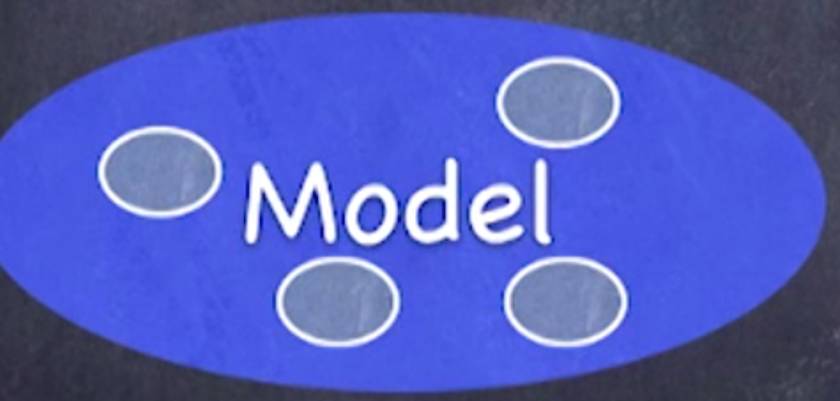
MVC & CocoaPods

Elijah

1. Model-View-Controller

MVC

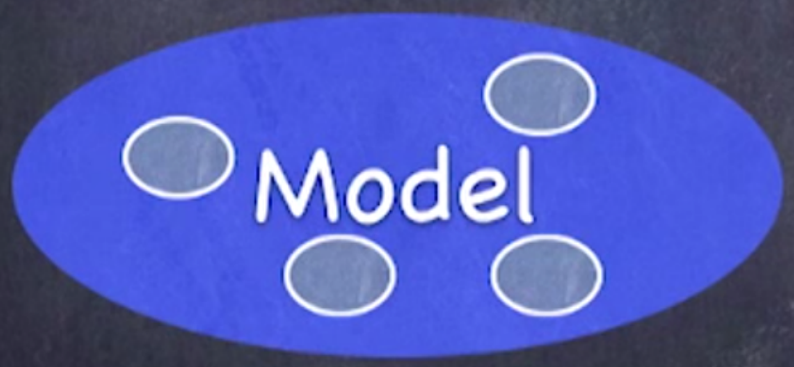
Controller



Divide objects in your program into 3 "camps."

