



Stanford CS193p

Developing Applications for iOS
Winter 2017



Today

- MVC

Object-Oriented Design Pattern

- Continuation of Calculator Demo

Computed Properties, MVC, Laying out the UI to work with different devices



MVC



Controller



Model



View

Divide objects in your program into 3 "camps."



MVC

Controller

Model

View

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



MVC



Controller



Model



View

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



MVC

Controller

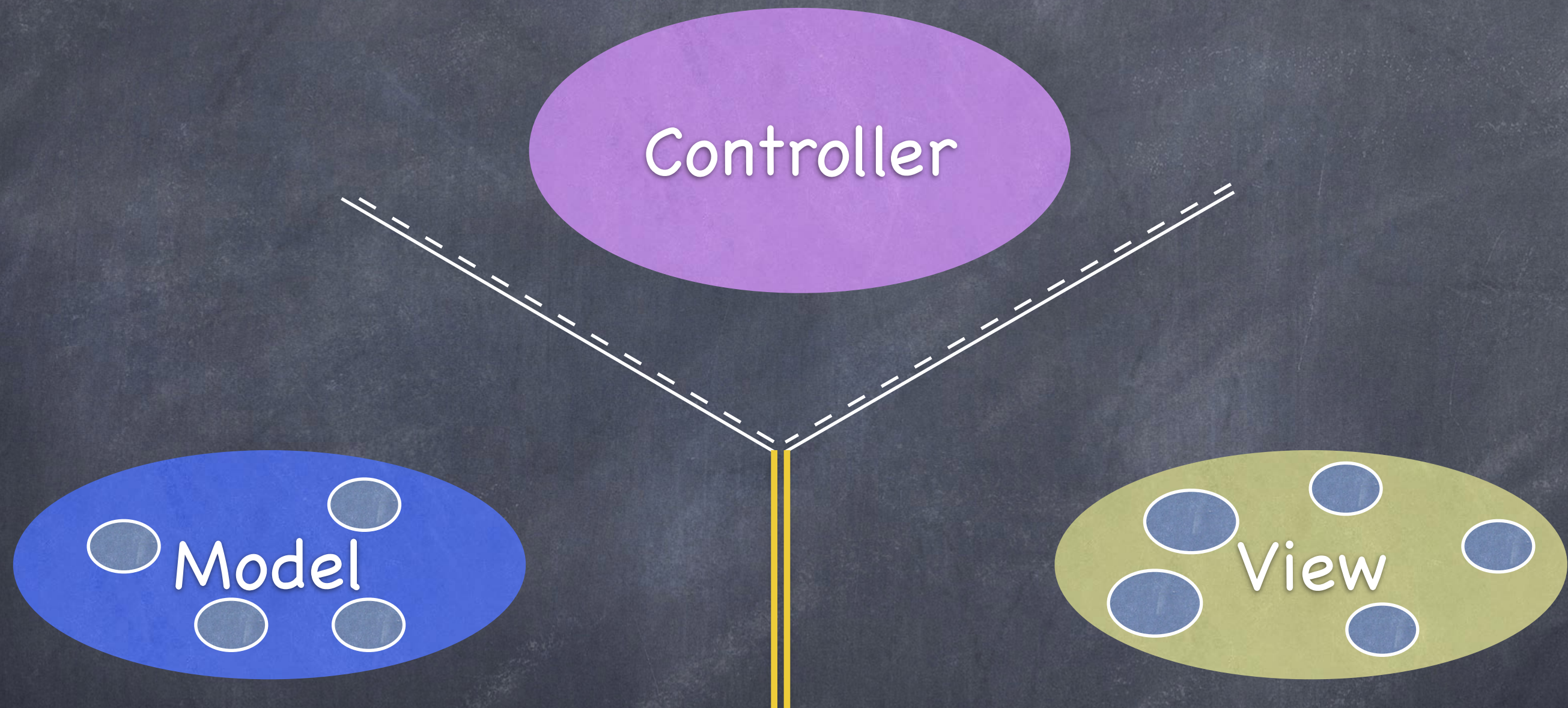
Model

View

View = Your Controller's minions



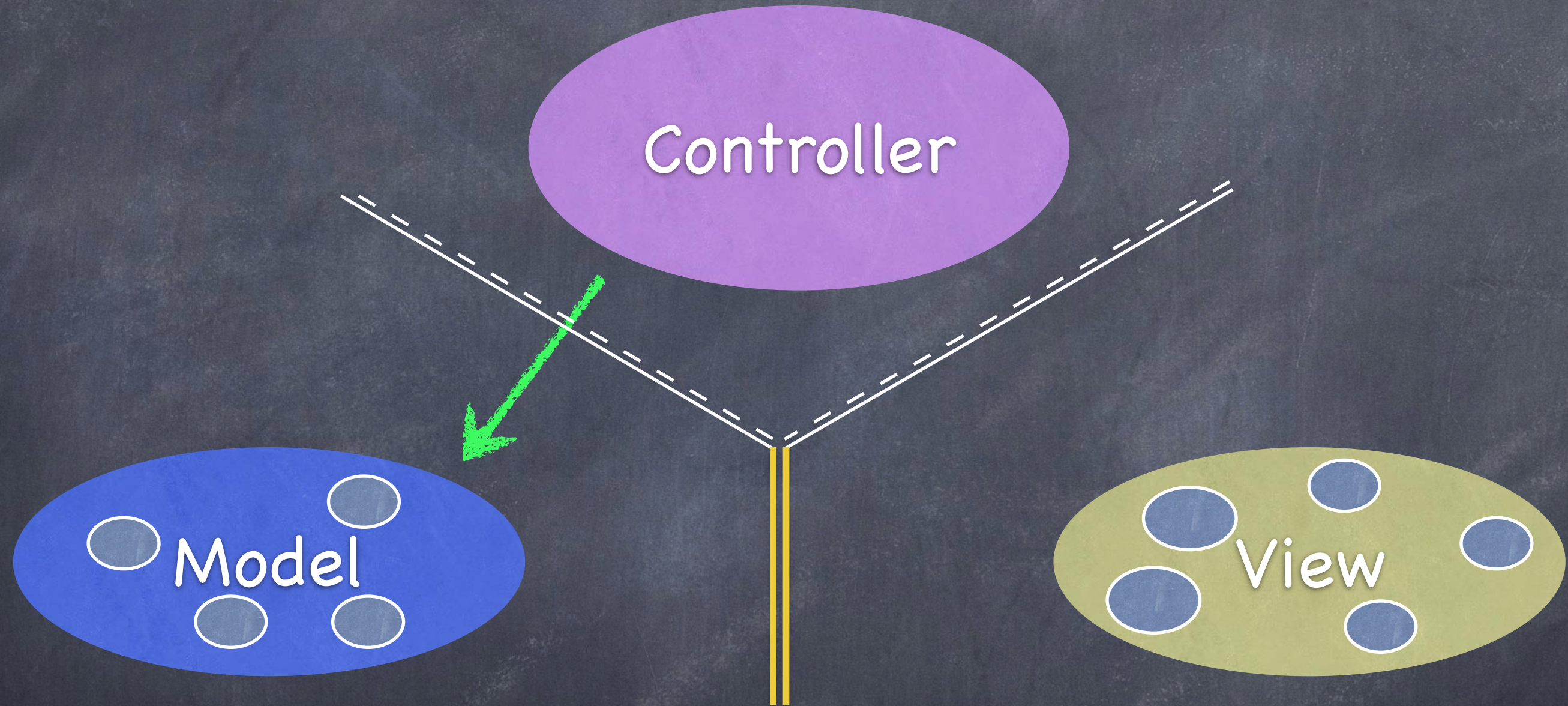
MVC



It's all about managing communication between camps



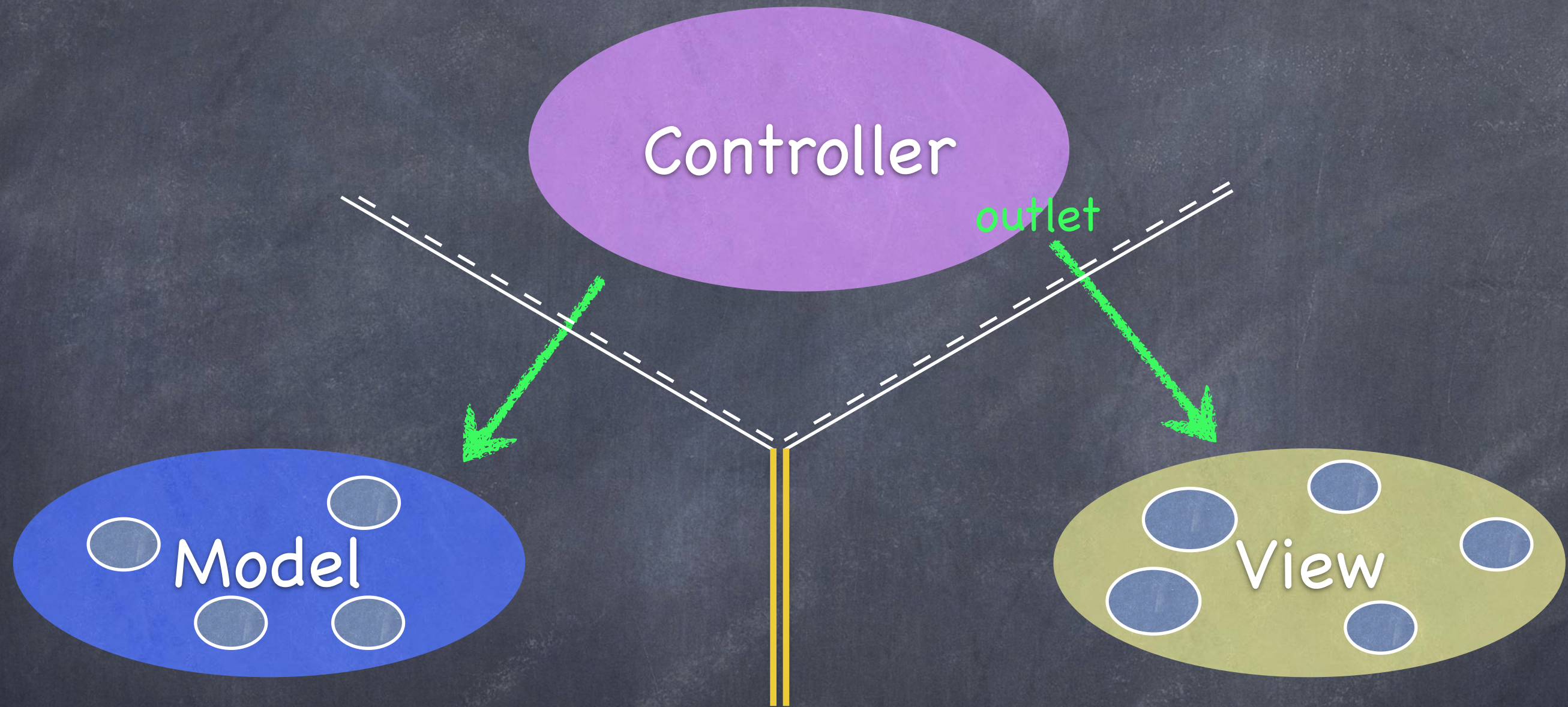
MVC



Controllers can always talk directly to their Model.



