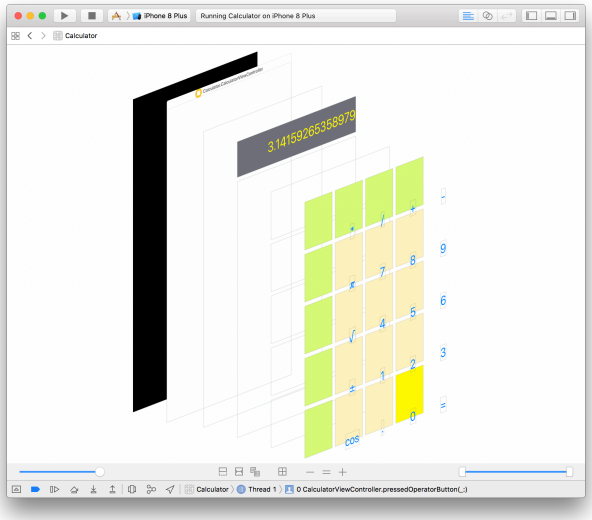
CS 326
Gestures and More Swift

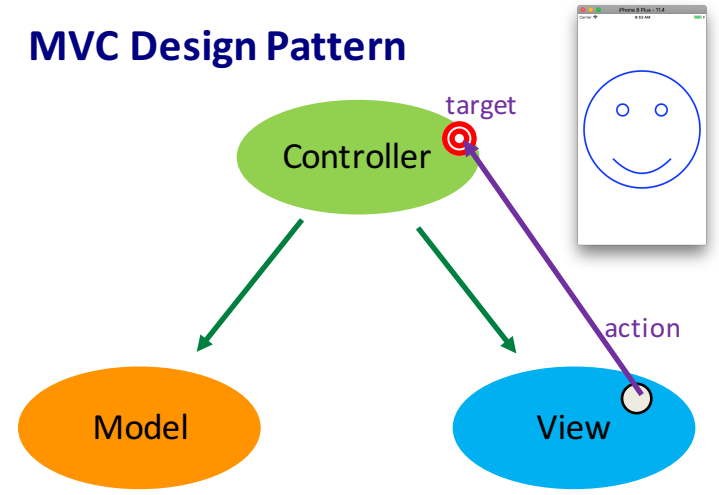
Stephen Freund



UIView Attributes

- Need to force a **`UIView`** to be redrawn when device orientation changes.
  - In Attributes Inspector, set **`UIView`**'s "Content Mode" to "redraw"
- Others
  - hidden
  - backgroundColor
  - transparency
- Experiment!

MVC Design Pattern



Model won't exactly match View's representation of a face...

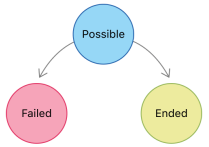
Gestures

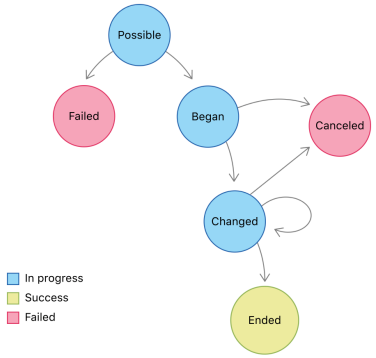
- UIGestureRecognizer subclasses:
  - UITapGestureRecognizer
  - UIPanGestureRecognizer
  - UIPinchGestureRecognizer
  - ...
- Two parts:
  - Ask view to add recognizer
  - provide "handler" for gesture recognizer
    - method in Controller if gesture affects model.
    - method in View if only affects the view (rare).

Quick Demo

UIGestureRecognizer States

- Discrete (tap, swipe, ...)
- Continuous (pan, pinch)





Recognizer Properties/ Methods

- Recognizers have type-specific ways of interacting with them
  - UIPanGestureRecognizer
    - func translation(in: UIView?) -> CGPoint
    - func setTranslation(CGPoint, in: UIView?)
  - UIPinchGestureRecognizer
    - scale
    - velocity
  - **See docs for more**

View Controller Lifecycle

- viewDidLoad()
  - Called after the view controller has loaded its view hierarchy into memory.
  - Good place to initialize model, do other bookkeeping work.

