iOS Programming Course Intro

Andrejus Belovas

What will I learn in this course?

- How to build cool apps
  Easy to build even very complex applications.
  Result lives in your pocket or backpack!
  Very easy to distribute your application through the AppStore.
  Vibrant development community.
- Real-life Object-Oriented Programming
  The heart of Cocoa Touch is 100% object-oriented.
  Application of MVC design model.
  Many computer science concepts applied in a commercial development platform

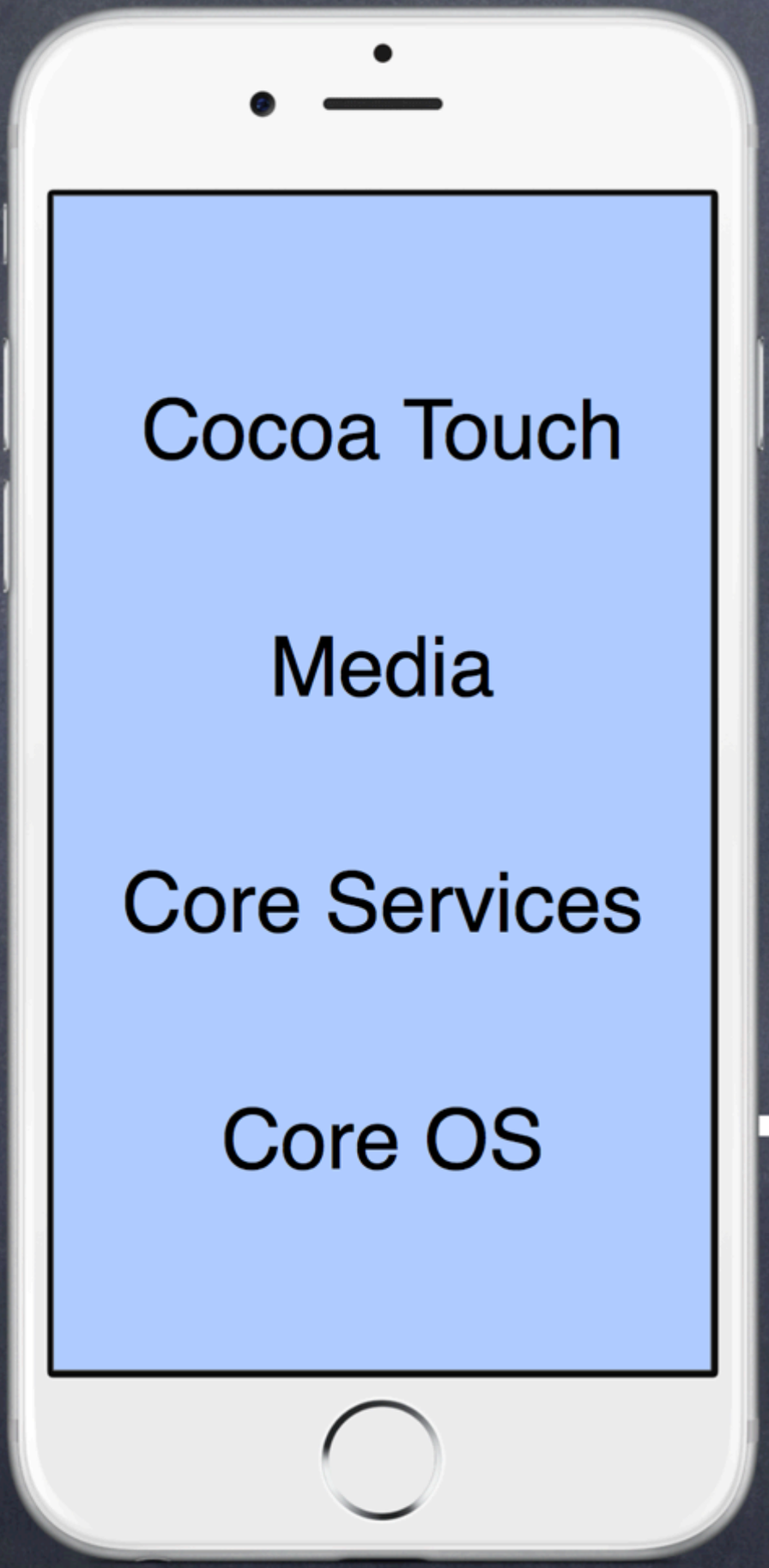
Prerequisites

- Prior Coursework
  Object-Oriented Programming experience mandatory.

- You should know well the meaning of these terms ...
  Class (description/template for an object)
Instance (manifestation of a class)
Message (sent to object to make it act)
    Method (code invoked by a Message)
Instance Variable (object-specific storage)
Superclass/Subclass (Inheritance)
If you are not very comfortable with all of these, this is probably not the class for you!

- Programming Experience
  If you have never written a program where you had to design and implement more than a handful of classes, this will be a big step up in difficulty for you.

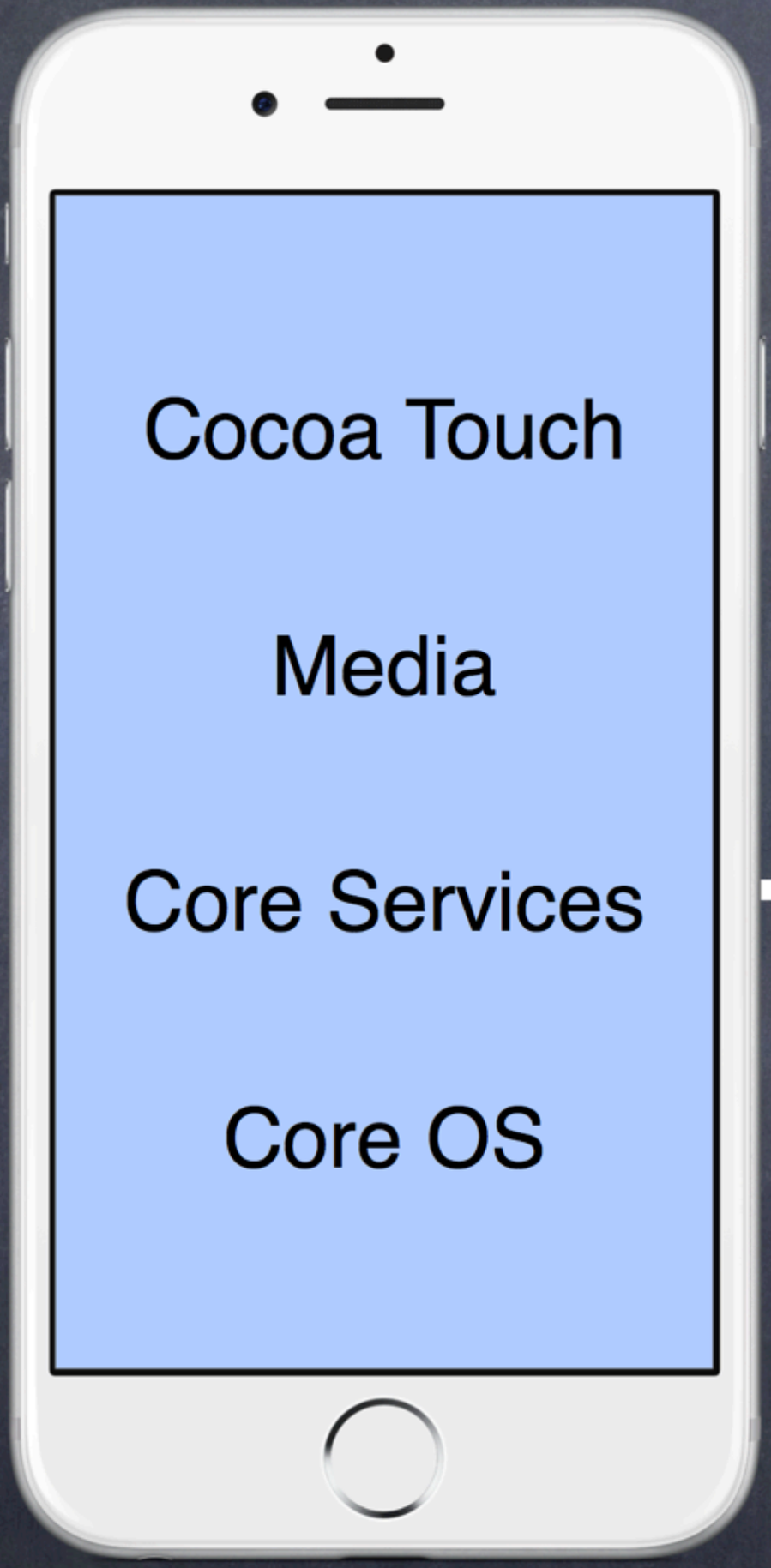
What's in iOS?



Core OS

OSX Kernel
Power Management
Mach 3.0
Keychain Access
BSD
Certificates
Sockets
File System
Security
Bonjour

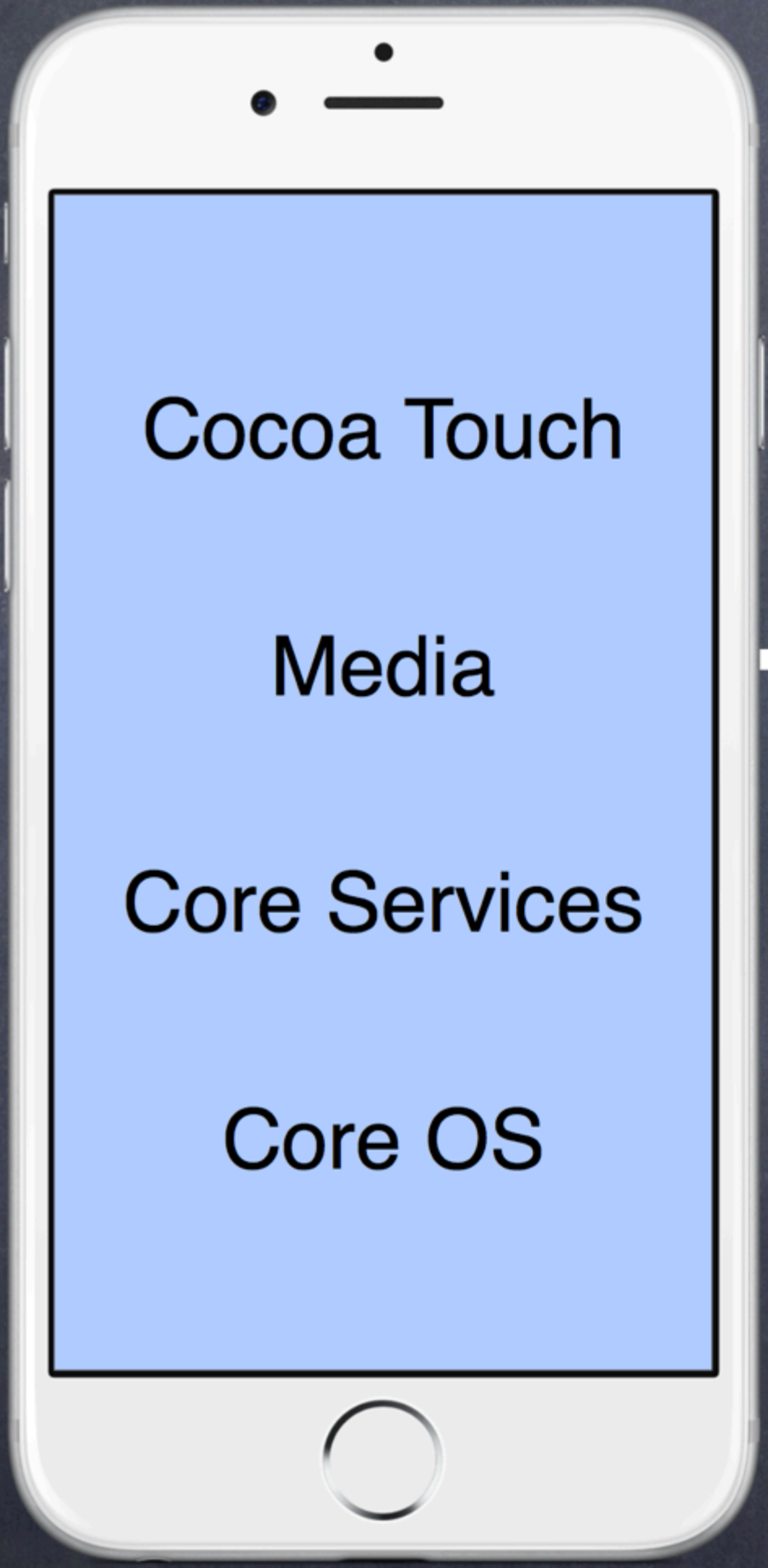
What's in iOS?



Core Services

- Collections
- Core Location
- Address Book
- Net Services
- Networking
- Threading
- File Access
- Preferences
- SQLite
- URL Utilities

What's in iOS?