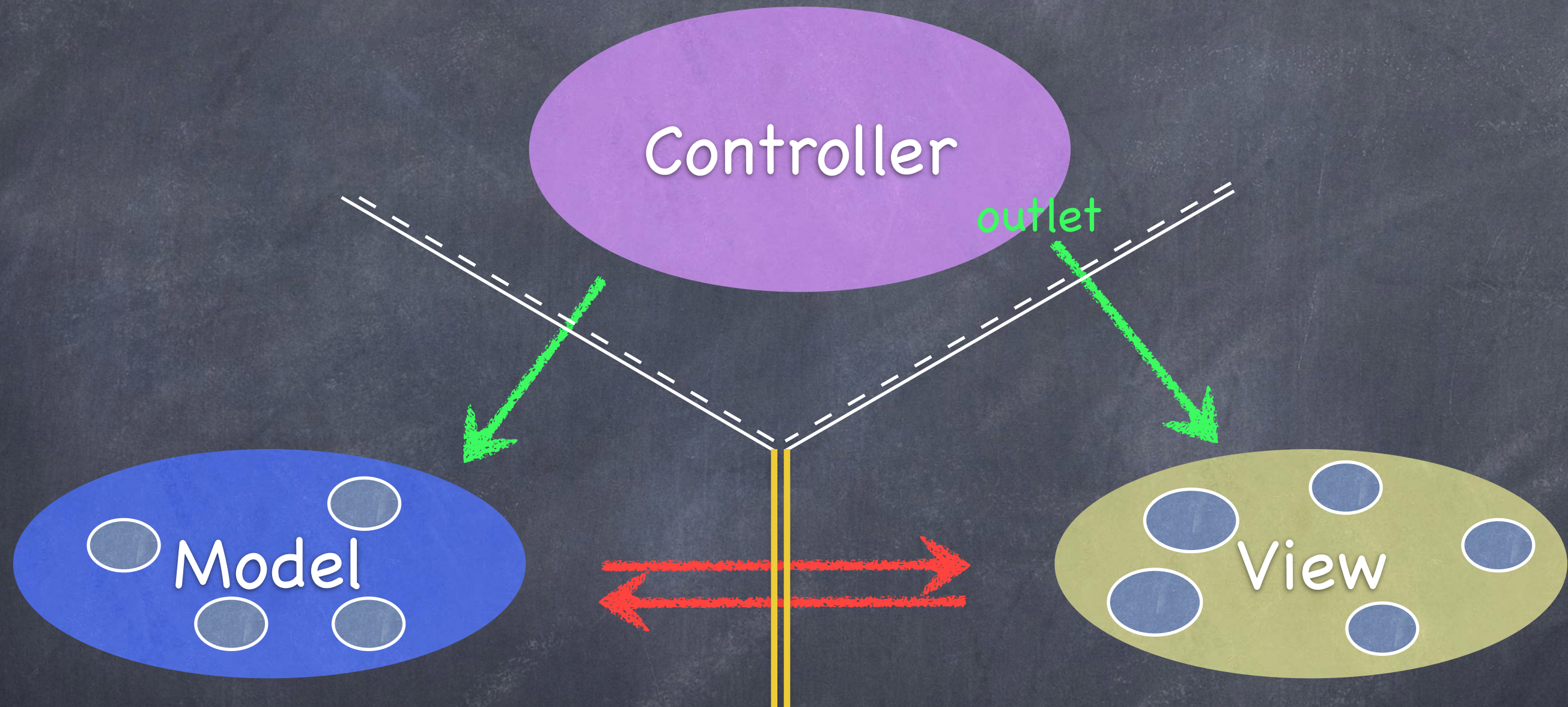
MVC



Controllers can also talk directly to their **View**.



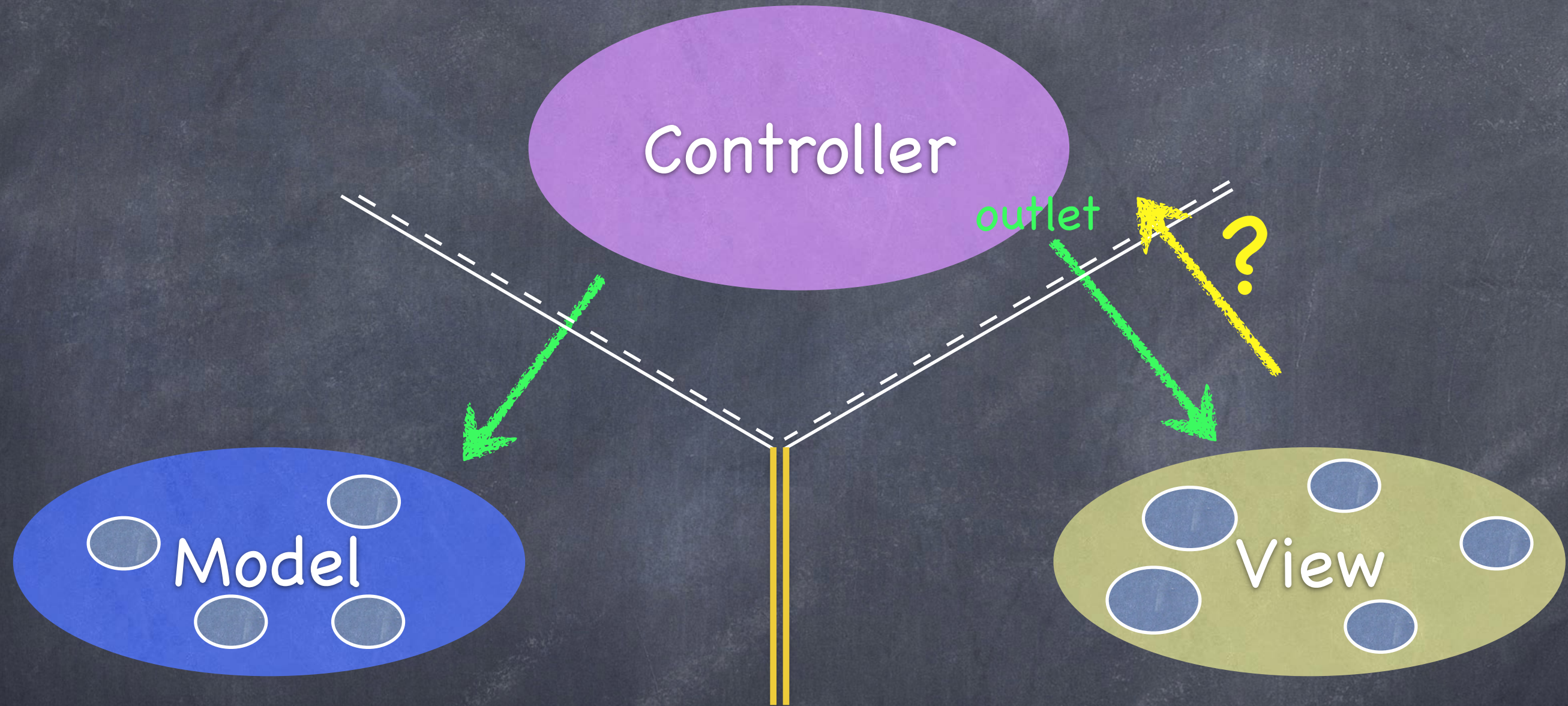
MVC



The **Model** and **View** should never speak to each other.



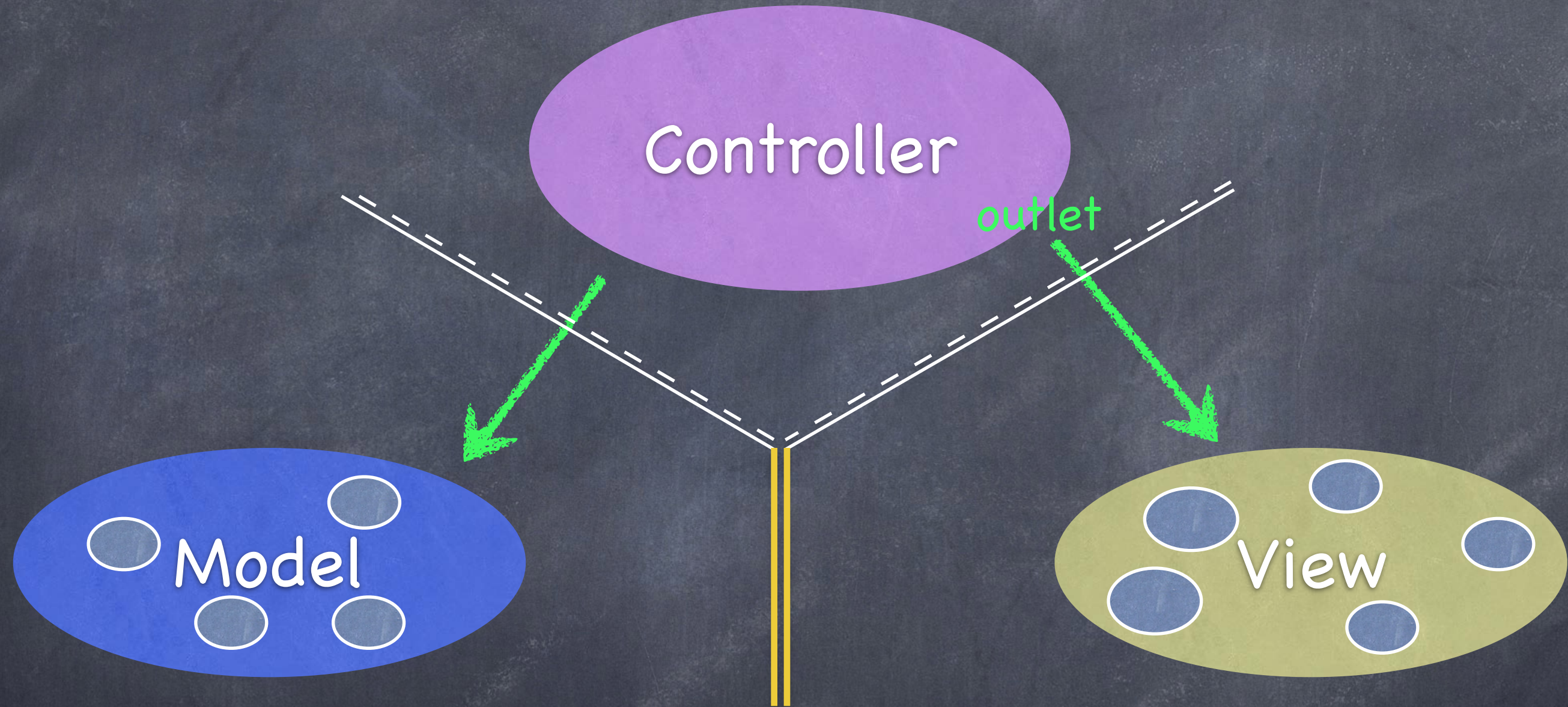
MVC



Can the **View** speak to its **Controller**?



