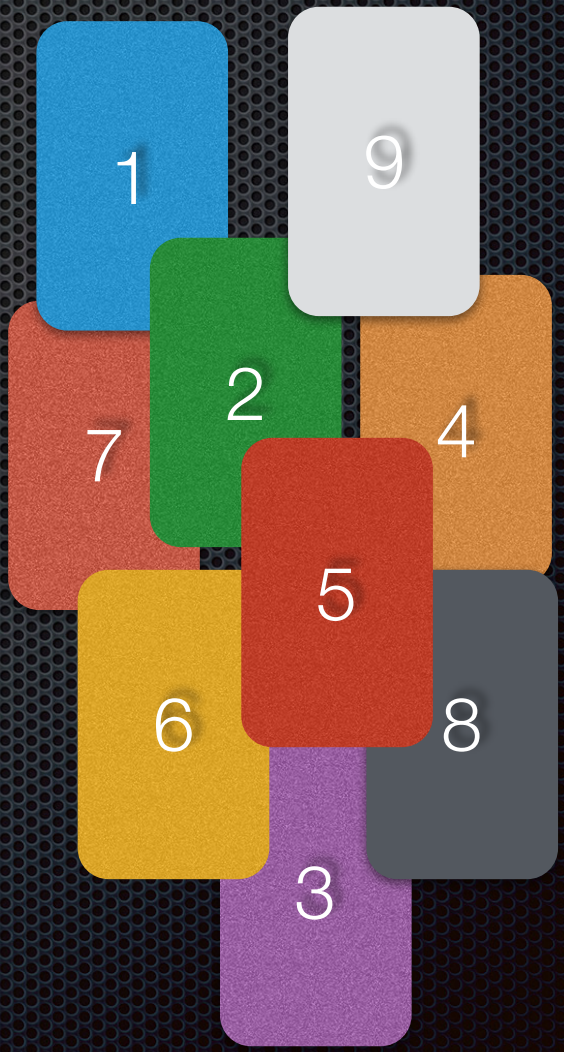
Mobile Application Programming

Data Models


SHALL WE PLAY A GAME?

Numbers

- Two players
- On the table, there are nine cards numbered from 1 to 9
- Players draw alternately
- The objective is to make a “book” – a set of three cards that adds to 15
- You can take more than three cards

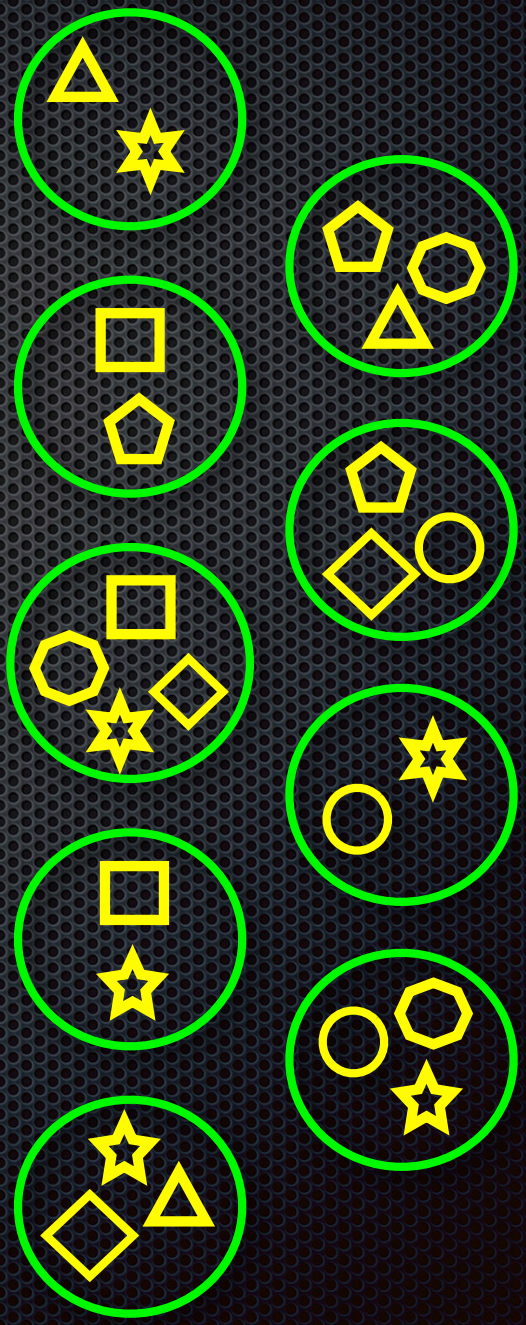


Player 1

Player 2

Shapes

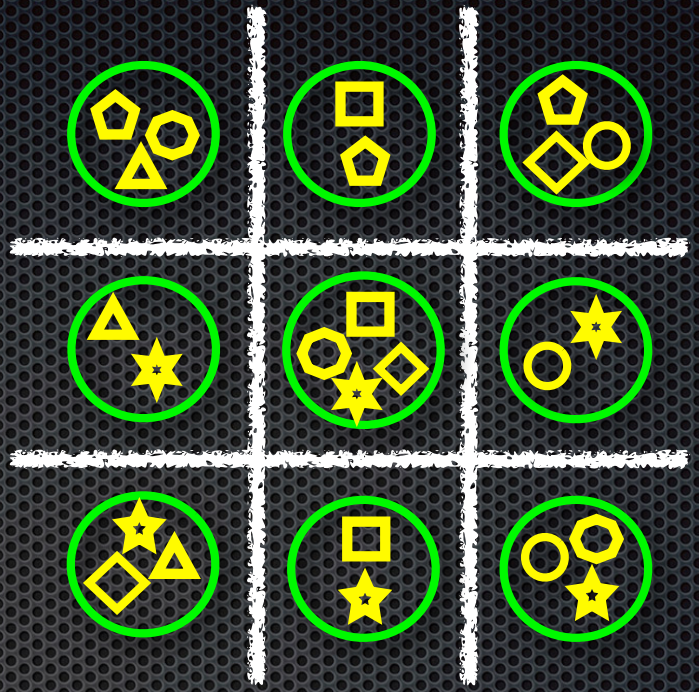
- Two players alternate choosing groups of shapes
- The first player to get 3 groups that contain the same shape wins

