pointsOfALine 用于存储 "一个笔划" 的点
4.在touchMoved中，获取"当前笔划" 并添加的 "当前点" 然后调用drawRect方法
5.在drawRect方法中实现 "笔划绘制"

B.返回功能
1.在自定的View添加back方法，在bak方法实现 移除最后一条线的点，并重绘

C.清除功能
1.在自定的View添加clear方法，在clear方法实现 移除所有线的点，并重绘

D.设置当前绘图的颜色
1.在自定View里添加一个数组colorsOfAllLine,用于存储所有笔划的颜色
2.在自定View里添加一个数组currentColor,用于存储当前笔划的颜色
3.在touchBegin方法，往colorsOfAllLine添加当前笔划的颜色

4.在drawRect方法，对应笔画获取对应颜色，并设置当前上下文的颜色

E.图片保存
1.调用之前写好的截图工具类即可

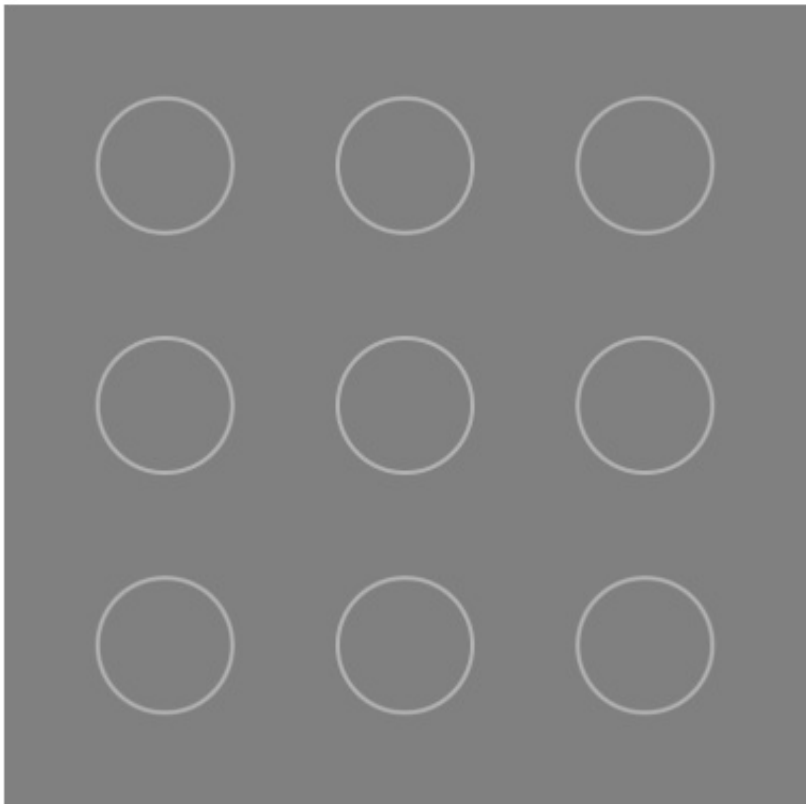
三、途鸦(UIBezierPath)
```
/**目标
 * 掌握UIBezierPath的使用,UIBezierPath 贝塞尔曲线
 * UIBezierPath是对Quartz2d 中路径封装的 OC对象
 */
```

UIBezierPath的使用步骤
1.创建一个UIBezierPath对象
```
*UIBezierPath *path = [UIBezierPath bezierPath];
```
2.设置起点
```
*[path moveToPoint:location];
```
3.设置连接点
```
[path addLineToPoint:location];
```

4.设置线宽，头尾样式、连接样式
```
//调用UIBezierPath的相应属性即可
*path.lineWidth = 2;
*path.lineJoinStyle = kCGLineJoinRound;
*path.lineCapStyle = kCGLineCapRound
```
4.渲染
```
*[path stroke]
```