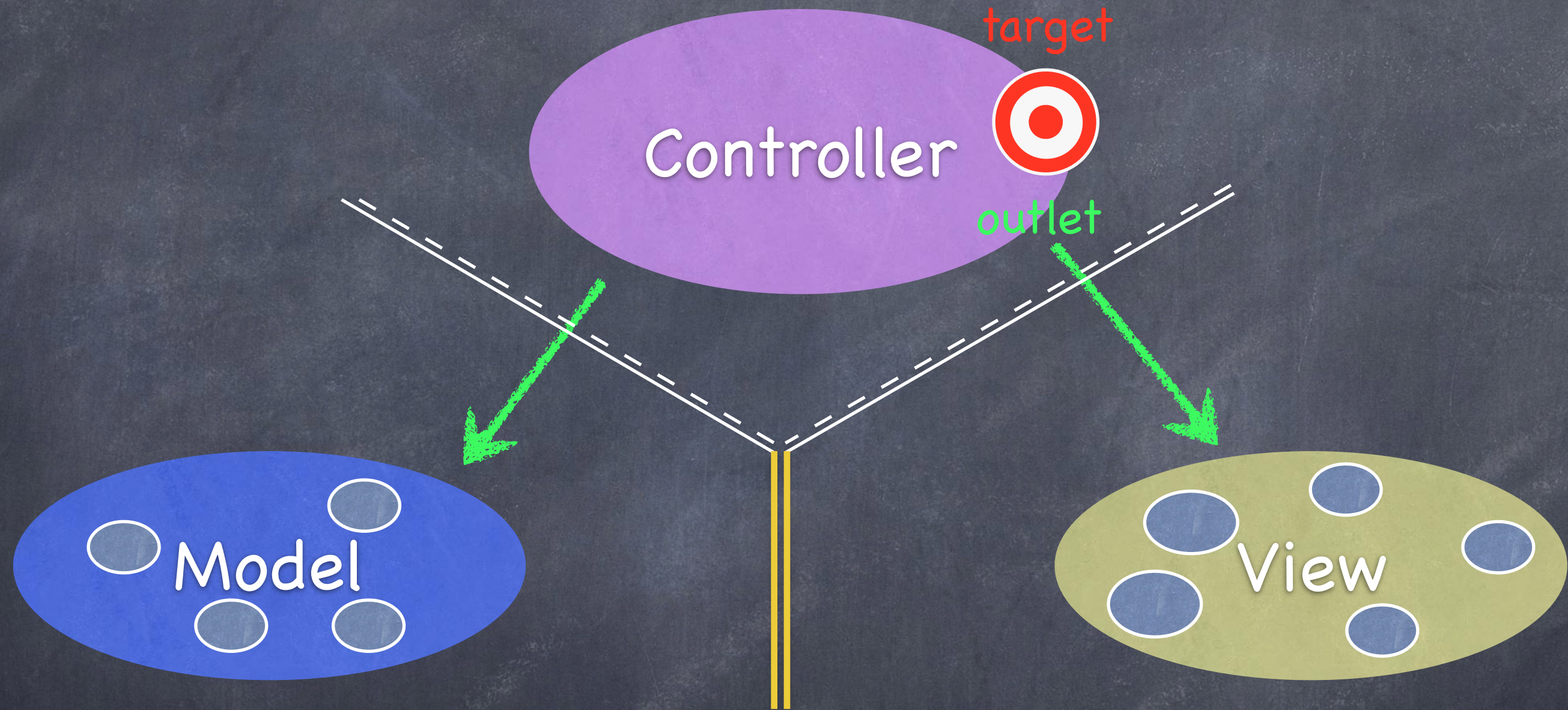
MVC



Sort of. Communication is "blind" and structured.



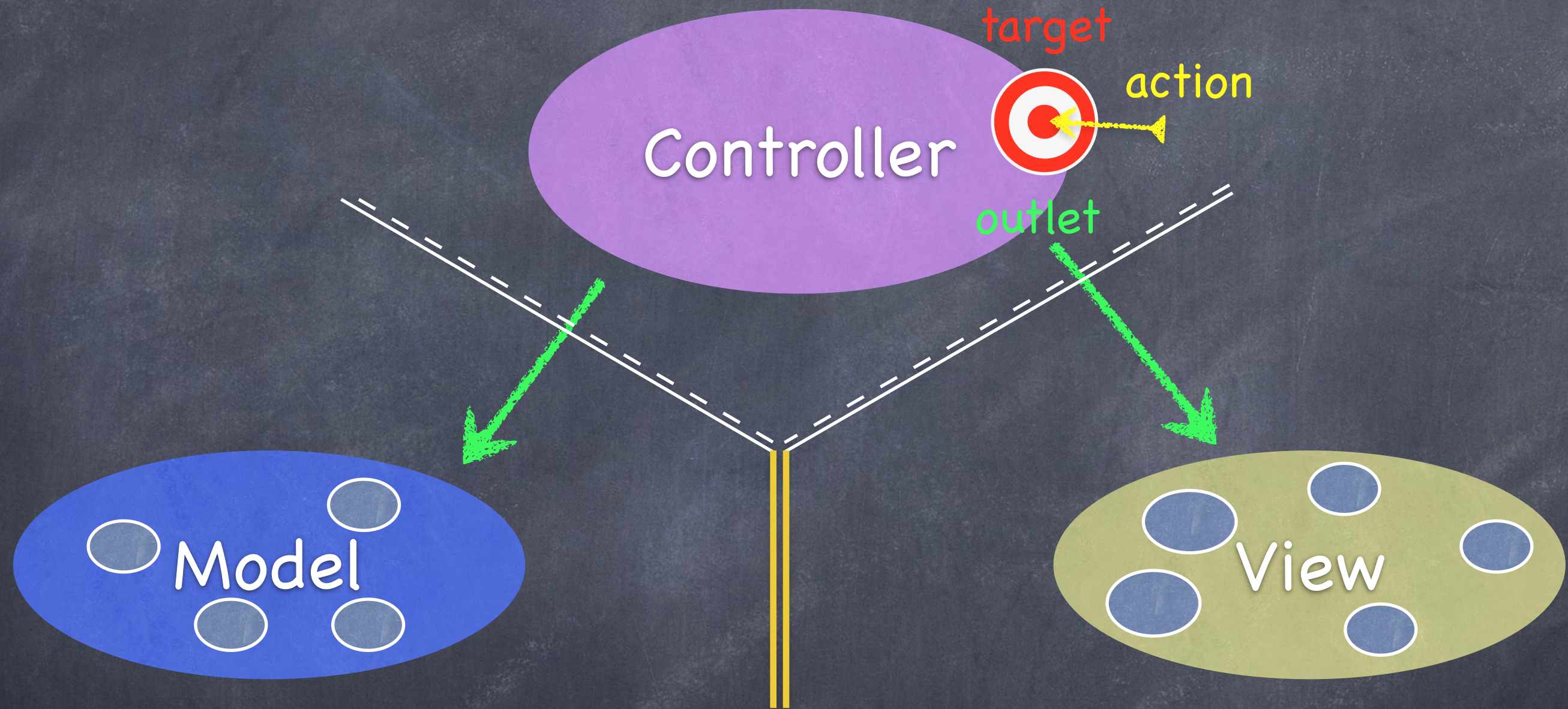
MVC



The Controller can drop a target on itself.



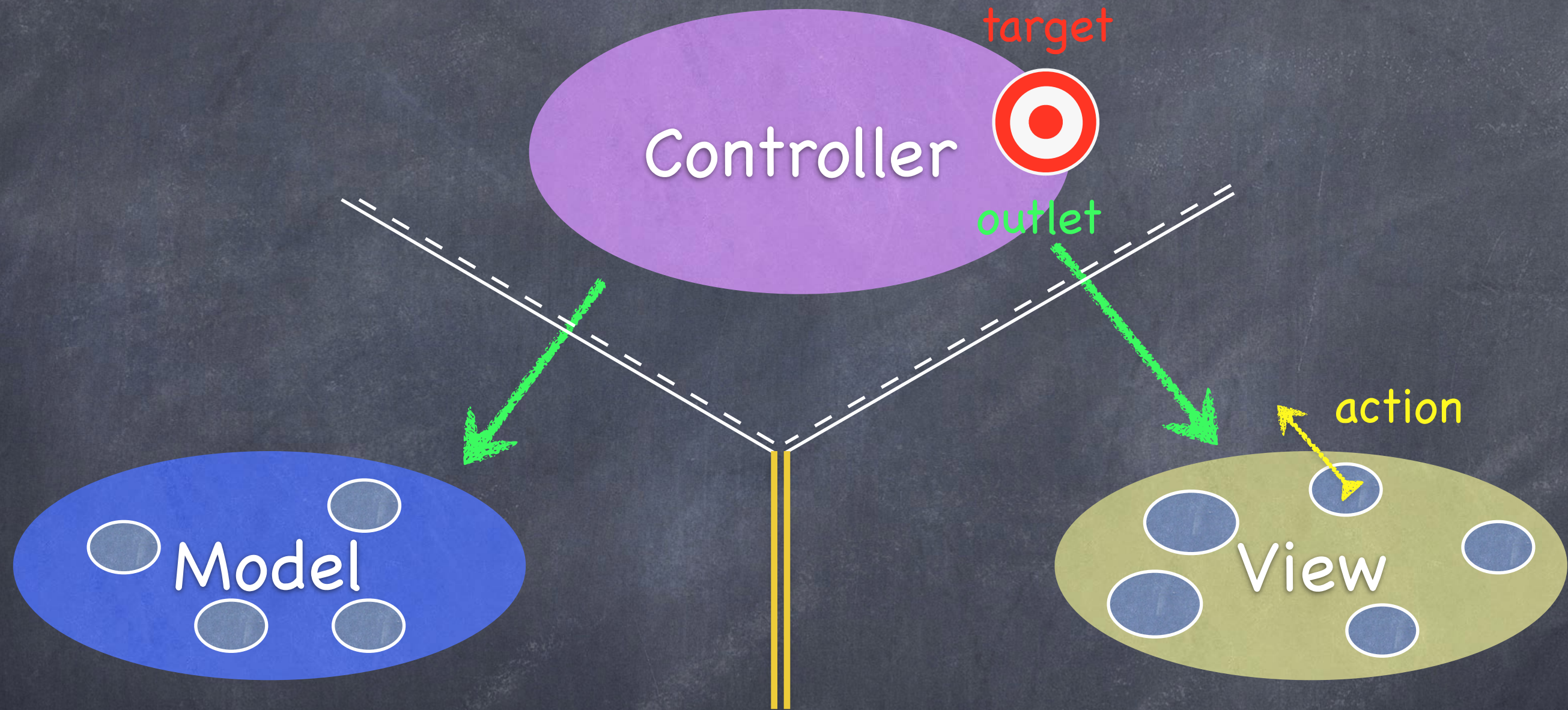
MVC



Then hand out an **action** to the **View**.



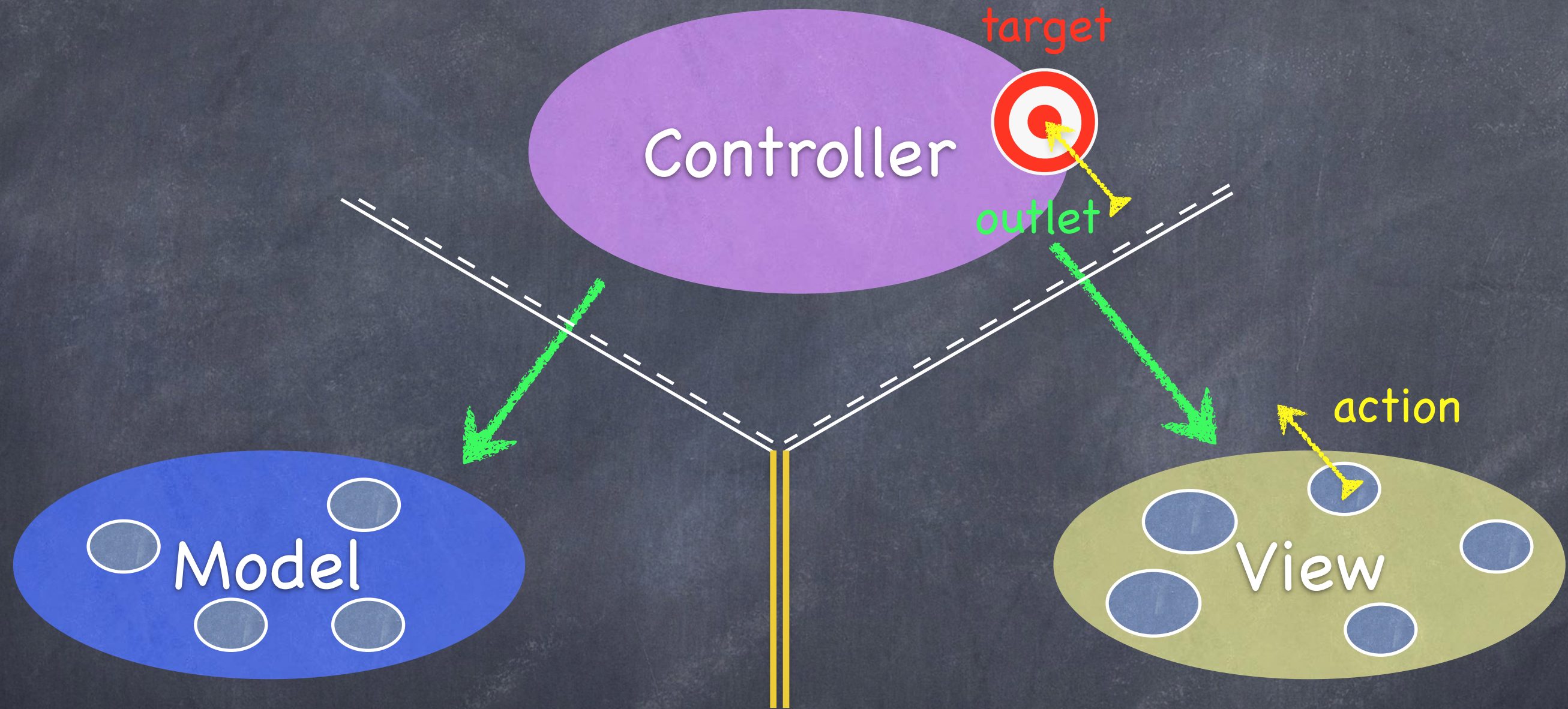
MVC



Then hand out an **action** to the **View**.