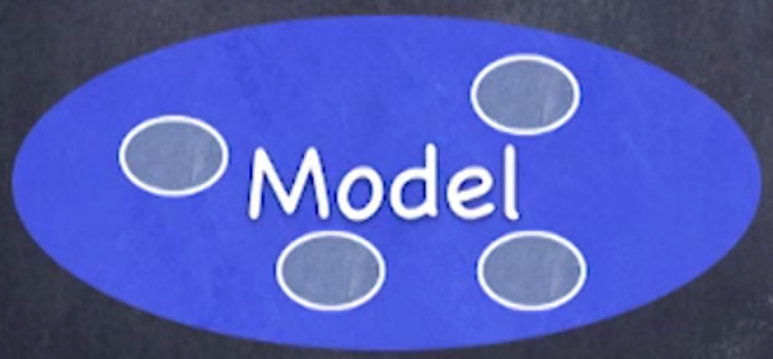
MVC

Controller



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

MVC



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

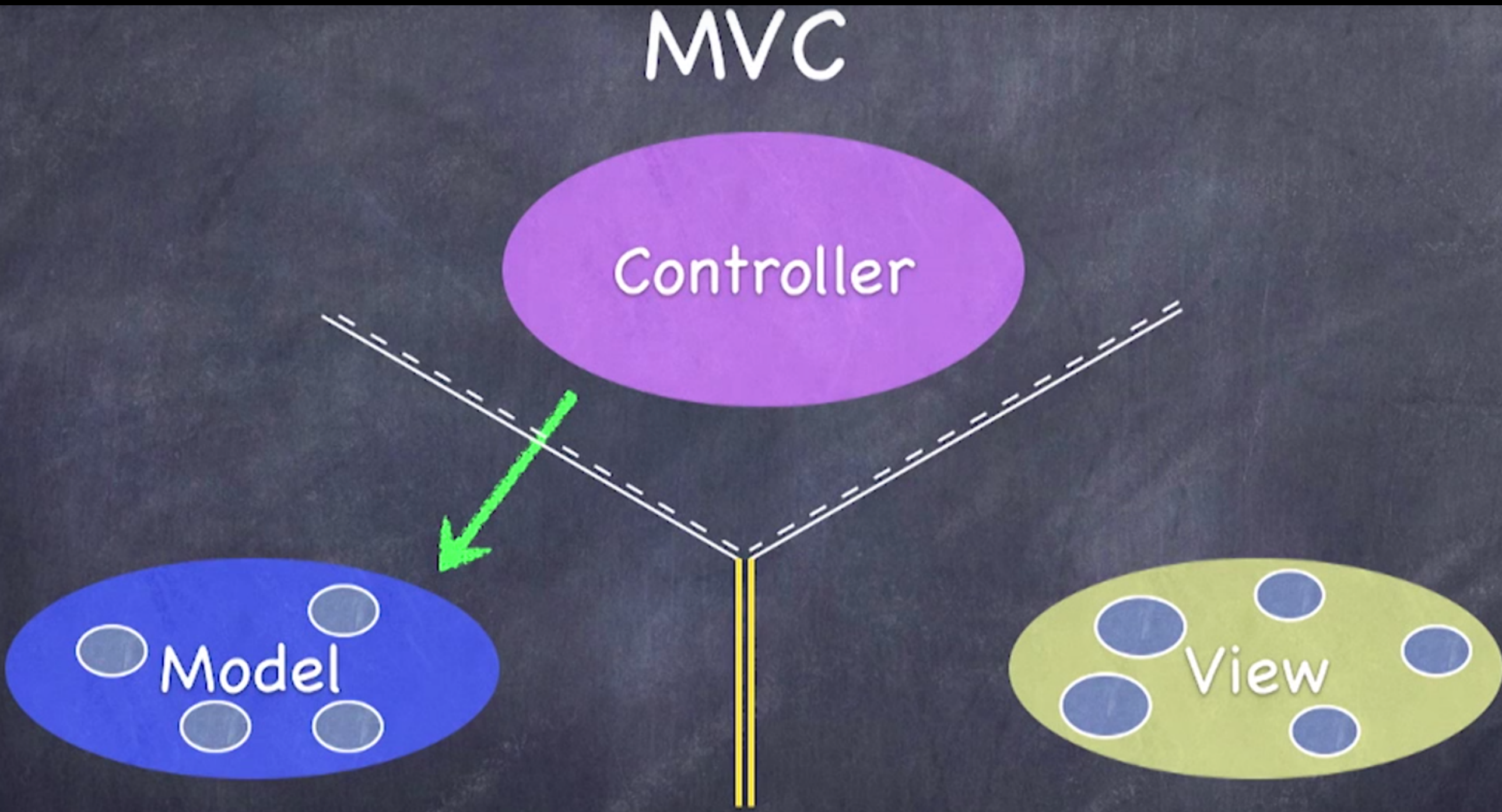
MVC

Controller

Model

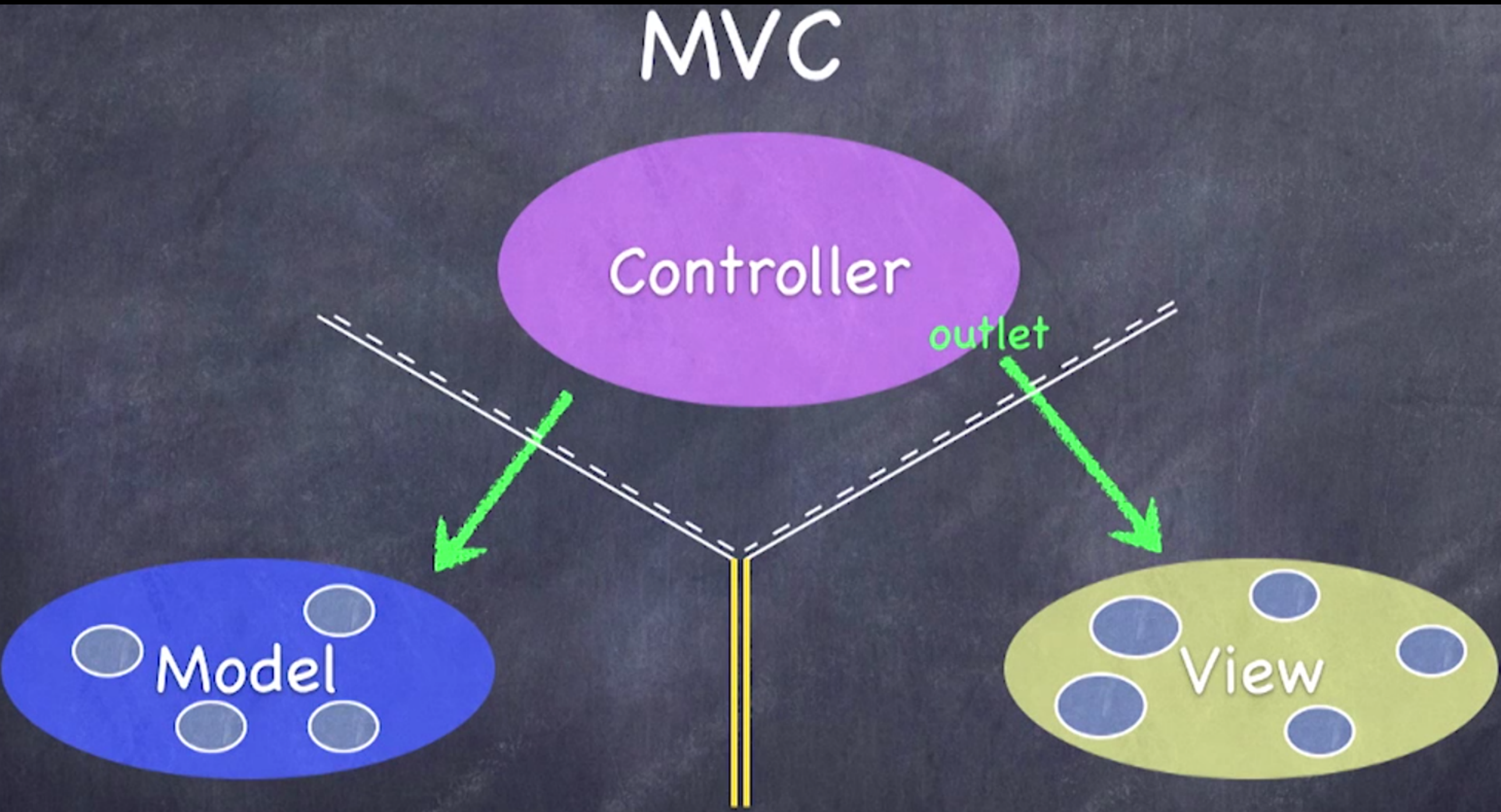
View = Your Controller's minions

MVC



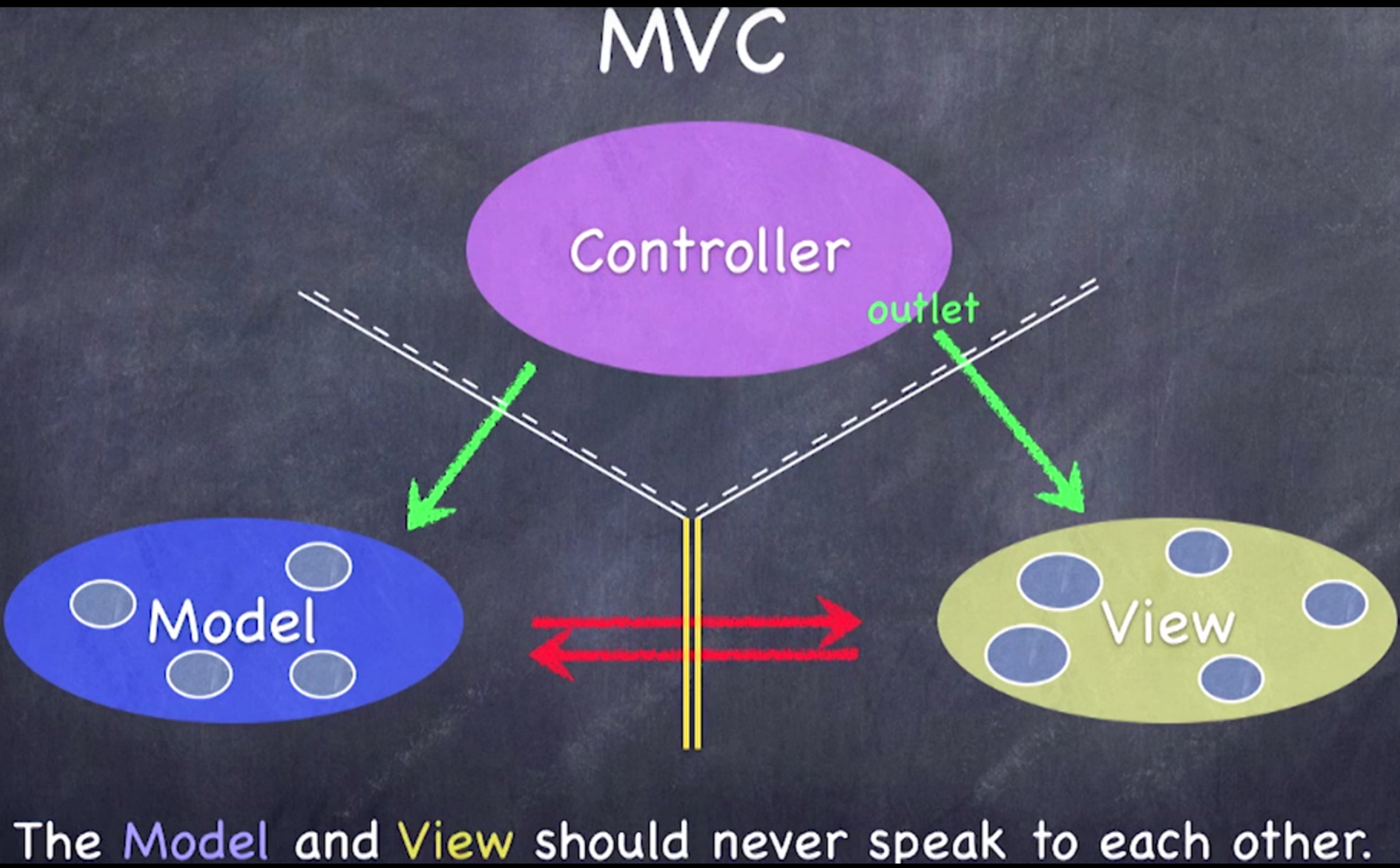
Controllers can always talk directly to their Model.