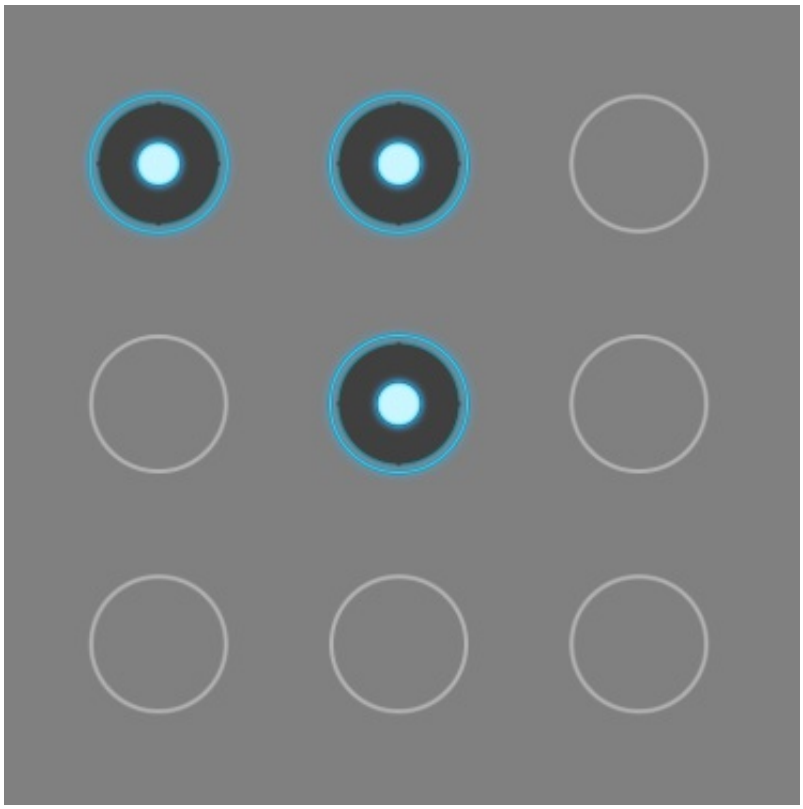
四、手势解锁(圆选中)
```
/**
 * 掌握在平时开发中，掌握的手势解锁原理
 */
```

步骤
A:排版
1.自定义一个View
2.在自定义View的initWithFrame方法中，初始化9个按钮，并设置 "正常状态下的图片"
3.在layoutSubviews对9个按钮重新设置位置与尺寸