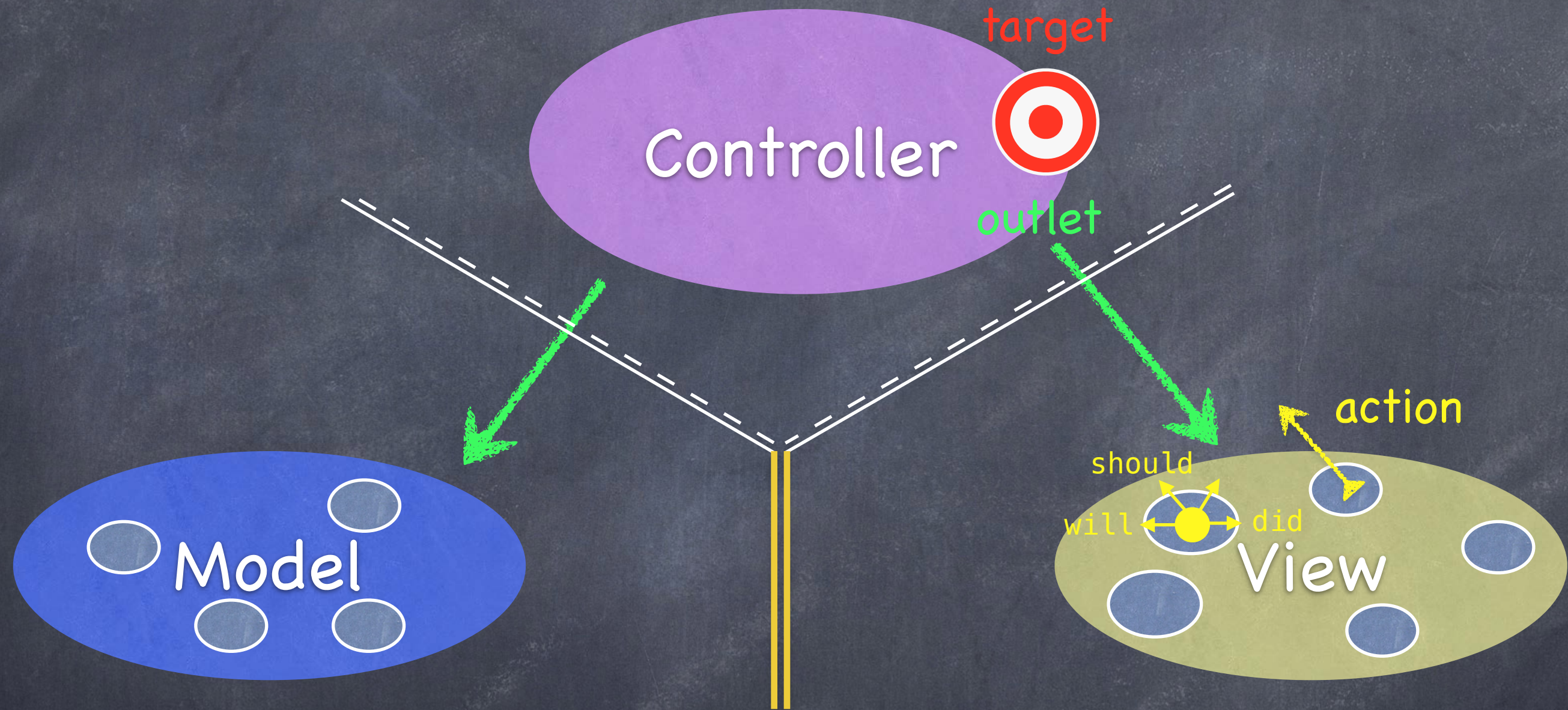
MVC



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



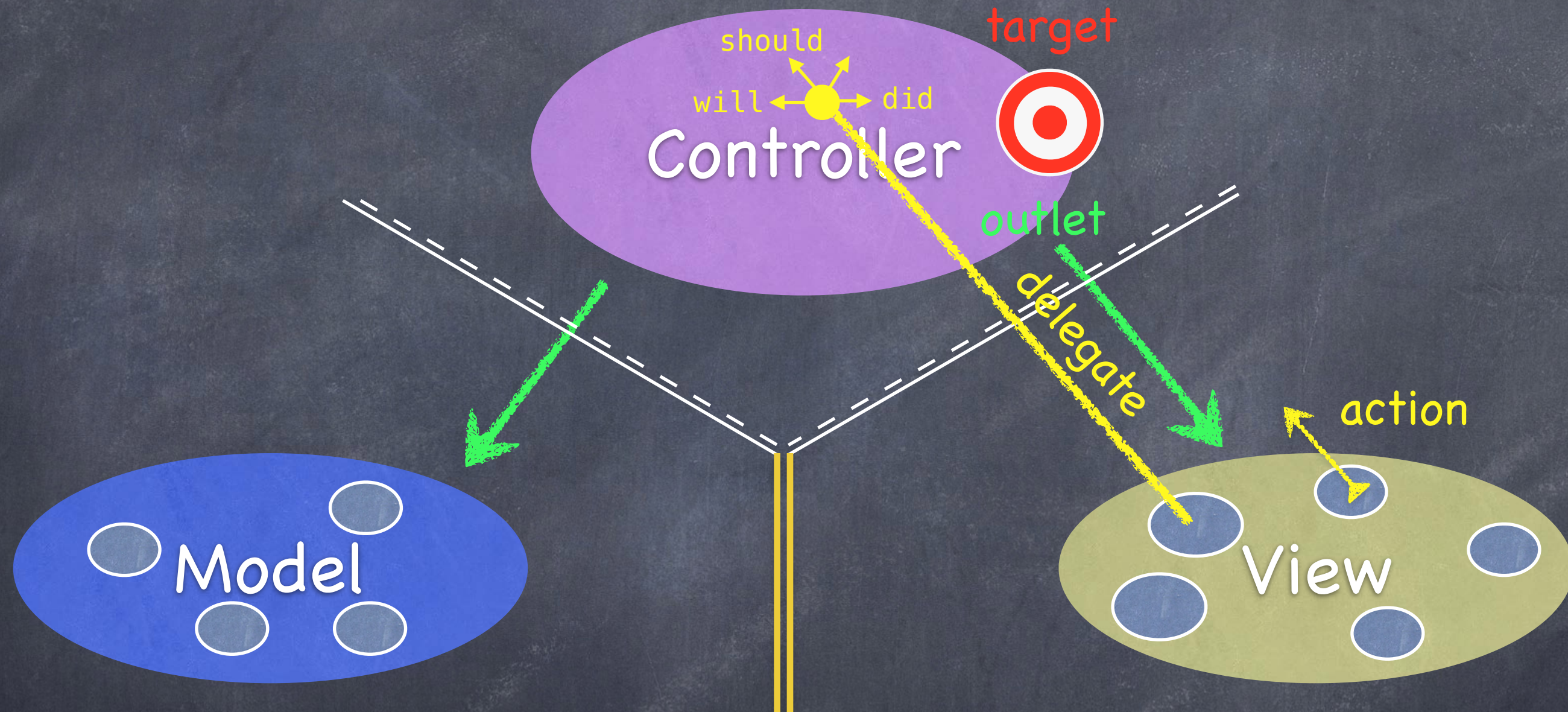
MVC



Sometimes the **View** needs to synchronize with the **Controller**.



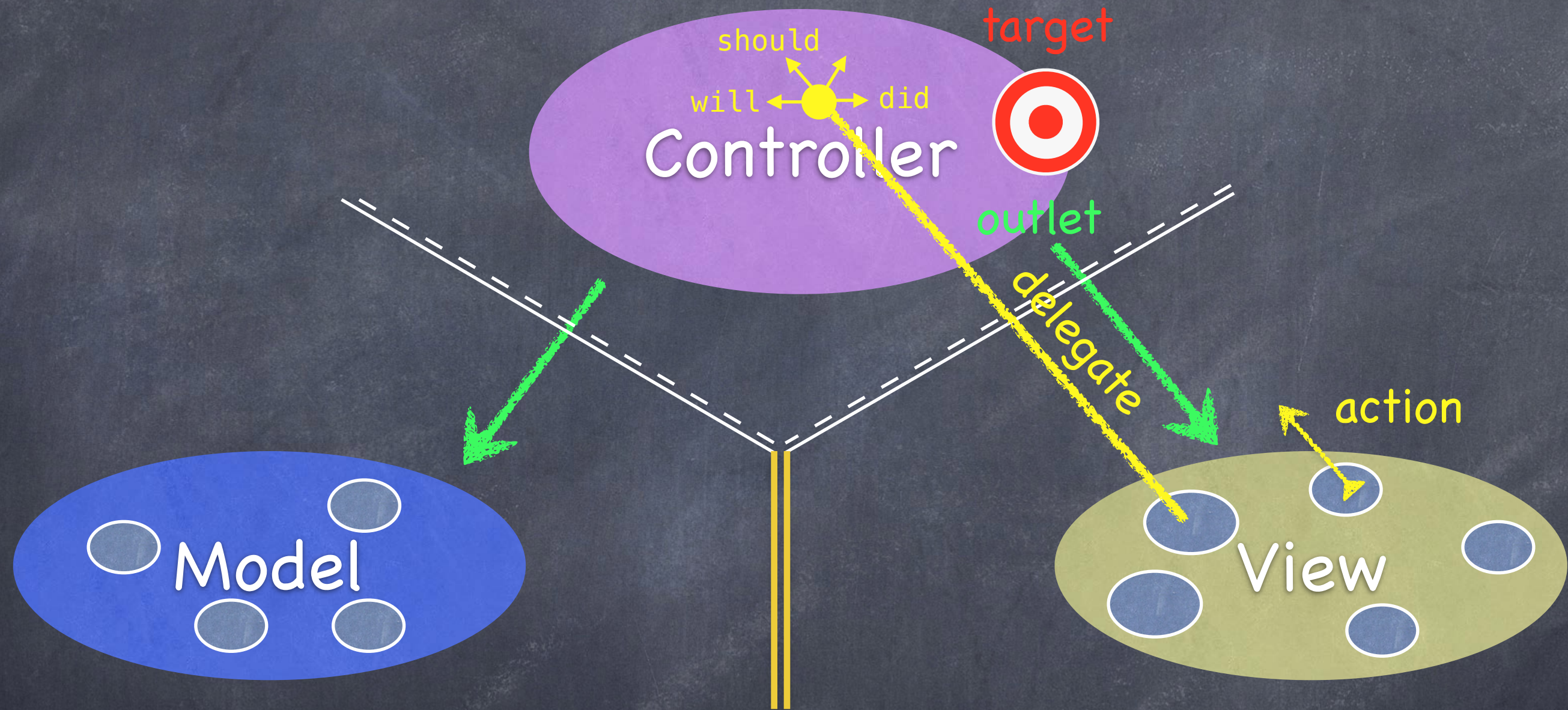
MVC



The **Controller** sets itself as the **View's** delegate.