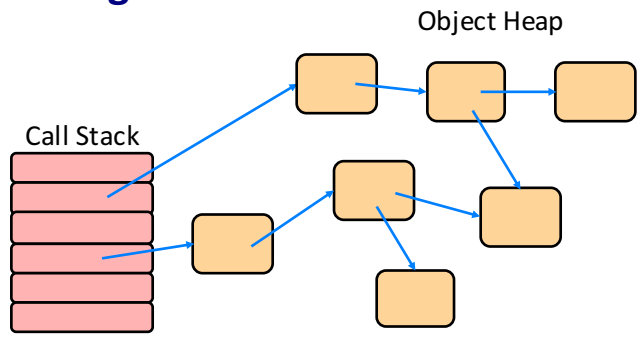
UserDefaults

- Lightweight database that persists between launches.
- Map from keys to values
  - key: String
  - value: String, Date, Int, Bool, Double, array, dictionary
- You update the map
- Changes occasionally saved by your app.

UserDefaults

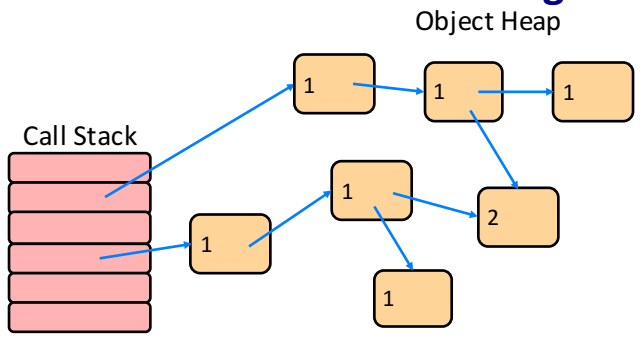
```
let defaults = UserDefaults.standard


defaults.set(3.14, forKey: "pi")
defaults.set("Springer", forKey: "cow")
defaults.set(nil, forKey: "pi")


let d = defaults.double(forKey: "pi") ?? 0.0
let s = defaults.string(forKey: "cow") ?? "..."
```

Garbage Collection



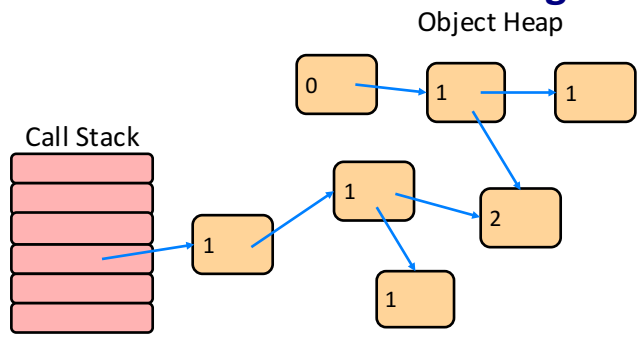
- When can you free memory for an object?

Automatic Reference Counting



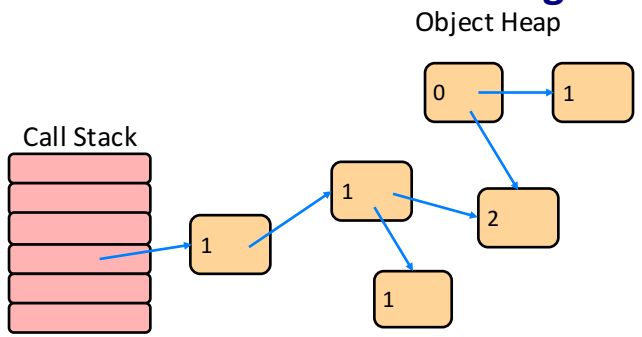
- Count number of references to each object
- Free object when count is 0

Automatic Reference Counting



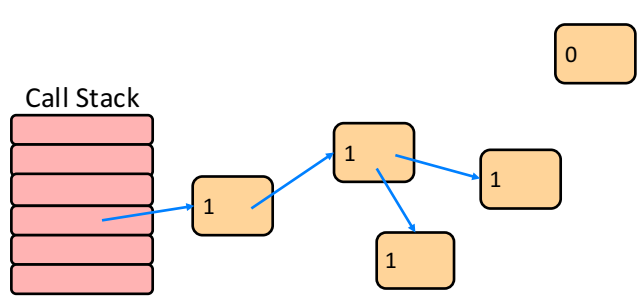
- Count number of references to each object
- Free object when count is 0

Automatic Reference Counting



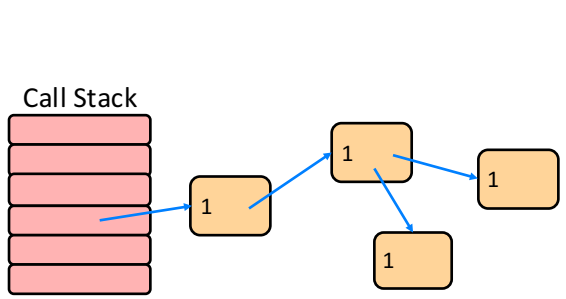
- Count number of references to each object
- Free object when count is 0