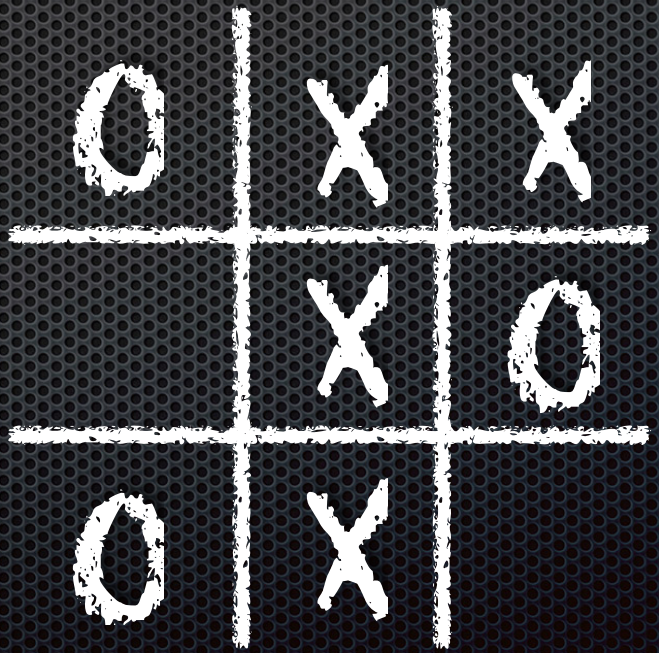


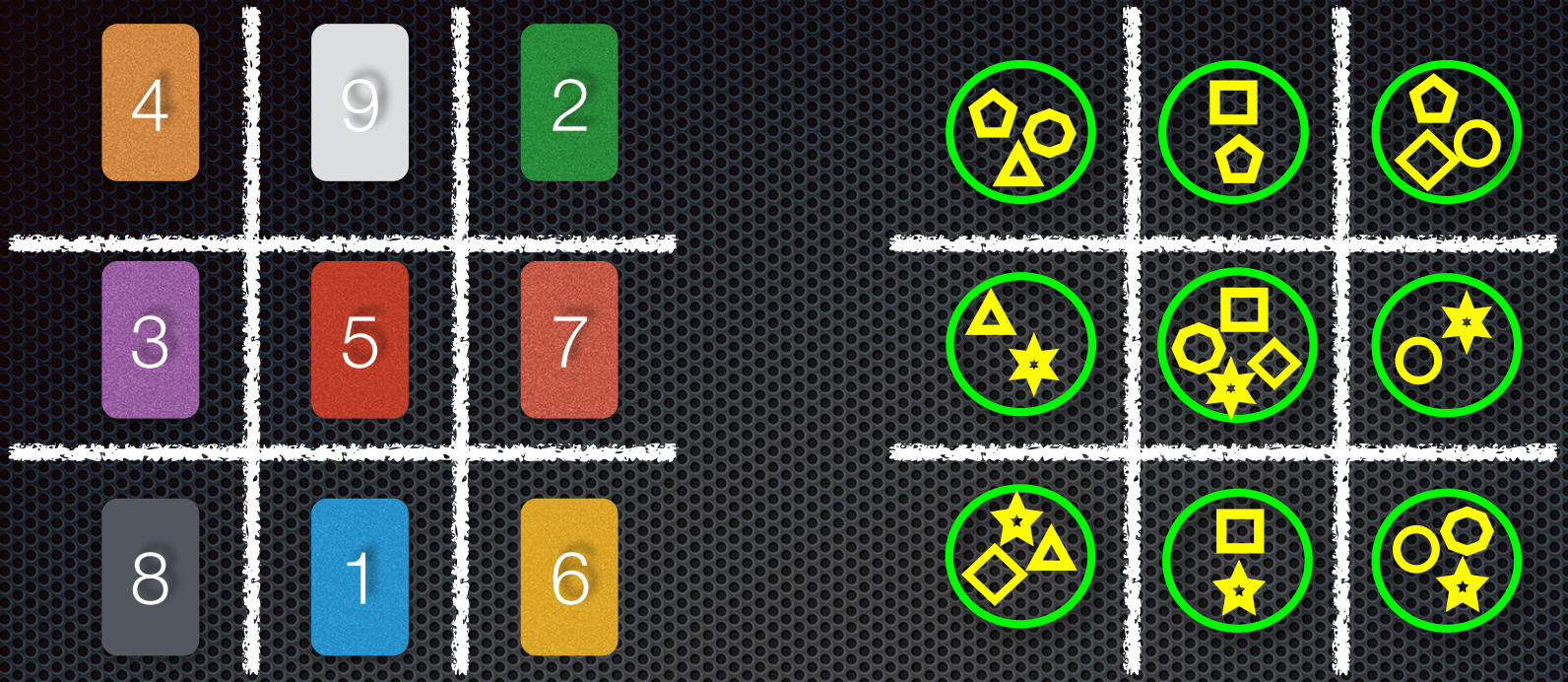
Player 1

Player 2

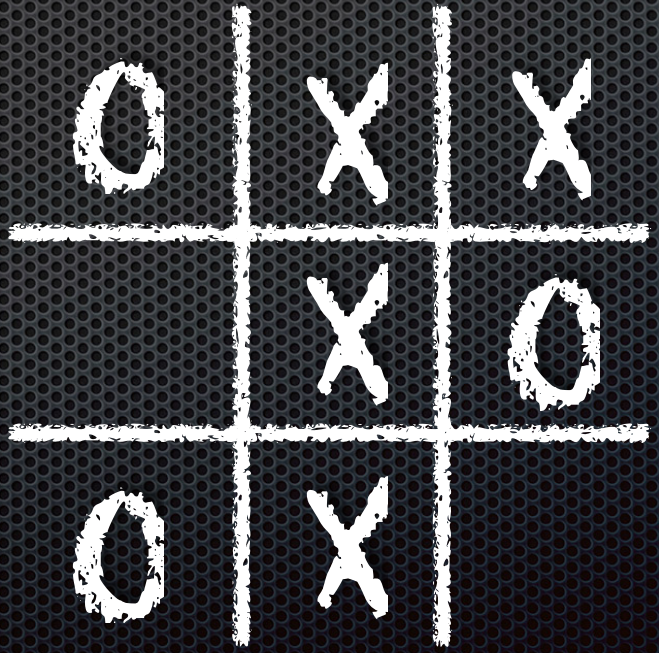

4
9
2
3
5
7
8
1
6

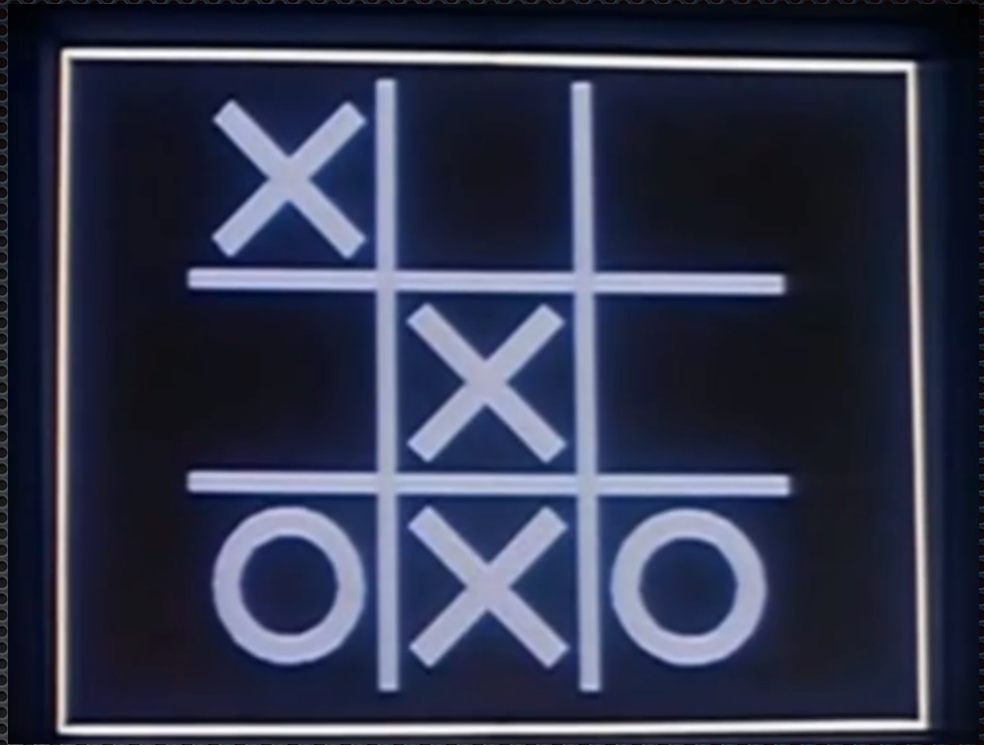


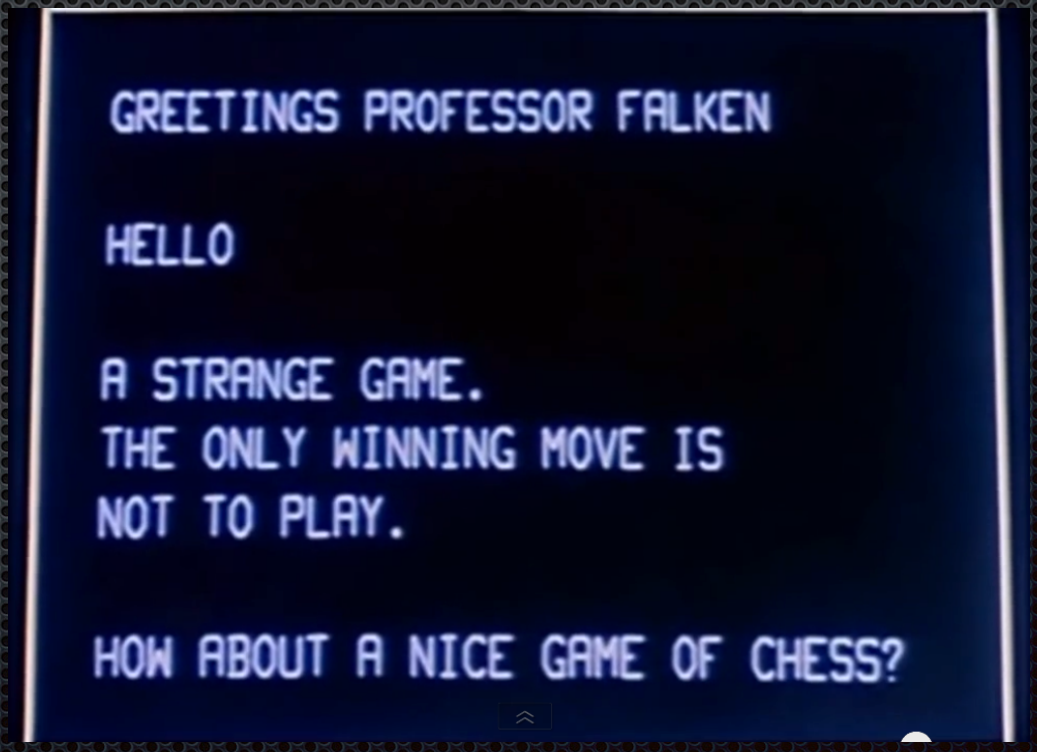

O
X
X
X
O
O
X

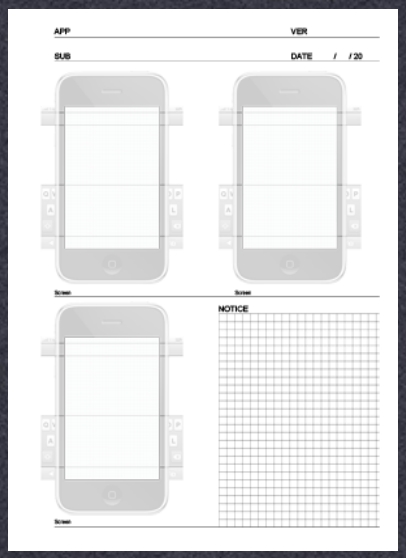


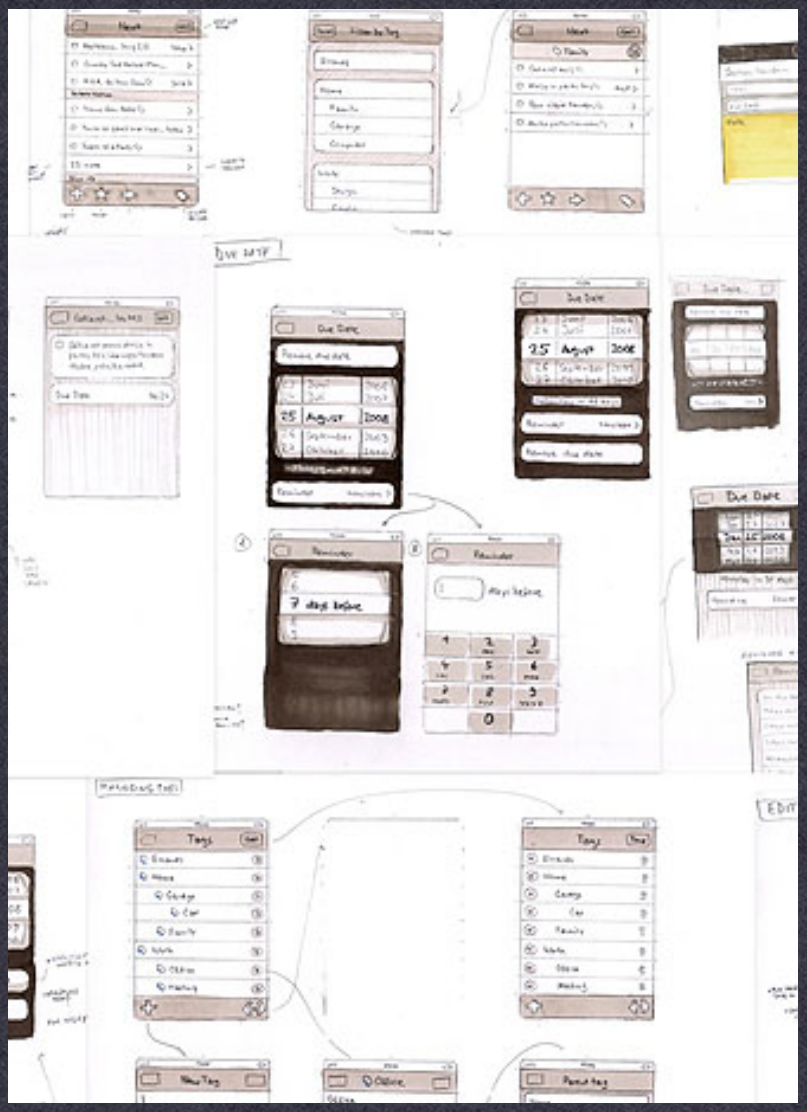
Representation Matters

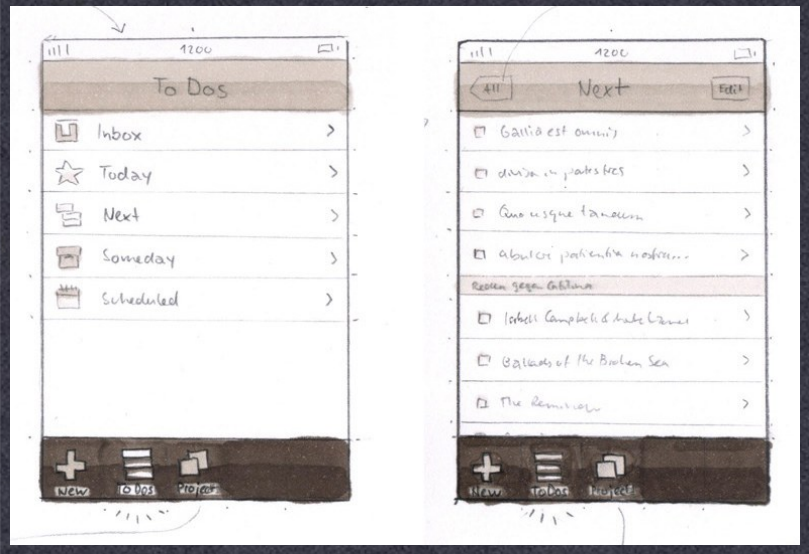


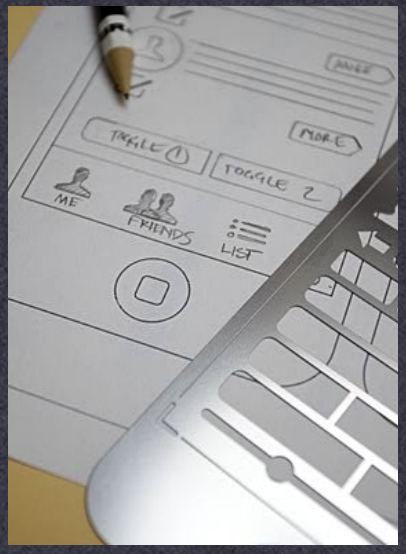



GREETINGS PROFESSOR FALKEN
HELLO
A STRANGE GAME.
THE ONLY WINNING MOVE IS
NOT TO PLAY.
HOW ABOUT A NICE GAME OF CHESS?