Media

Core Audio

OpenAL

Audio Mixing

Audio Recording

Video Playback

JPEG, PNG, TIFF

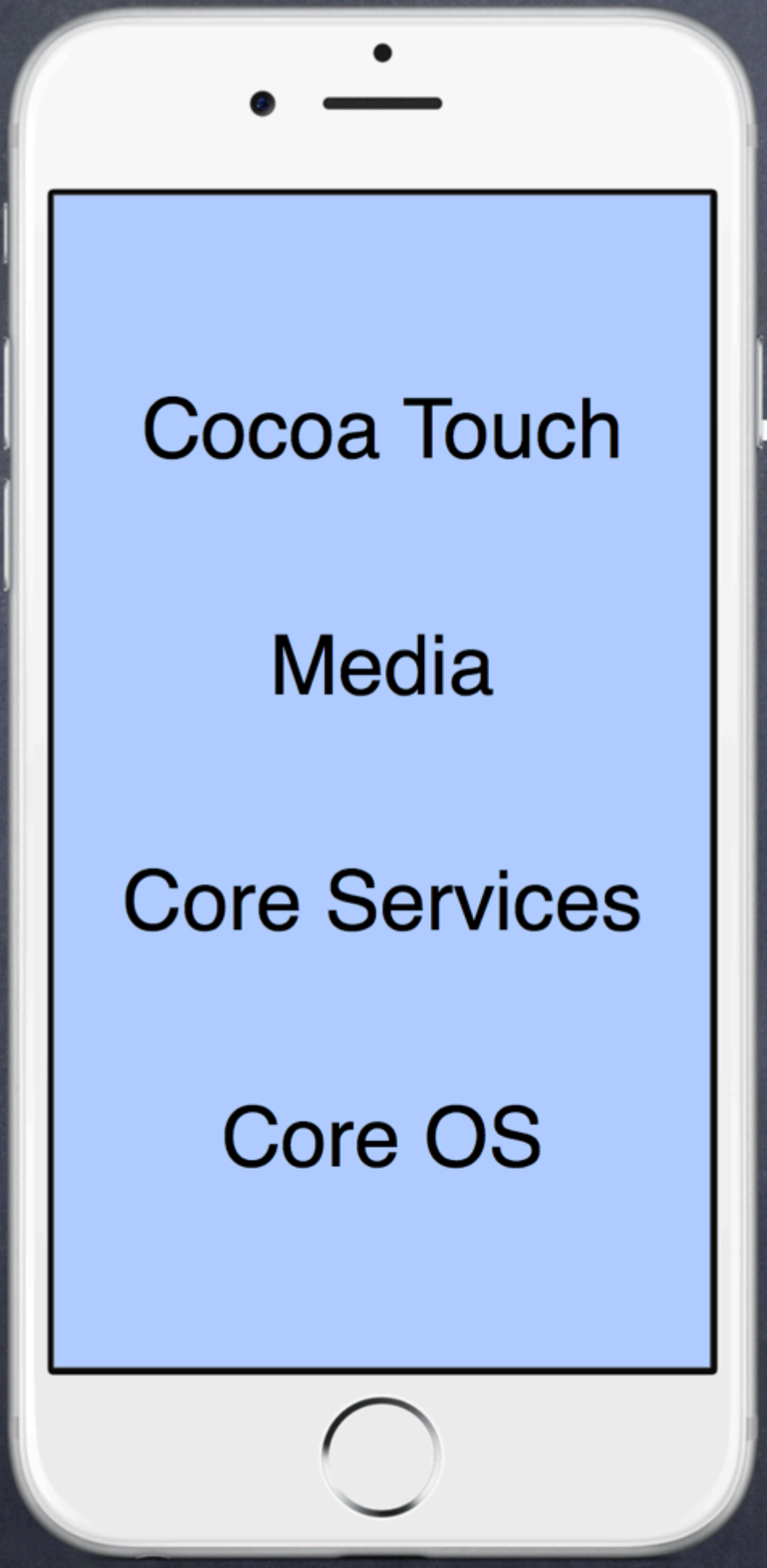
PDF

Quartz (2D)

Core Animation

OpenGL ES

What's in iOS?



Cocoa Touch

- Multi-Touch
- Core Motion
- View Hierarchy
- Localization
- Controls
- Alerts
- Web View
- Map Kit
- Image Picker
- Camera

Platform Components

- Tools

Xcode 6

Instruments

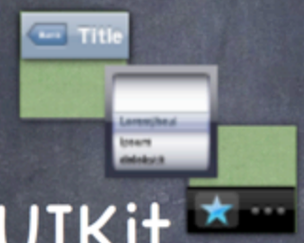
- Language(s)

```
let value = formatter.numberFromString(display.text!)?.doubleValue
```

- Frameworks

Foundation

Core Data



UIKit

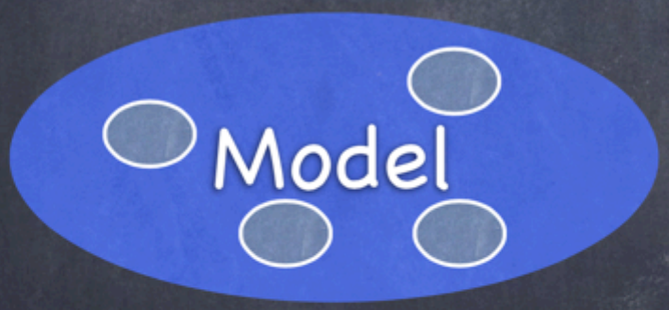
Core Motion

Map Kit

- Design Strategy

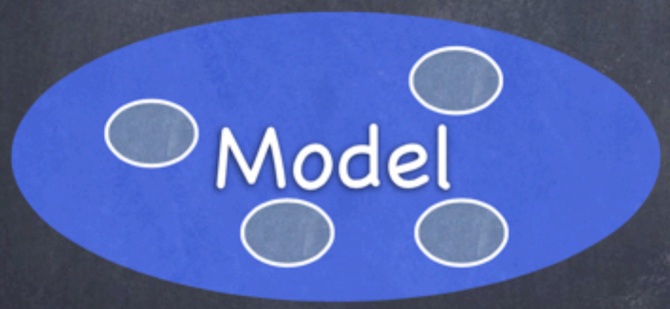
MVC

MVC



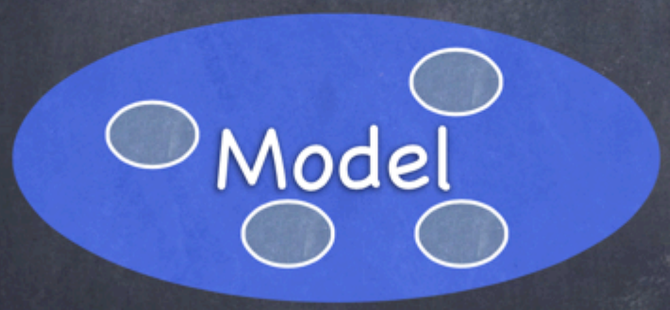
Divide objects in your program into 3 "camps."

MVC



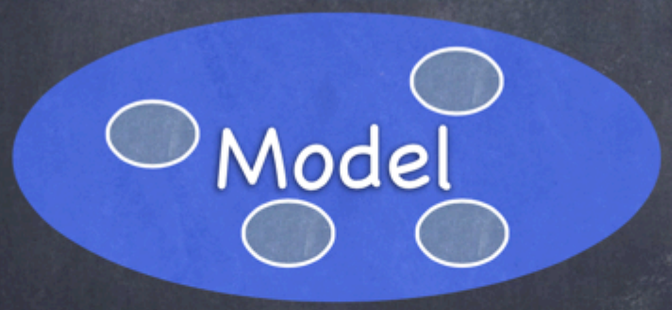
Model = What your application is (but not how it is displayed)

MVC



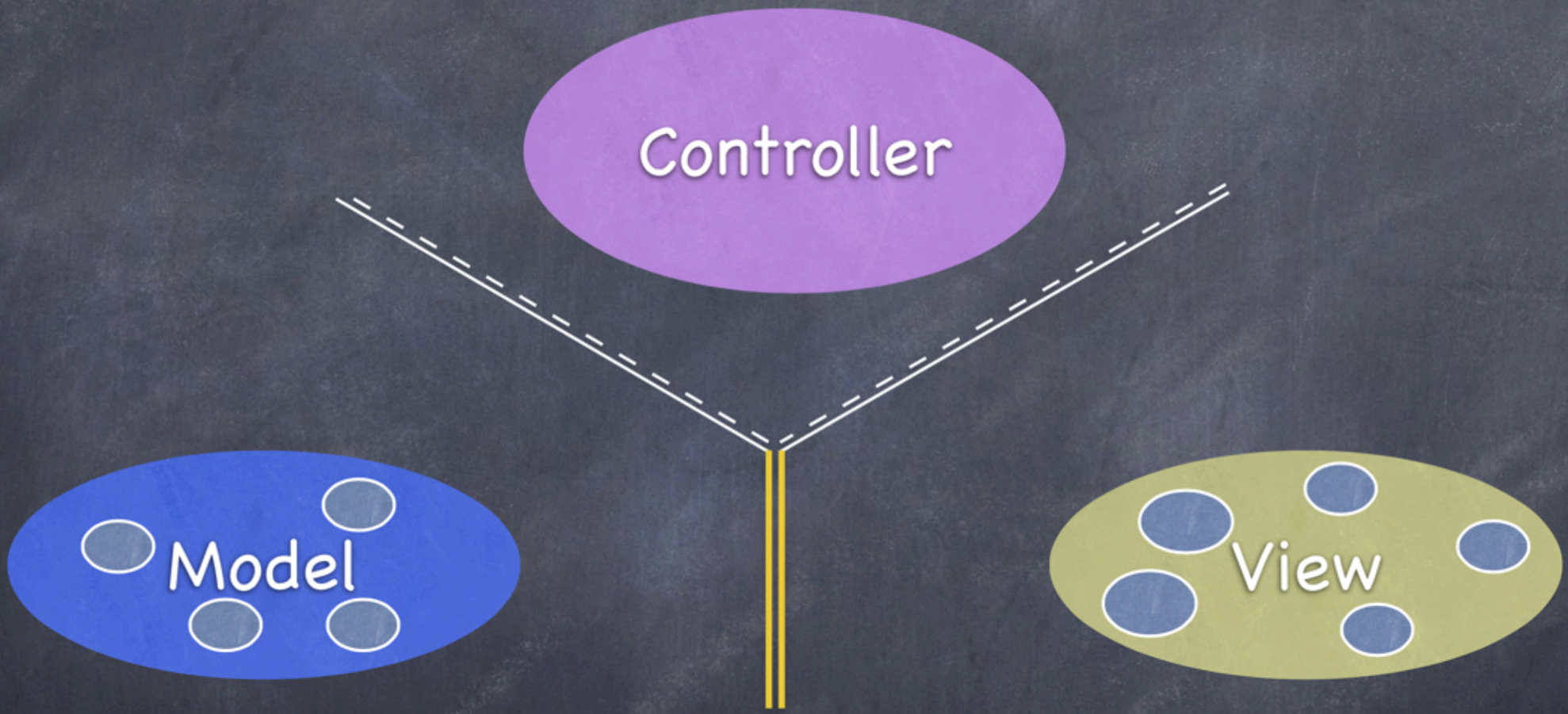
Controller = How your Model is presented to the user (UI logic)

