# HSB Color Picker

Apple bilibili bug swift

---

BUG: 每次选择颜色会增加两个 VC
将之前的移入 `Supporting Files`，更新语法

- 忽略 fileprivate 的更改
  新建
- Indicator: UIView
- IndicatableUIControl: UIControl
- HueGradientLayer: CAGradientLayer
- HuePicker: IndicatableUIControl
- ColorPicker: IndicatableUIControl
- HSBViewController: UIViewController

取消原来`Storyboard`中的`Segue`
另外拖入并连接一个VC，设置为HSBVC，向其中添加一个Button，三个View

## @IBDesignable Indicator: UIView

小白点

```
    var radius: CGFloat{
        return min(bounds.width, bounds.height) / 2
    }
```

```
    // viewinit
    override init(frame: CGRect) {
        if frame.size == .zero{
            super.init(frame: CGRect(origin: frame.origin, size: CGSize(width: 20, height: 20)))
        } else {
            super.init(frame: frame)
        }
        setup()
    }

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setup()
    }

    private func setup(){
        backgroundColor = .white
        layer.borderColor = UIColor.black.cgColor
        layer.borderWidth = 1
```

```swift
        layer.cornerRadius = radius
        layer.masksToBounds = true
        fit()
    }
```

```swift
    func fit(){
        if let top = superview{
            if !top.bounds.intersetcs(center){
                setCenter(to: center.fitting(in: top))
            }
        }
    }

extension CGPoint{
    func fitting(in view: UIView) -> CGPoint{
        return fitting(in: view.bounds)
    }
    func fitting(in rect: CGRect) -> CGPoint{
        let fittingX = max(min(x, rect.maxX), rect.minX)
        let fittingY = max(min(y, rect.maxY), rect.minY)
        return CGPoint(x: fittingX, y: fittingY)
    }
}

extension CGRect{
    public func intersetcs(_ point: CGPoint) -> Bool{
        return self.minX <= point.x && self.maxX >= point.x &&  self.minY <= point.y &&
self.maxY >= point.y
    }
}
```

```swift
    func setCenter(to newCenter: CGPoint){
        setCenter(xTo: newCenter.x, yTo: newCenter.y)
    }

    func setCenter(xTo newX: CGFloat? = nil, yTo newY: CGFloat? = nil){
        if let x = newX {
            frame.origin.x = x - radius
        }
        if let y = newY {
            frame.origin.y = y - radius
        }
        fit()
    }
```

## @IBDesignable IndicatableUIControl: UIControl

sendAction 和 storyboard 的两种 event binding

```swift
    let indicator = Indicator()

    override func didMoveToSuperview() {
        super.didMoveToSuperview()
        addSubview(indicator)
        indicator.setCenter(to: center)
    }
```

```
    private func update(for touch: UITouch){
        indicator.setCenter(xTo: touch.location(in: self).x)
        sendActions(for: .valueChanged)
    }

    // indtrack
    func beginTracking(_:with:) -> Bool {
        sendActions(for: .touchDown)
        update(for: touch)
    }
    func continueTracking(_:with:) -> Bool {
        update(for: touch)
    }
    func endTracking(_:with:) {
        guard touch != nil else {sendActions(for: .touchCancel);return}
        update(for: touch!)
        sendActions(for: .touchUpInside)
    }
```

## @IBDesignable HueGradientLayer: CAGradientLayer

讲解坐标，左下角(0,0)，右上角(1,1)，默认 locations 是 [0,1]，start -> end 是 (.5, 0 -> 1)

```
    private func setup(){
        var stops = [Double]()
        stops.append(contentsOf: stride(from: 0, through: 1, by: 0.1))
        self.locations = stops
        self.colors = stops.map { UIColor(hue: CGFloat($0), saturation: 1, brightness: 1, 
alpha: 1).cgColor }

        self.startPoint = CGPoint(x: 0, y: 0.5)
        self.endPoint = CGPoint(x: 1, y: 0.5)
    }

    // gradinit
    override init() {
        super.init()
        setup()
    }

    override init(layer: Any) {
        super.init(layer: layer)
        setup()
    }

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setup()
    }
```

## @IBDesignable HuePicker: IndicatableUIControl

```swift
    override open class var layerClass: Swift.AnyClass {
        return HueGradientLayer.self
    }
    var value: CGFloat {
        return convert(indicator.center, to: self).x / bounds.width
    }
```

## @IBDesignable ColorPicker: IndicatableUIControl

```swift
    var hue: CGFloat = 0{
        didSet{
            setNeedsDisplay()
            sendActions(for: .valueChanged)
        }
    }

    var value: UIColor{
        return UIColor(
            hue: hue,
            saturation: convert(indicator.center, to: self).y / bounds.height,
            brightness: convert(indicator.center, to: self).x / bounds.width,
            alpha: 1
        )
    }
```

```swift
    override func draw(_ rect: CGRect) {

        let context = UIGraphicsGetCurrentContext()
        context?.move(to: .zero)
        let period = Int(bounds.height)
        let stops: [CGFloat] = [0.0, 1.0]

        for i in 0..<period{

            let saturation = CGFloat(i)/CGFloat(period)
            let colors = stops.map { UIColor(hue: hue, saturation: saturation, brightness:
$0, alpha: 1).cgColor }
            let gradient = CGGradient(colorsSpace: CGColorSpaceCreateDeviceRGB(), colors:
colors as CFArray, locations: stops)

            context?.saveGState()
            UIBezierPath(rect: CGRect(x: 0, y: CGFloat(i), width: bounds.width, height:
1)).addClip()
            context?.drawLinearGradient(
                gradient!,
                start: CGPoint(x: 0, y: i),
                end: CGPoint(x: bounds.maxX, y: CGFloat(i)),
                options: []
            )
            context?.restoreGState()

        }
        super.draw(rect)
    }
```

## HSBViewController: UIViewController

```
    override var prefersStatusBarHidden: Bool{
        return true
    }

    @IBOutlet weak var preview: UIView!
    @IBOutlet weak var huePicker: HuePicker!
    @IBOutlet weak var colorPicker: ColorPicker!

    // hsb
    private var hue: CGFloat = 0
    private var saturation: CGFloat = 0
    private var brightness: CGFloat = 0


    var value: UIColor{
        get{
            return colorPicker.value
        }
        set{
            newValue.getHue(&hue, saturation: &saturation, brightness: &brightness, alpha: nil)
        }
    }
```

```
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        value = manager.lineColor
        huePicker.indicator.setCenter(xTo: hue * huePicker.bounds.width)
        colorPicker.indicator.setCenter(to: CGPoint(x: saturation * colorPicker.bounds.height, y: brightness * colorPicker.bounds.width))
        colorPicker.hue = hue
        huePicker.addTarget(self, action: #selector(ColorPickerViewController.hueValueChanged), for: .valueChanged)
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)
        huePicker.removeTarget(self, action: #selector(ColorPickerViewController.hueValueChanged), for: .valueChanged)
        manager.lineColor = value
    }

    func hueValueChanged(){
        colorPicker.hue = huePicker.value
    }

```

```swift
    @IBAction func colorValueChanged() {
        preview.backgroundColor = value
    }
```