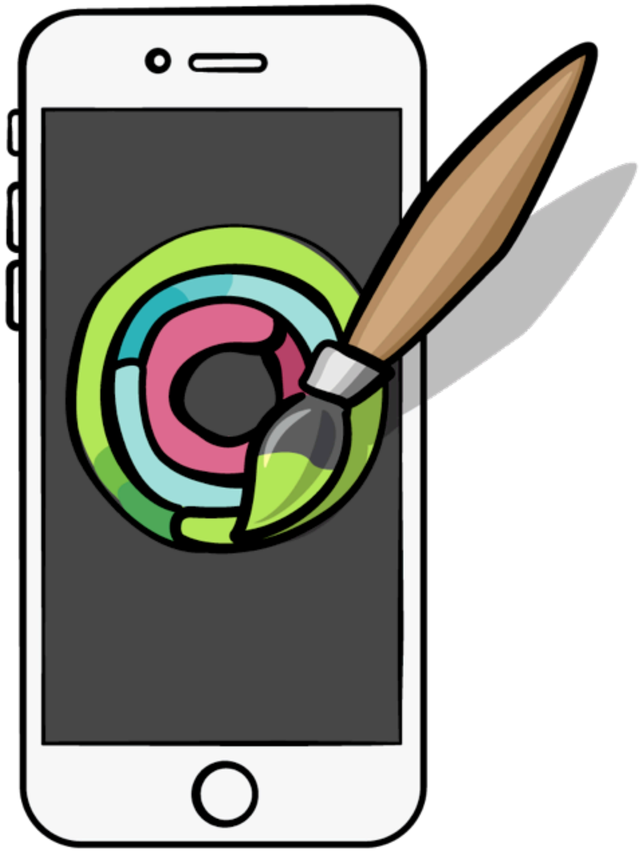
CUSTOM CONTROLS IN iOS



Part 11: Gesture Recognizers


Carrier
12:43 AM
Fill

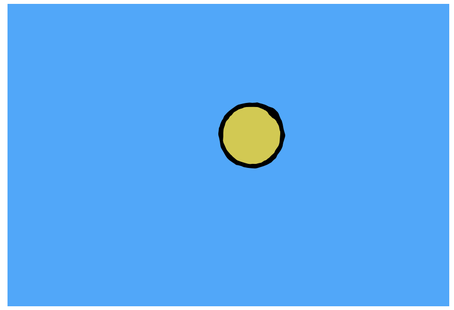
Gesture Recognizers

```
tapGestureRecognizer = TapGestureRecognizer(
  target: self,
  action: #selector(self.handleTap)
)
addGestureRecognizer(tapGestureRecognizer!)
```

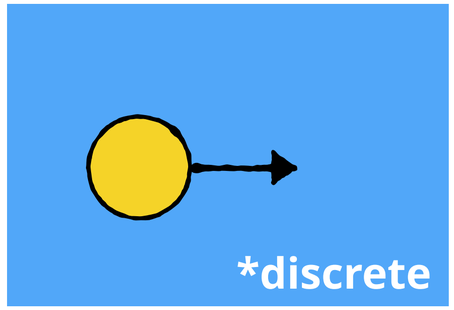
```
extension UIGes
  ...
  open func touchesBegan(  touches: Set<UITouch>, with event: UIEvent)
  open func touche                                        ent: UIEvent)
  open func touche                                        ent: UIEvent)
  open func touche                                      h event: UIEvent)
}
```

```
func handleTap() {
  switch tapGestureRecognizer.state {
  case .began:
    ...
  case .ended:
    ...
  default:
    ...
  }
}
```

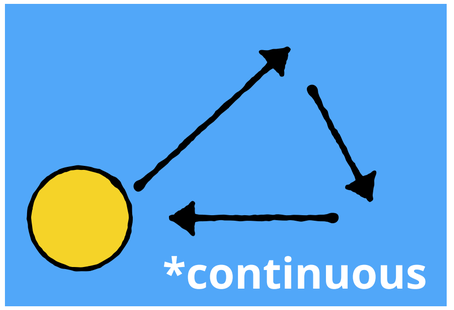
Gesture Recognizer Subclasses



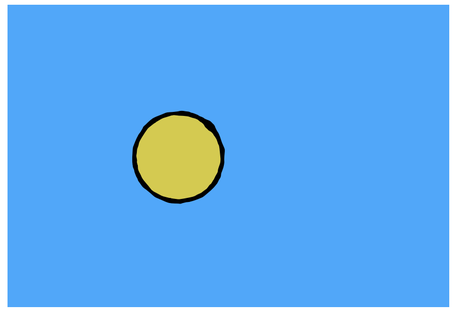
`UITapGestureRecognizer`



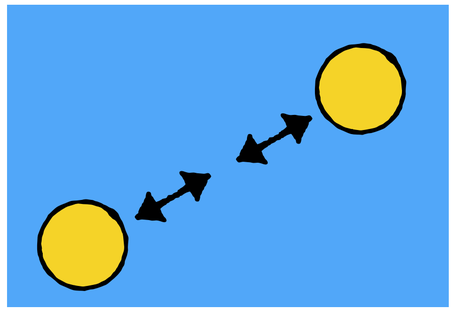
`UISwipeGestureRecognizer`



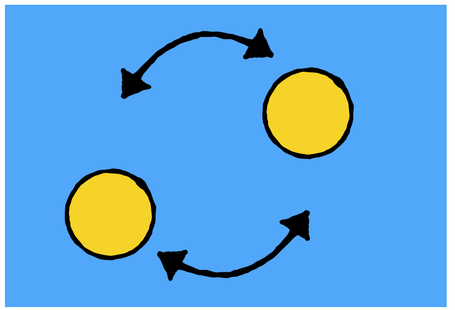
`UIPanGestureRecognizer`



`UILongPressGestureRecognizer`



`UIPinchGestureRecognizer`



`UIRotationGestureRecognizer`

Using Multiple Gesture Recognizers

```
final class TapGestureRecognizer: UITapGestureRecognizer, GestureRecognizer {
  ...
}
```

```
extension PanGestureRecognizer: GestureRecognizer {
  func handleLargeRadiusTouch(yPositionInView: CGFloat) {
   ...
  }
}
```

```
protocol GestureRecognizer: class {
  var selection: Selection {get set}
  var selectionTouchRadiusCrossover: CGFloat! {get set}
}

extension GestureRecognizer where Self: UIGestureRecognizer {
  func handleLargeRadiusTouch(yPositionInView: CGFloat) {}

  func setSelection(touches: Set<UITouch>) {
   ...
  }
}
```

Challenge Time!


```
enum Selection {
  case
    color(float3),
    valueDelta(Float)
}
```

```
if touch.majorRadius < selectionTouchRadiusCrossover {
  selection = .color(
    UnitCube.getColor(
      positionInView: positionInView,
      viewSize: view!.bounds.size
      )
    )
} else {
  selectValueDelta(yPositionInView: positionInView.y)
}
```