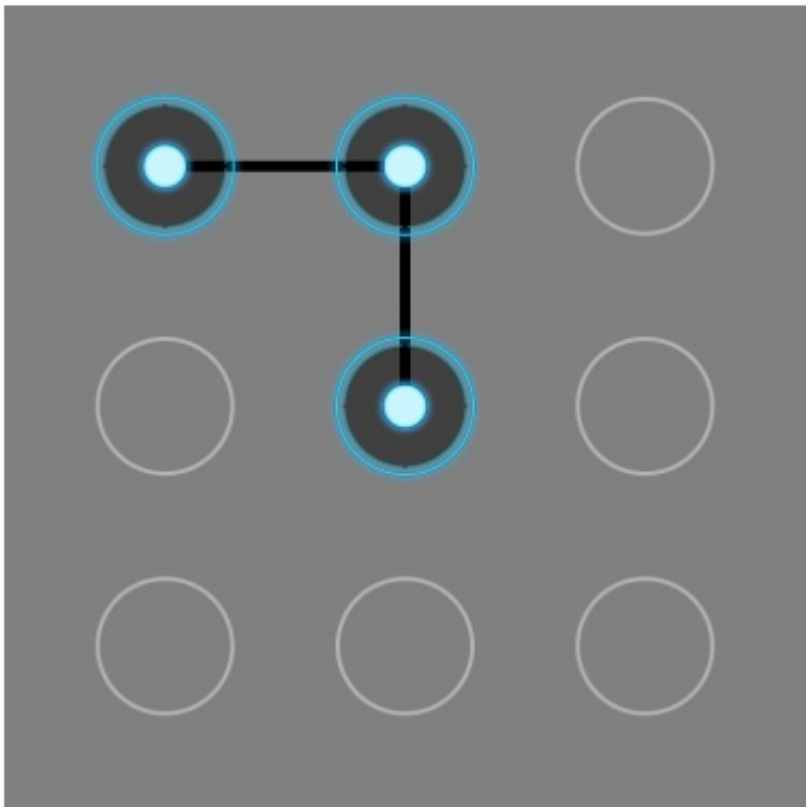


B:监听按钮选中状态

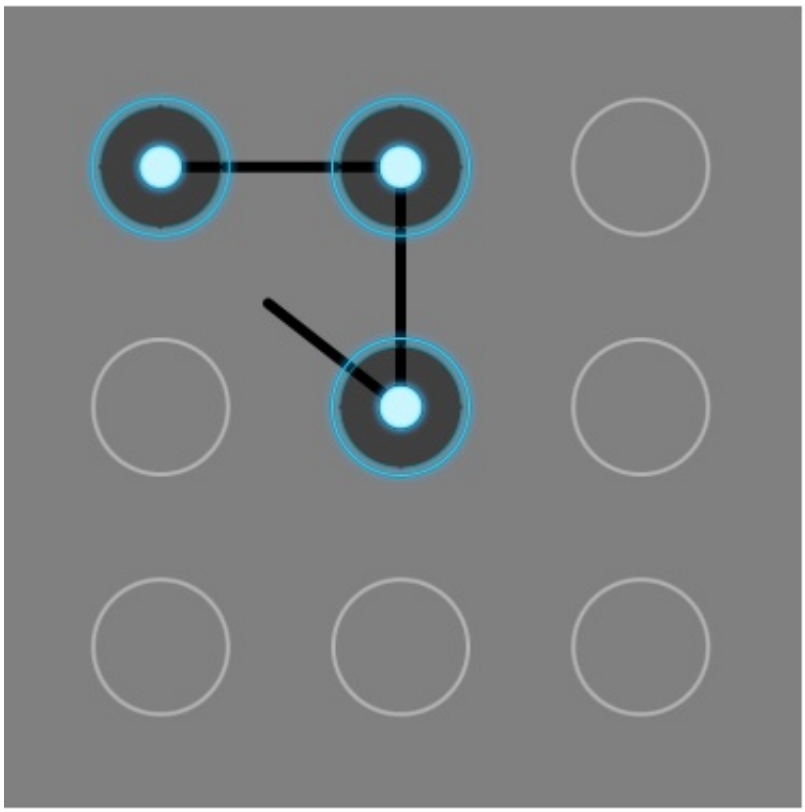
1.在初始化按钮时，设置按钮不交互userInteractionEnabled，并设置按钮的选中状态图片
2.在touchBegin和touchMove中都要实现下而两个步骤
*获取当前触摸点
*遍历所有按钮，判断当前 "触摸点" 是否在按钮范围内，获取 "触摸" 点下按钮
*定义一个选中按钮的数组属性【seletedBtns】,将选中的的按钮放入一个数组
*如果当前触摸到的按钮已经在数组中，就不需要添加
3.在touchEnd中，取消所有按钮的选中状态，并将 "选中按钮" 的所有元素 "移除"



C:画线
1.在drawRect中根据选中按钮的中心点画线
2.在touchesMoved 重新绘制



D:绘制 不在按钮内的触摸点的线
1.在touchesMoved中，如果不在按钮中，添加一个结束位置的位置，用于绘制线的路径



F:告知解锁密码
*通过代理的方式

注：自定义的view一定要设置个背影颜色，不然渲染会出问题

五、触摸事件的传递

```
/*目标
 *1.掌握事件的传递
 */
```

"1、事件是怎么的产生和传递"
①.当手指触摸屏幕后会产生 '触摸事件', 然后将事件加入UIApplication的管理事件队列中
②.UIApplication会取出事件队列中 '最前面的事件' 分发下去，事先分发给应用程序的主窗口中 'keyWindow'
③.主窗口接收到事件后，分发给自己的子控件，寻找最适合的接收事件的控件
④.找到 '最适合' 接收的控件后，调用控件的touchesBegin/touchesMoved/touchesEnded方法

"2、如何找到最合适的控件来处理事件？"
①.判断自己是否能接收触摸事件?

```
  /*
   控件不接收触摸事件的三种情况
   1> 不接收用户交互 userInteractionEnabled=NO
   2> 隐藏 hidden = YES
   3> 透明 alpha = 0.0 ~ 0.01
   */
```

②.判断触摸点是否在自己身上?

```
    /*
     * 判断触摸点在不在自己身上，view有一个方法【-(BOOL)pointInside:withEvent:】
     * 返回NO,就代表不在自己身边上，那不再遍历子控件
     * 返回YES，代表点在自己向上，那继续遍历子控件
     */
```

③.从后往前遍历子控件，重复前面的两个步骤
④.如果没有符合条件的子控件，那么就自己最适合处理的控件
⑤.找到最适合的控件后就调用touchesBegin/touchesMoved/touchesEnded方法