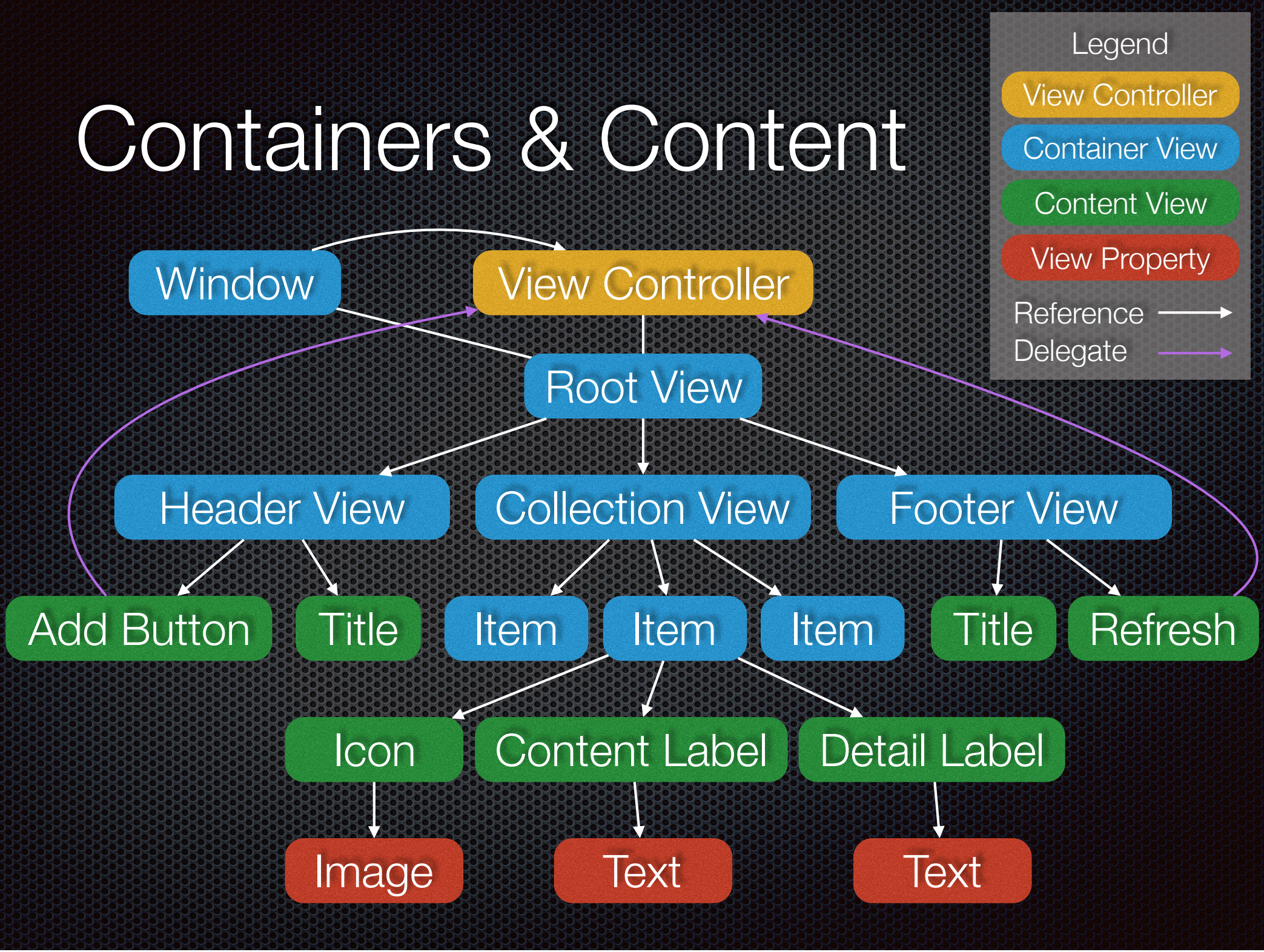




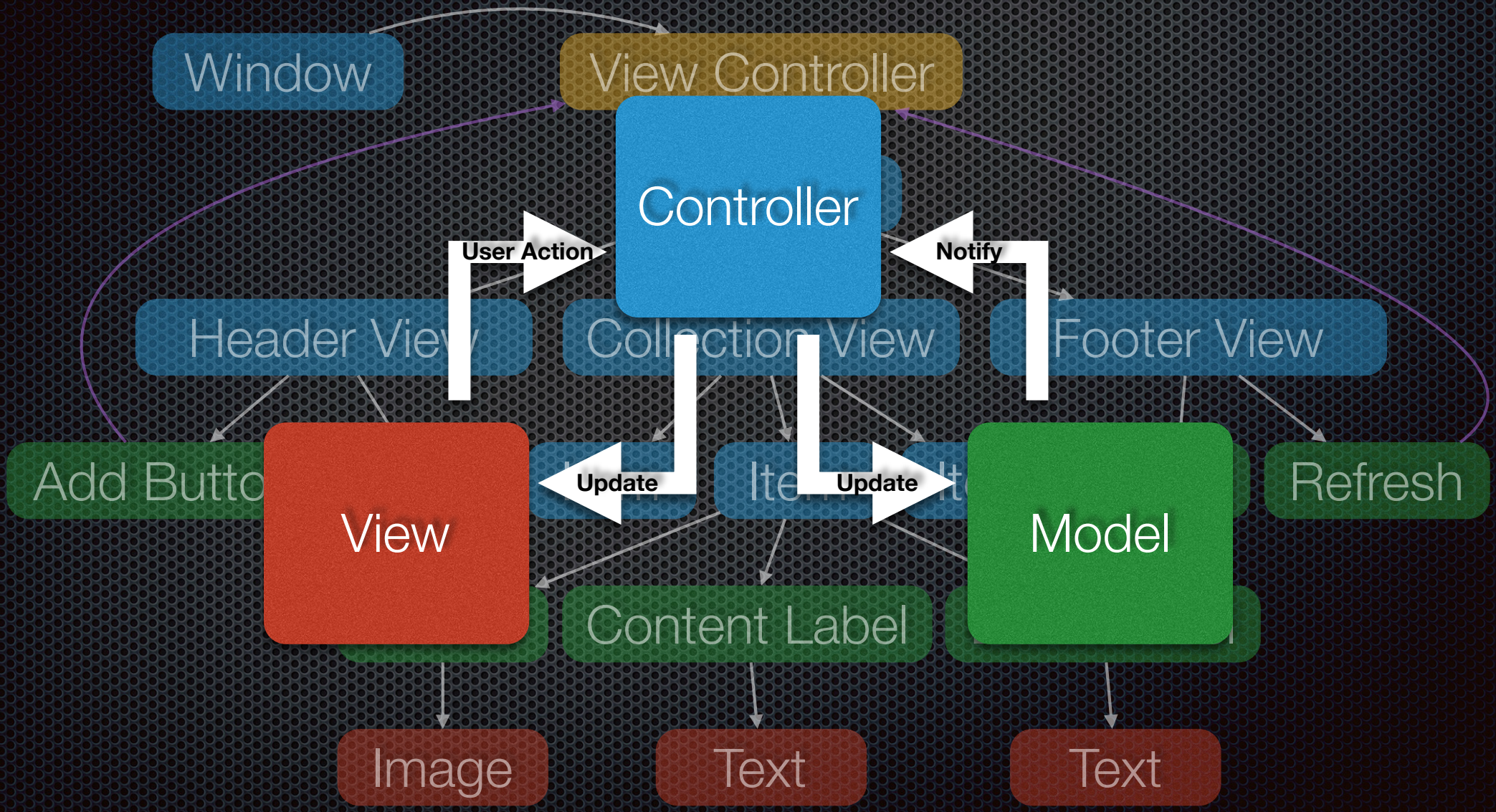
Tic Tac Toe isomorphs and their relation to user interface design
introduced to me by Lorenzo Swank. WarGames references added by me.
Original idea from Scott Klemmer (Stanford) and Susan Brennan (Stony Brook)

DATE	INSTRUCTOR
11 JANUARY 2010	**LORENZO SWANK**

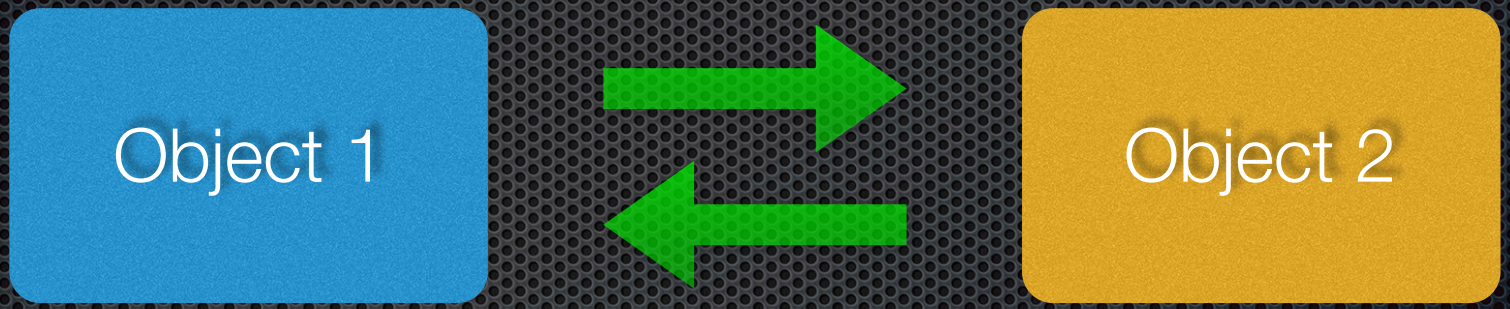
Containers & Content



How Does This Fit With MVC?



Problem: 2 Objects Talking



Delegates

- A delegate is an object that performs actions on the behalf of another object
- A common use is a data model object alerting a controller of changes to its data, which then tells view objects about the change
- Another use of them is a view object having a controller object interact with the program data model on its behalf when the user triggers events
- 6 bits of code are required to properly set up both sides of a delegate connection between two objects

Delegator

1. Delegate Protocol
2. Delegate Property
3. Delegate Invocation

```
import UIKit

protocol KnobDelegate: class
{
    func knob(knob: Knob, rotatedToAngle angle: Float)
}

class Knob : UIView
{
    private var _knobRect: CGRect = CGRectZero
    private var _angle: Float = 3.0 * Float(M_PI) / 2.0

    var angle: Float
    {
        get { return _angle }
        set
        {
            _angle = newValue
            setNeedsDisplay()
        }
    }

    weak var delegate: KnobDelegate? = nil

    override func touchesMoved(touches: NSSet, withEvent event: UIEvent)
    {
        let touch: UITouch = touches.anyObject() as UITouch
        let touchPoint: CGPoint = touch.locationInView(self)
        let touchAngle: Float = atan2f(
            Float(touchPoint.y - _knobRect.midY),
            Float(touchPoint.x - _knobRect.midX))

        angle = touchAngle
        delegate?.knob(self, rotatedToAngle: angle)
    }

    override func drawRect(rect: CGRect)
    {
    }
}
```

```
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, KnobDelegate
{
    var window: UIWindow?

    func application(application: UIApplication,
       didFinishLaunchingWithOptions l: [NSObject: AnyObject]?) -> Bool
    {
        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        window?.makeKeyAndVisible()

        var knob: Knob = Knob(frame: window!.frame)
        knob.backgroundColor = UIColor.darkGrayColor()
        knob.delegate = self
        window?.addSubview(knob)

        return true
    }

    func knob(knob: Knob, rotatedToAngle angle: Float)
    {
        println("Knob rotated to angle: \(angle)")
    }
}
```

Delegator

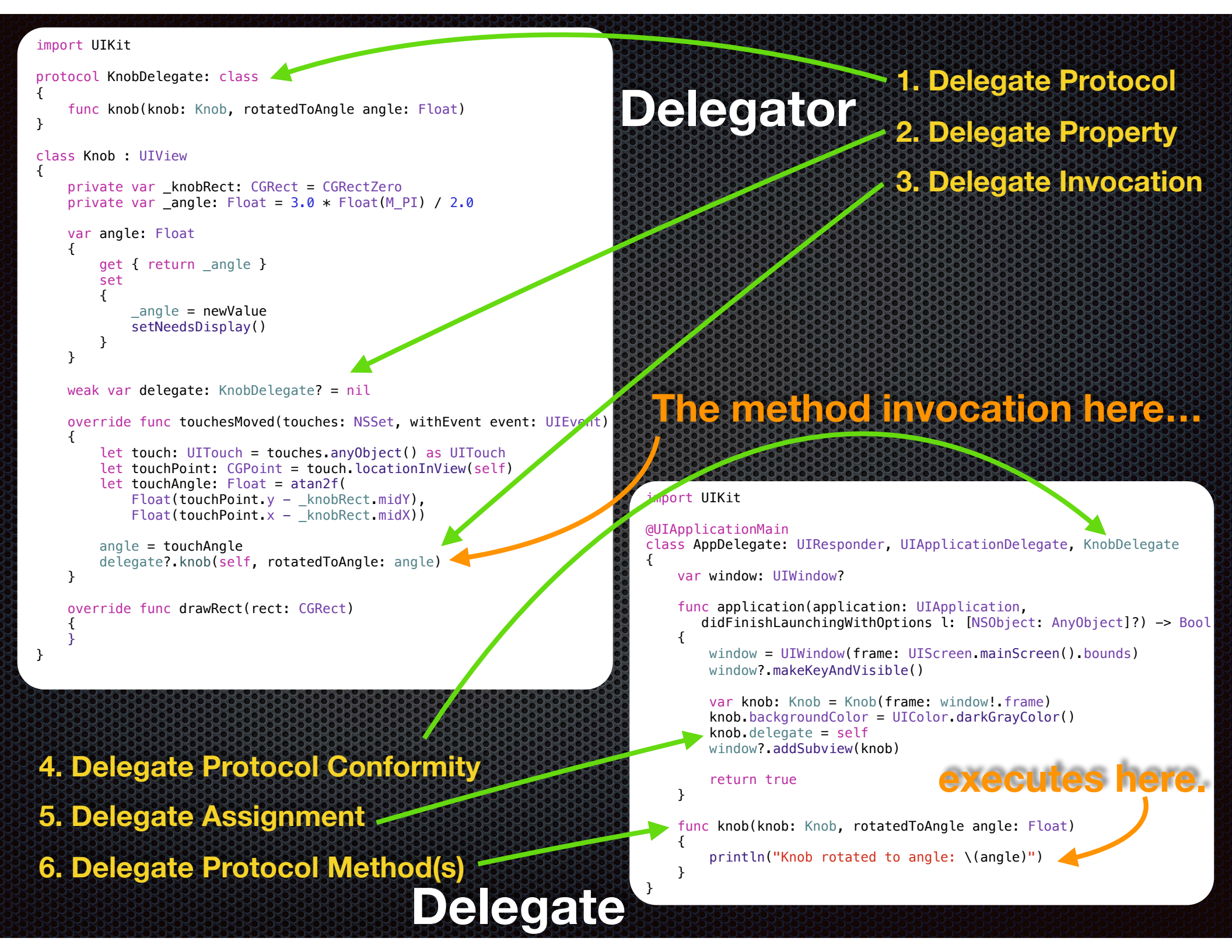
1. Delegate Protocol
2. Delegate Property
3. Delegate Invocation

```swift
import UIKit

protocol KnobDelegate: class
{
    func knob(knob: Knob, rotatedToAngle angle: Float)
}

class Knob : UIView
{
    private var _knobRect: CGRect = CGRectZero
    private var _angle: Float = 3.0 * Float(M_PI) / 2.0

    var angle: Float
    {
        get { return _angle }
        set
        {
            _angle = newValue
            setNeedsDisplay()
        }
    }

    weak var delegate: KnobDelegate? = nil

    override func touchesMoved(touches: NSSet, withEvent event: UIEvent)
    {
        let touch: UITouch = touches.anyObject() as UITouch
        let touchPoint: CGPoint = touch.locationInView(self)
        let touchAngle: Float = atan2f(
            Float(touchPoint.y - _knobRect.midY),
            Float(touchPoint.x - _knobRect.midX))

        angle = touchAngle
        delegate?.knob(self, rotatedToAngle: angle)
    }

    override func drawRect(rect: CGRect)
    {
    }
}
```