MVC



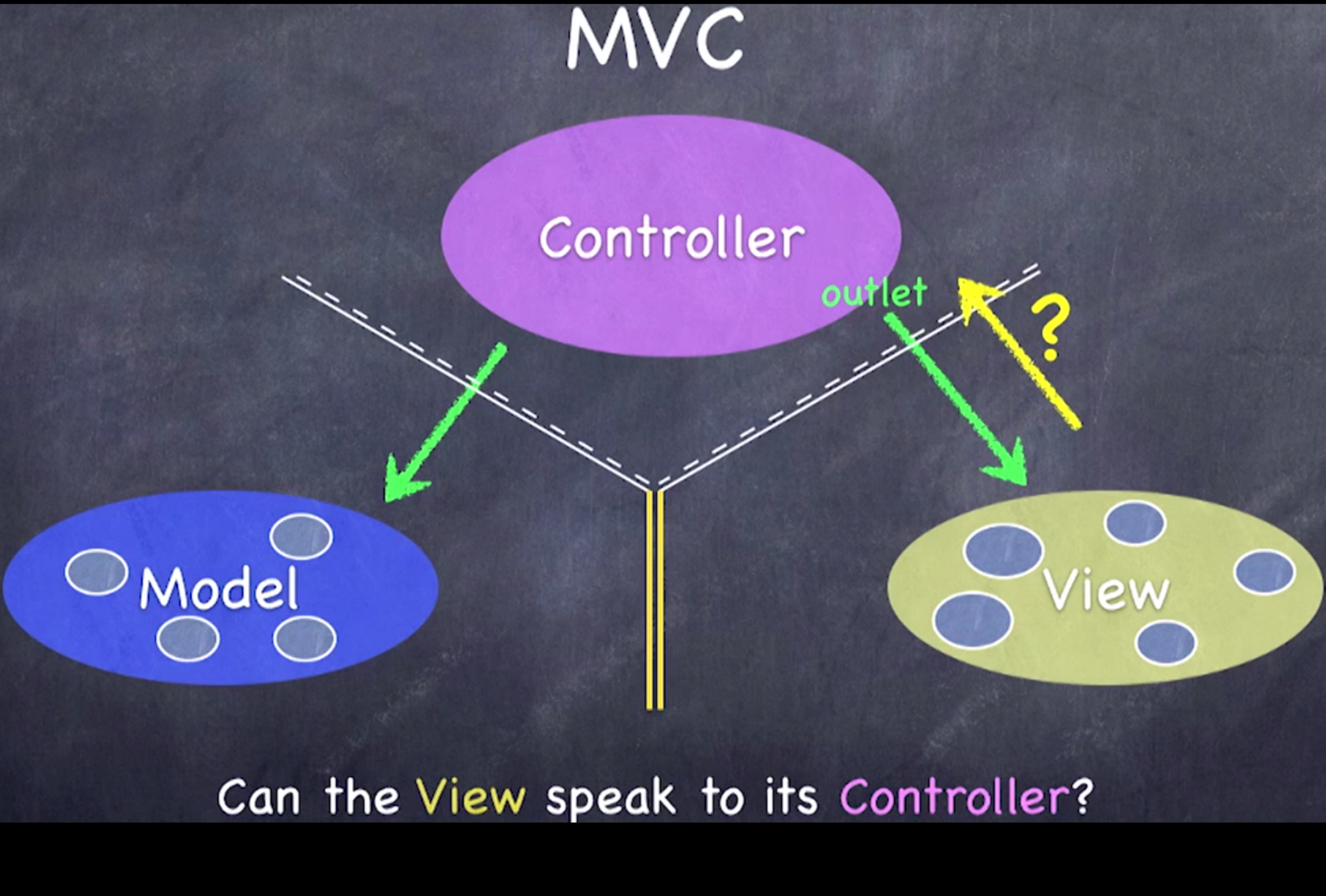
Controllers can also talk directly to their View.

MVC



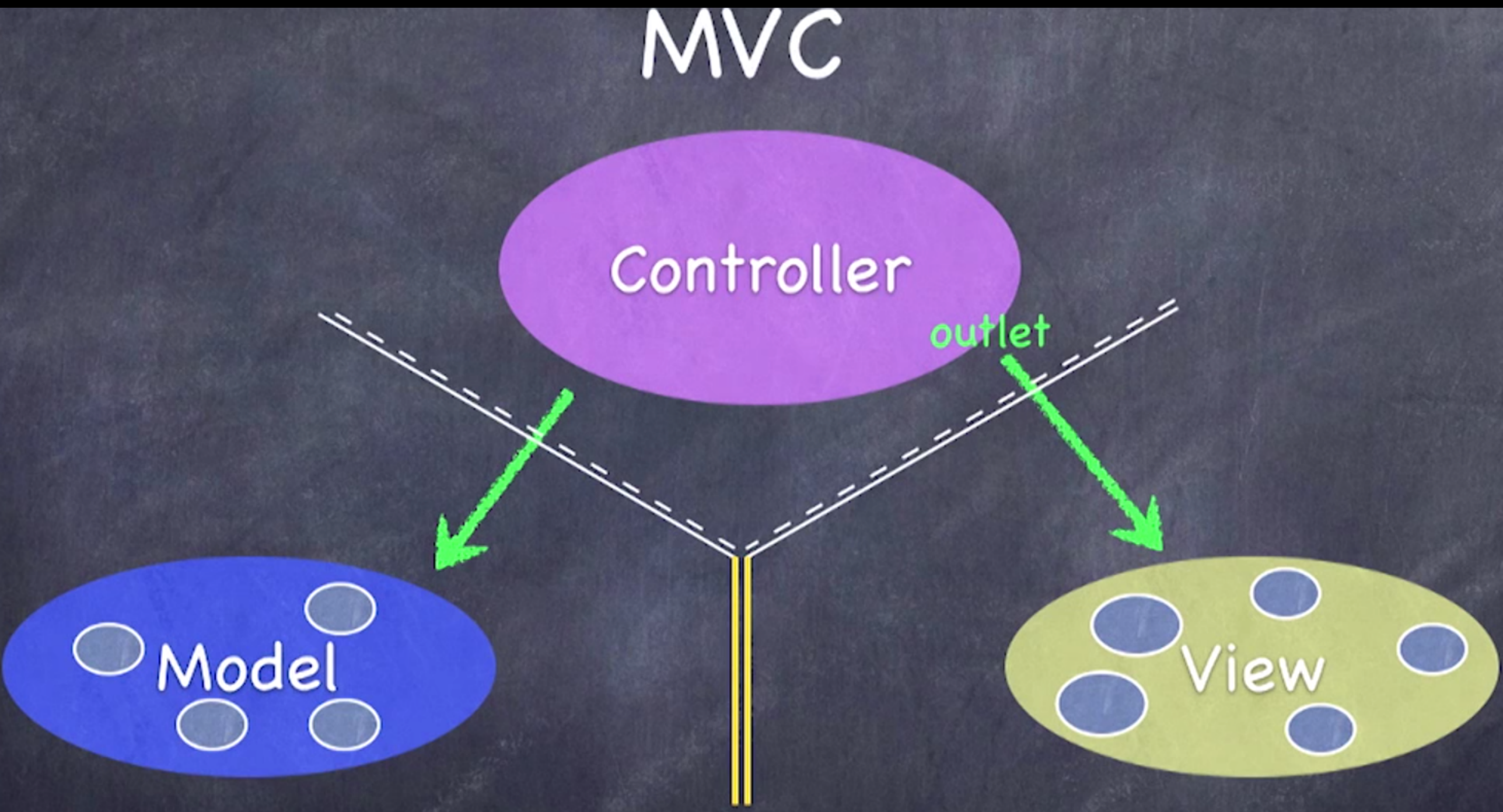
The Model and View should never speak to each other.