MVC



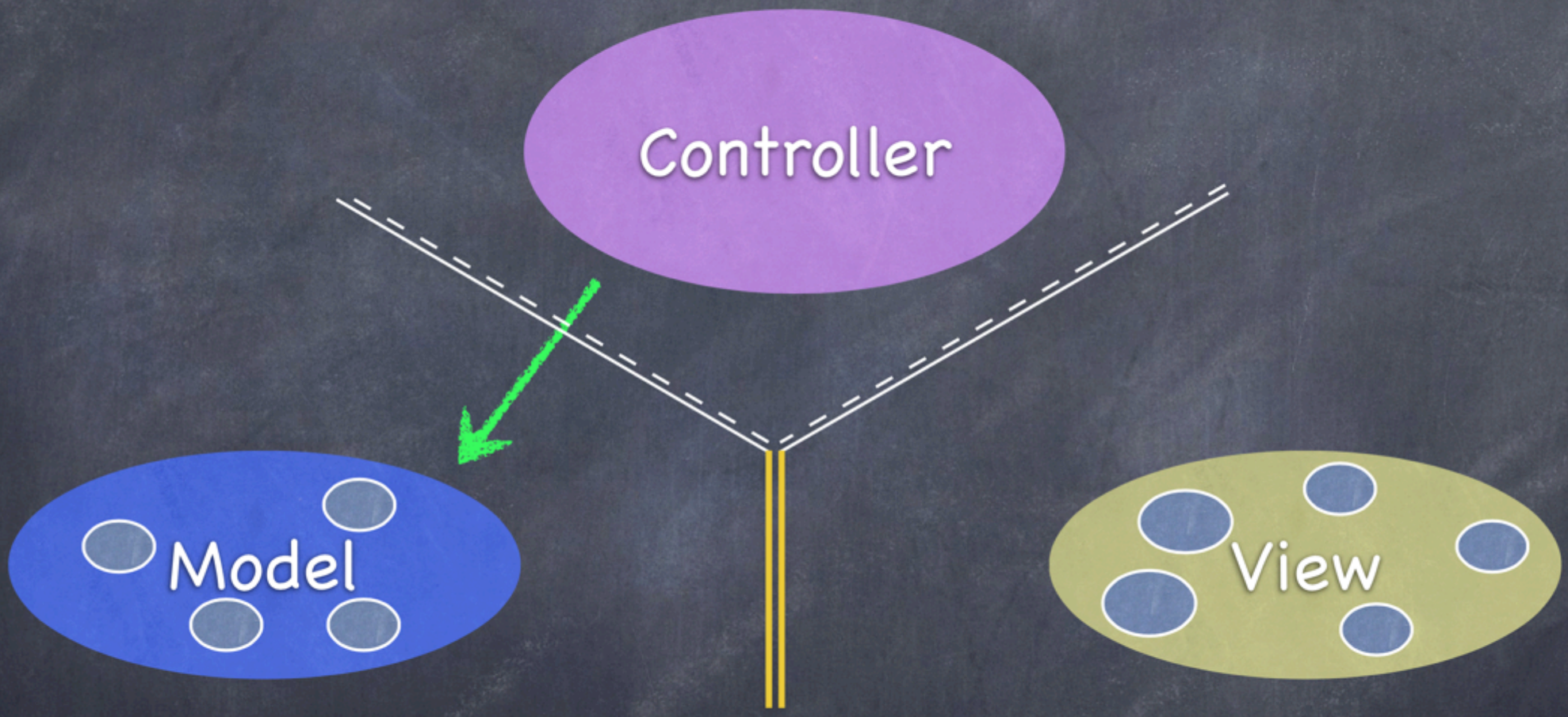
View = Your Controller's minions

MVC



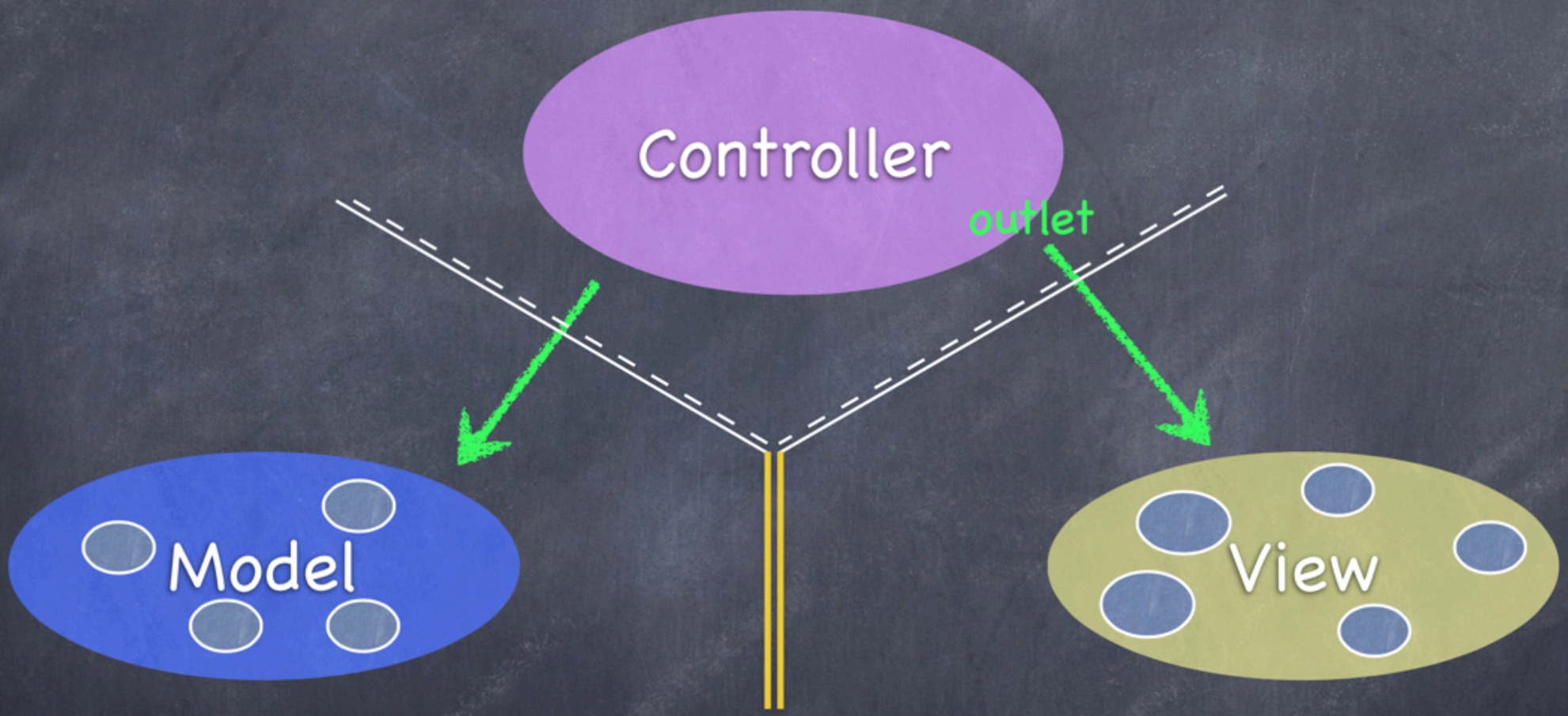
It's all about managing communication between camps

MVC



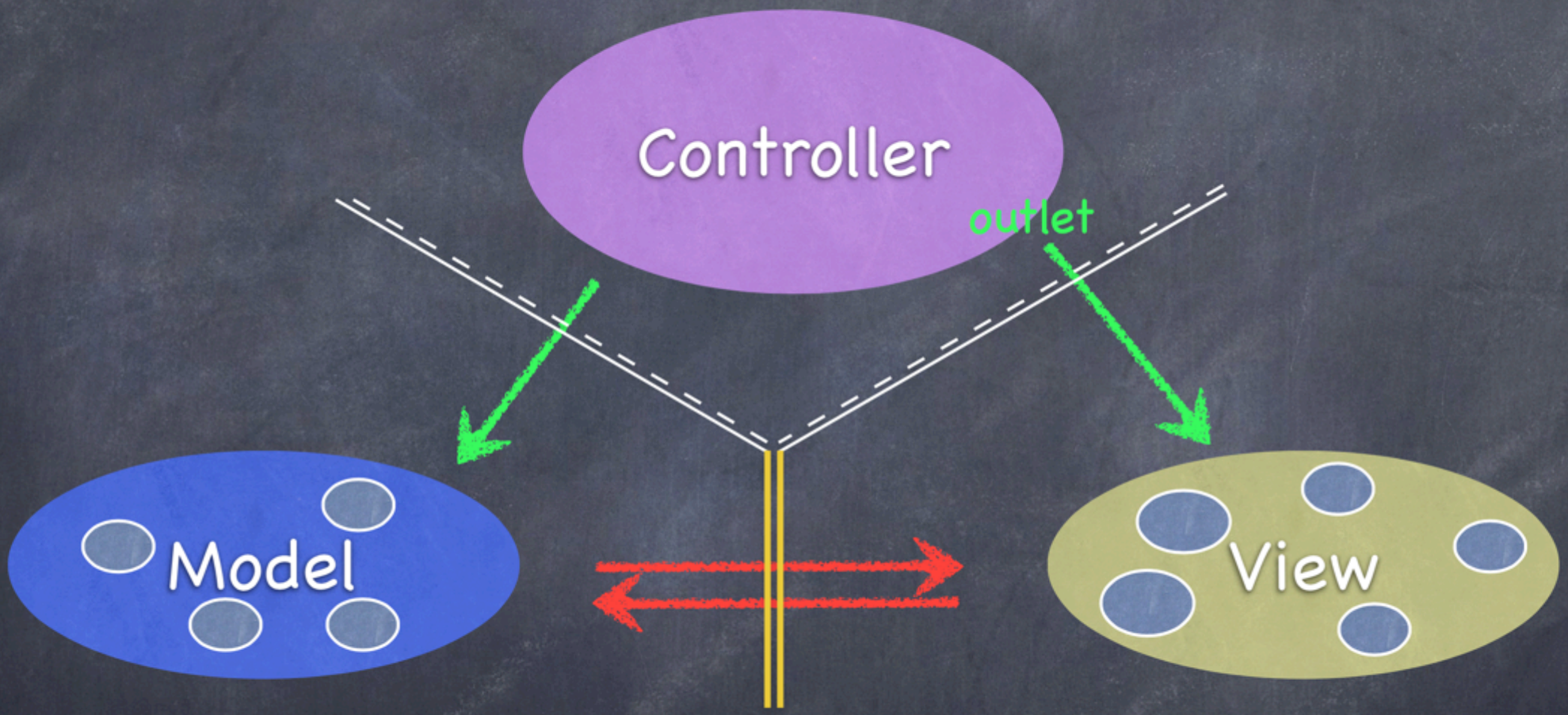
Controllers can always talk directly to their Model.

MVC



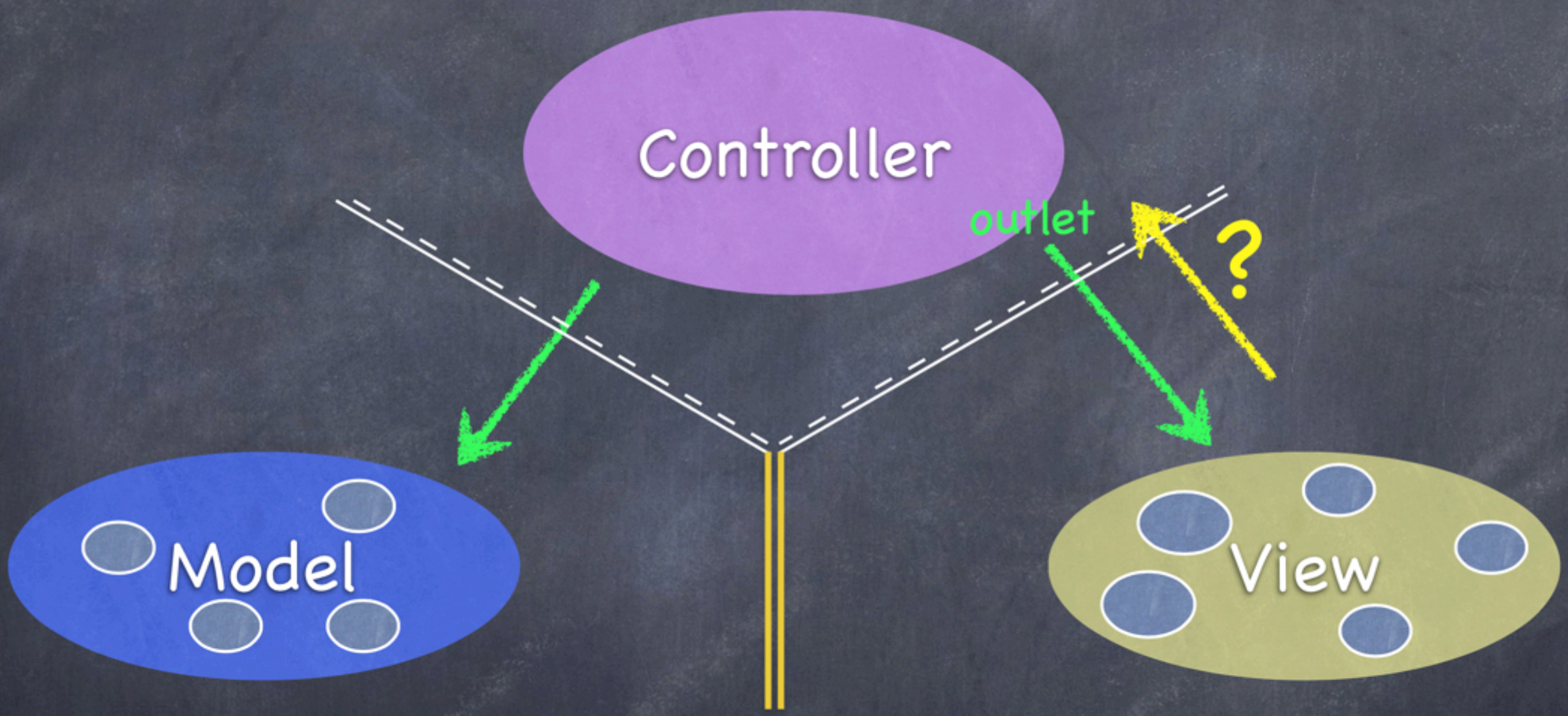
Controllers can also talk directly to their View.

MVC



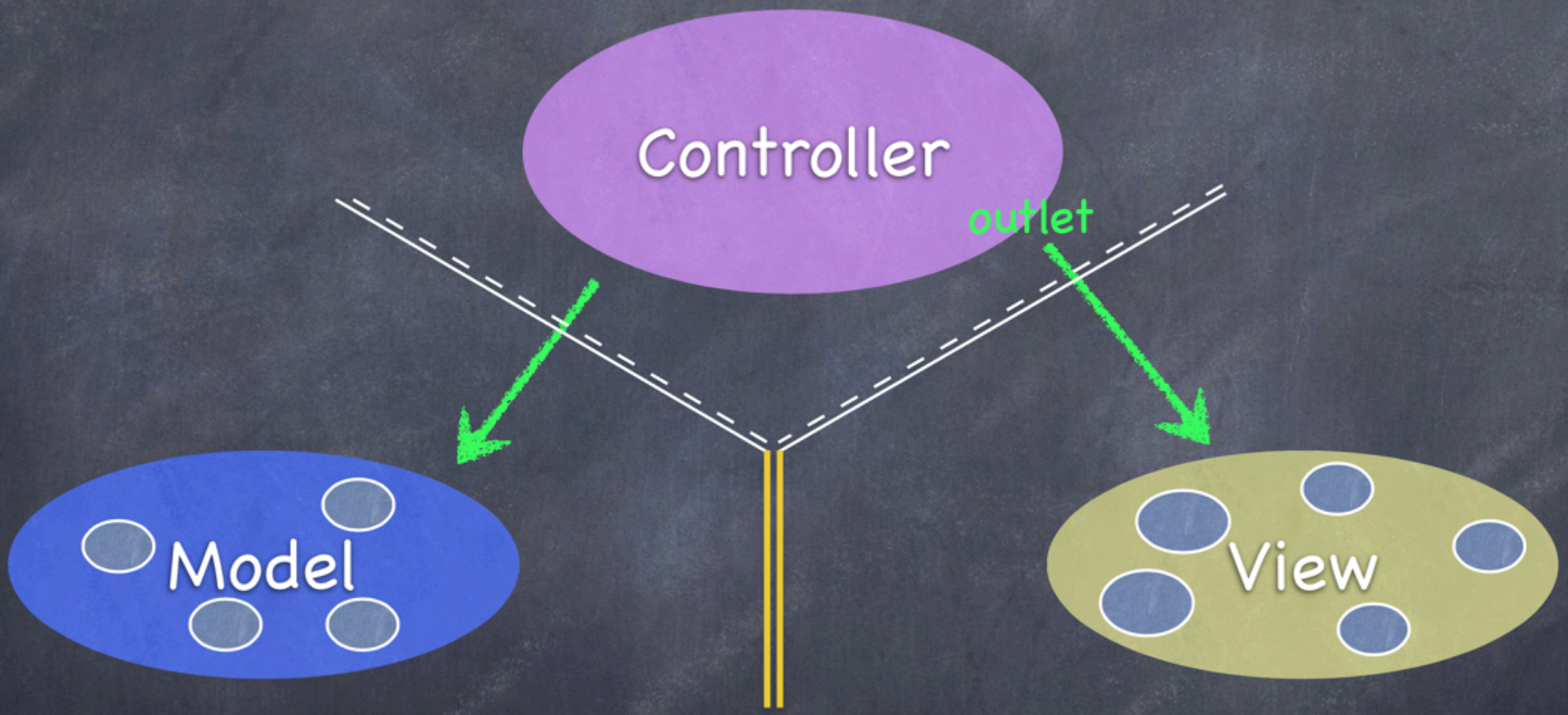
The Model and View should never speak to each other.