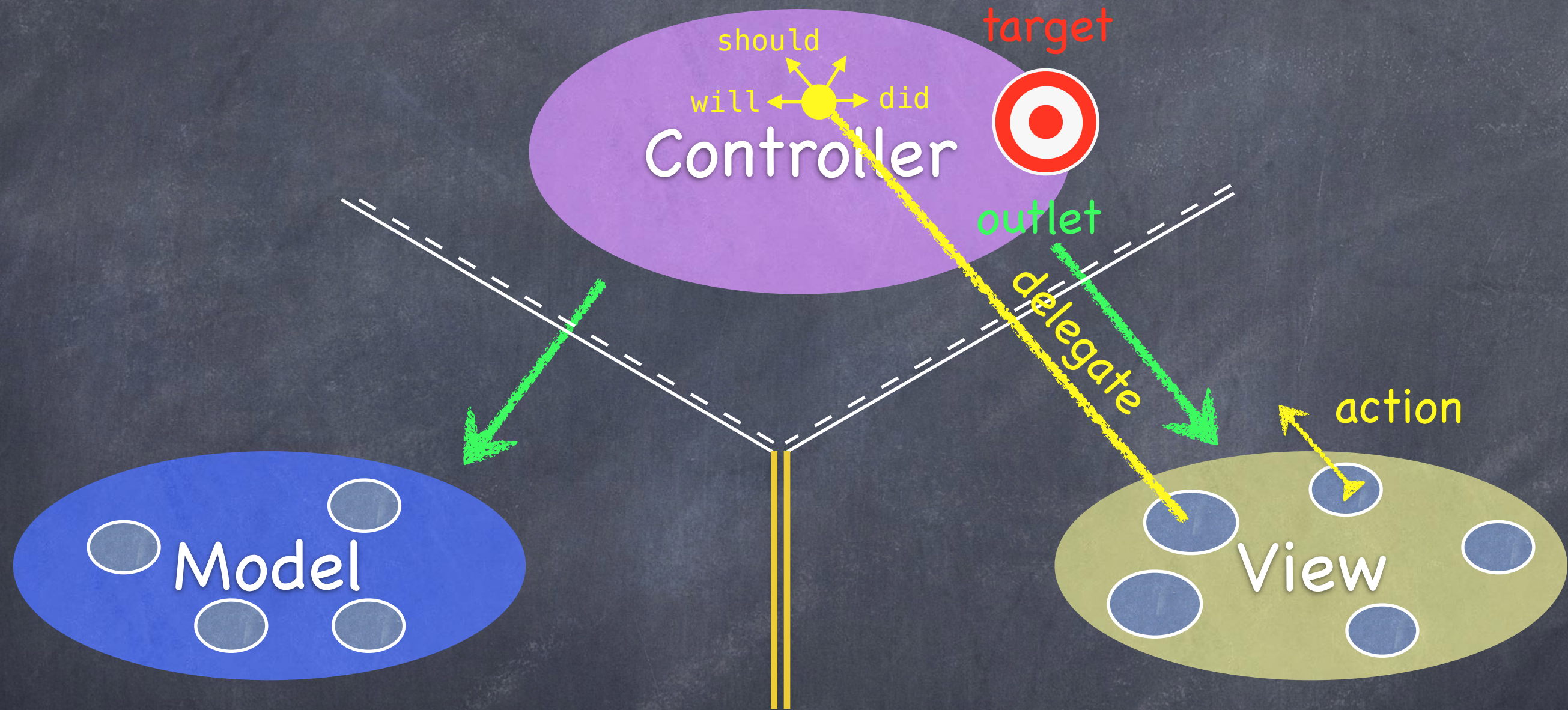
MVC



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



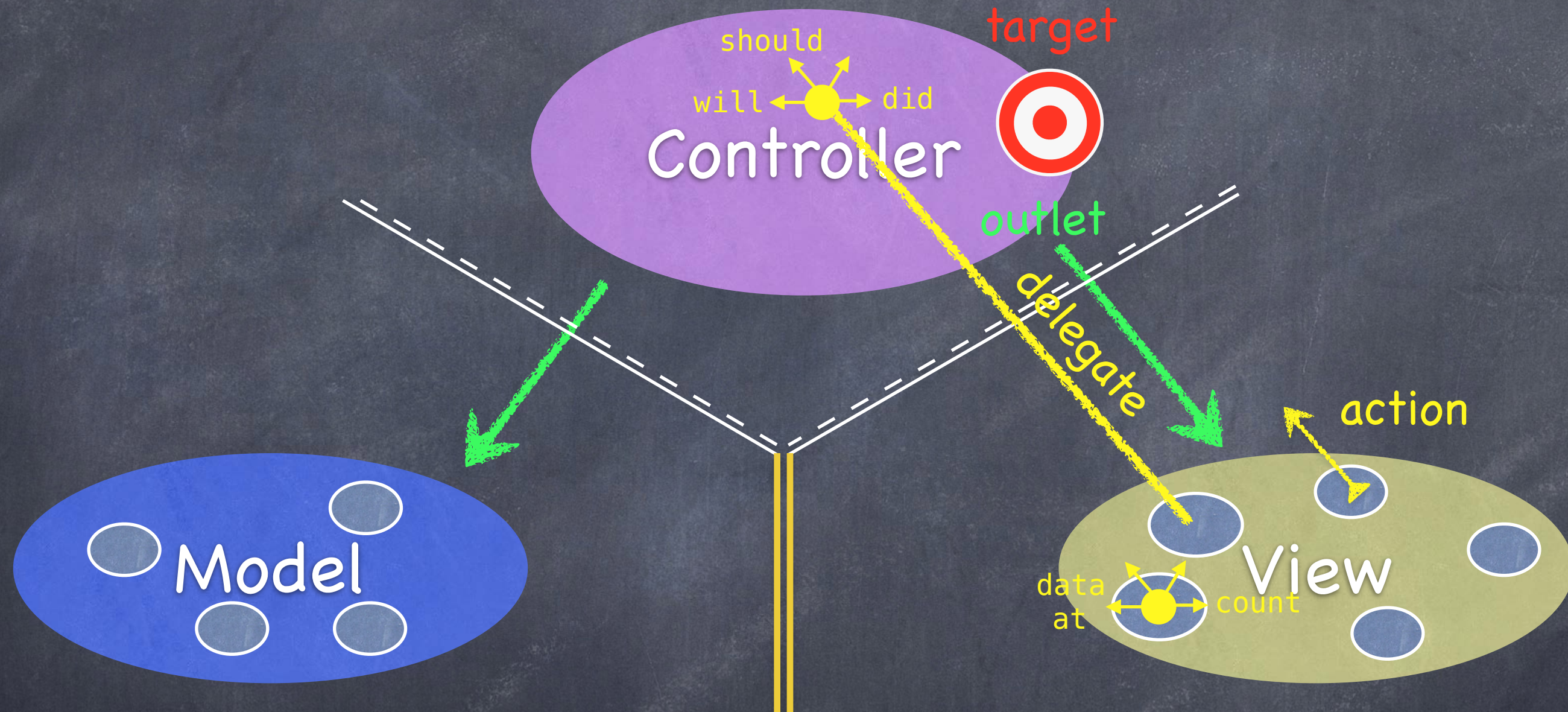
MVC



Views do not own the data they display.