MVC



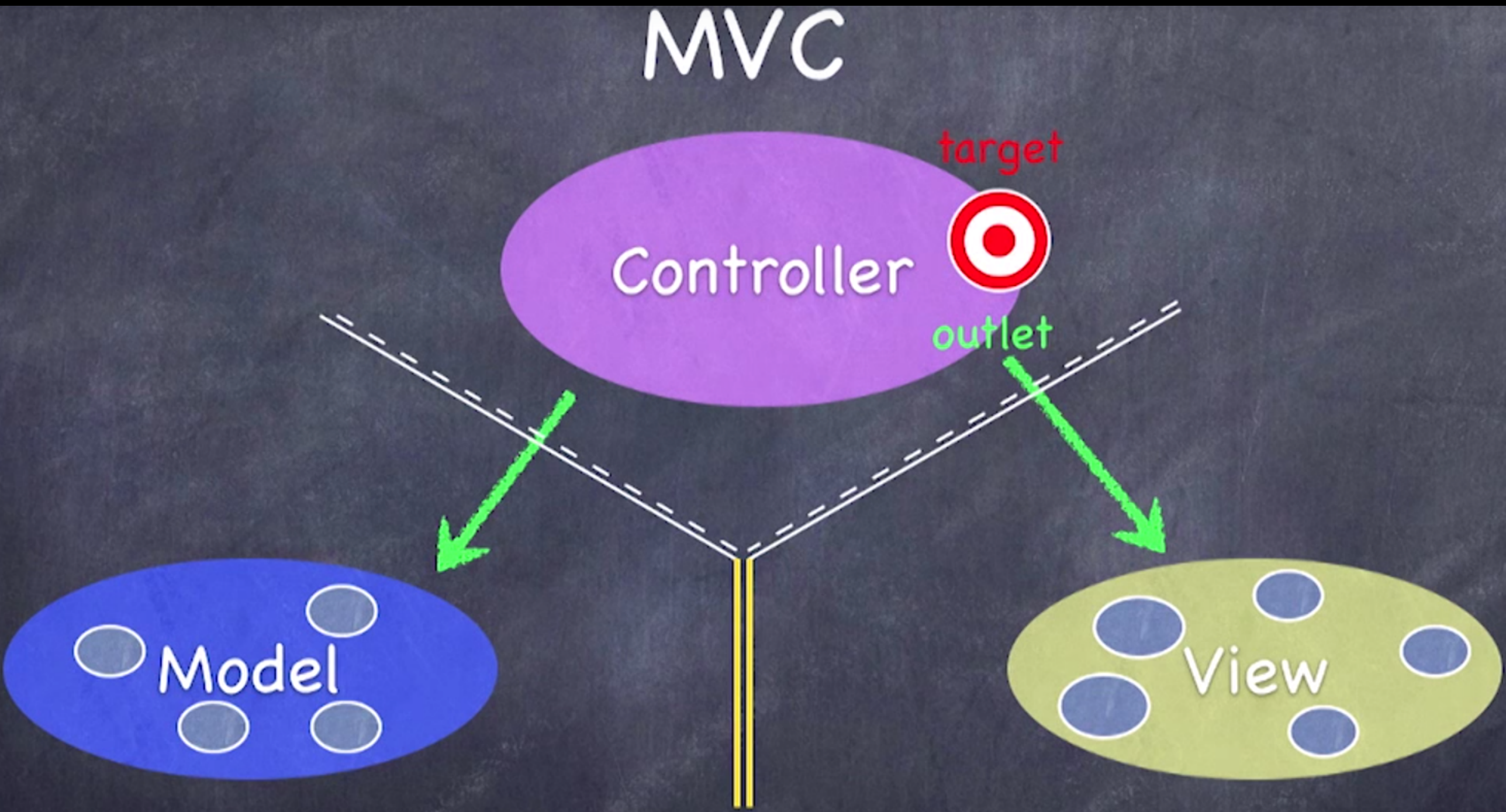
Can the View speak to its Controller?