MVC



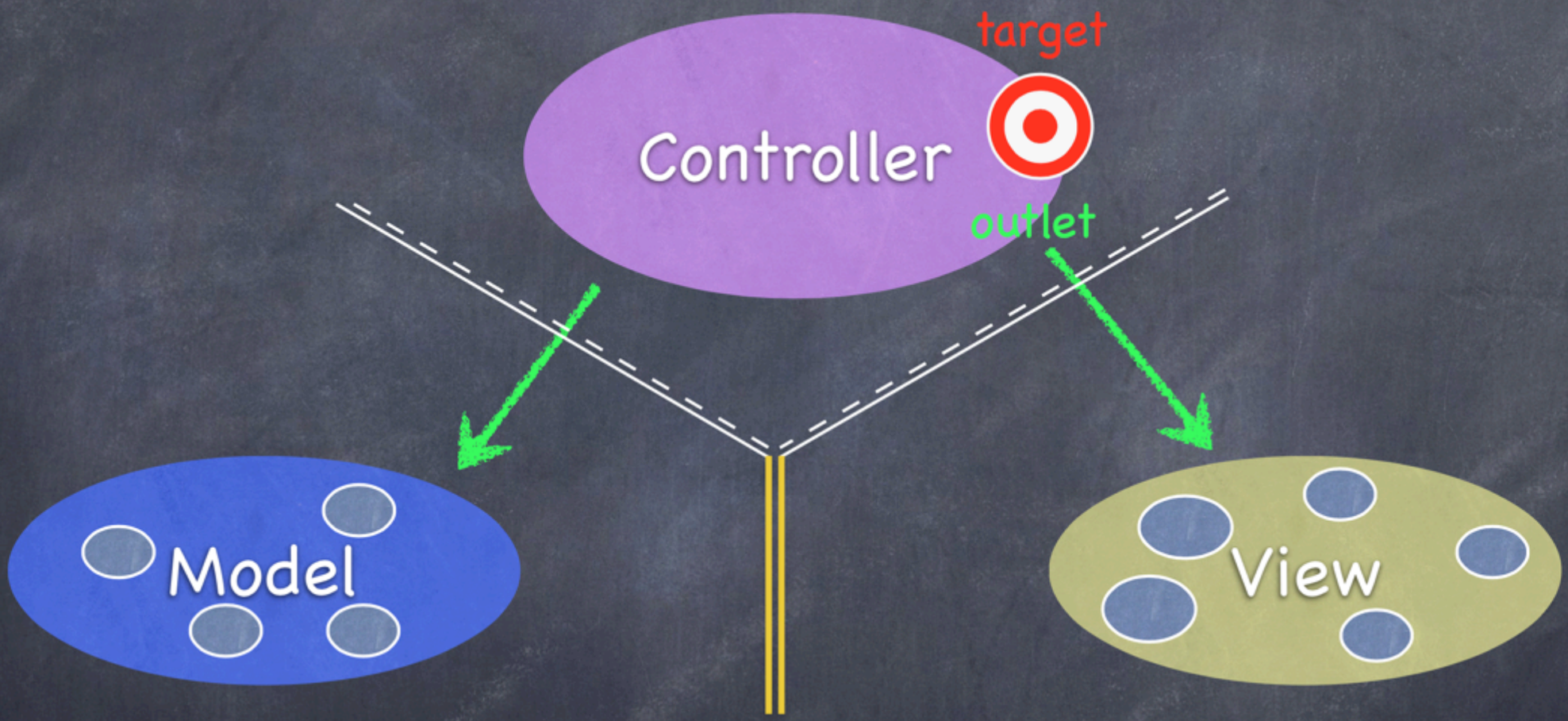
Can the View speak to its Controller?

MVC



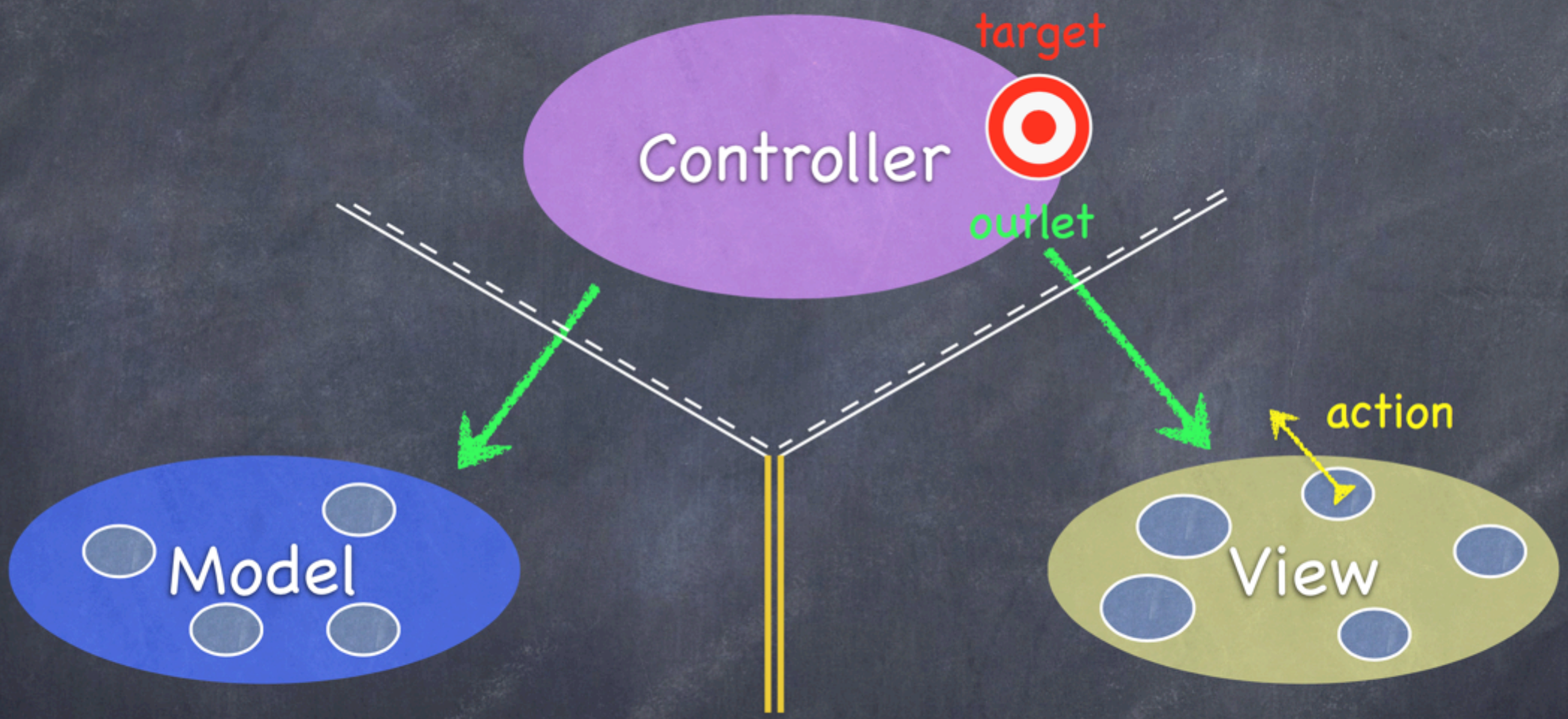
Sort of. Communication is "blind" and structured.

MVC



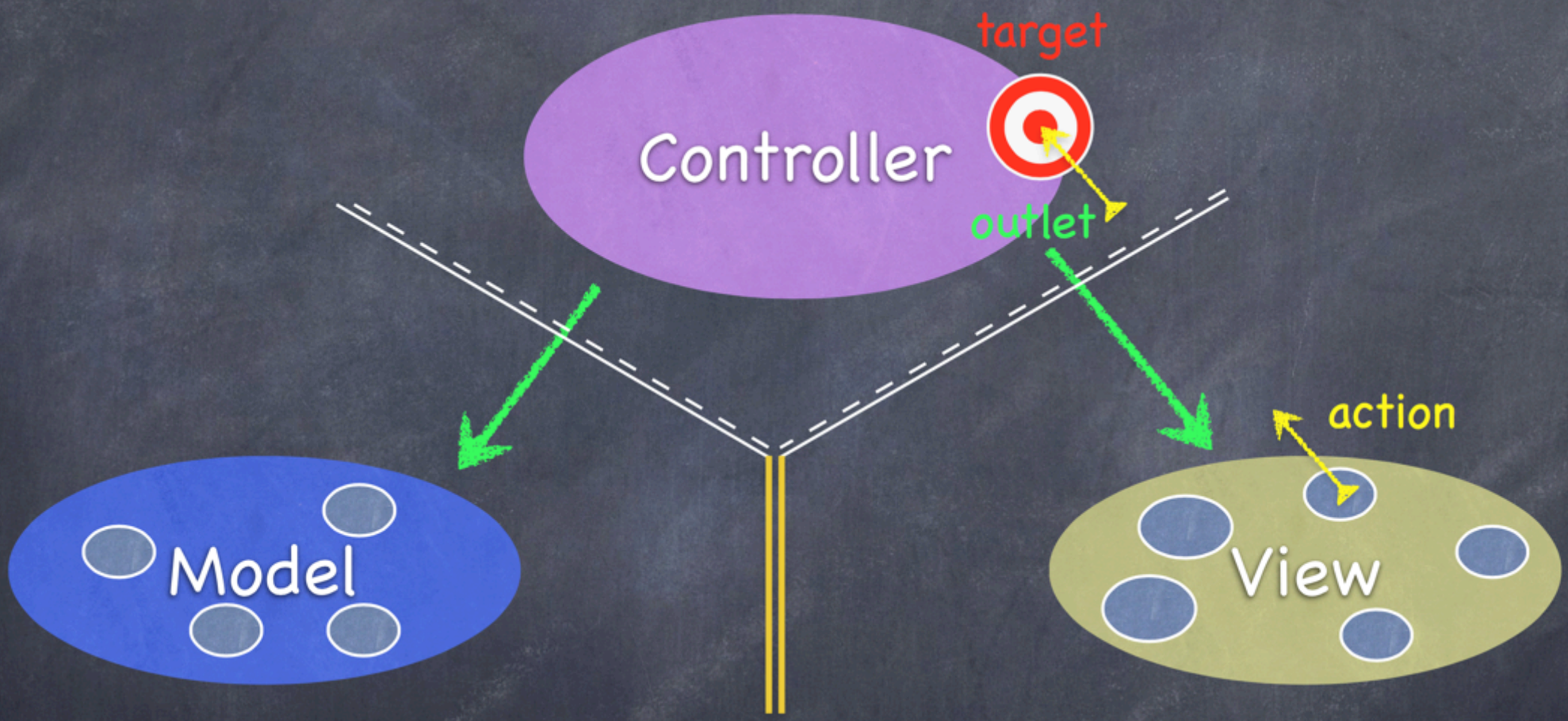
The Controller can drop a target on itself.

MVC



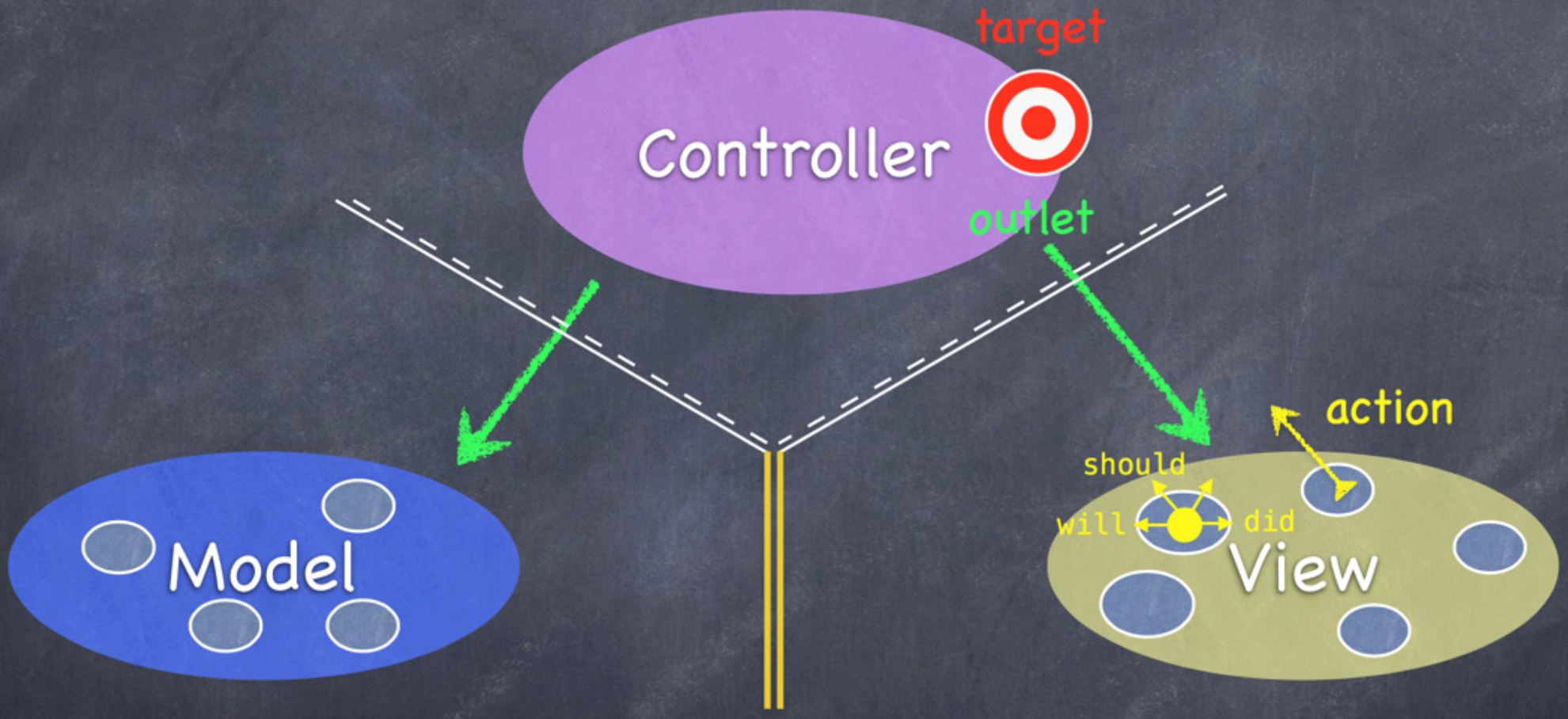
Then hand out an action to the View.