4. Delegate Protocol Conformity
5. Delegate Assignment
6. Delegate Protocol Method(s)

Delegate

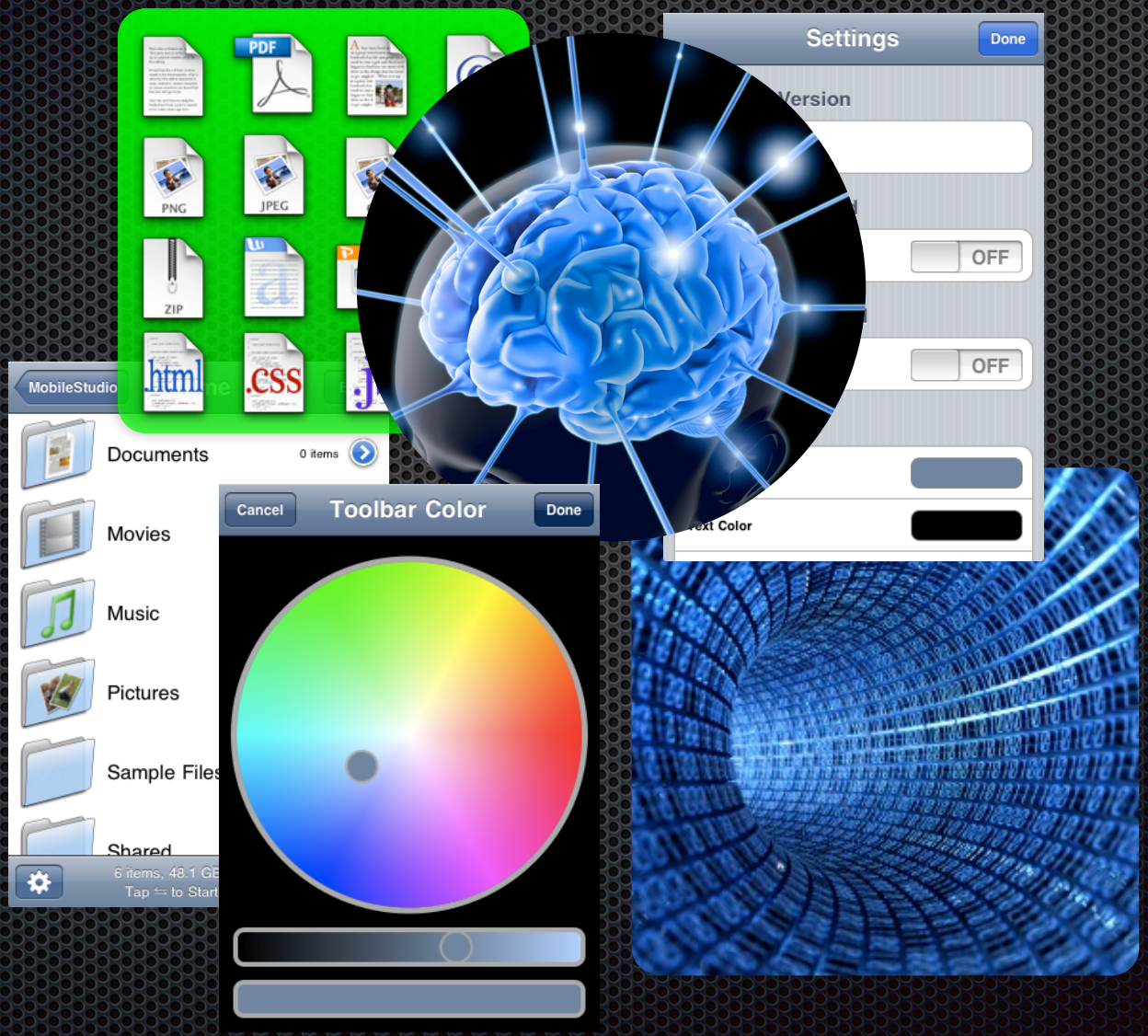
```swift
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, KnobDelegate
{
    var window: UIWindow?

    func application(application: UIApplication,
       didFinishLaunchingWithOptions l: [NSObject: AnyObject]?) -> Bool
    {
        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        window?.makeKeyAndVisible()

        var knob: Knob = Knob(frame: window!.frame)
        knob.backgroundColor = UIColor.darkGrayColor()
        knob.delegate = self
        window?.addSubview(knob)

        return true
    }

    func knob(knob: Knob, rotatedToAngle angle: Float)
    {
        println("Knob rotated to angle: \(angle)")
    }
}
```



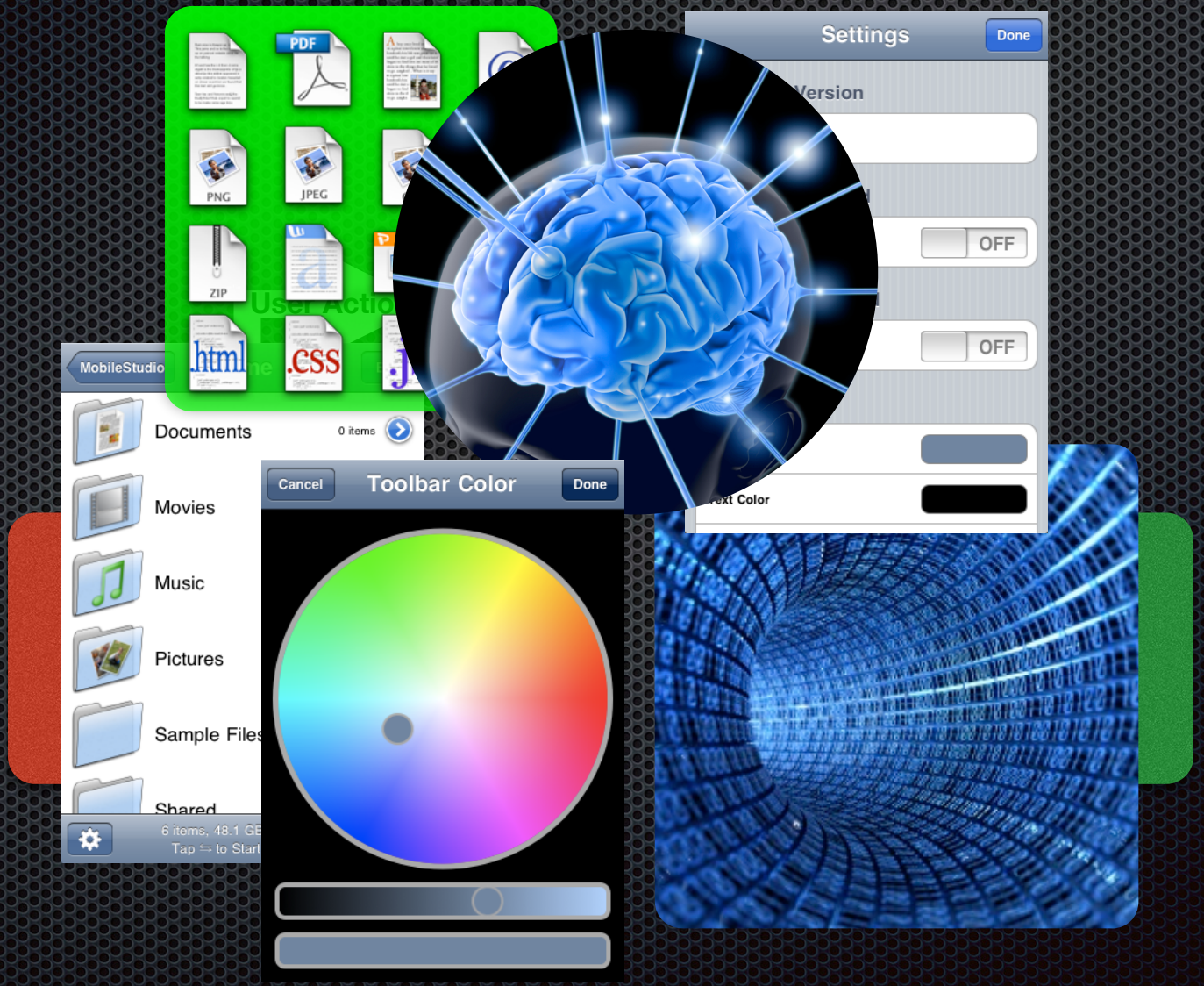

```swift
import UIKit

protocol KnobDelegate: class
{
    func knob(knob: Knob, rotatedToAngle angle: Float)
}

class Knob : UIView
{
    private var _knobRect: CGRect = CGRectZero
    private var _angle: Float = 3.0 * Float(M_PI) / 2.0

    var angle: Float
    {
        get { return _angle }
        set
        {
            _angle = newValue
            setNeedsDisplay()
        }
    }

    weak var delegate: KnobDelegate? = nil

    override func touchesMoved(touches: NSSet, withEvent event: UIEvent)
    {
        let touch: UITouch = touches.anyObject() as UITouch
        let touchPoint: CGPoint = touch.locationInView(self)
        let touchAngle: Float = atan2f(
            Float(touchPoint.y - _knobRect.midY),
            Float(touchPoint.x - _knobRect.midX))

        angle = touchAngle
        delegate?.knob(self, rotatedToAngle: angle)
    }

    override func drawRect(rect: CGRect)
    {
    }
}
```

```swift
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, KnobDelegate
{
    var window: UIWindow?

    func application(application: UIApplication,
       didFinishLaunchingWithOptions l: [NSObject: AnyObject]?) -> Bool
    {
        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        window?.makeKeyAndVisible()

        var knob: Knob = Knob(frame: window!.frame)
        knob.backgroundColor = UIColor.darkGrayColor()
        knob.delegate = self
        window?.addSubview(knob)

        return true
    }

    func knob(knob: Knob, rotatedToAngle angle: Float)
    {
        println("Knob rotated to angle: \(angle)")
    }
}
```