MVC



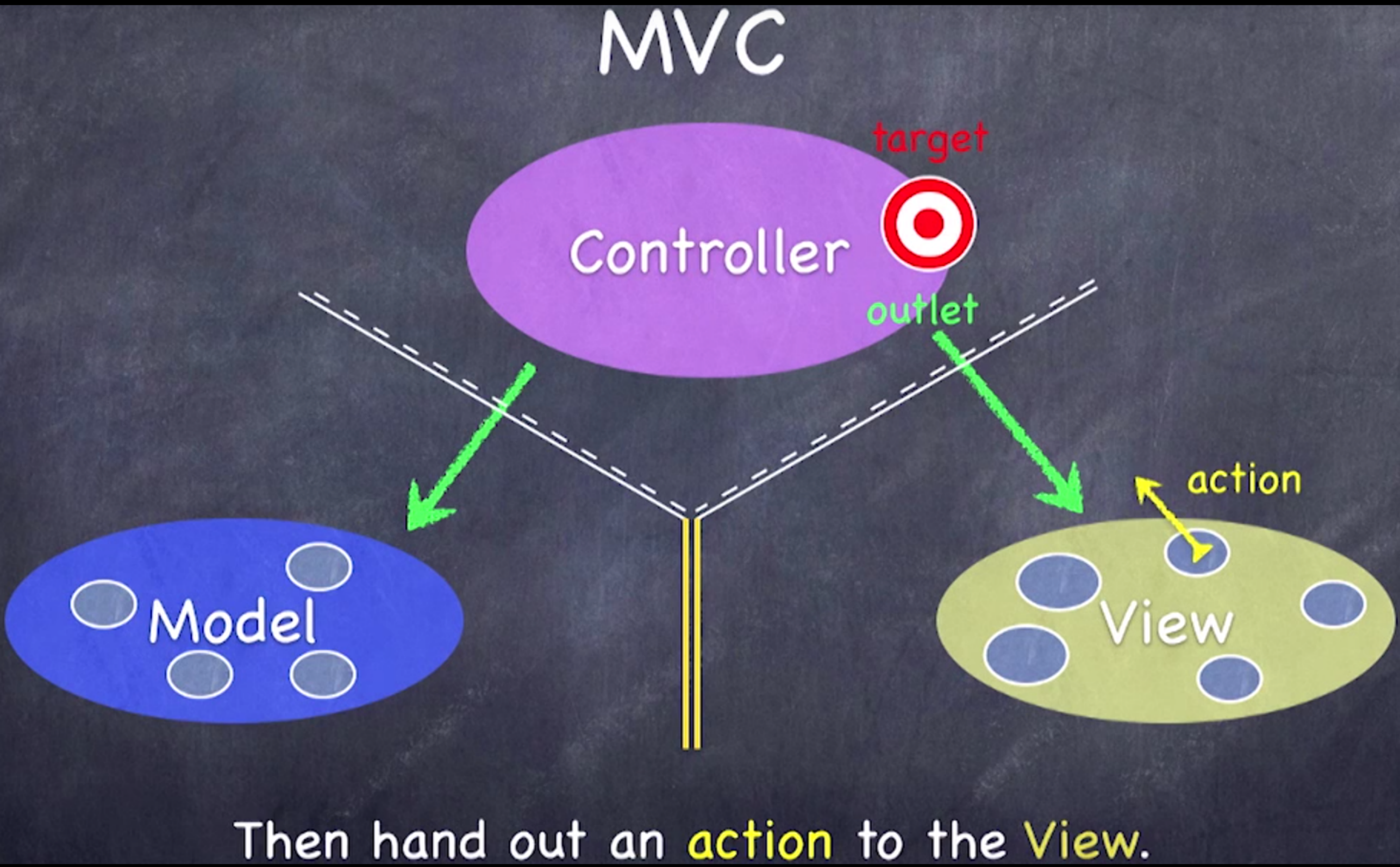
Sort of. Communication is "blind" and structured.

MVC



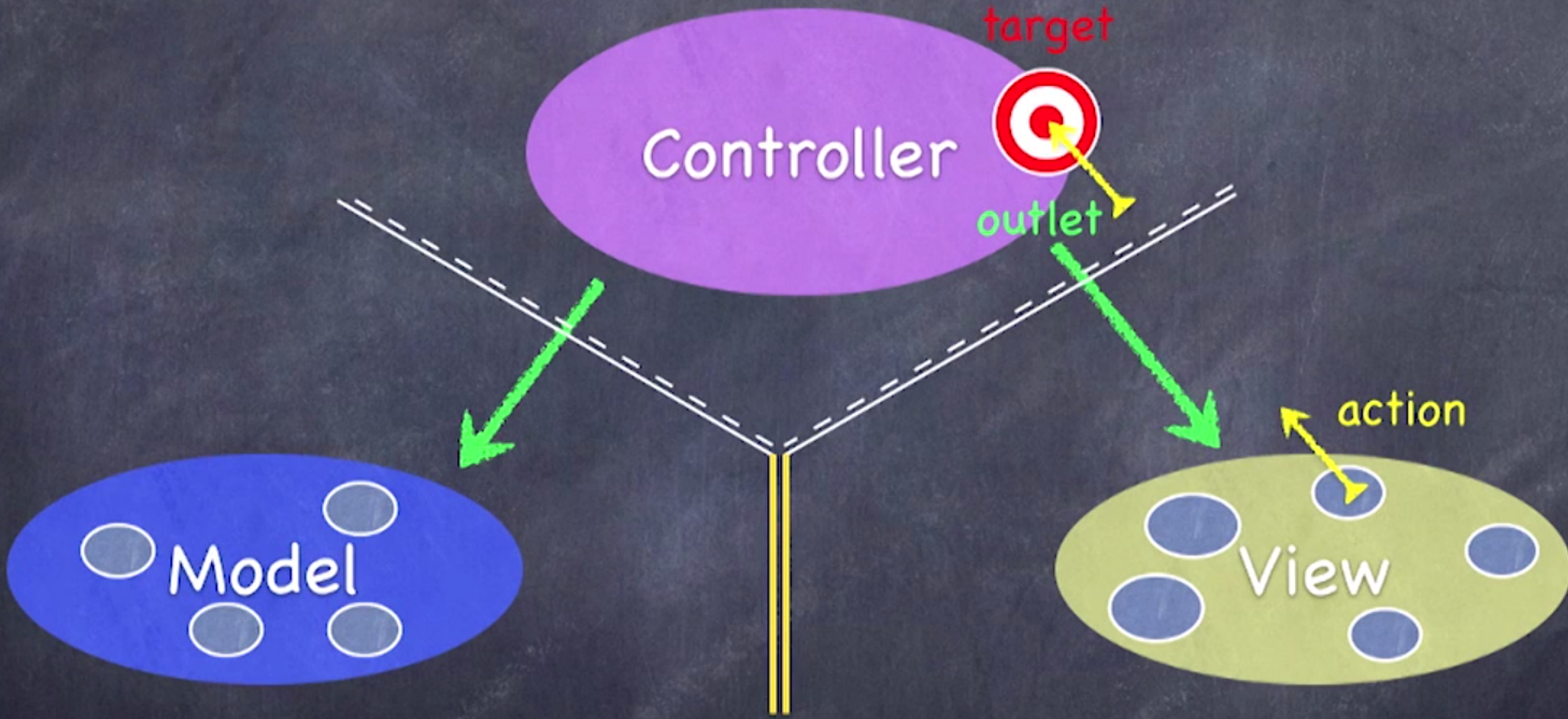
The Controller can drop a target on itself.