MVC



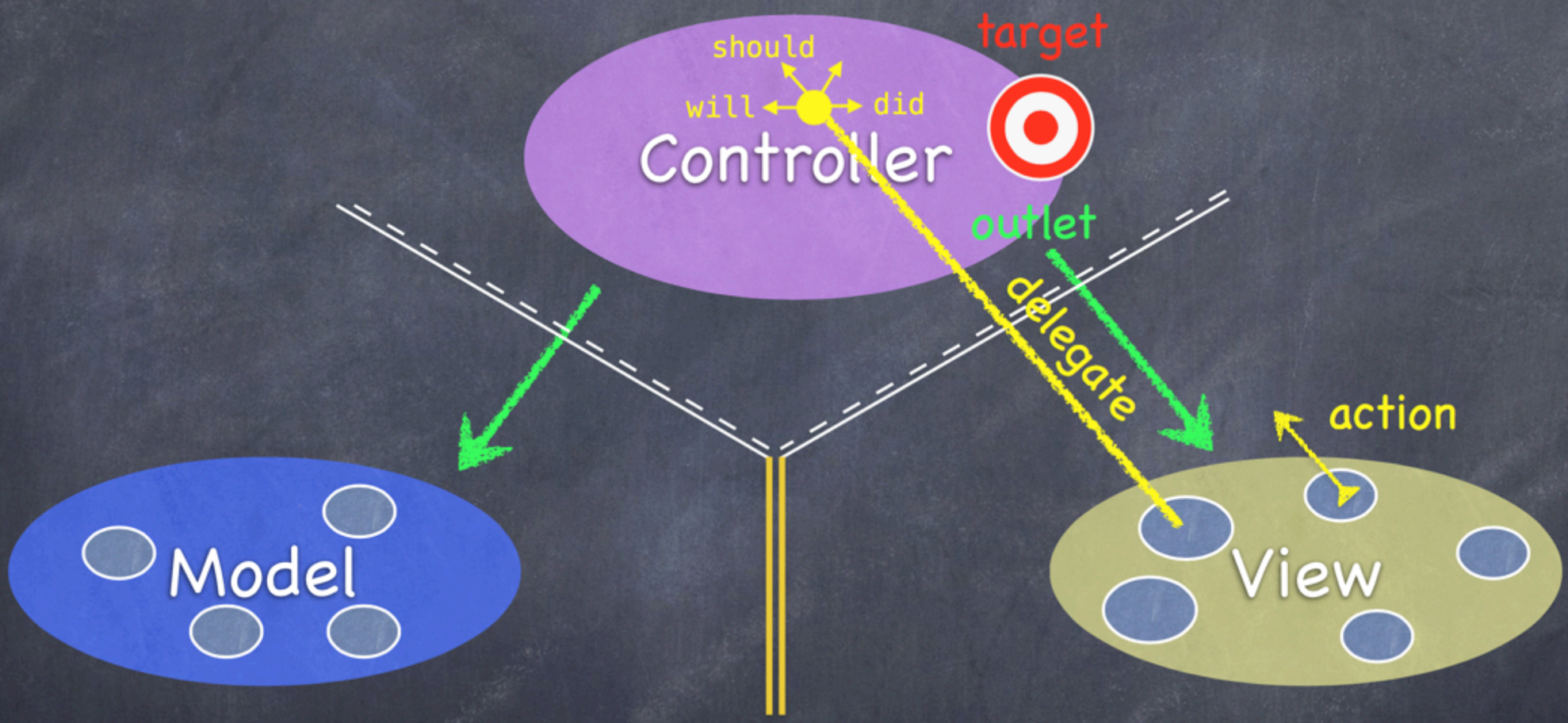
The View sends the action when things happen in the UI.

MVC



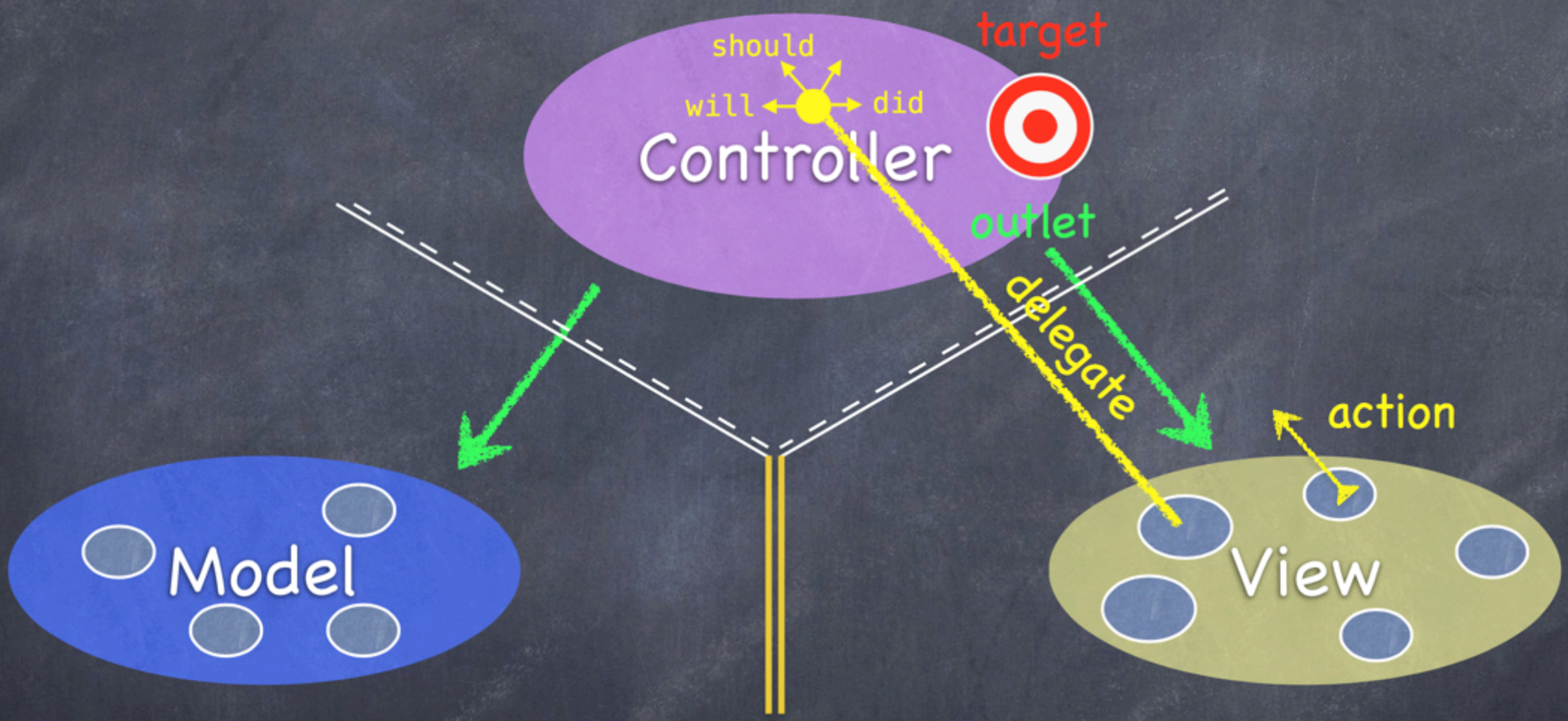
Sometimes the View needs to synchronize with the Controller.

MVC



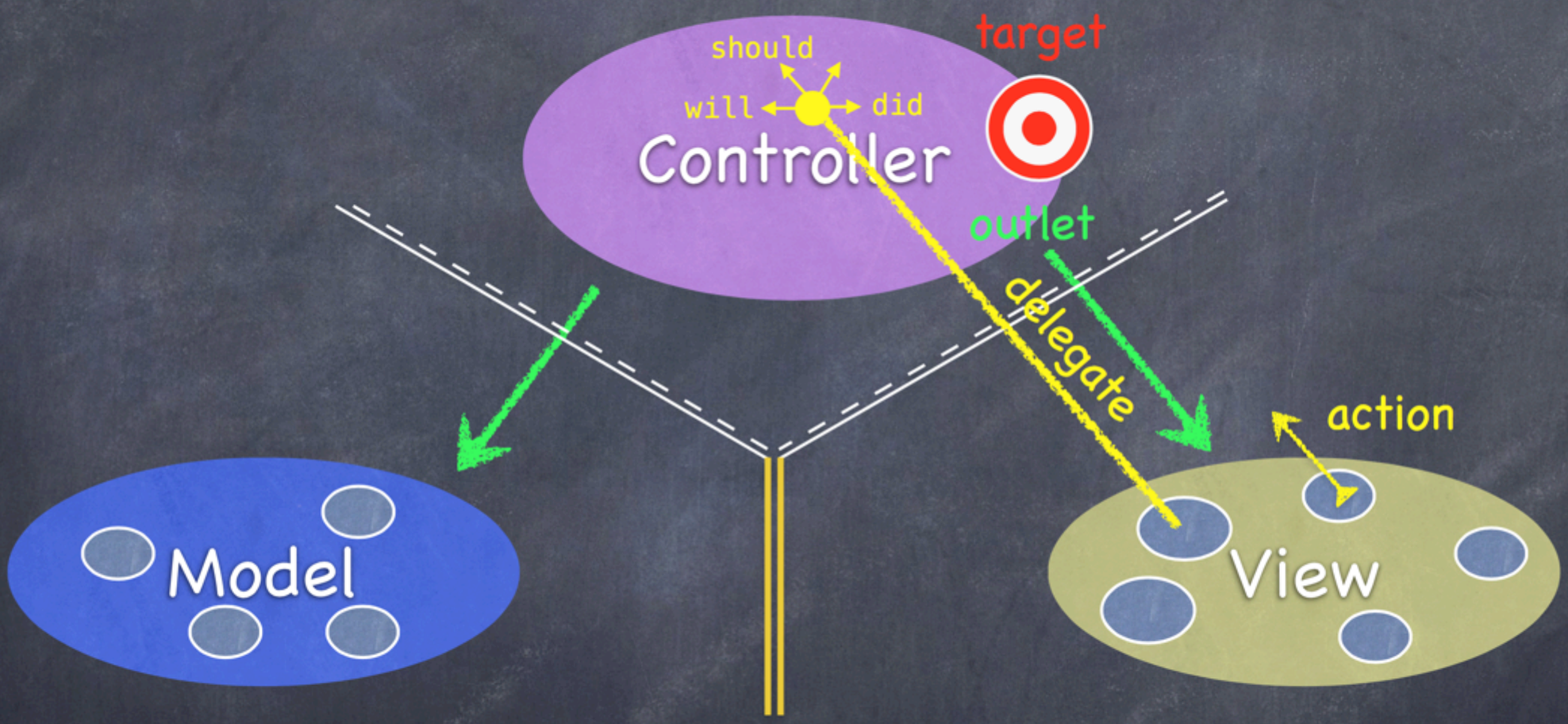
The Controller sets itself as the View's delegate.

MVC



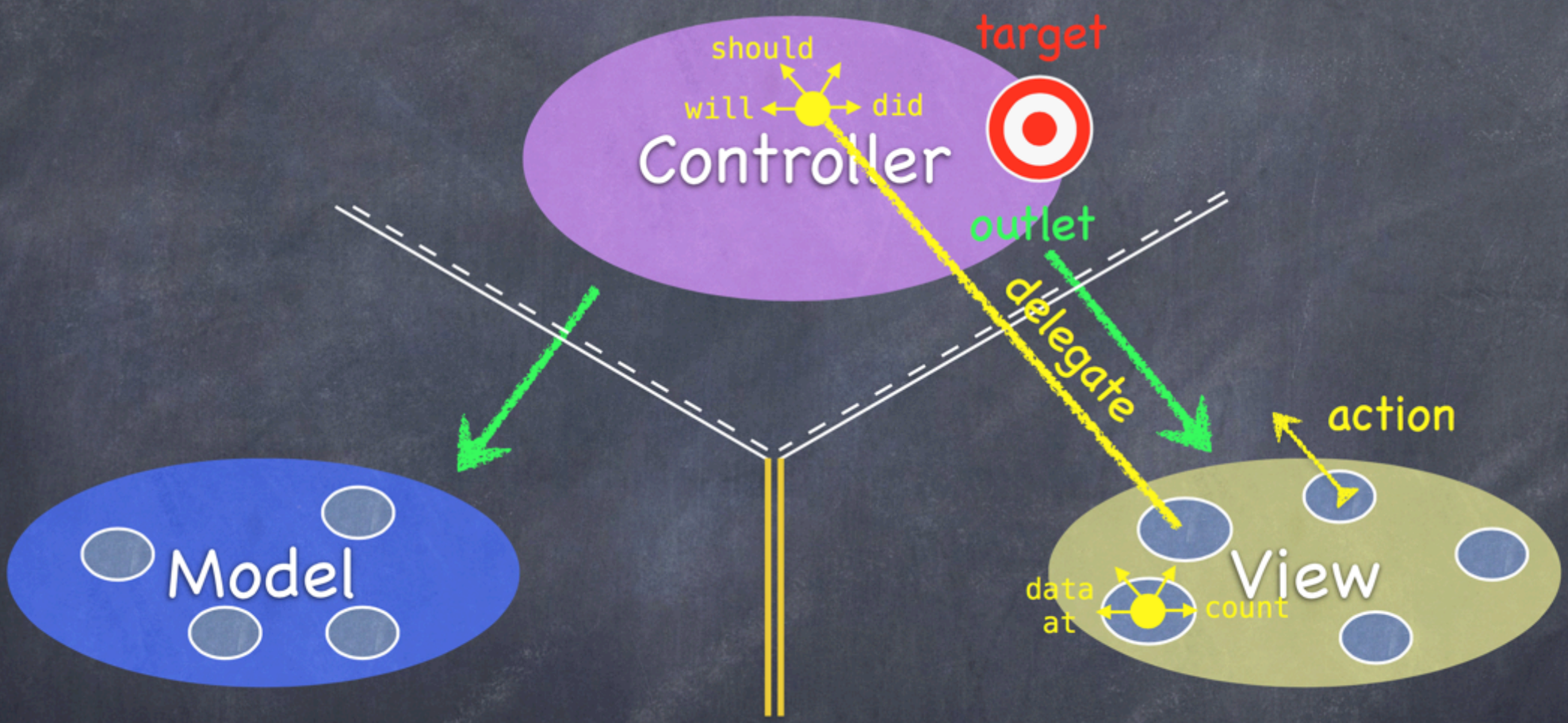
The delegate is set via a protocol (i.e. it's "blind" to class).