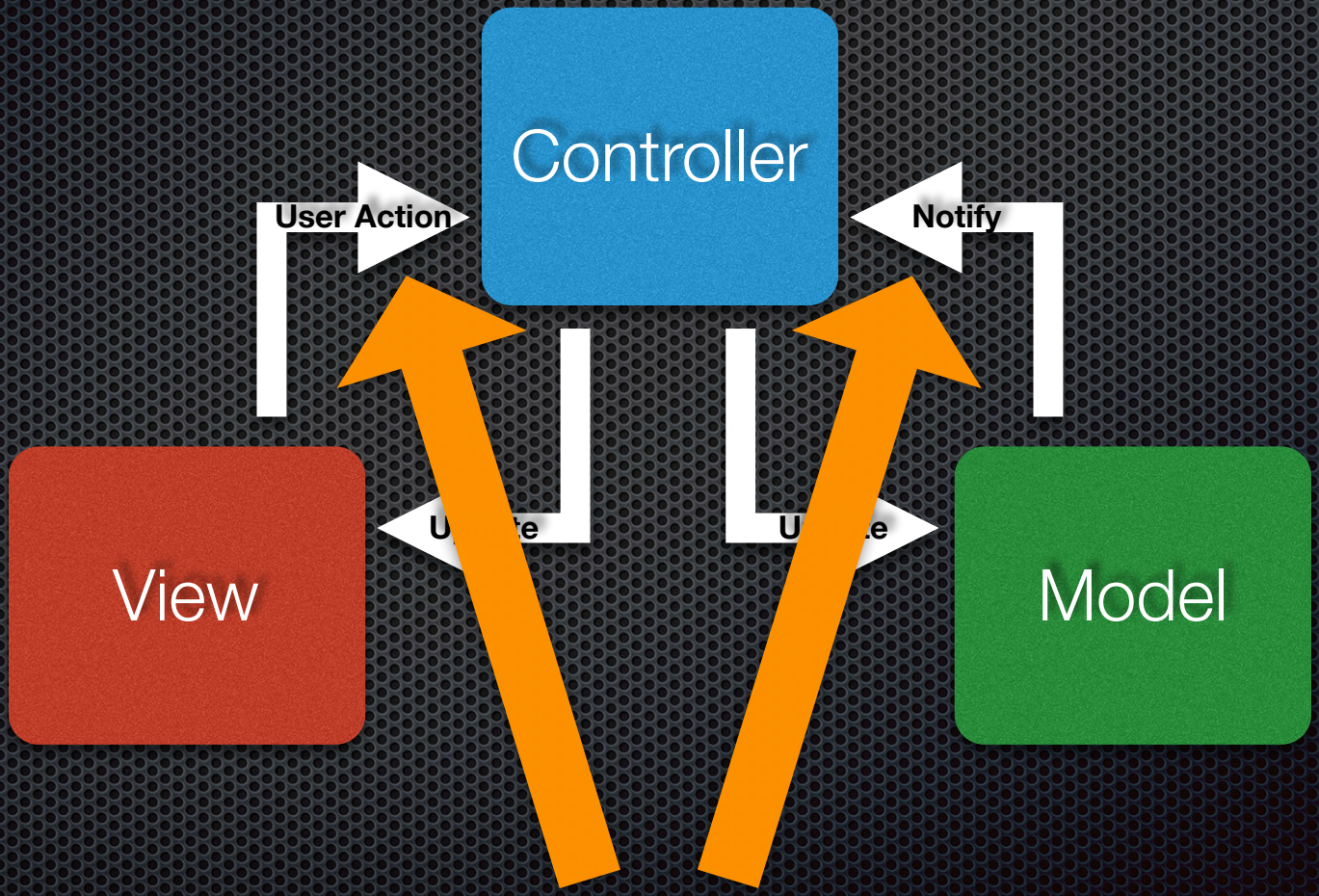
Application



Application Controller (MVC)

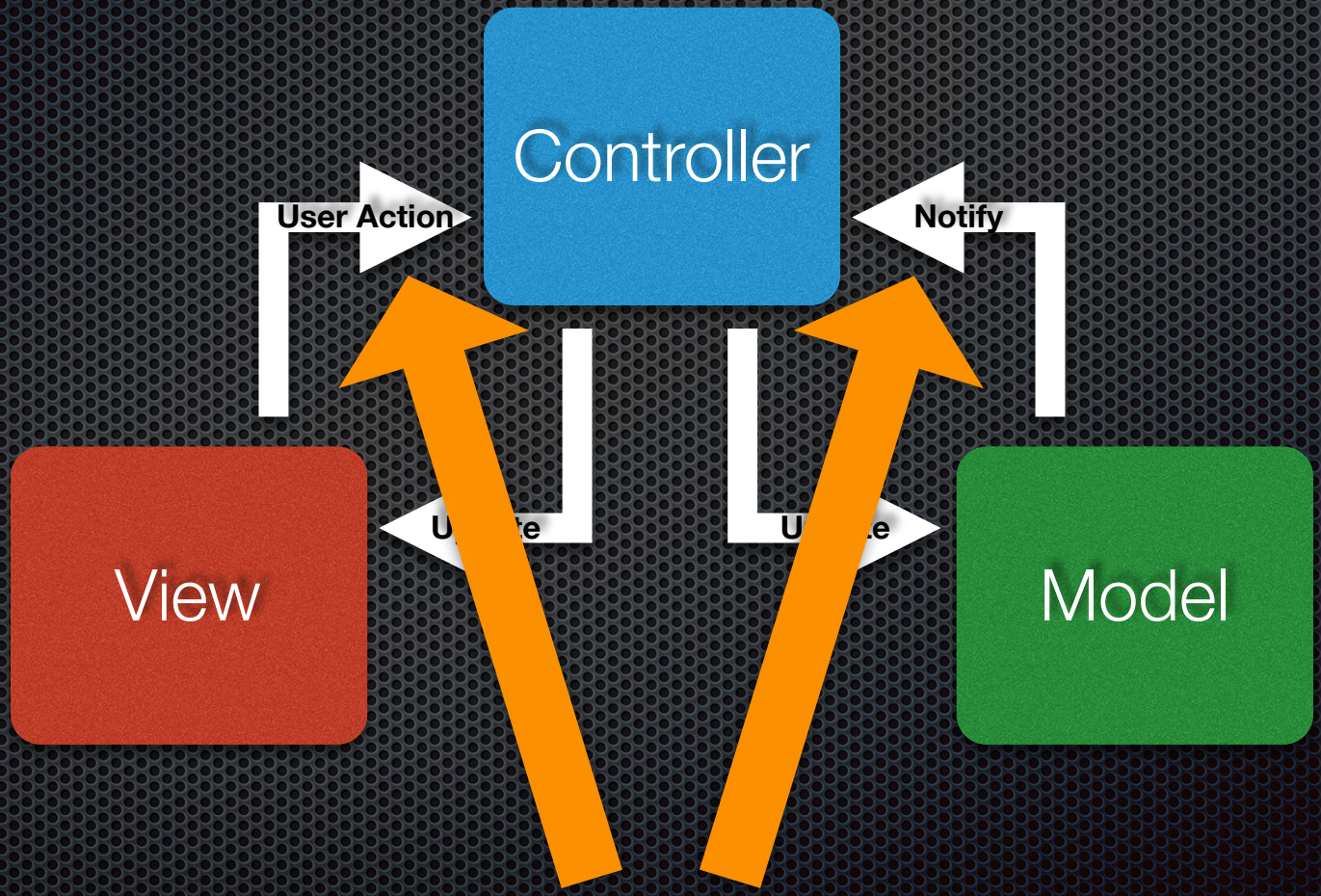


Notification



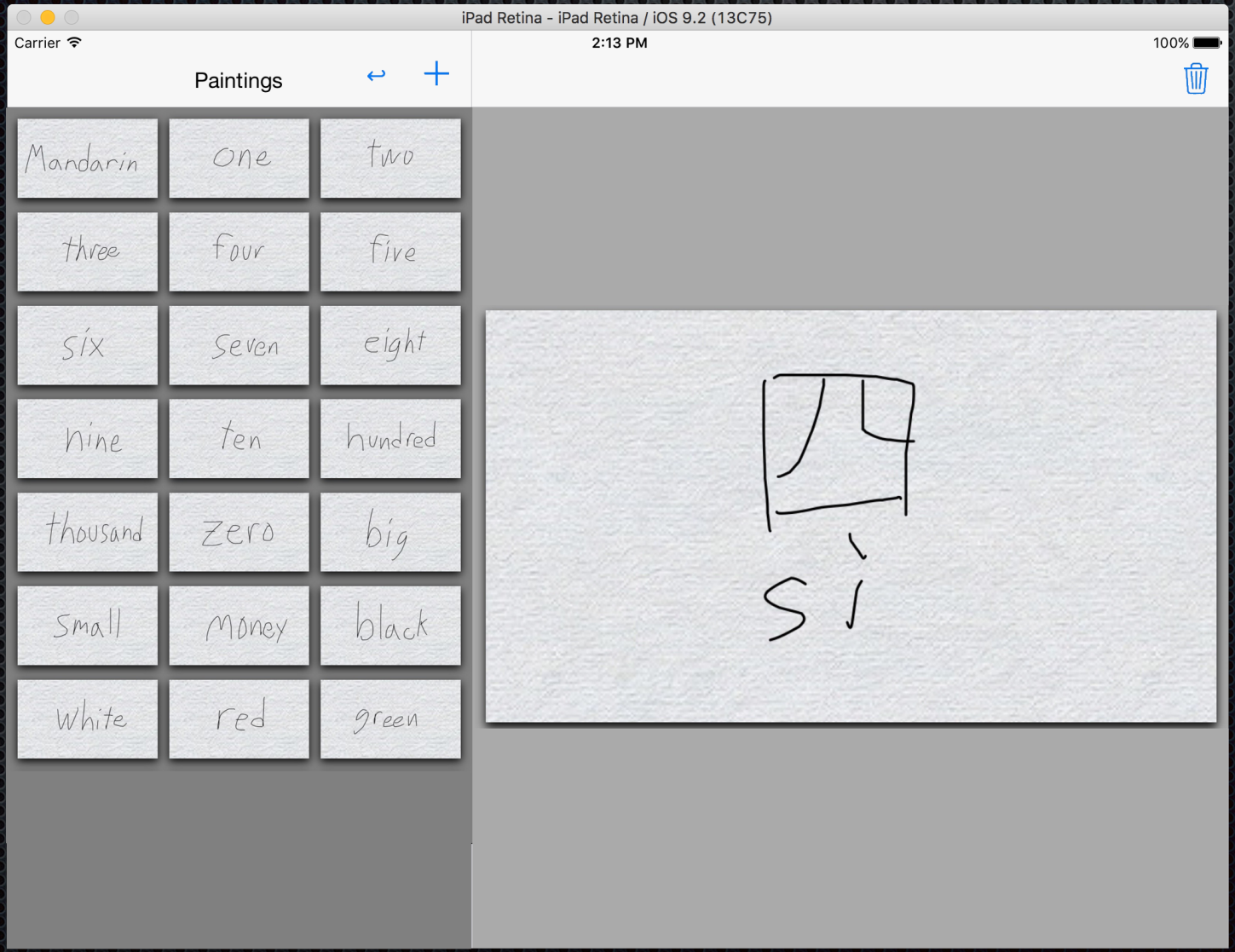
How do these happen?