MVC



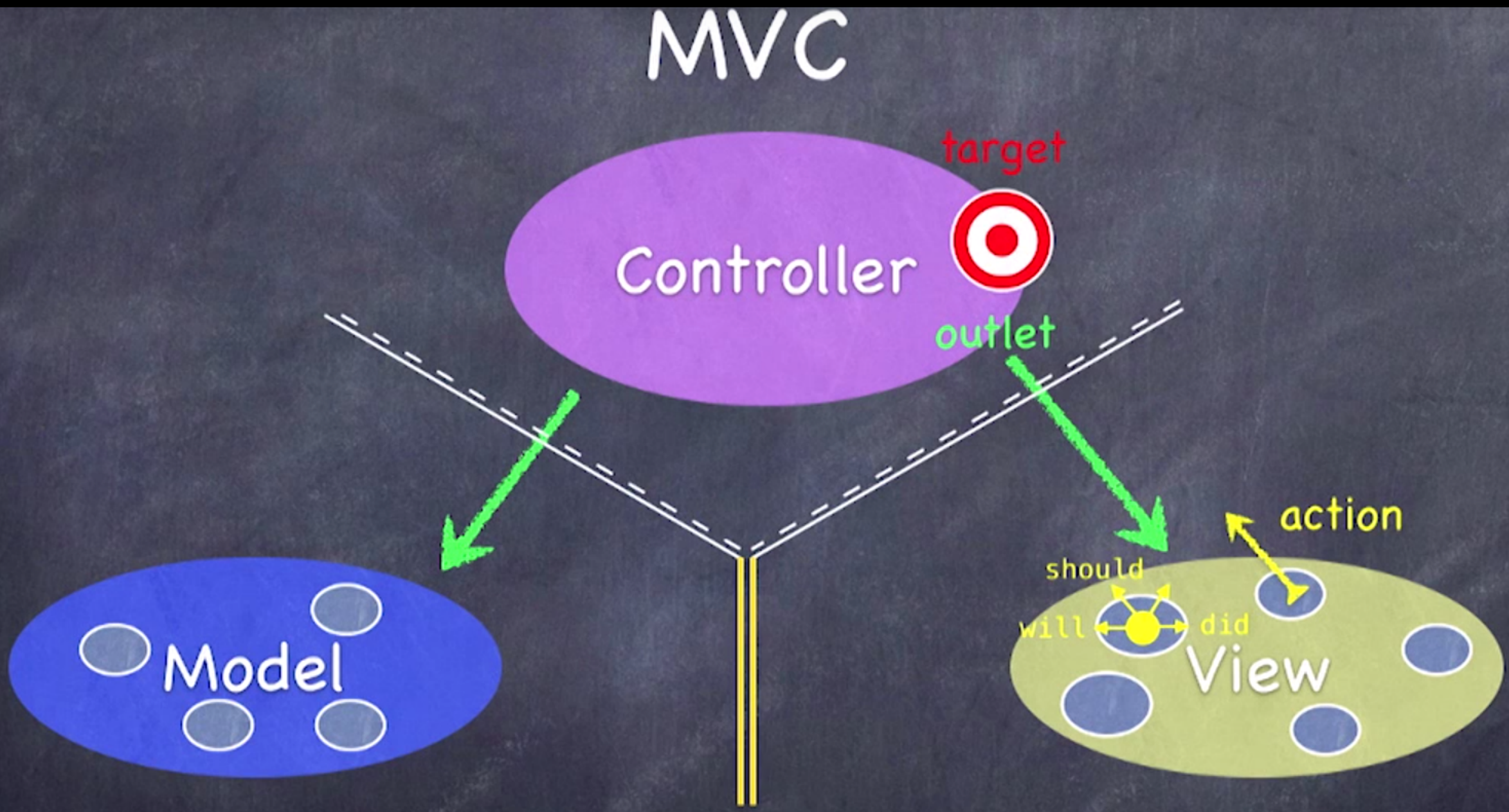
Then hand out an action to the View.

MVC



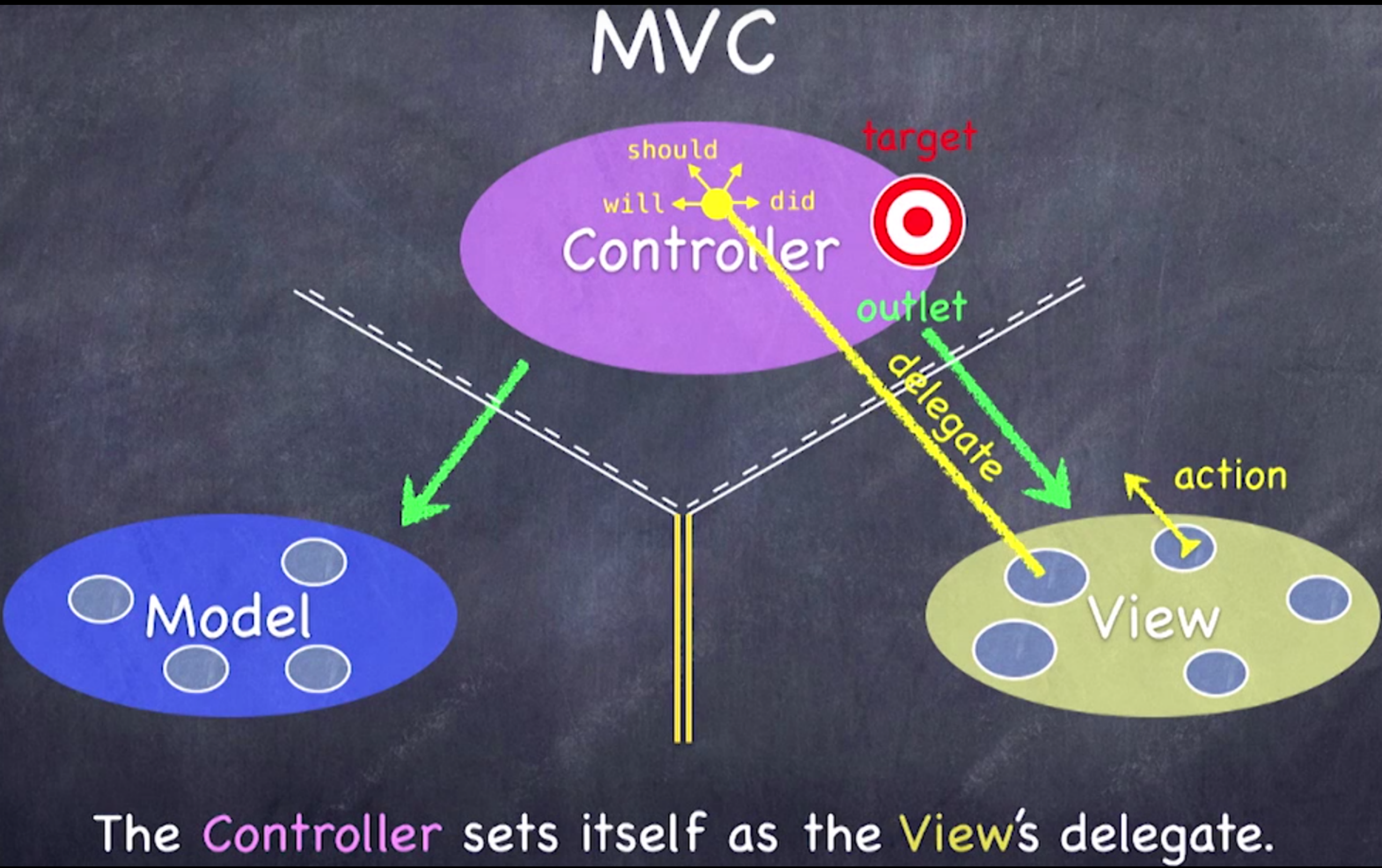
The View sends the action when things happen in the UI.