Automatic Reference Counting



- Count number of references to each object
- Free object when count is 0

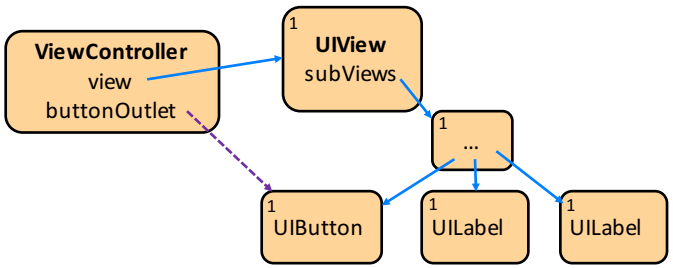
Automatic Reference Counting



- Count number of references to each object
- Free object when count is 0

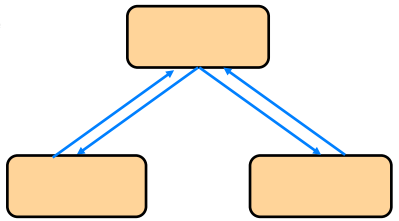
Weak References

- Don't count weak references to object
  - `weak var buttonOutlet : UIButton?`
  - weak references may become nil at any time...



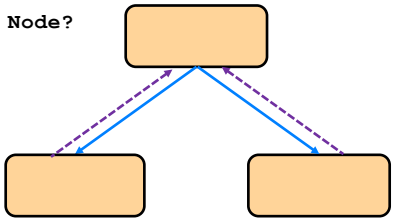
Reference Cycle

```
class Node {
  let left: Node
  let right: Node
  var parent : Node
  ...
}
```



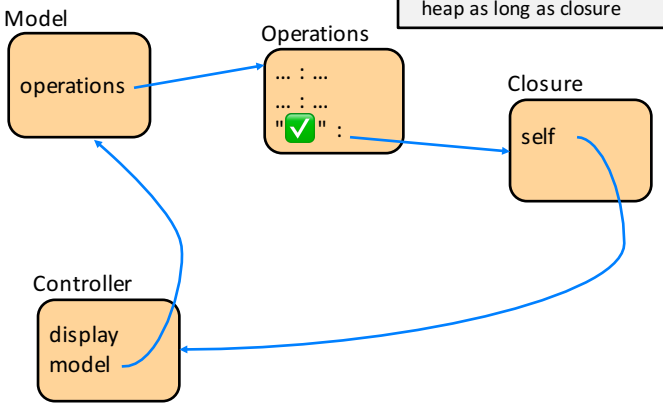
Reference Cycle

```
class Node {
  let left: Node
  let right: Node
  weak var parent : Node?
  ...
}
```

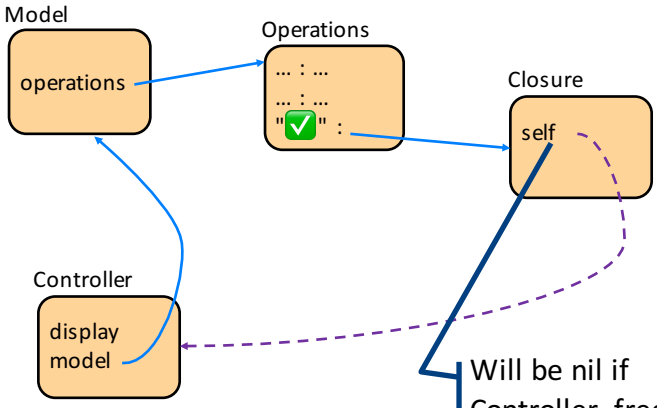


Reference Cycle

Closures
- "Capture" all objects referenced inside their code
- Those objects need to stay in heap as long as closure



Weak References



Weak Variable Capture

```
model.addUnaryOp(symbol: "✅") { [weak t = self] in
   t?.display.textColor = UIColor.green
   return sqrt($0)
}


model.addUnaryOp(symbol: "✅") { [weak self] in
   self?.display.textColor = UIColor.green
   return sqrt($0)
}
```

Swift Access Levels

- `open`
- **`public`**
- **`internal`**
  - default
- `file-private`
- **`private`**