Notification

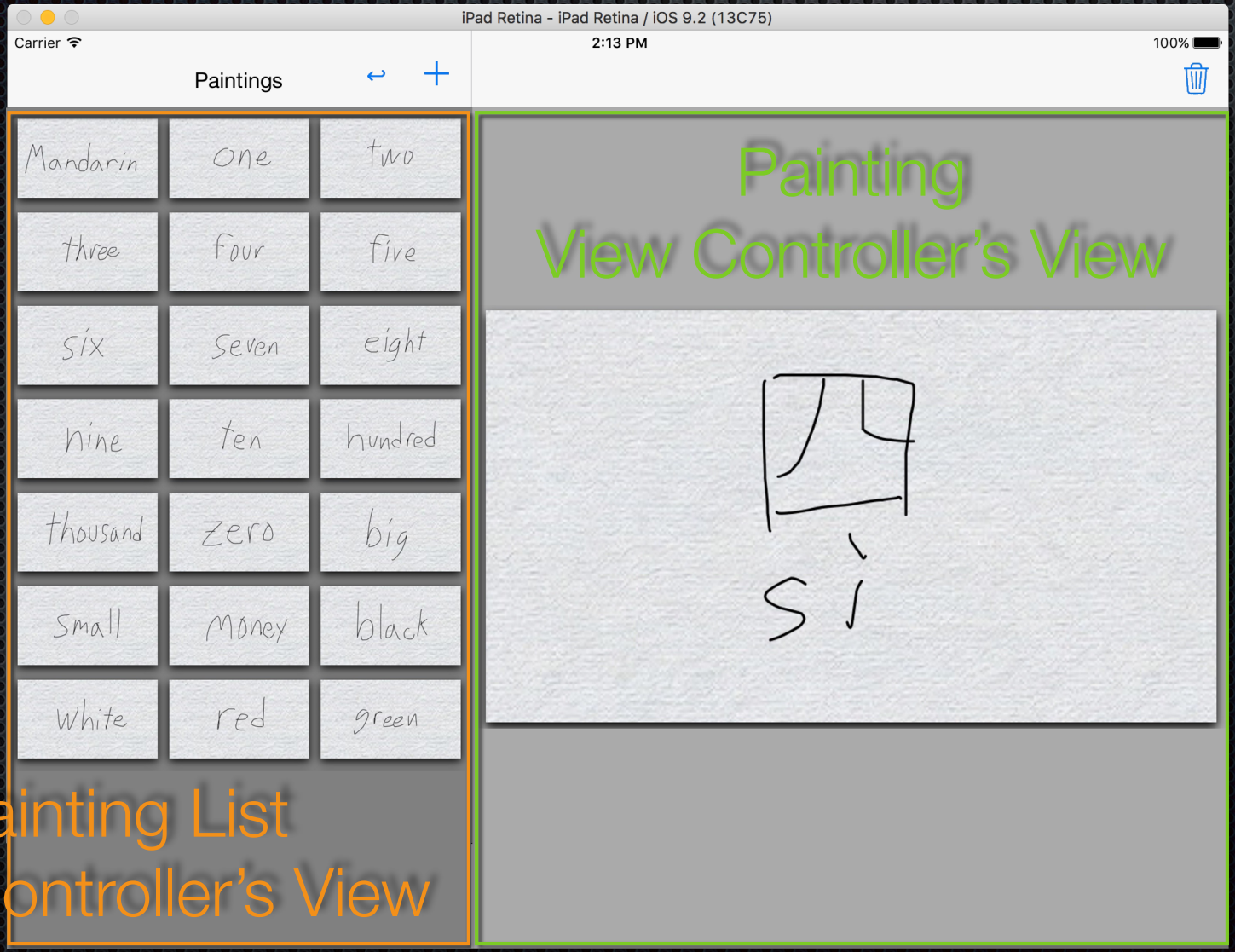


How do these happen? Delegation

Example: Paintings



Example: Paintings



Has

Weak Reference

Delegate

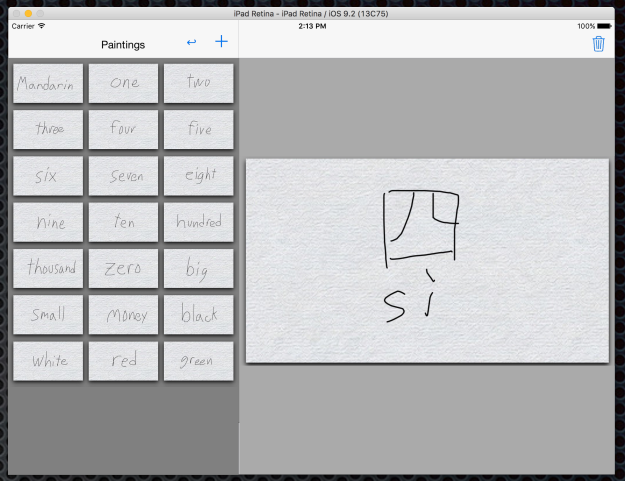
Controller

Model

View

Main Controller

App Delegate
having a Split View Controller
attached to a window



Painting List View Controller

Painting View Controller

UICollectionView

Painting Collection

PaintingView

Has

Weak Reference

Delegate

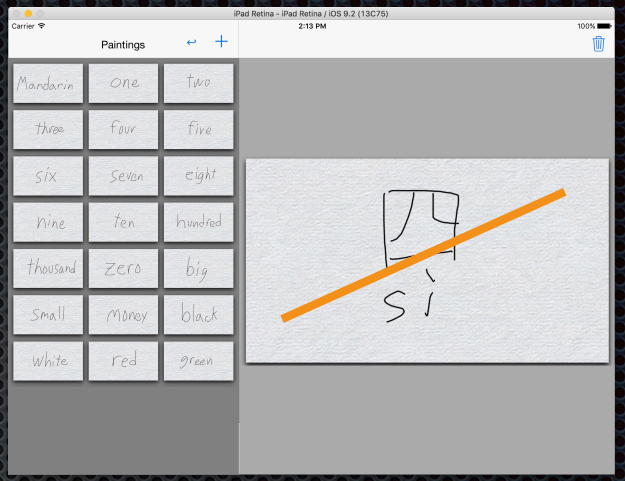
Controller

Model

View

Main Controller

App Delegate
having a Split View Controller
attached to a window



Painting List View Controller

Painting View Controller

UICollectionView

Painting Collection

PaintingView

User adds a stroke by dragging their finger then lifting it

Has

Weak Reference

Delegate

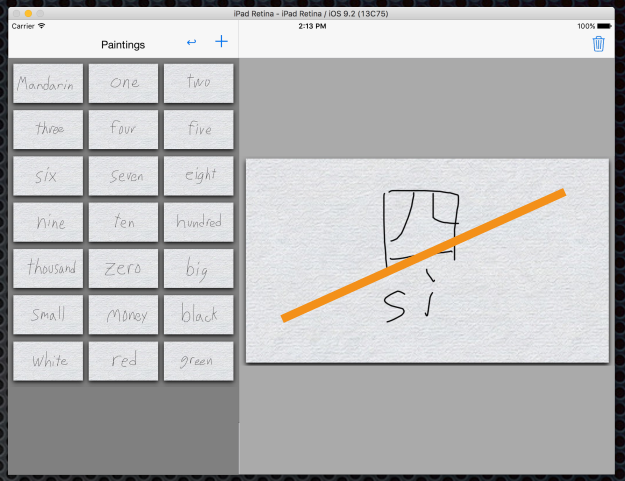
Controller

Model

View

Main Controller

App Delegate
having a Split View Controller
attached to a window



Painting List View Controller

Painting View Controller

PaintingView notifies delegate of addition

UICollectionView

Painting Collection

PaintingView

Has
Weak Reference
Delegate

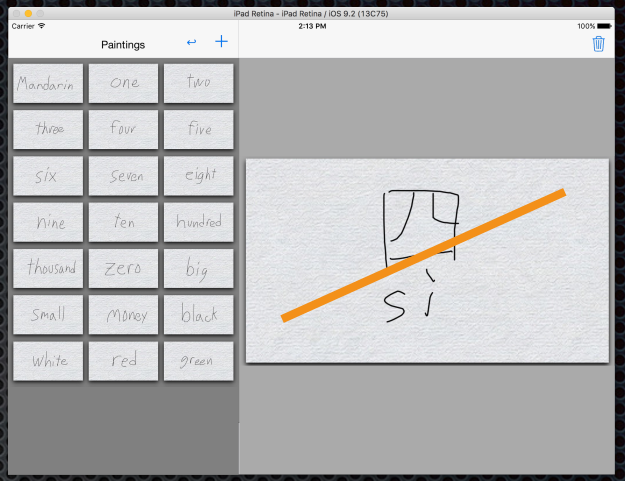
Controller

Model

View

Main Controller

App Delegate
having a Split View Controller
attached to a window



Painting View Controller requests addition of the stroke to the painting in the Painting Collection

Painting List View Controller

Painting View Controller

UICollectionView

Painting Collection

PaintingView

Has

Weak Reference

Delegate

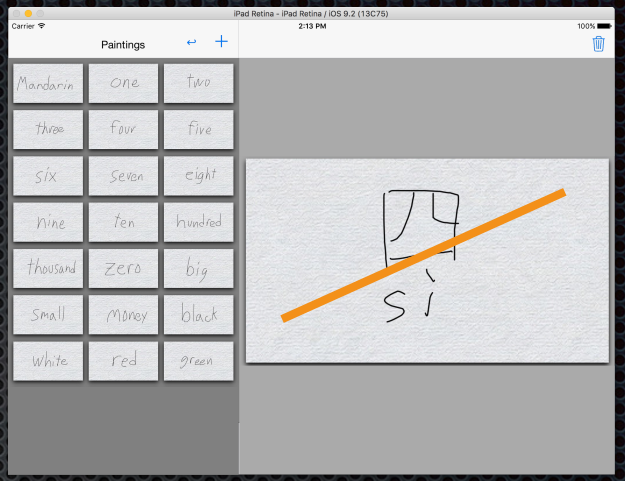
Controller

Model

View

Main Controller

App Delegate
having a Split View Controller
attached to a window



Painting List View Controller

Painting View Controller

UICollectionView

Painting Collection

PaintingView

The Painting Collection alerts its delegate of the change to its data

Has

Weak Reference

Delegate

Controller

Model

View

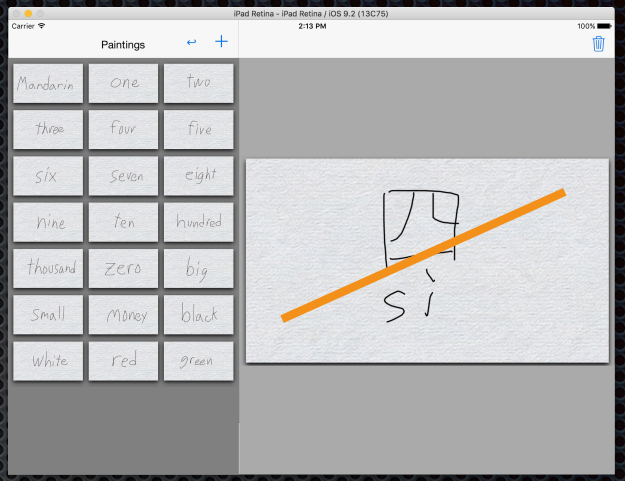
Main Controller

The Main Controller tells all view controllers that they should update their views

App Delegate

having a Split View Controller

attached to a window



Painting List View Controller

Painting View Controller

UICollectionView

Painting Collection

PaintingView

Has

Weak Reference

Delegate

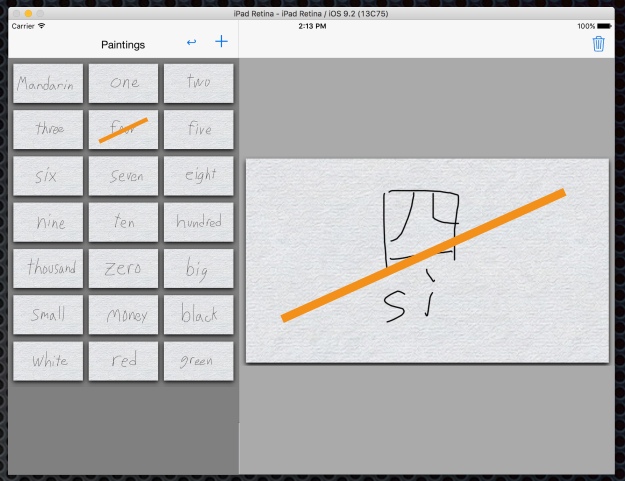
Controller

Model

View

Main Controller

App Delegate
having a Split View Controller
attached to a window



Painting List View Controller

Painting View Controller

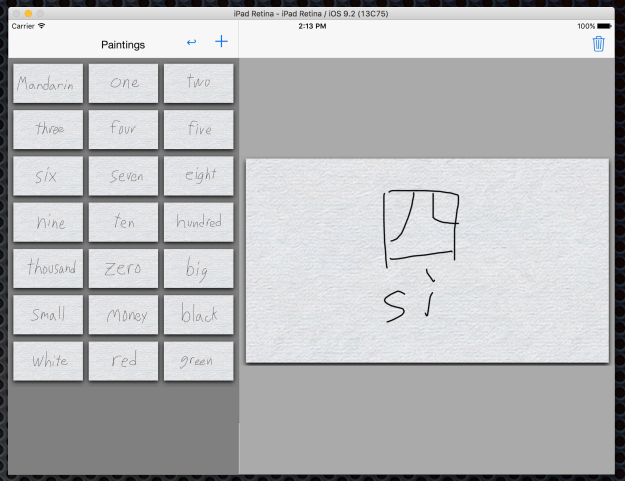
The view controllers update their views by asking the Painting Collection for the latest data