MVC



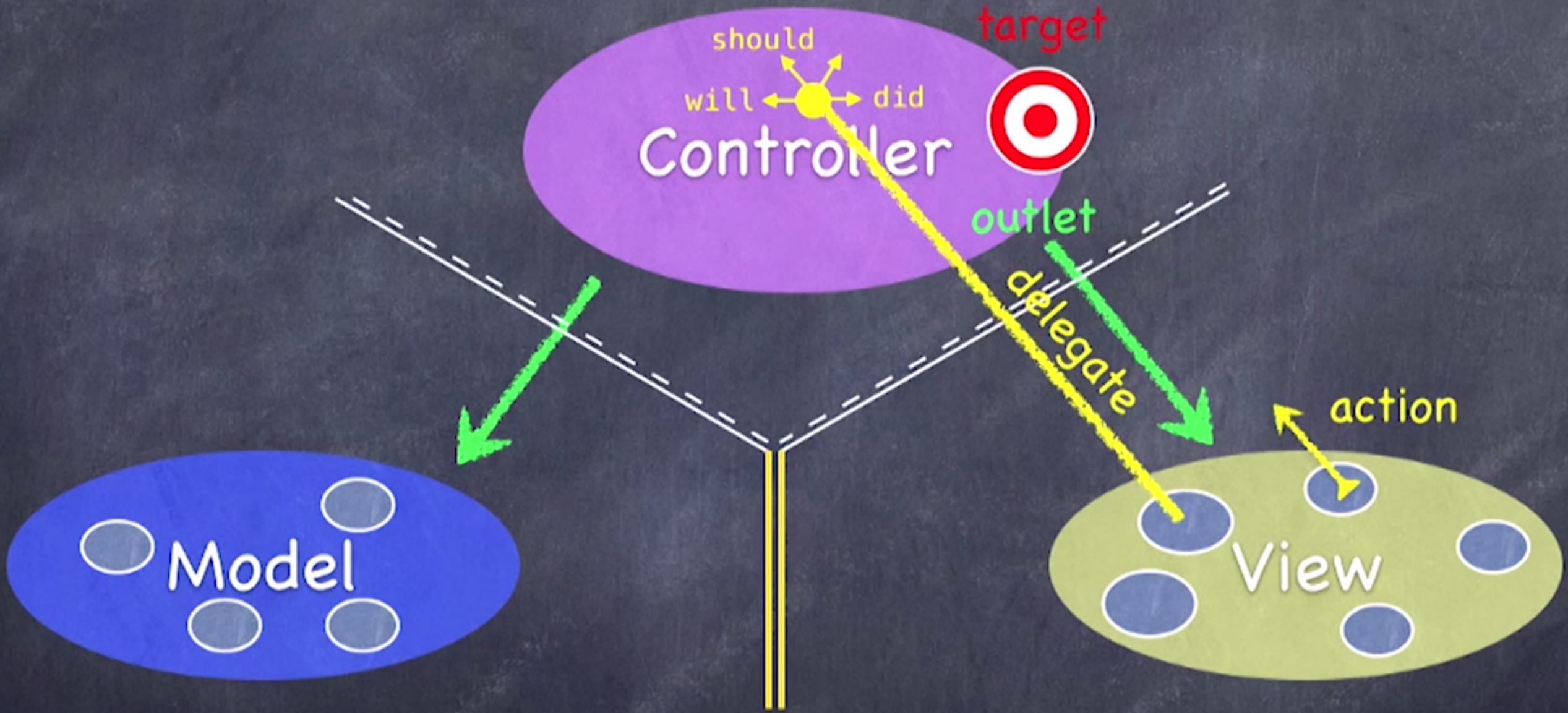
Sometimes the View needs to synchronize with the Controller.

MVC



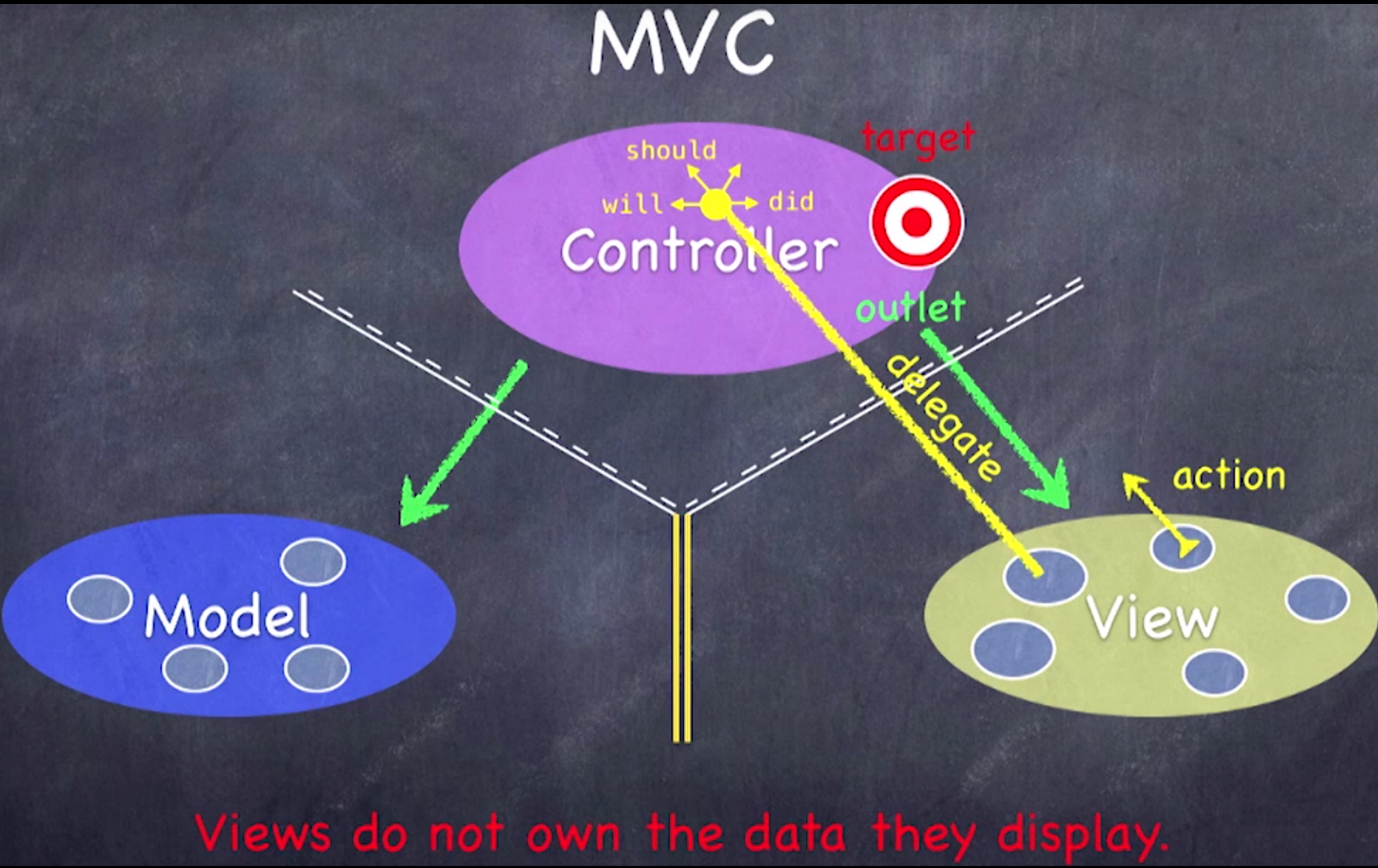
The Controller sets itself as the View's delegate.