MVC



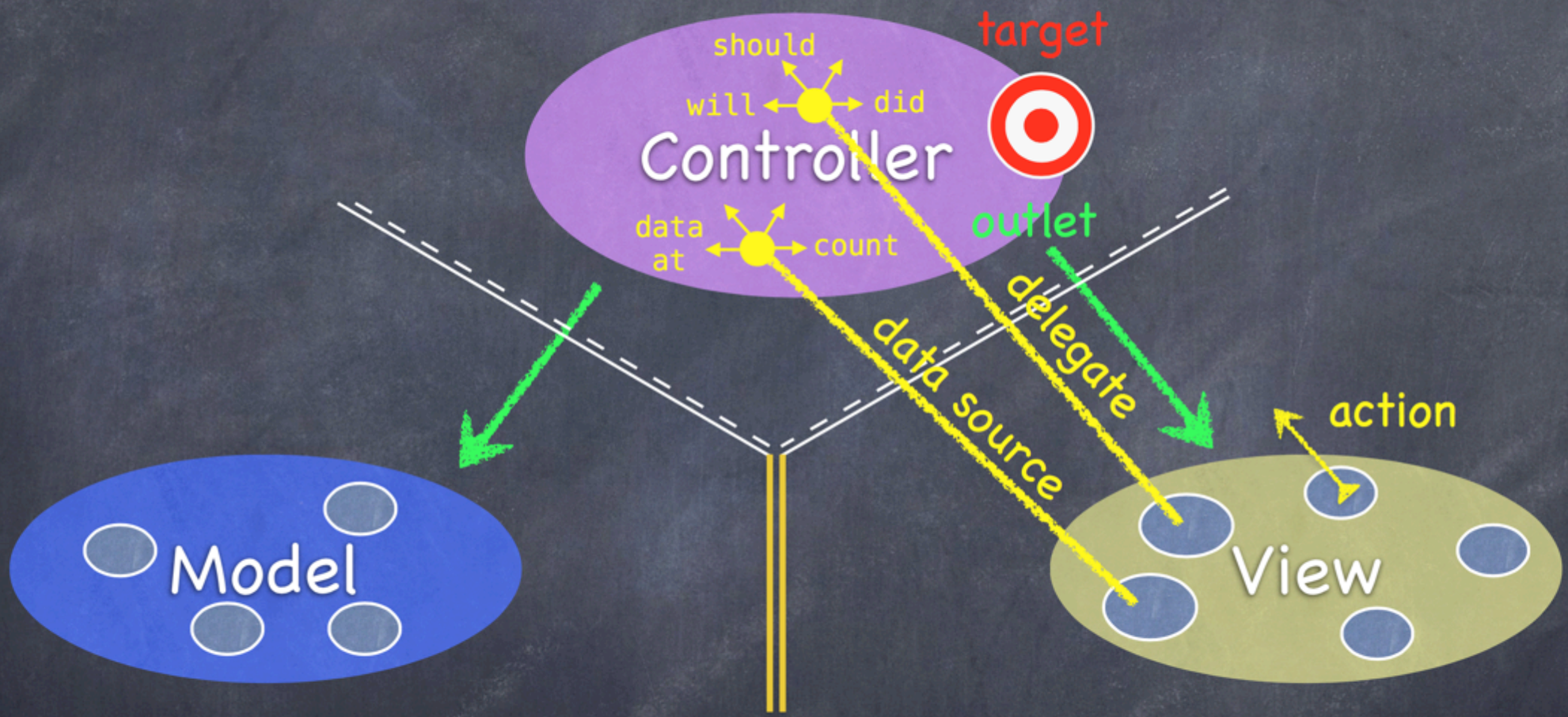
Views do not own the data they display.

MVC



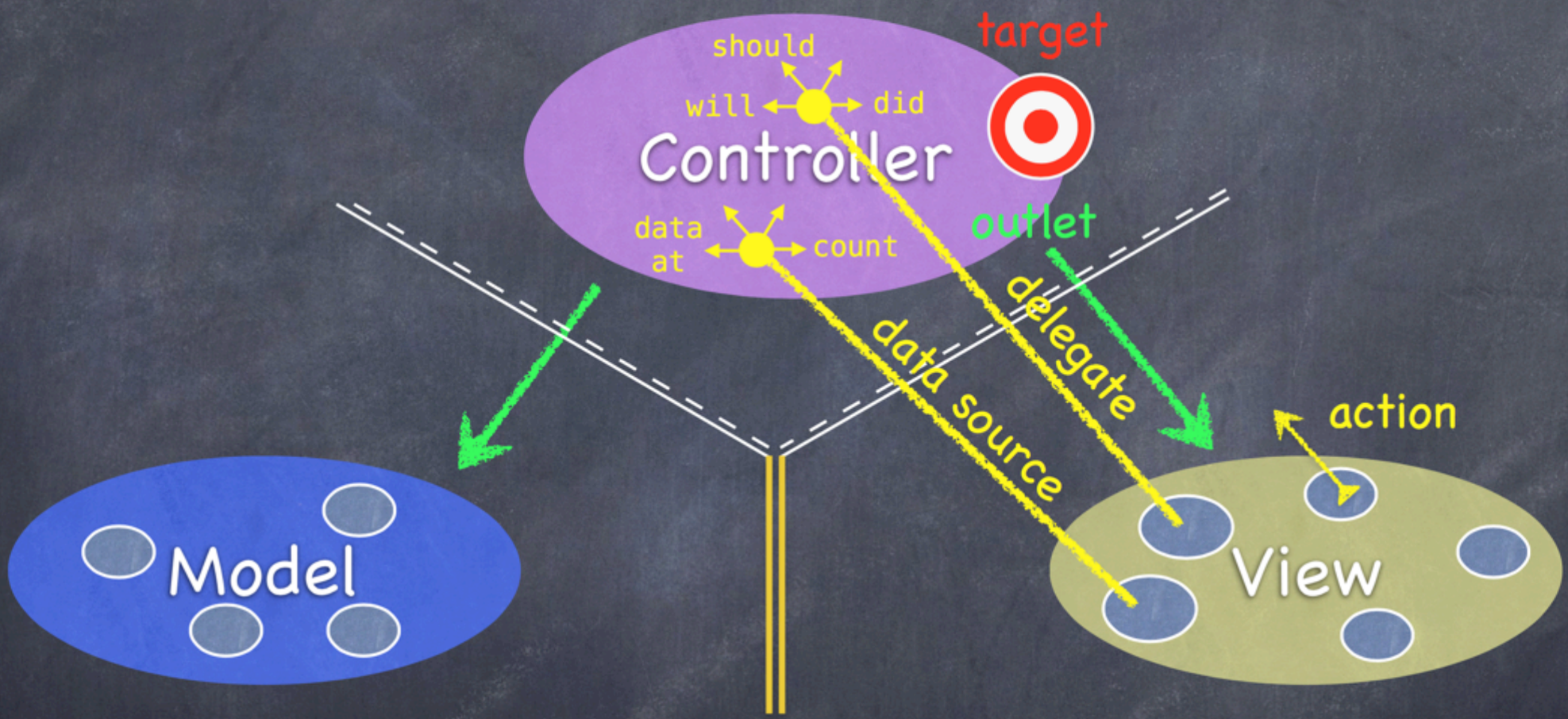
So, if needed, they have a protocol to acquire it.

MVC



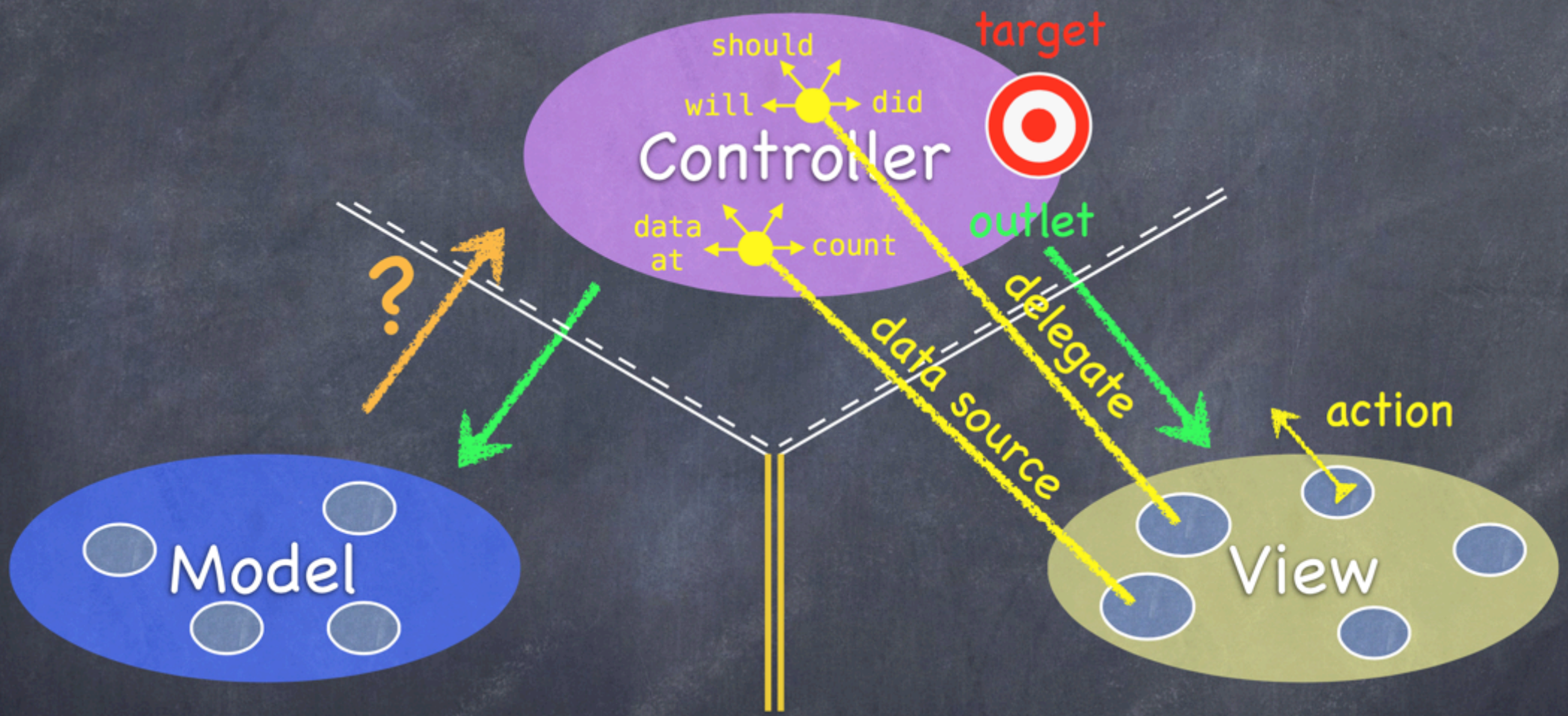
Controllers are almost always that data source (not Model!).

MVC



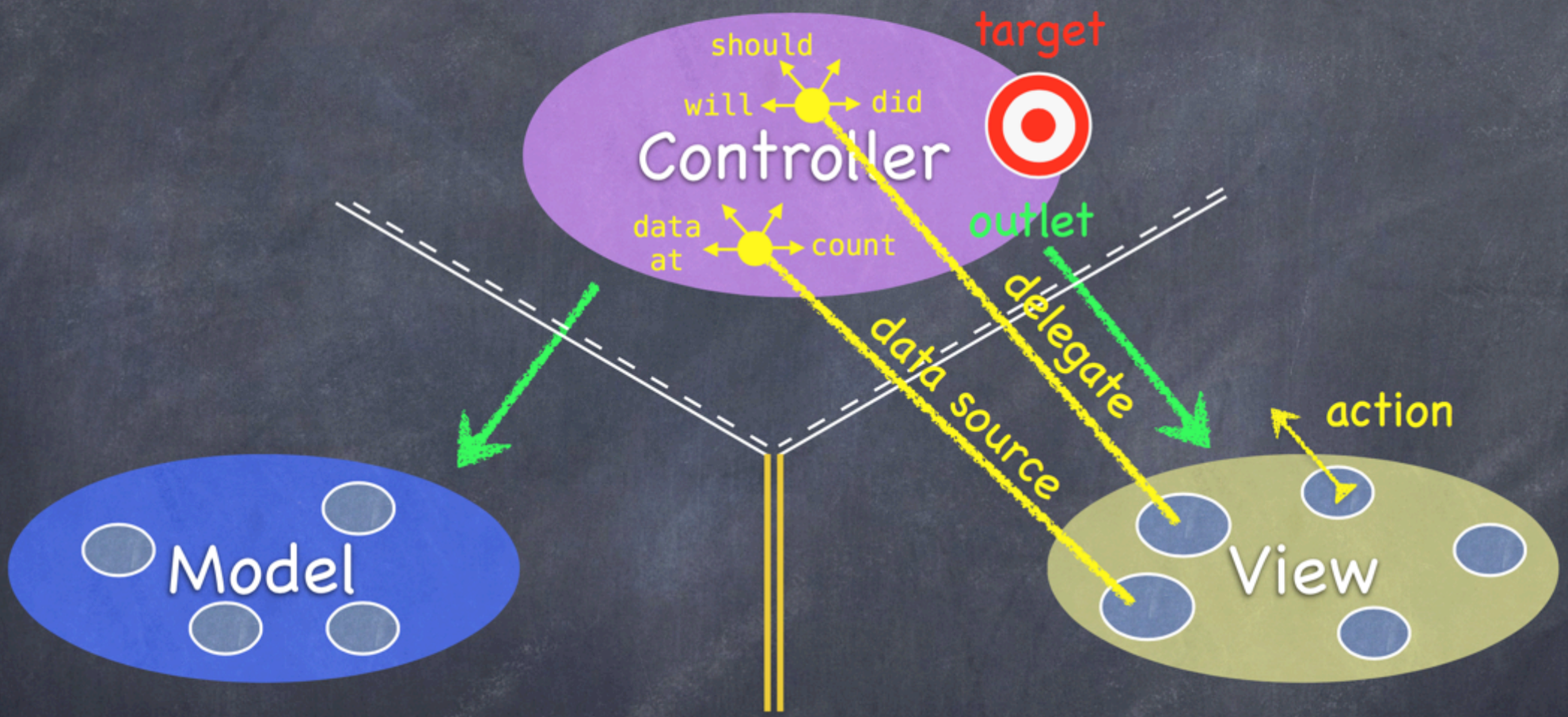
Controllers interpret/format Model information for the View.