UICollectionView

Painting Collection

PaintingView

Data Conversion

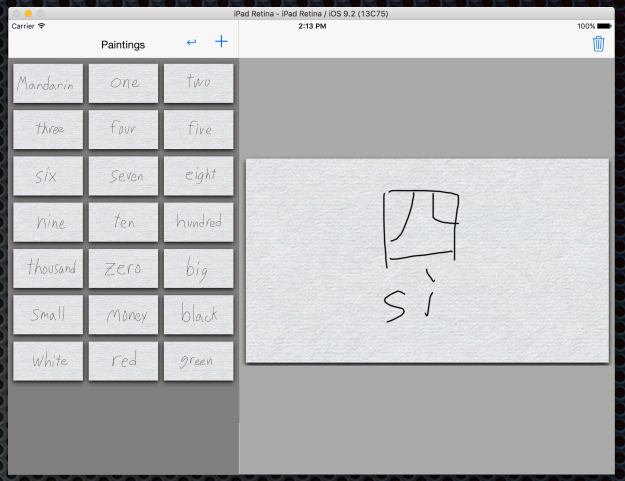


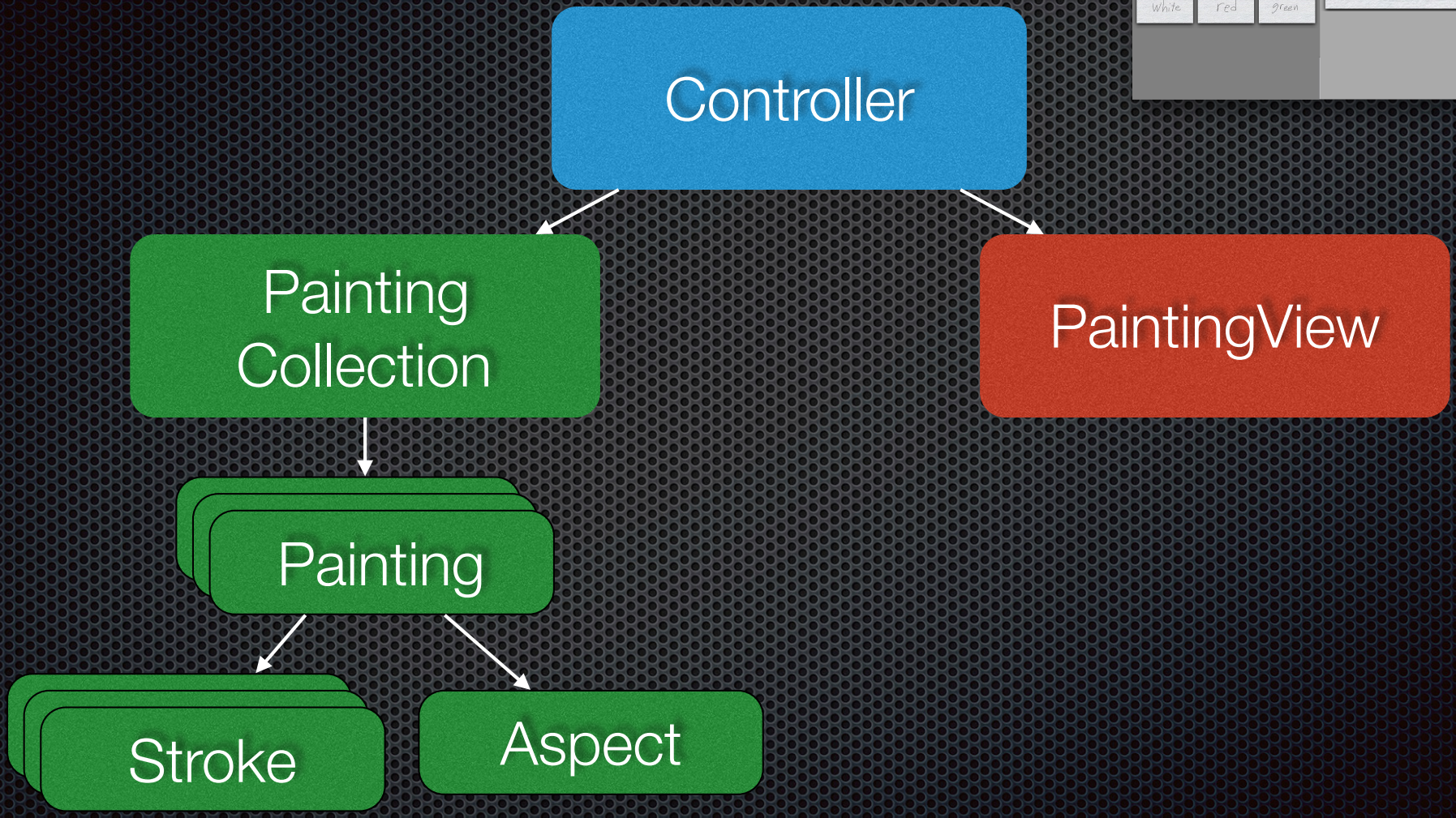
Controller

Painting Collection

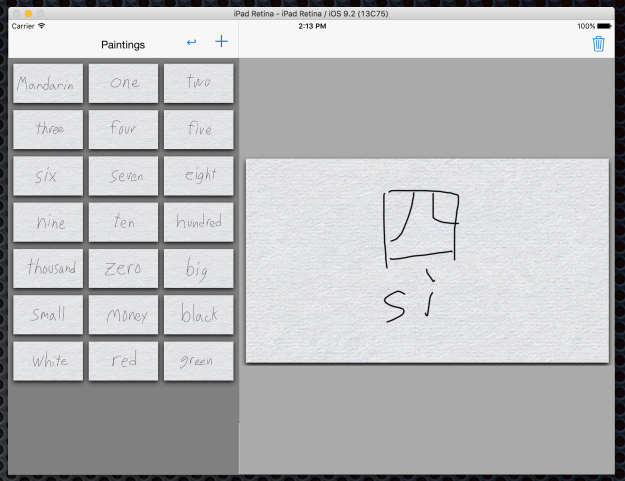
PaintingView

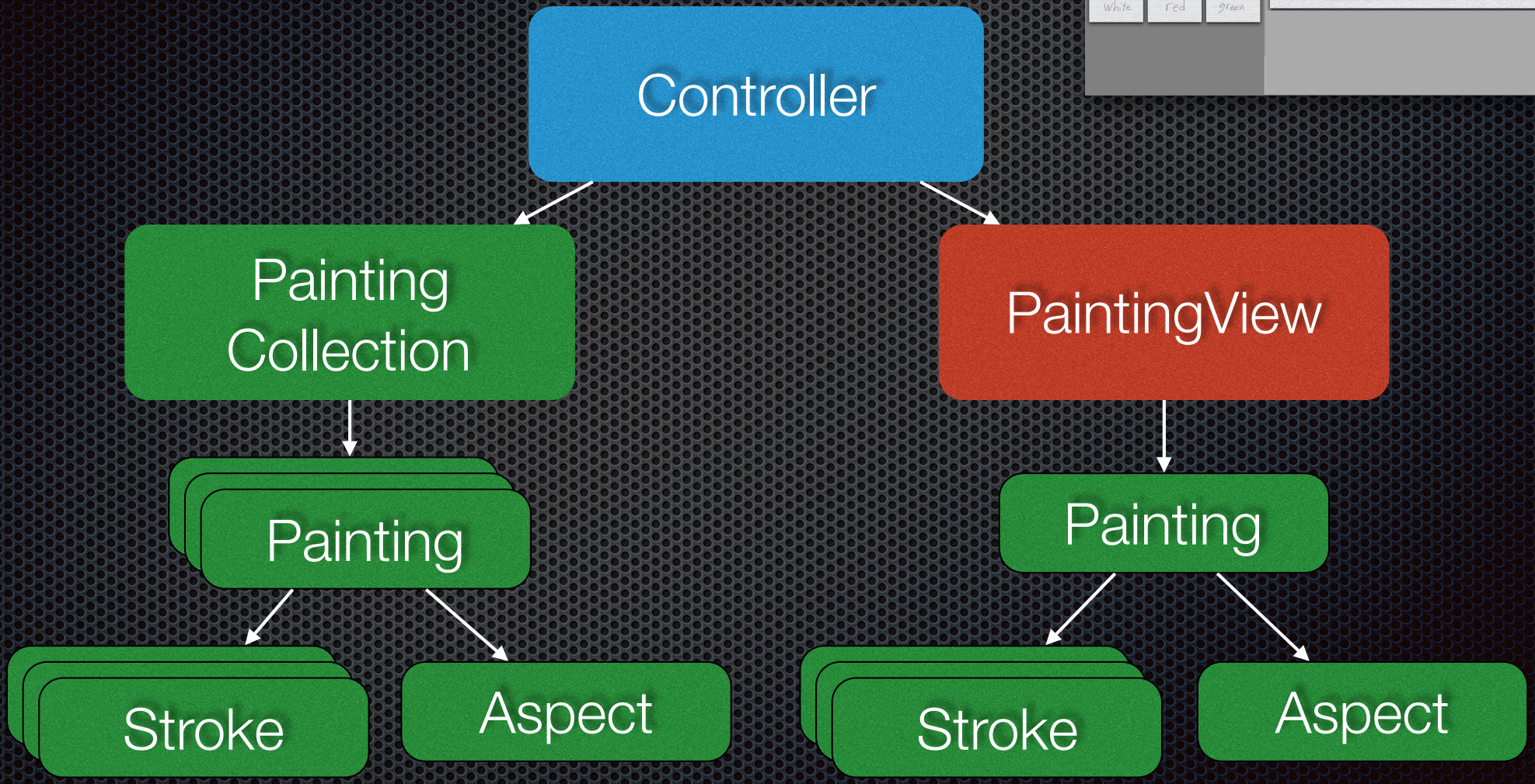
Data Conversion



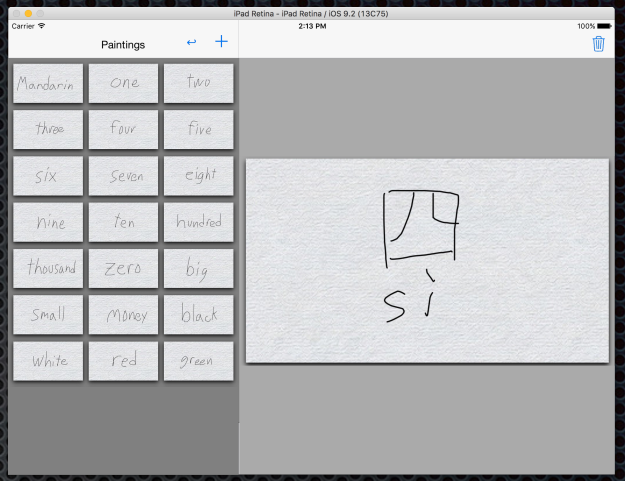


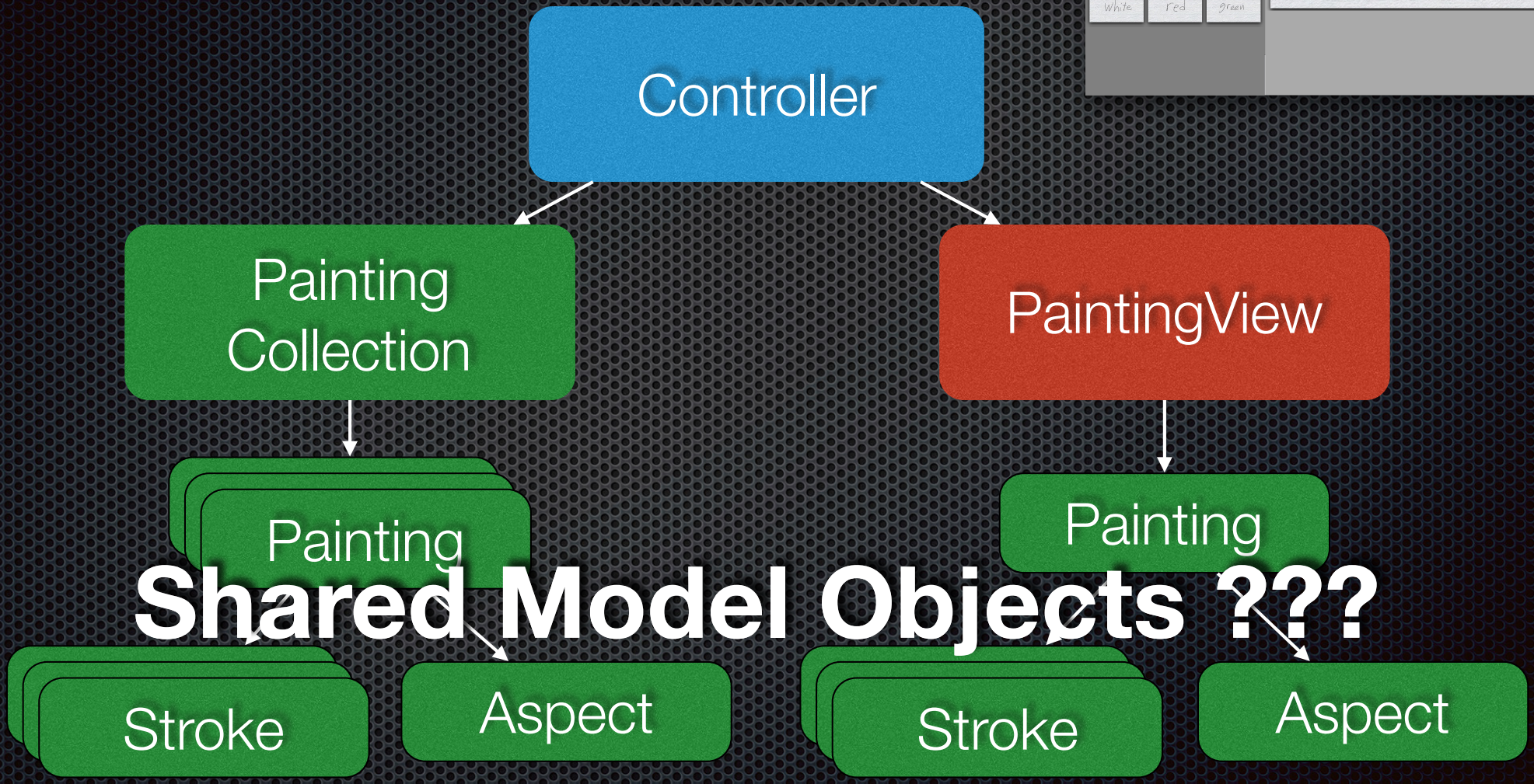
Data Conversion



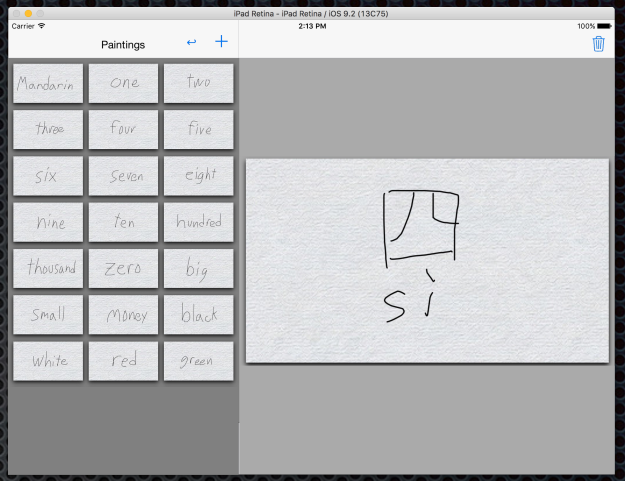


Data Conversion





Data Conversion



Controller

Painting Collection

PaintingView

Painting

Painting

Shared Model Objects ???

Stroke

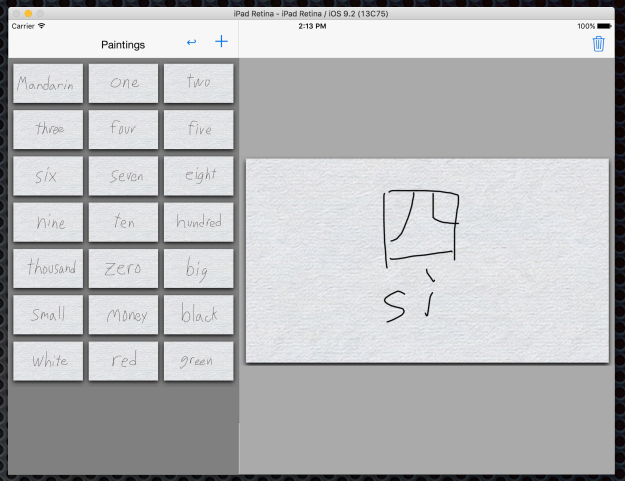
Aspect

Stroke

Aspect

Strange Data?

Data Conversion



Controller

Painting Collection

PaintingView

Persistence?

Painting

Painting

Shared Model Objects ???

Stroke

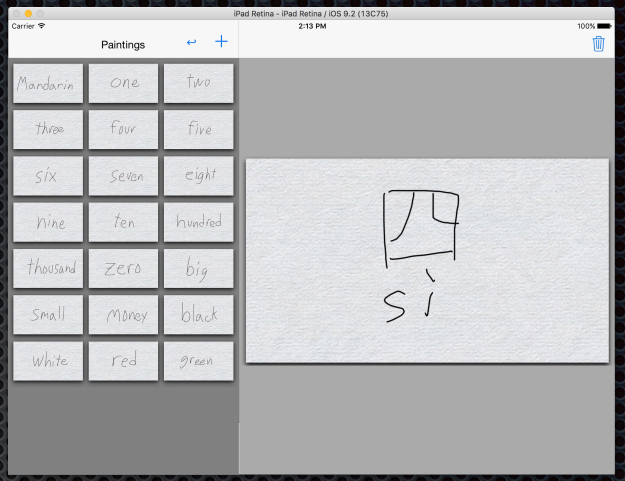
Aspect

Stroke

Aspect

Strange Data?

Data Conversion



Server Updates?

Controller

Painting Collection

PaintingView

Persistence?

Painting

Painting

Shared Model Objects ???

Stroke

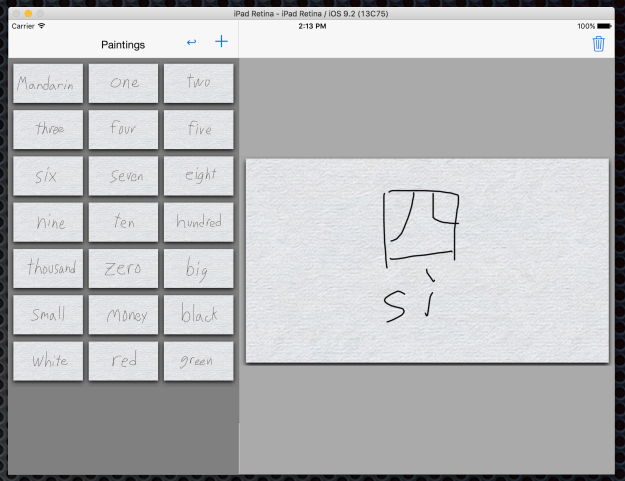
Aspect

Stroke

Aspect

Strange Data?

Data Conversion



Server Updates?

Controller

Painting Collection

PaintingView

Persistence?

Painting

Painting