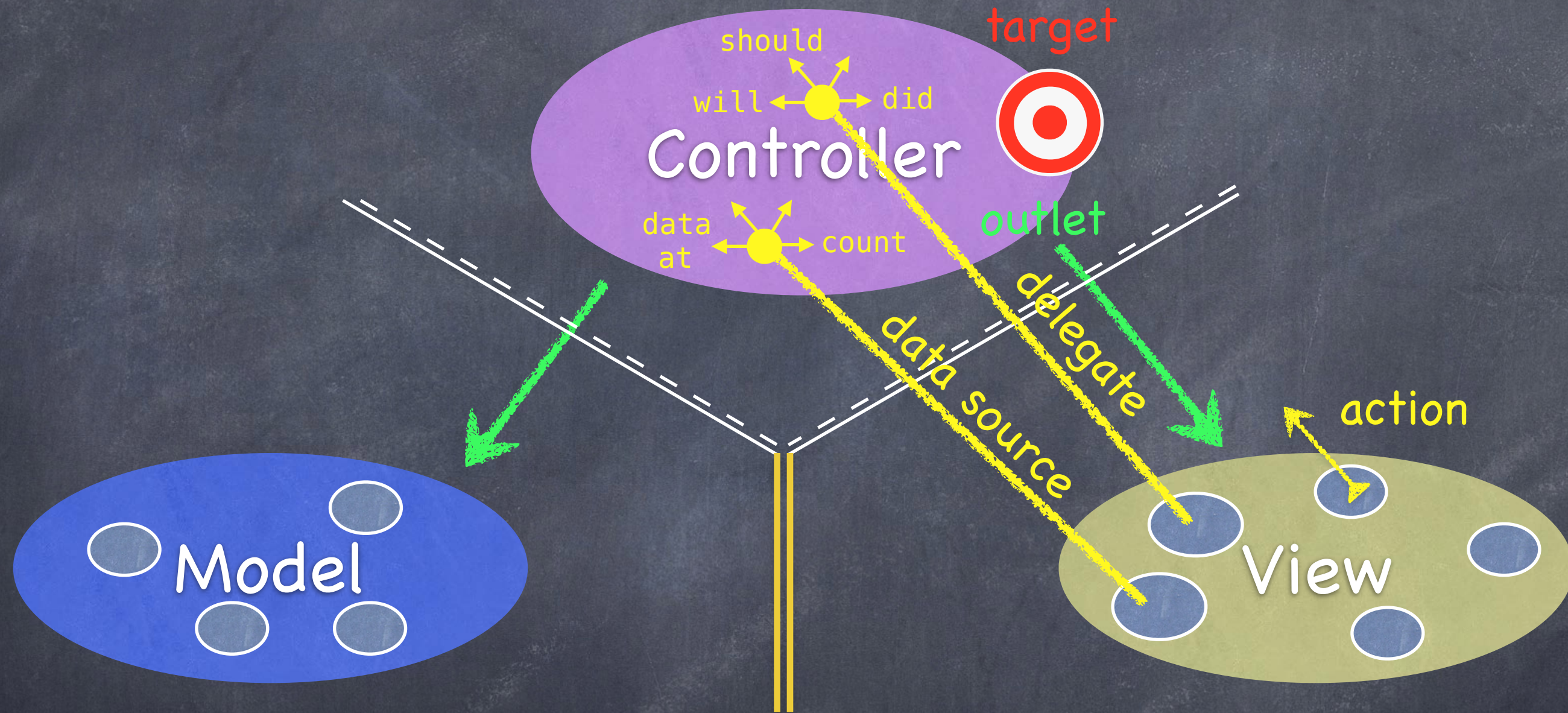
MVC



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



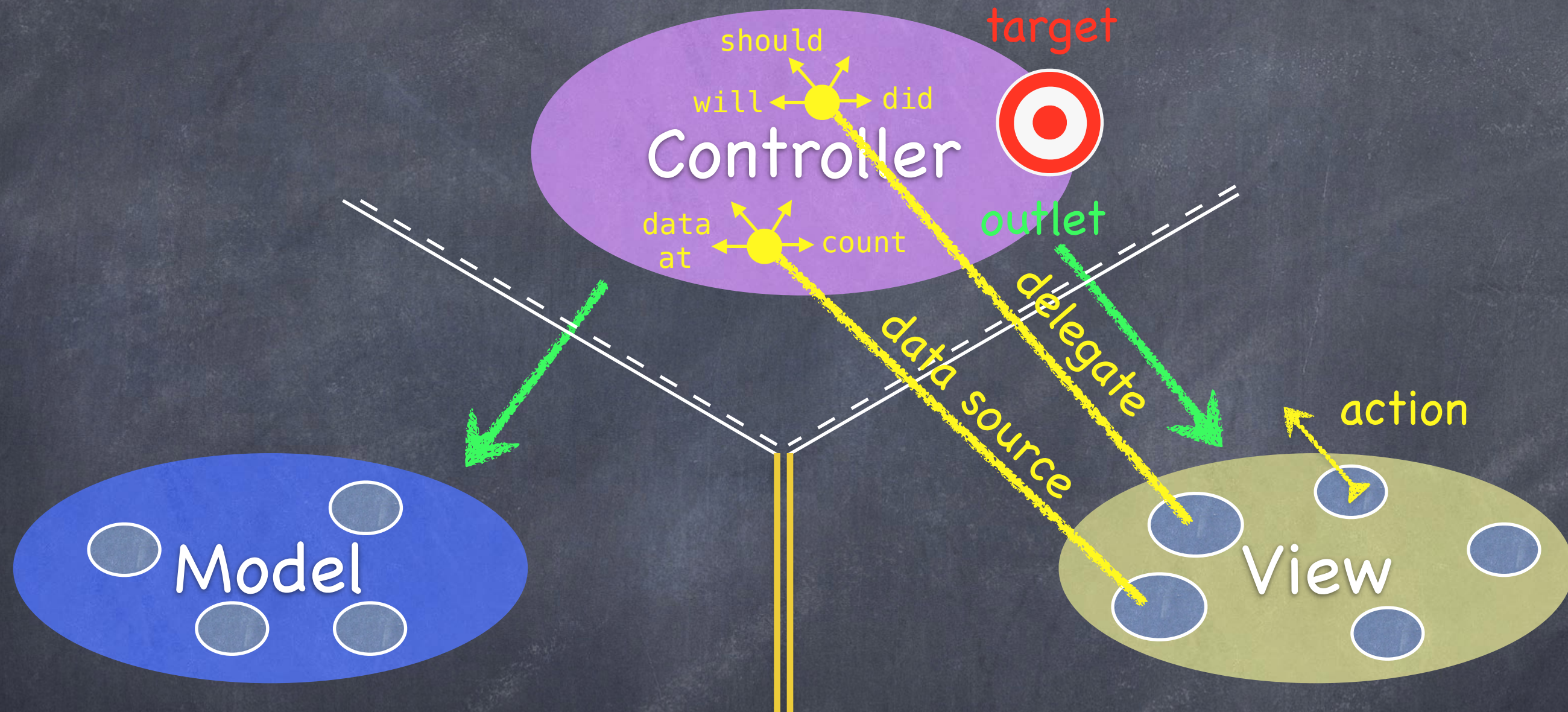
MVC



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



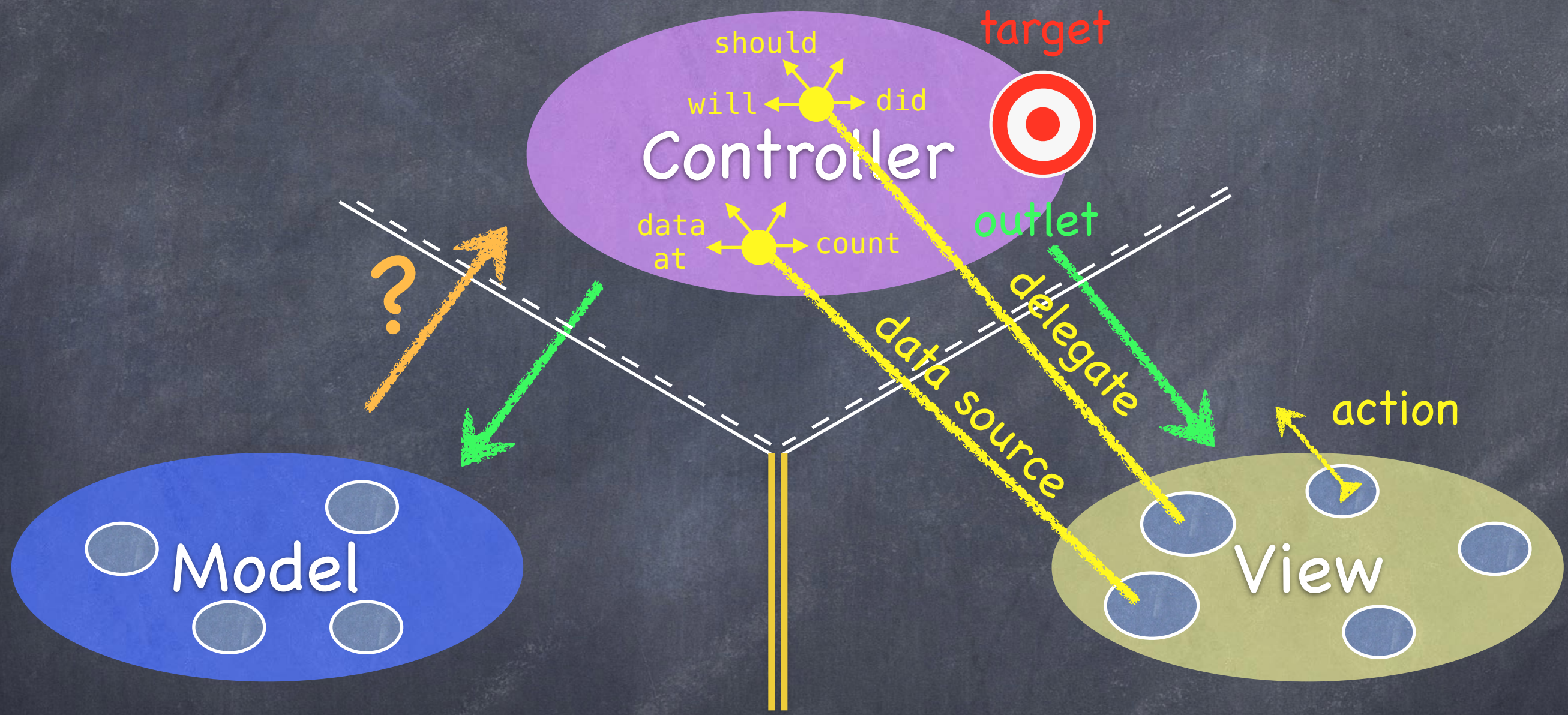
MVC



Controllers interpret/format Model information for the View.

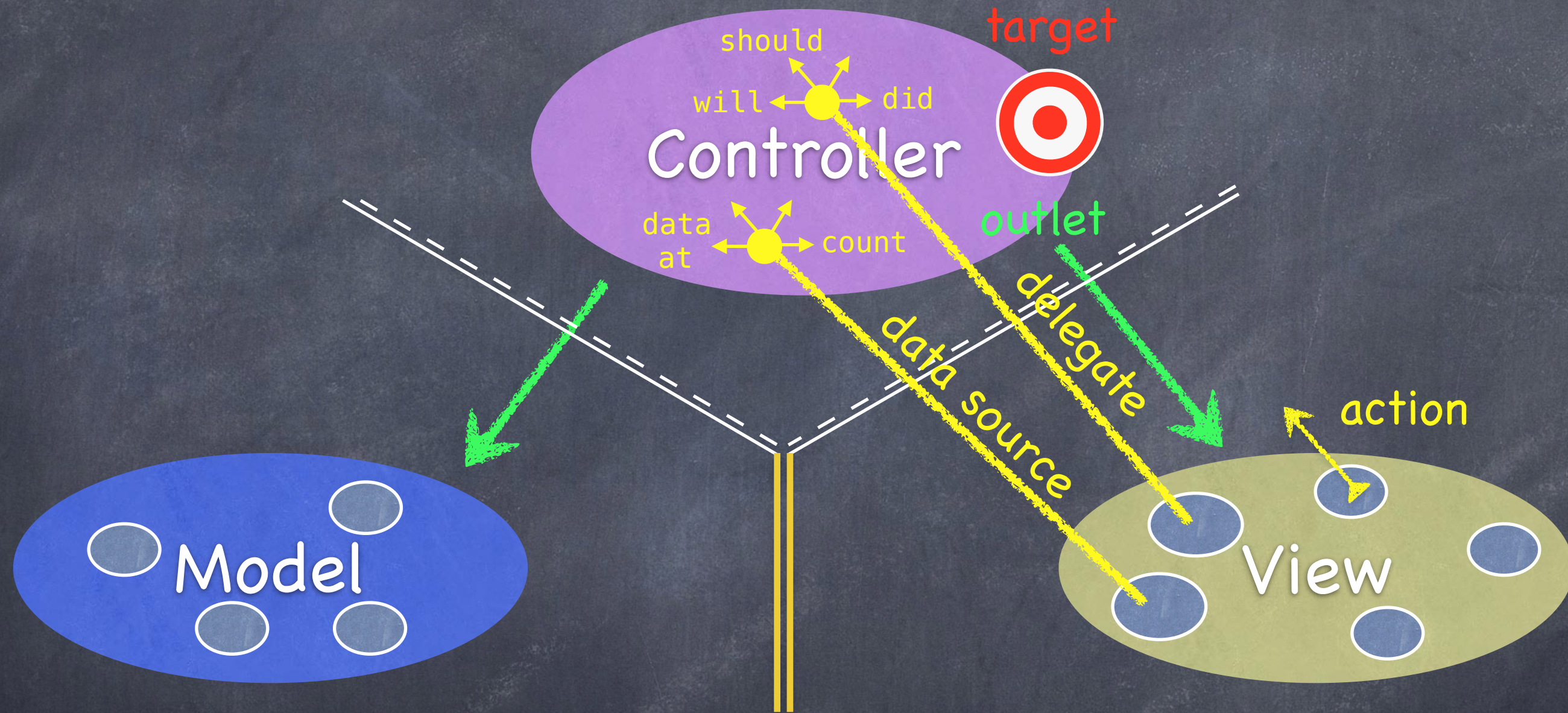


MVC



Can the Model talk directly to the Controller?