MVC



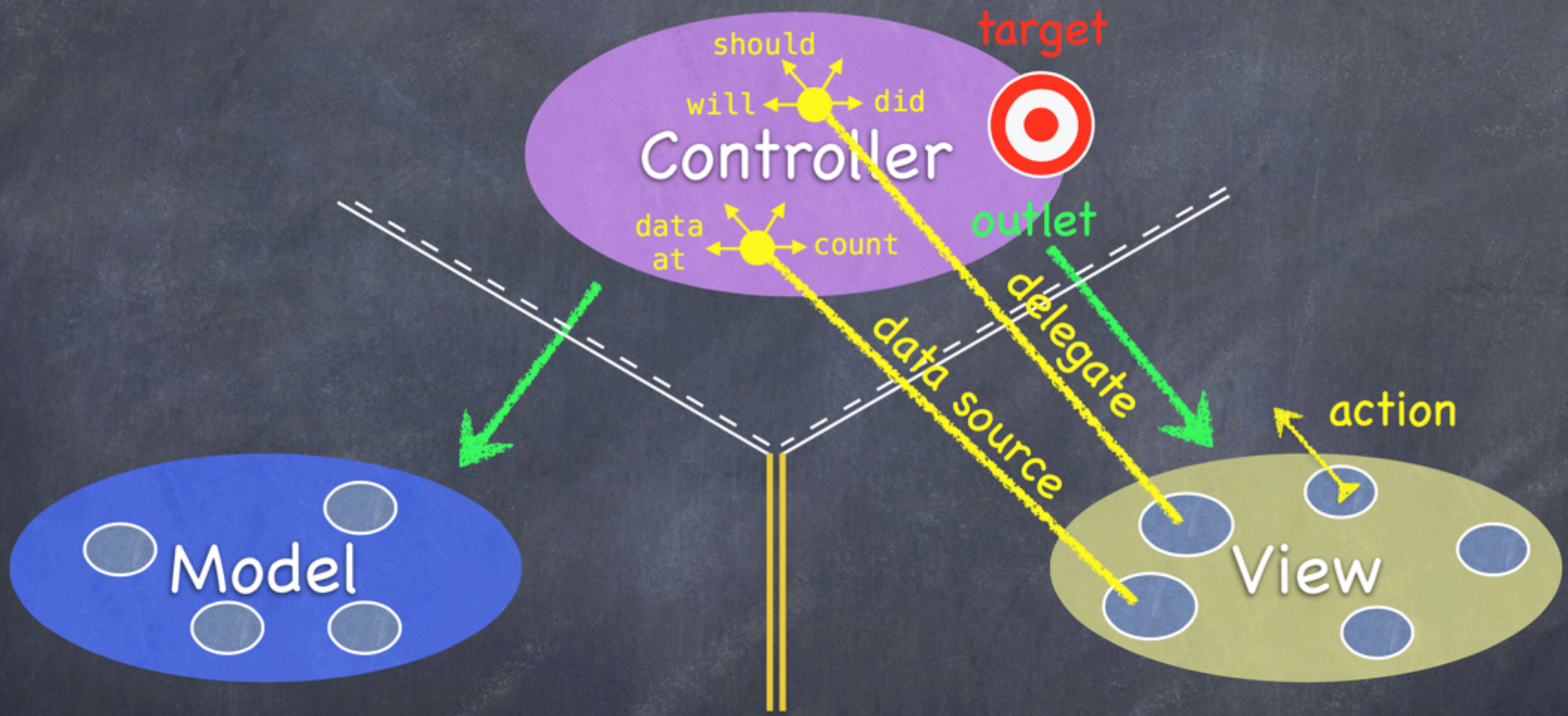
Can the Model talk directly to the Controller?

MVC



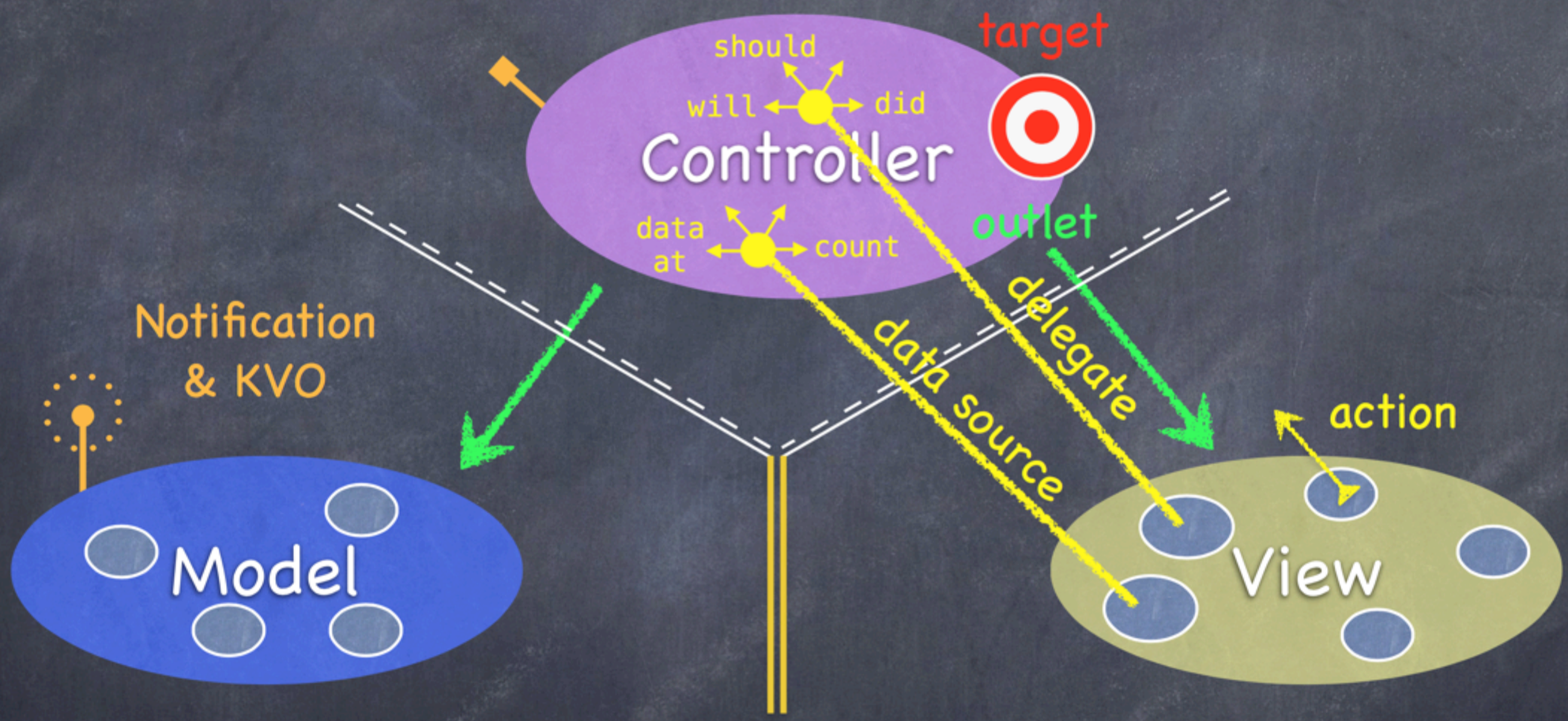
No. The Model is (should be) UI independent.

MVC



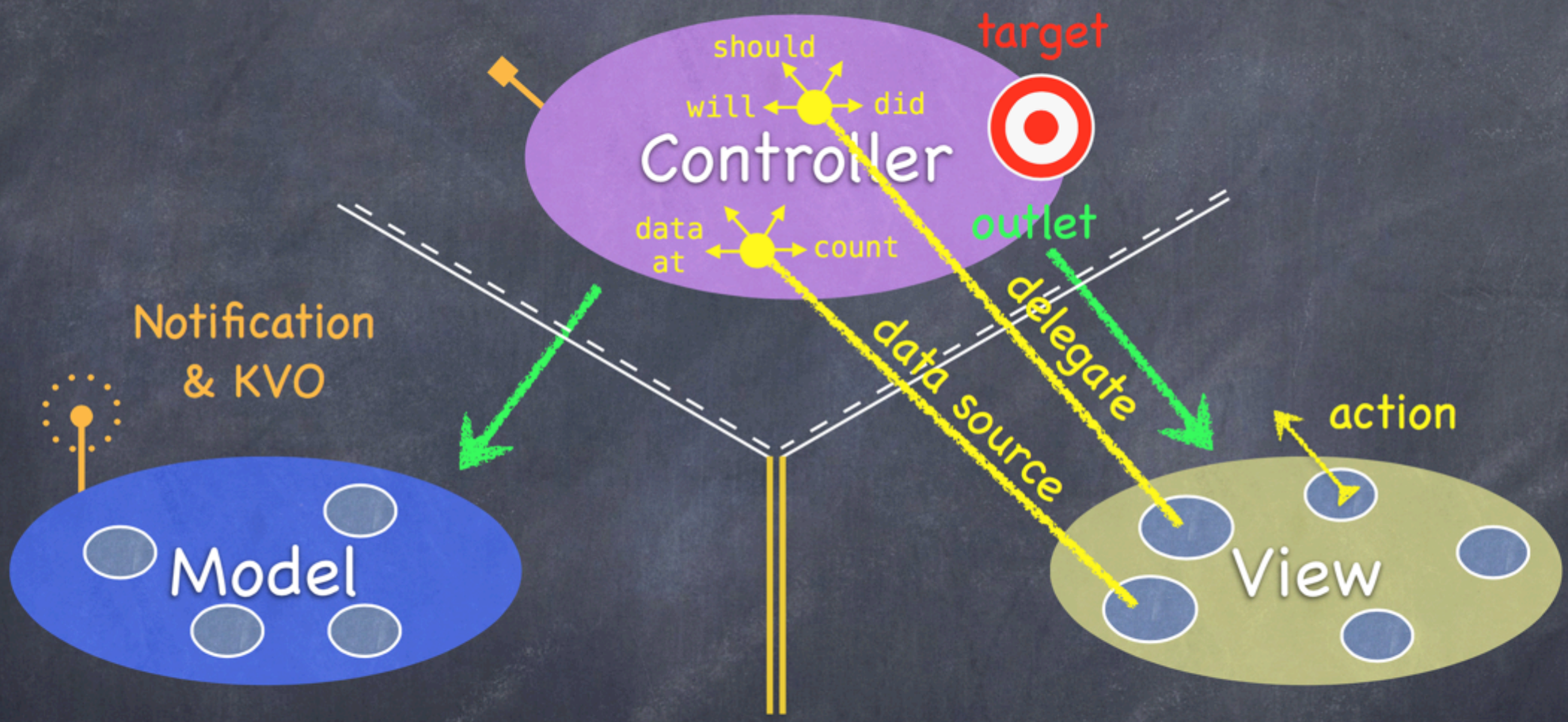
So what if the Model has information to update or something?

