# p1: 改进

## Swift 2 -> 3

可以点击 `Convert`

如果点击了 `Later`，可以通过 `Edit -> Convert -> Current Swift Syntax` 来转化

> PPT

## Line -> UIBezierPath

```
// 我们定义 Line, 更新数组，删除 lastPoint
typealias Line = (path: UIBezierPath, color: UIColor)

var lines = [Line]()
var trash = [Line]()
// var lastPoint
```

## drawRect

```
// lineCap, Width 等是 UIBezierPath 的属性
// 我们通过循环设置，绘制
for line in lines{
    line.color.setStroke()
    line.path.lineJoinStyle = .round
    line.path.lineCapStyle = .round
    line.path.lineWidth = 5
```

```
    line.path.stroke()
}
```

## touches 系列简化

```
// 当用户开始触摸，，将起始点移动到当前触摸位置
func touchesBegan(_:_:){
    super.touchesBegan(_:_:)
    // 我们可以不判断 touches, 而是触摸的位置是否为空
    // 如果能够使得 location 为
    // touches 中第一个在 view 里的位置
    // 而不是nil
    if let location = touches.first?.location(in: self){
        // 那么新建 UIBezierPath 并添加到 lines
        lines.append((UIBezierPath(), lineColor))
        //
        lines.last?.move(to: location)
    }
}
```

```
// 将更新线位置变为一个统一的函数
private func update(for touches: Set<UITouch>){
    if let location = touches.first?.location(in: self){
        // 延长Path到当前触摸位置
        lines.last?.addLine(to: location)
        // 更新界面
        setNeedsDisplay()
    }
}
```

```
// 快捷键 touches
// 所以我们在 touchesMoved, ended, cancelled 执行 update(for:
touches) 函数
```

```
func touchesXXX(:_:_){
    super.touchesXXX(:_:_)
    update(for: touches)
}
```

> 运行测试

# p2: 调色板

## Segue

> PPT
>
> 添加返回按钮，讲解如何新建 `Segue`

## Manager

```
// 新建 PathManager
import UIKit
typealias Line = (path: UIBezierPath, color: UIColor)
let manager = PathManager.shared
class PathManager{
    static let shared = PathManager()
    private init(){}
    var lines = [Line]()
    var trash = [Line]()
    var lineColor = UIColor.black()
    func clear(){
        lines.removeAll()
        trash.removeAll()
    }
    func undo(){
        if !lines.isEmpty{
            trash.append(lines.removeLast())
        }
    }
    func redo(){
        if !trash.isEmpty{
            lines.append(trash.removeLast())
```

```
        }
    }

    // 为了方便使用，我们添加两个
    func newLine(){
        lines.append((UIBezierPath(), lineColor))
    }
    // 为了能够时刻更新，我们使用 Computed Property
    // > PPT
    var current: (line: Line?, path: UIBezierPath?, lineColor:
UIColor){
        get{
        // 当我们在获取的时候，返回数组中当前的最后一个元素
            return (lines.last, lines.last?.path, lineColor)
        }
    }
}
```

## 更改 DrawView

```
for line in manager.lines {}

touchesBegan(_:_:){
    manager.newLine()
    manager.current.path?.move(to: location)
}

update(_:){ manager.current.path?.addLine(to: location) }
```

# RGBAViewController

新建 RGBAViewController，删除原本代码

链接 — `[rgba]Label`, `[rgba]Slider`, `preview`

## 加快摆放控件

```
// Label: [Text:Black, Shadow:(1,1)] White;width=height=1
<Back preview
r -0- 255
g -0- 255
b -0- 255
a -0- 255
```

> PPT

## private RGBA

```
private var r: CGFloat = 0{
    willSet{
        rLabel.text = "\(newValue * 255)"
        rSlider.setValue(Float(newValue), animated: true)
    }
}
// 补全: defGBA
```

## value

```
var value: UIColor{
    get{
        return UIColor(red: r, green: g, blue: b, alpha: a)
    }
    set{
        // UnsafeMutablePointer<CGFloat>
```

```
        // getRed(_ red: inout CGFloat, ...)
        newValue.getRed(&r, green: &g, blue: &b, alpha: &a)
        preview.backgroundColor = newValue
    }
}
```

# 共用 IBAction

**设置几个 slider 的 tag 属性：`0`，`1`，`2`，`3`**

```
// 注意，各个 Slider 都要关联到
@IBAcion func valueChanged(_ sender: UIButton){
    switch sender.tag{
        case 0: r = CGFloat(sender.value)
        // 补全：casesGBA
    }
    preview.backgroundColor = value
}
```

# veiwWillXXX -> Controller

```
// RGBA
func viewWillDisappear(_:){
    super.viewWillDisappear(_:)
    manager.lineColor = value
}

viewWillAppear(_:){
    super
    preview.backgroundColor = manager.lineColor
}

// Draw，更新 StoryBoard 链接
```

```
func viewWillAppear(_:){
    super.viewWillAppear(_:)
    value = manager.lineColor
}
```

# ViewController

```
func touchUpInside(sender: UIButton){
    switch sender.tag{
        case 1:
            manager.clear()
        // 补全: tui
    }
    drawView.setNeedsDisplay()
}
```

> 运行，会报错，然后删除原来的关联
>
> 尝试旋转，更改为 redraw