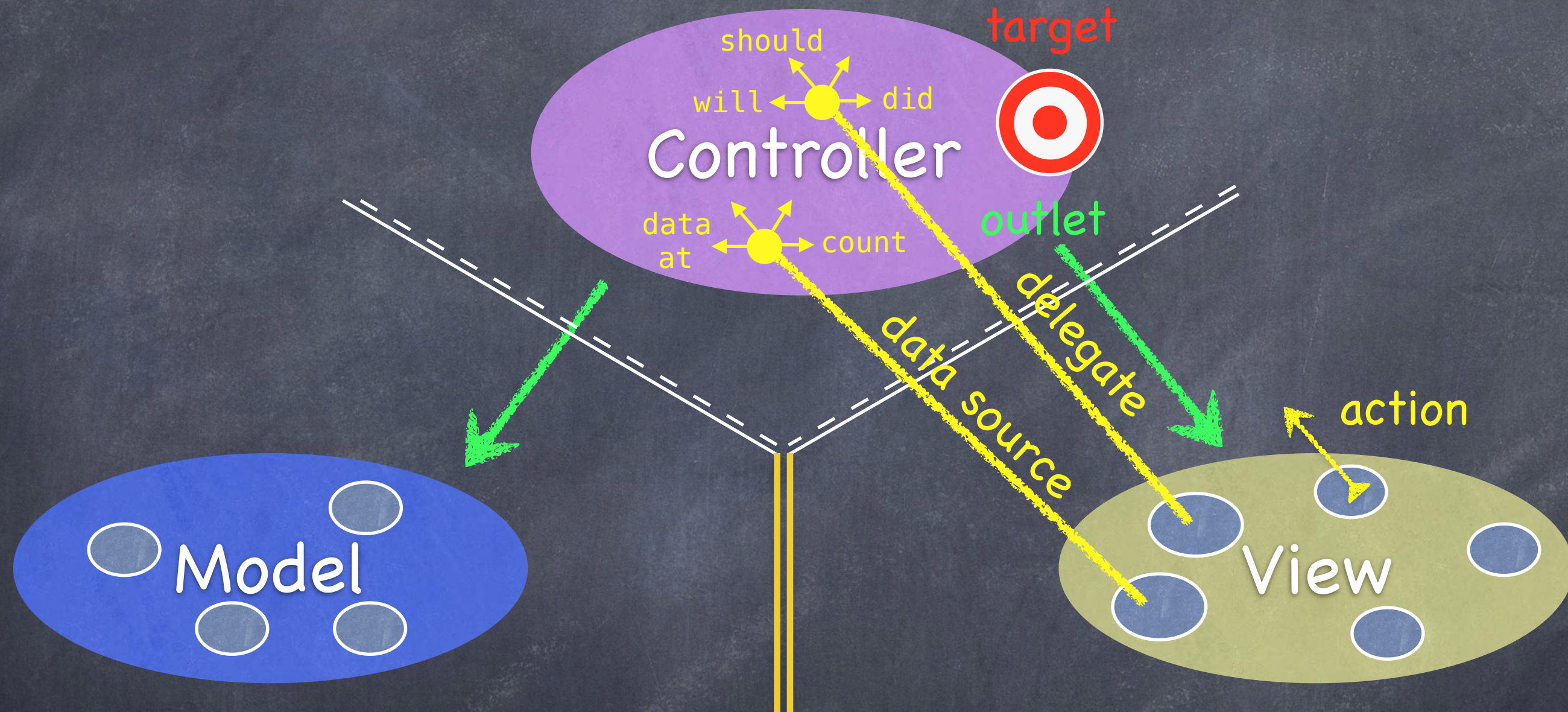
MVC



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



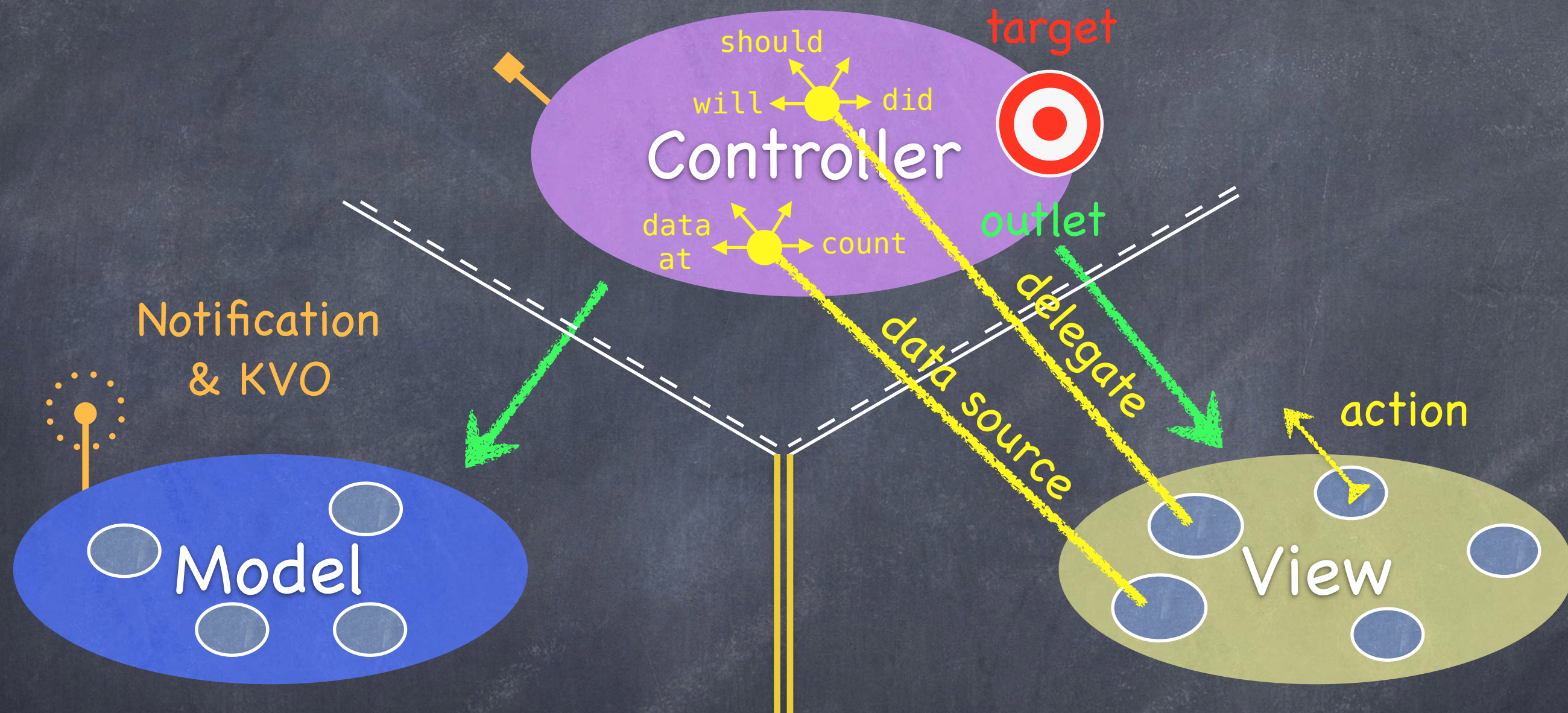
MVC



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



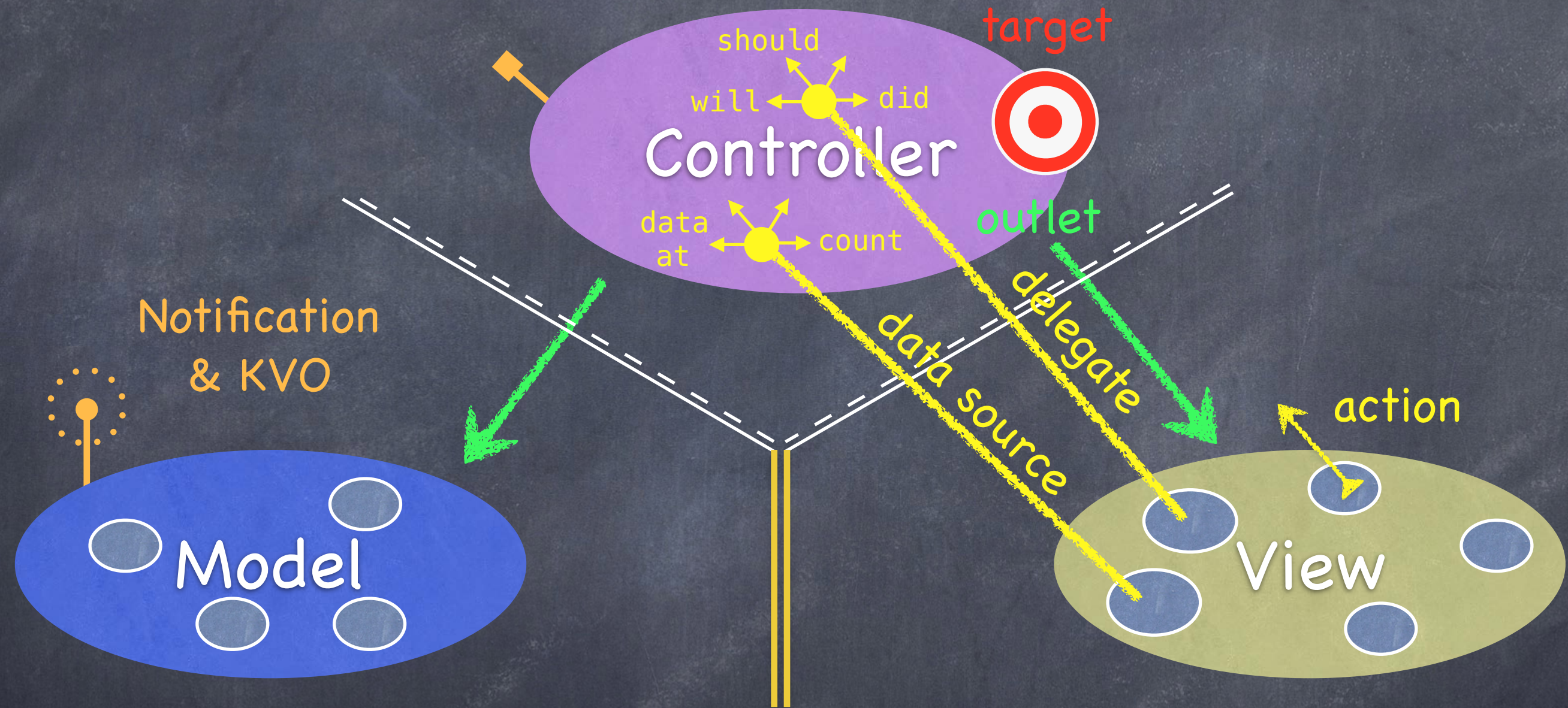
MVC



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



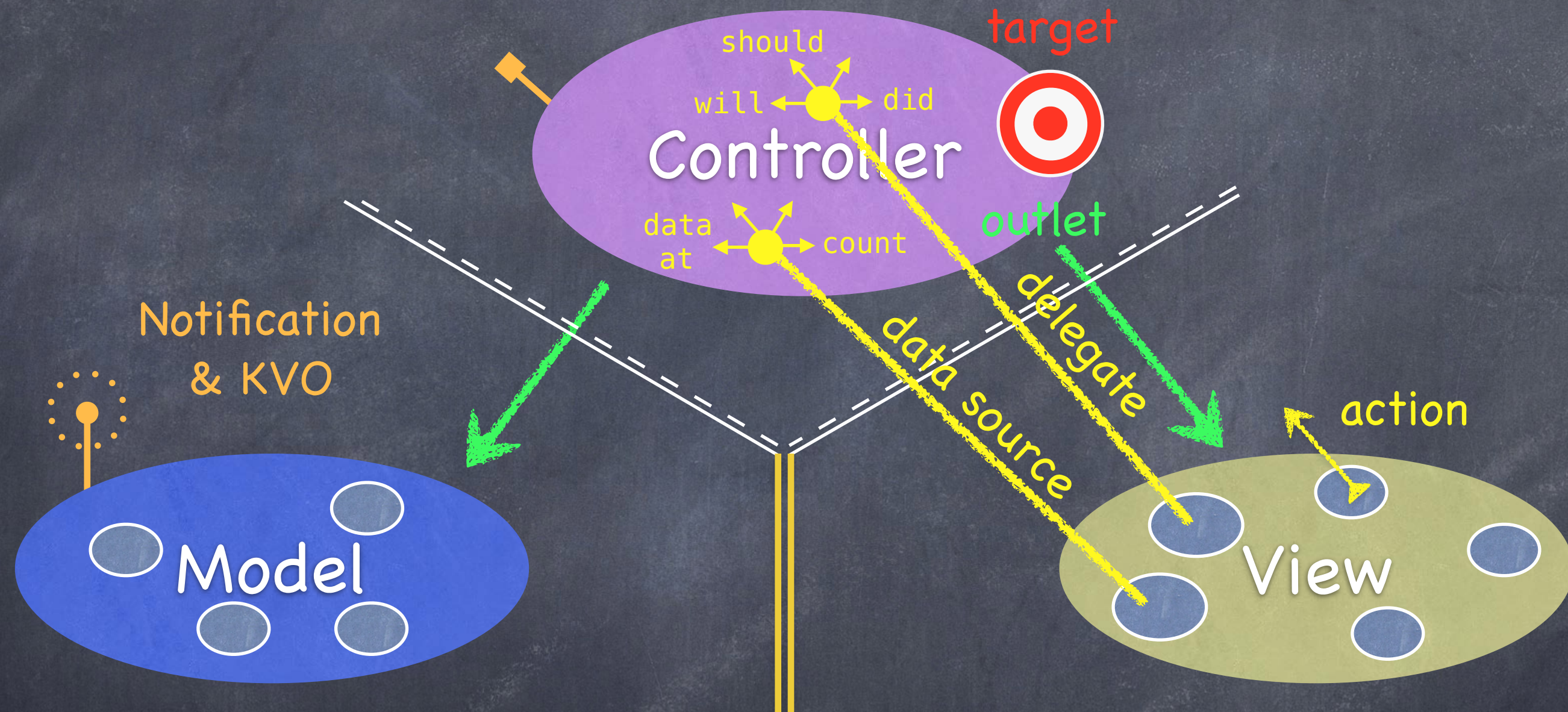
MVC



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



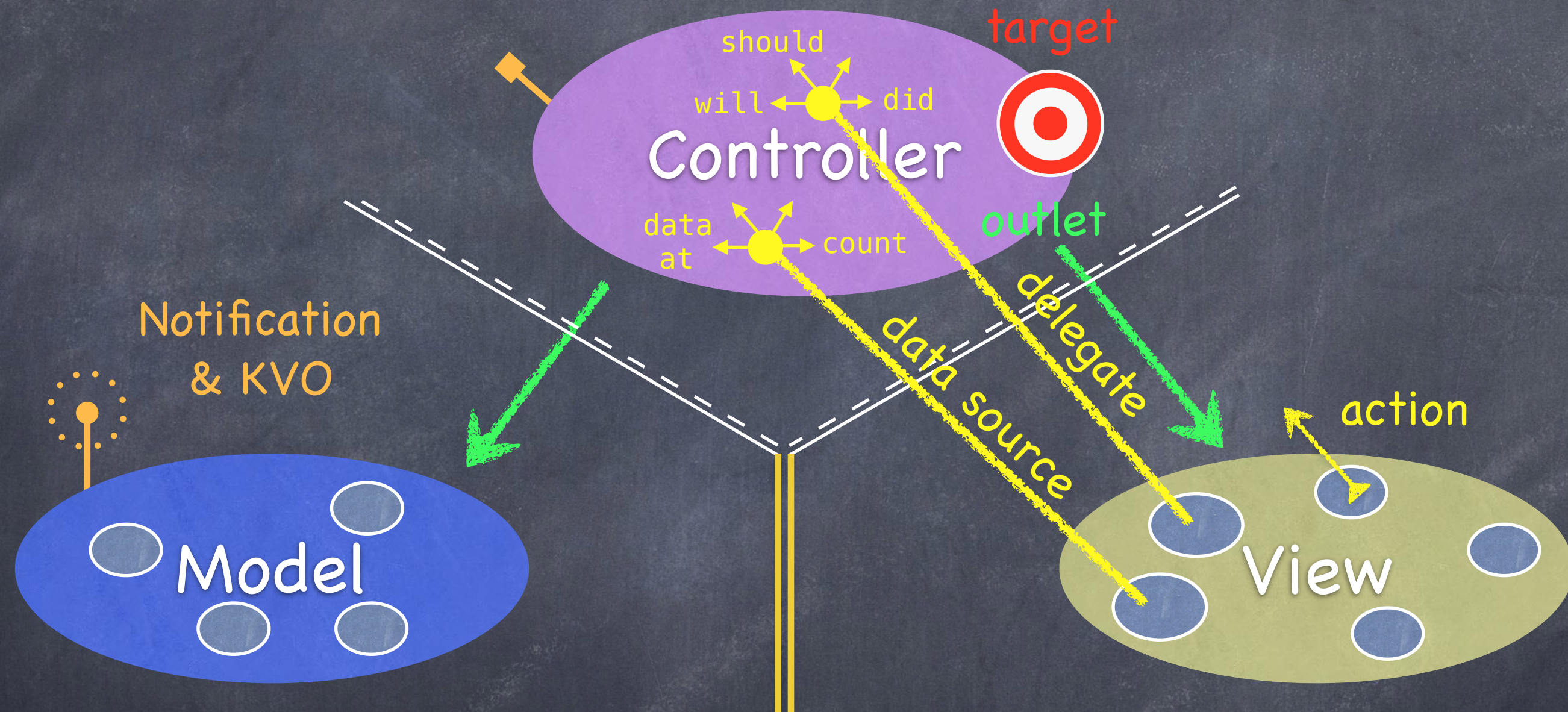
MVC



A **View** might “tune in,” but probably not to a **Model’s** “station.”