六、响应者链条

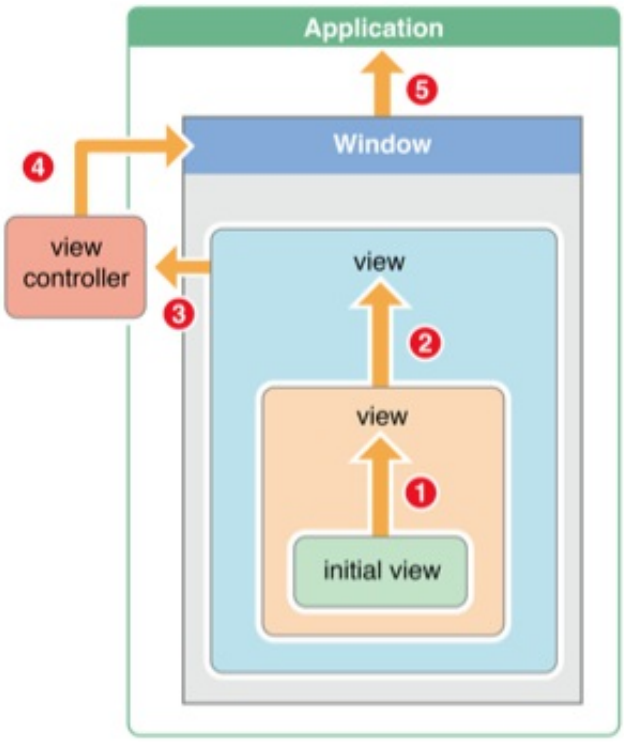
```
/*目标
 *1.掌握响应者链条
```

```
 */
```

"1、什么是响应者"
①.继承了UIResponder的对象就是响应者

"2、什么是响应者链条"
①.由多个响应者对象连接起来的链条叫做响应者链条，如下图
==========

响应者链条图



==========

②.什么是上一个响应者?
>如果当前这个view是控制器的view,控制器就是上一个响应者
>如果当前这个view不是控制器的view,那么父控件就是上一个响应者

③.利用响应者链条可以让多个控件处理同一个 "触摸事件"
"怎么让多个控件处理同一个‘触摸事件’事件?"
>在最后适合的控件里调用super的touchesBegan方法,这样就将事件传给上一个响应，上一个响应者也可以处理事件了

"3、小结:事件的完整处理过程"
①.先将事件对象由上往下传递(由父控制传给子控件)，找到最适合的控件来处理
②.调用最合适的控件的touches...方法
③.如果调用了[super touch…],就会将事件顺着响应都链往上传递，传递给上一个响应者
④.接着上一个响应者就会调用的touches...方法
⑤.如果没有找到最适合的控件来处理事件，则将事件传回来窗口，窗口不处理事件，将事件传给UIApplication
⑥.如果Applicatoin不能处理事件，则将其丢弃

```
//面试题:
/**
  响应者链条是什么?
  1.它是一种事件处理中制，由多个响应者对象连接起来的链条，使得事件可以沿着这些对象进行传递。
  2.如果一个响应者对象不能处理某个事件或动作消息,则将该事件消息重新发送给链中的下一个响应者。
  3.消息沿着响应者链向上、向更高级别的对象传递,直到最终被处理。
*/
```

七、手势识别

```
/*目标
 *掌握ios中的多种手势识别器的使用
 *掌握手势的使用步骤
 *1.创建手势对象
 *2.往控件添加手势对象
 *3.实现手势监听的方法
 */
```

ios3.2之后出现了UIGestureRecognizer手势，一般我们用其父类

"1、敲击手势UITapGestureRecognizer"

①敲击手势对象相关属性

*numberOfTapsRequired 敲击次数

*numberOfTouchesRequired 多点敲击

②代理方法 "实现左半边才接收触摸事件"

*【-gestureRecognizer:shouldReceiveTouch:】是否接收触摸事件

"2、长按手势UILongPressGestureRecognizer"

①长按手势对象相关属性

*minimumPressDuration 长按时间

*allowableMovement 触摸点周围多远的距离内长按有效

*state 手势状态，开始与结束 UIGestureRecognizerStateBegan/UIGestureRecognizerStateEnded

"3、轻扫手势UISwipeGestureRecognizer"

①轻扫手势对象相关属性

*direction 轻扫方向

```
//    UISwipeGestureRecognizerDirectionRight = 1 << 0,//向右轻扫
//    UISwipeGestureRecognizerDirectionLeft  = 1 << 1,//向左轻扫
//    UISwipeGestureRecognizerDirectionUp    = 1 << 2,//向上轻扫
//    UISwipeGestureRecognizerDirectionDown  = 1 << 3,//向下轻扫
```

"4、捏合手势UIPinchGestureRecognizer"

①捏合手势对象相关属性

*scale缩放比例【是一个累加的过程】

"5、旋转手势UIRotationGestureRecognizer"

①旋转手势对象相关属性

*rotation旋转角度【角度也是一个累加的过程】

②如果捏合和旋转想同时使用，设置其中一个手势的代理并实现代理的一个方法【-(BOOL)gestureRecognizer:shouldRecognizeSimultaneouslyWithGestureRecognizer:】并返回YES即可,允许跟其它手势一起发生

```
//Simultaneous:同時發生
```

"6、拖拽手势UIPanGestureRecognizer"

①旋转手势对象相关方法

*【recognizer translationInView:】移动的距离

*【recognizer setTranslation:inView:】

"7、Storybaord添加手势"