MVC



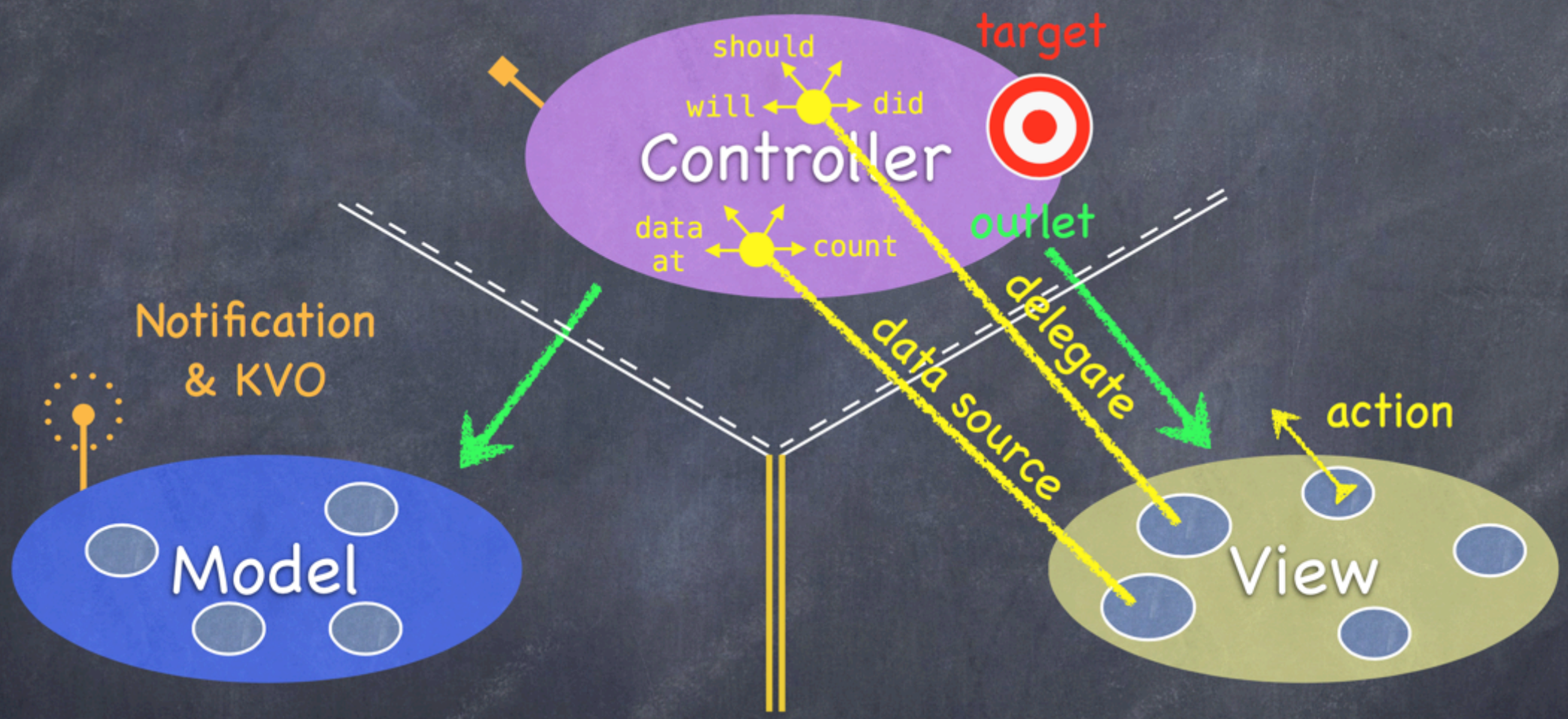
It uses a "radio station"-like broadcast mechanism.

MVC



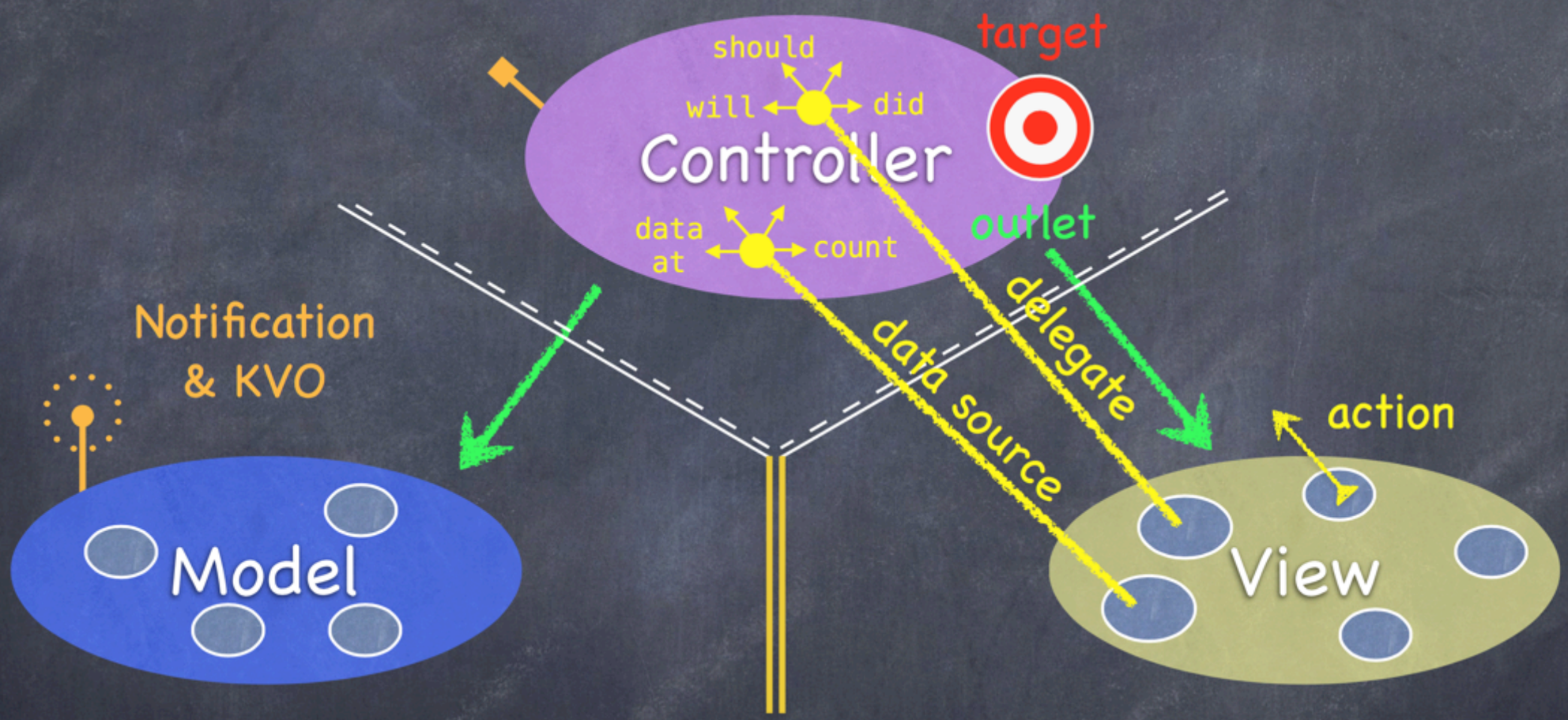
Controllers (or other Model) "tune in" to interesting stuff.

MVC



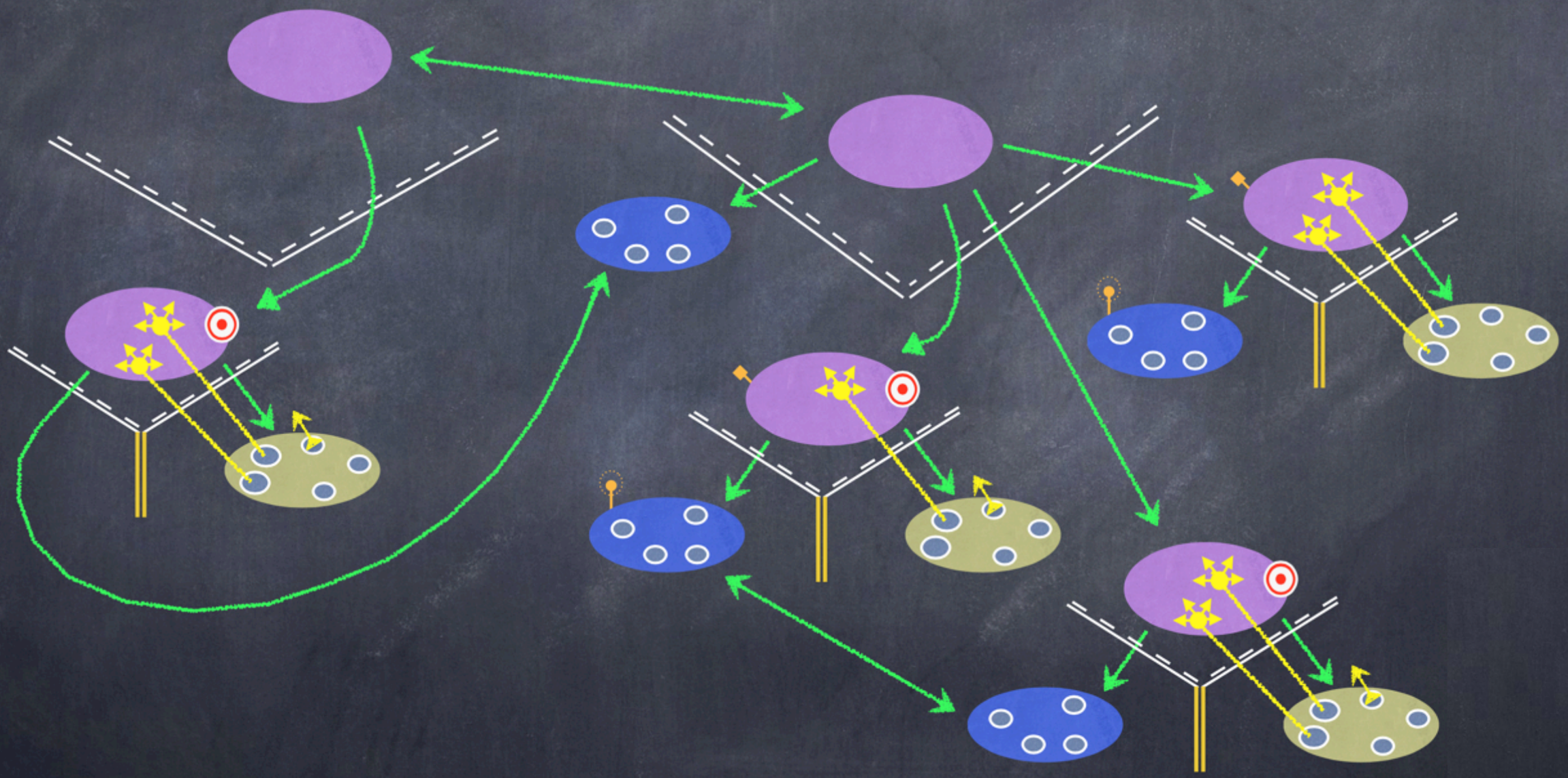
A View might "tune in," but probably not to a Model's "station."

MVC

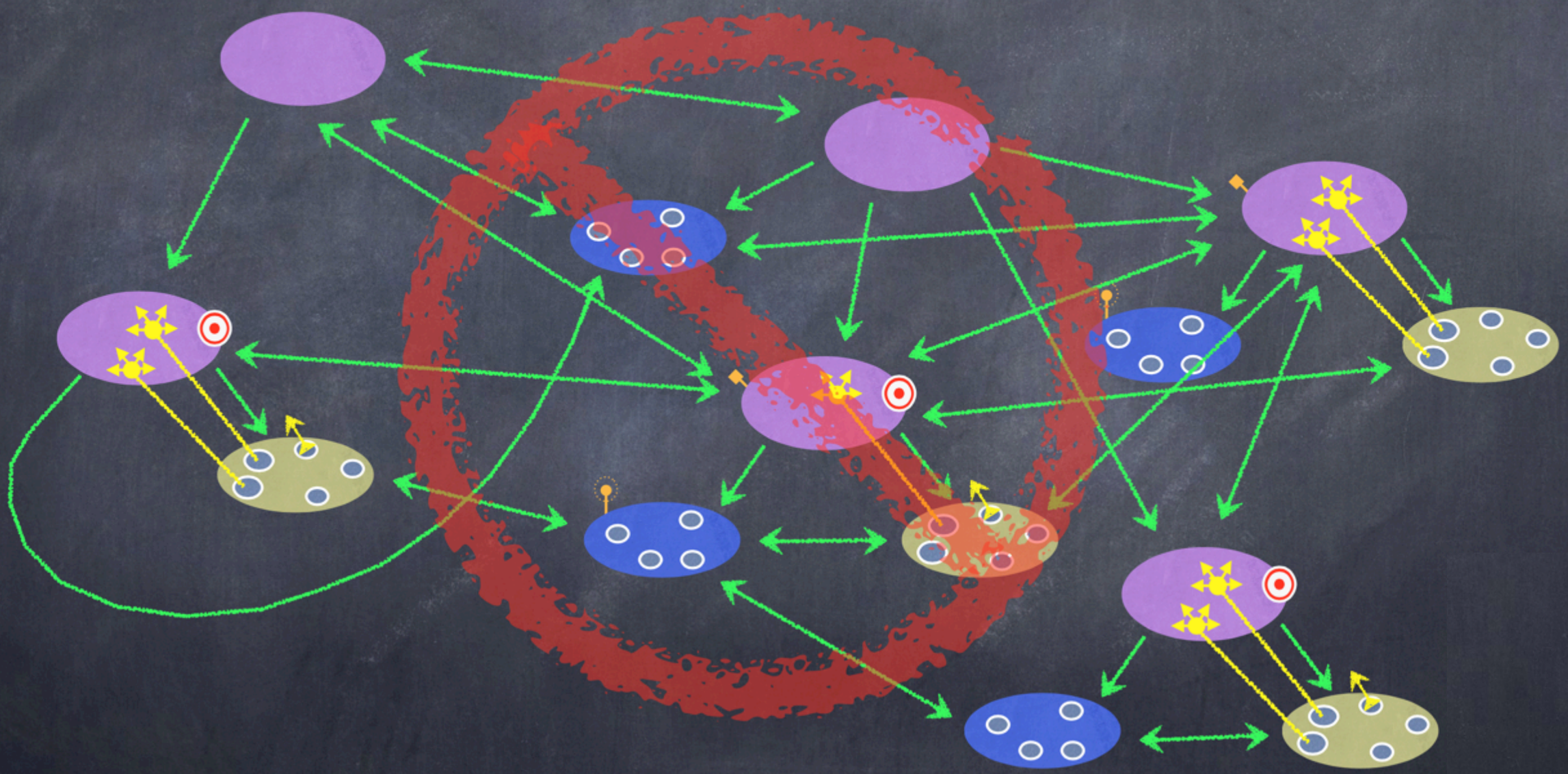


Now combine MVC groups to make complicated programs ...

MVCs working together



MVCs not working together



Demo

- Topics in the demo ...

Learn the basics of the Xcode integrated development environment (IDE), which you'll use to write, test and debug your iOS apps.

Use the Single View Application project template to quickly begin developing a new app.

Create a universal app that can run on iPhones and iPads.

Design an app's UI visually (without programming) using Interface Builder, storyboarding and auto layout.

Display text and an image in a UI.

Support both portrait and landscape orientations.

Edit the attributes of Cocoa Touch UI components.

Build and launch an app in the iOS simulator.

Make the app more accessible to visually impaired people by specifying string descriptions for use with iOS's VoiceOver.

Support internationalization so your app can display strings in different languages based on the user's device settings.