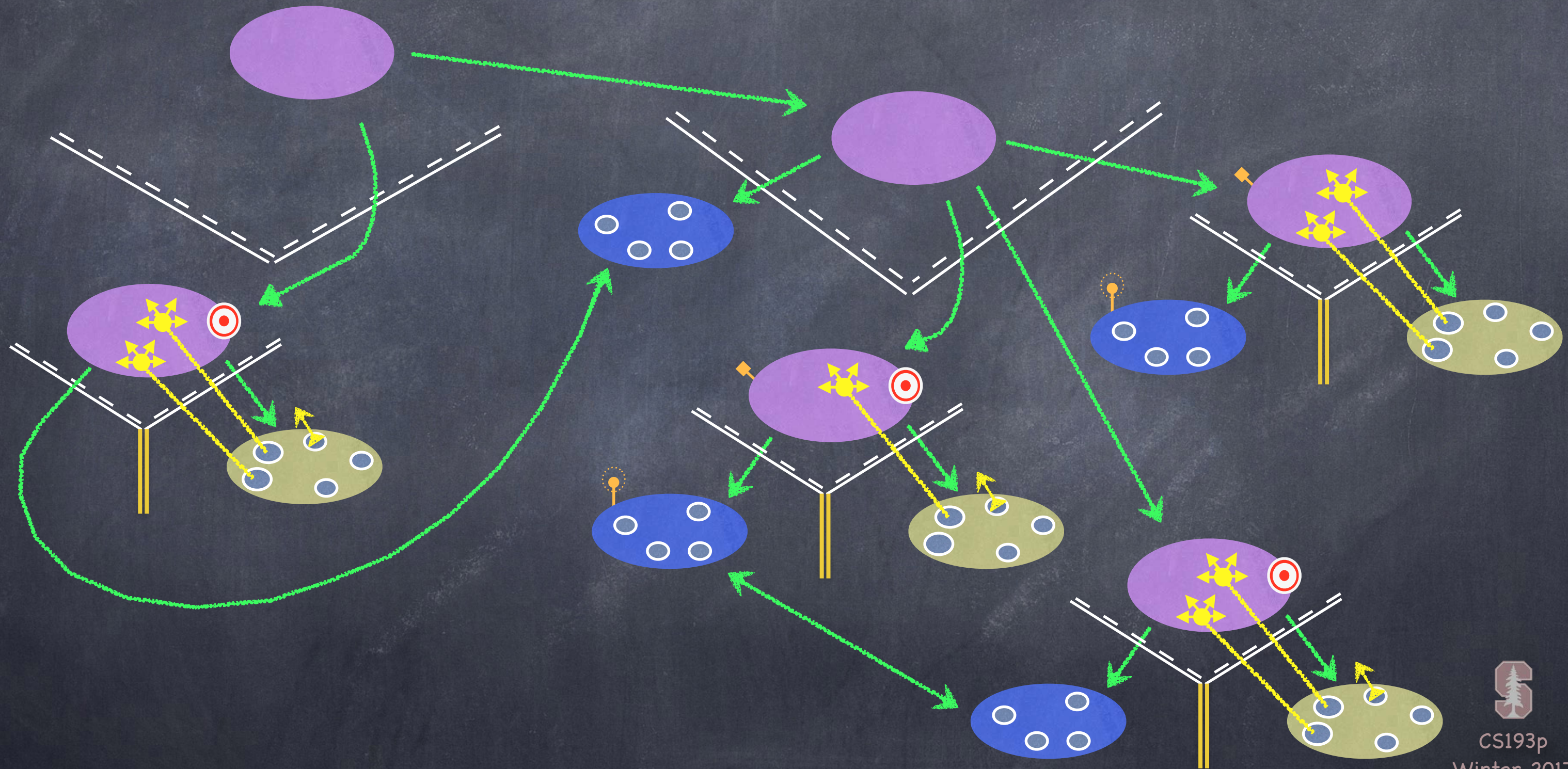
MVC



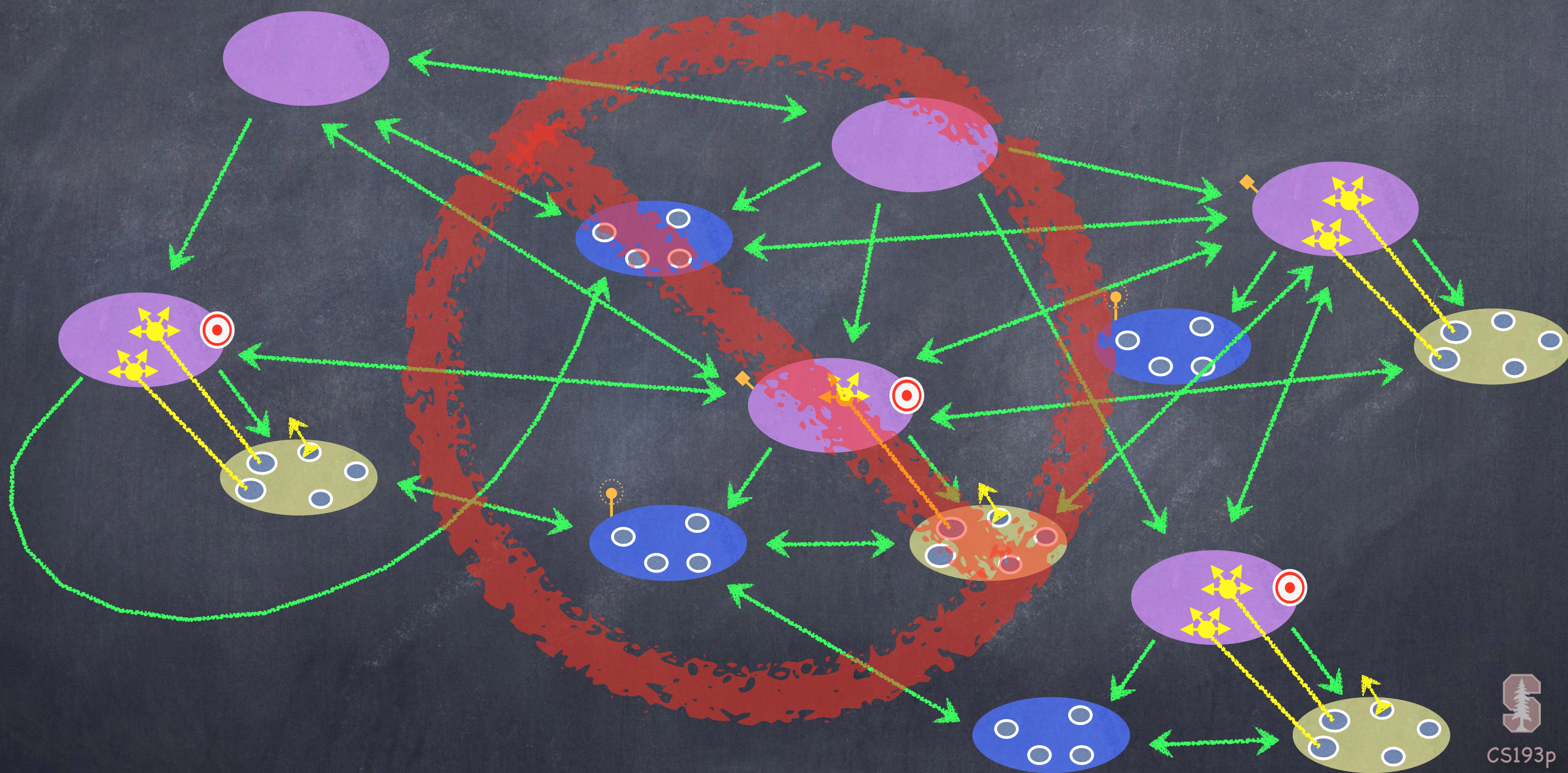
Now combine MVC groups to make complicated programs ...



MVCs working together



MVCs not working together



Demo

👁️ Calculator continued ...

MVC

`struct` vs. `class` (mutating, etc.)

public versus `private` API

more examples of Optional

`Dictionary<KeyType, ValueType>`

`enum`

associated values

`switch`

Functions as types

Closure syntax for defining functions “on the fly”

`UIStackView`

First peek at Autolayout (stick things to the edges)