Shared Model Objects ???

Stroke

Aspect

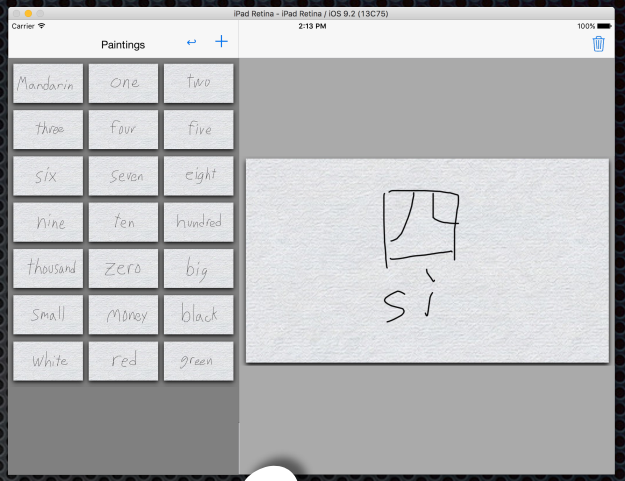
Stroke

Aspect

Mutability?

Strange Data?

Data Conversion



Server Updates?

Controller

Painting Collection

PaintingView

Persistence?

Changing Data?

Painting

Painting

Painting

Shared Model Objects ???

Concurrently?

Stroke

Aspect

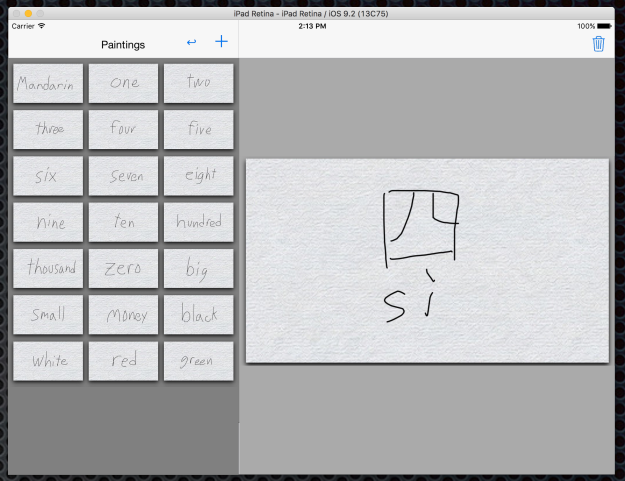
Stroke

Aspect

Mutability?

Strange Data?

Data Conversion



Controller

Painting Collection

Painting

Stroke

Aspect

PaintingView

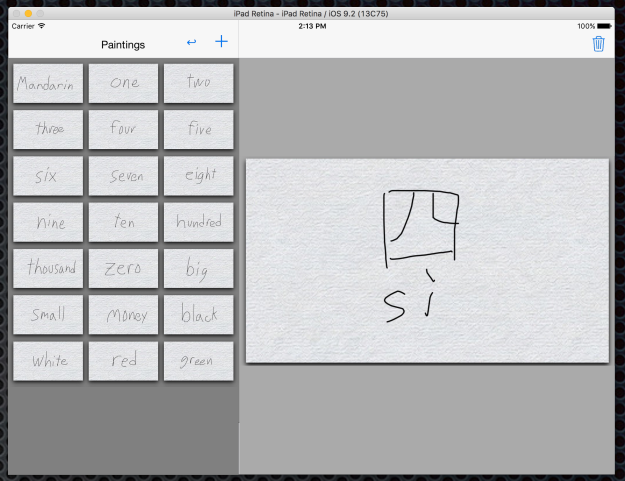
Polyline

Brush

Current Polyline

Brush

Data Conversion



Controller

Painting Collection

Painting

Stroke

Aspect

x:0->1, y:0->1

16/9

PaintingView

Polyline

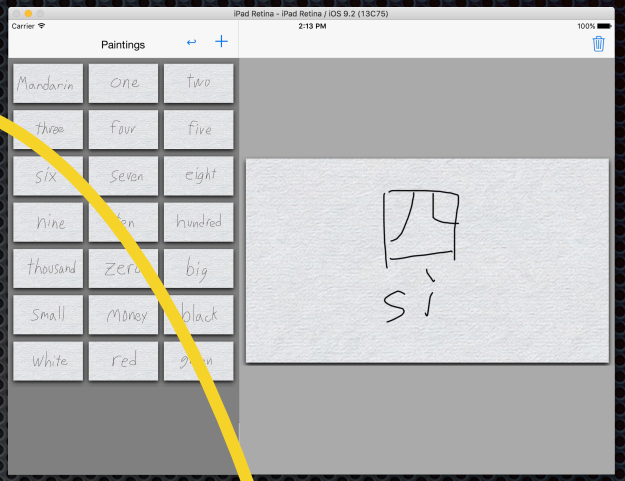
Brush

x:0->width, y:0->height

Current Polyline

Brush

Data Conversion



Controller

Painting Collection

Painting

Stroke

Aspect

PaintingView

Polyline

Brush

x:0->width, y:0->height

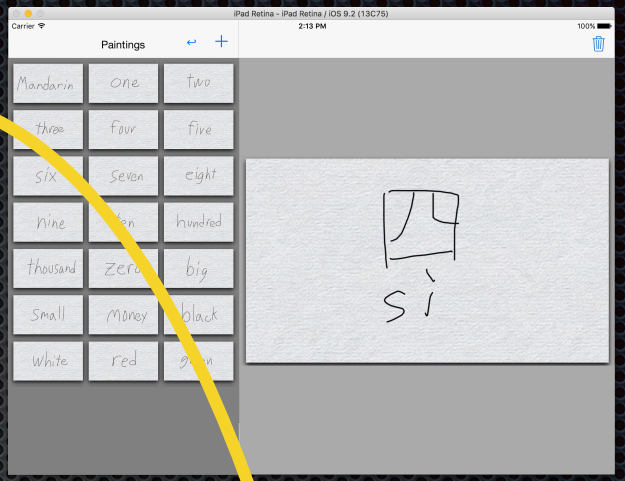
Current Polyline

Brush

x:0->1, y:0->1

16/9

Data Conversion



model → view

model ← view

Controller

Painting Collection

PaintingView

Painting

Polyline

x:0->width, y:0->height

Current Polyline

Stroke

Aspect

Brush

Brush

x:0->1, y:0->1

16/9