MVC



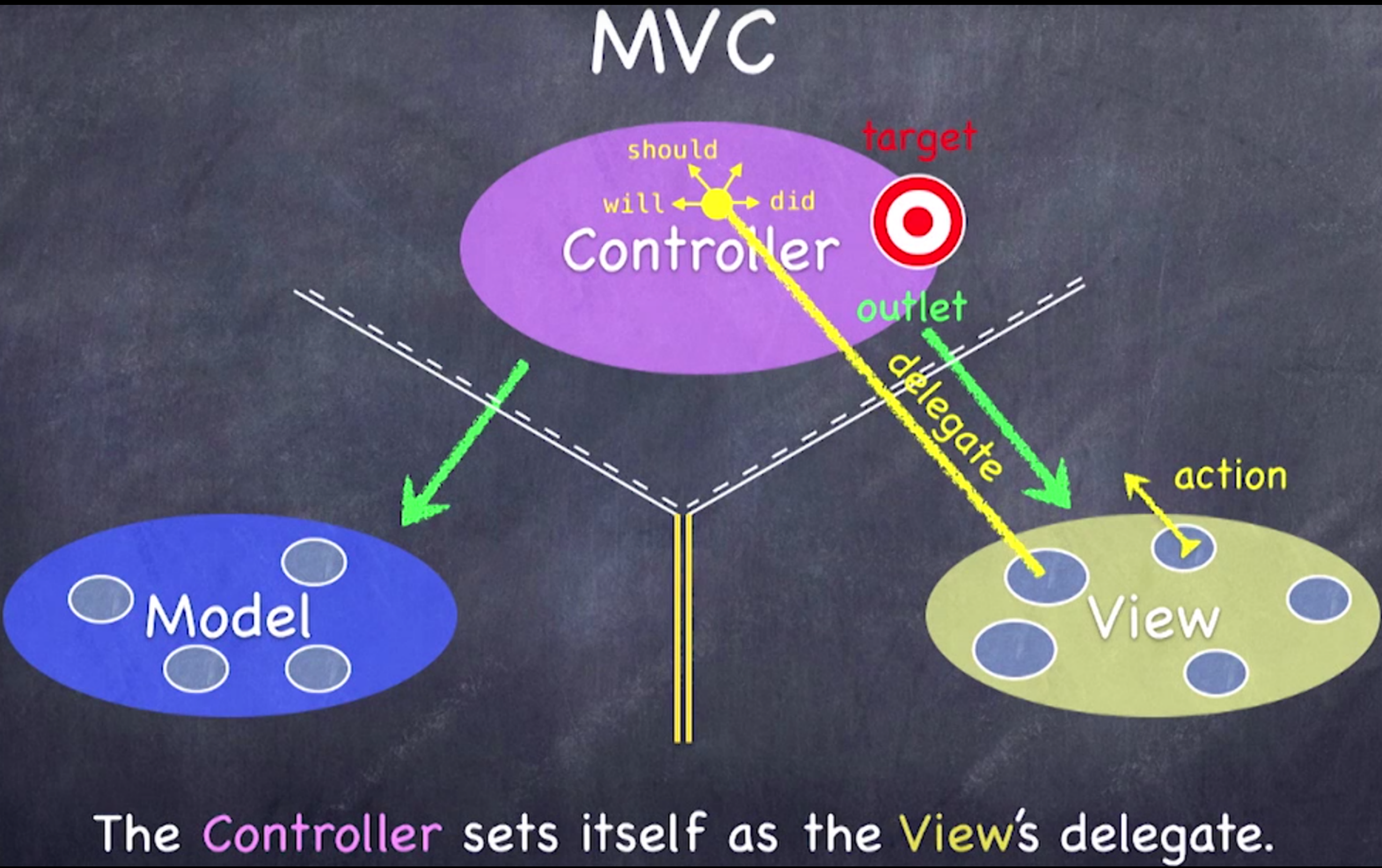
The delegate is set via a protocol (i.e. it's "blind" to class).

MVC



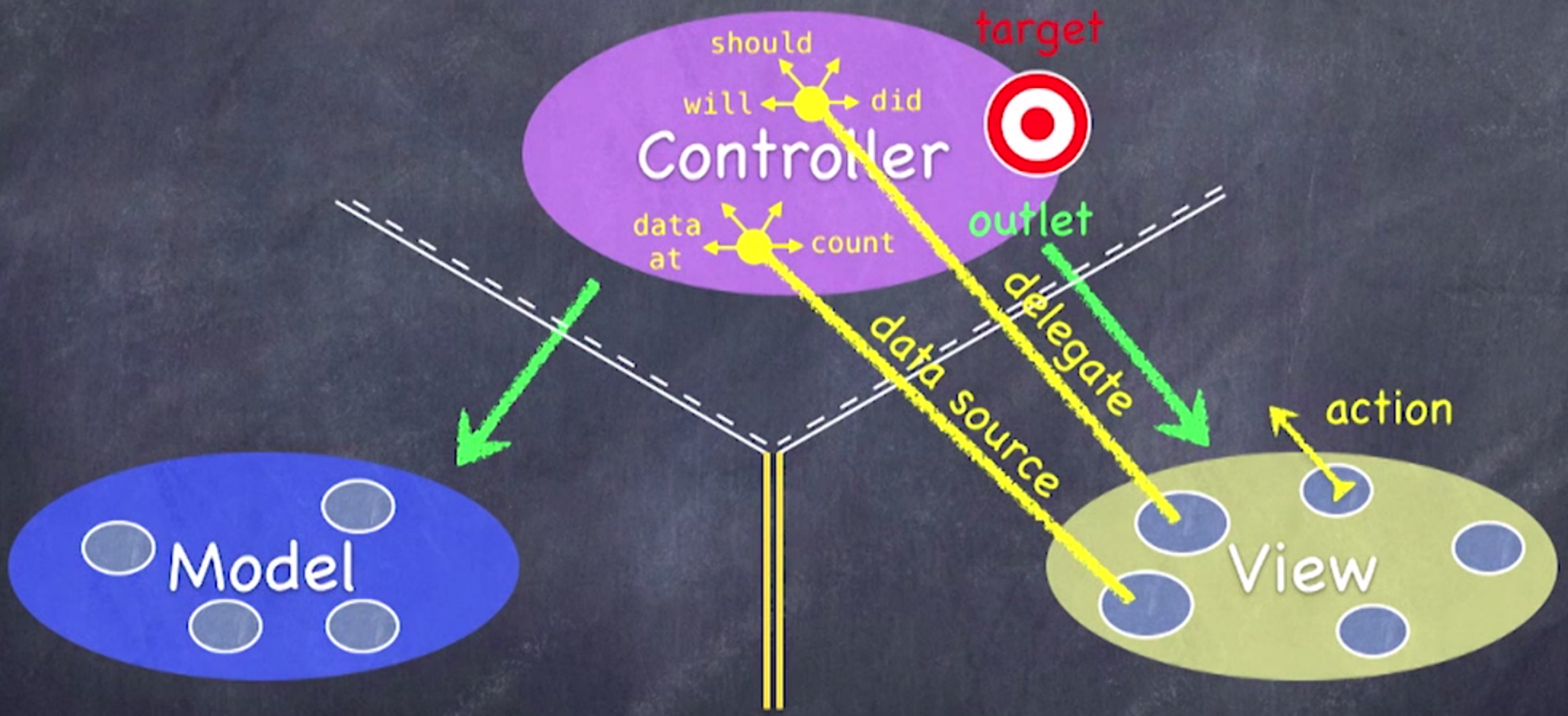
Views do not own the data they display.

MVC



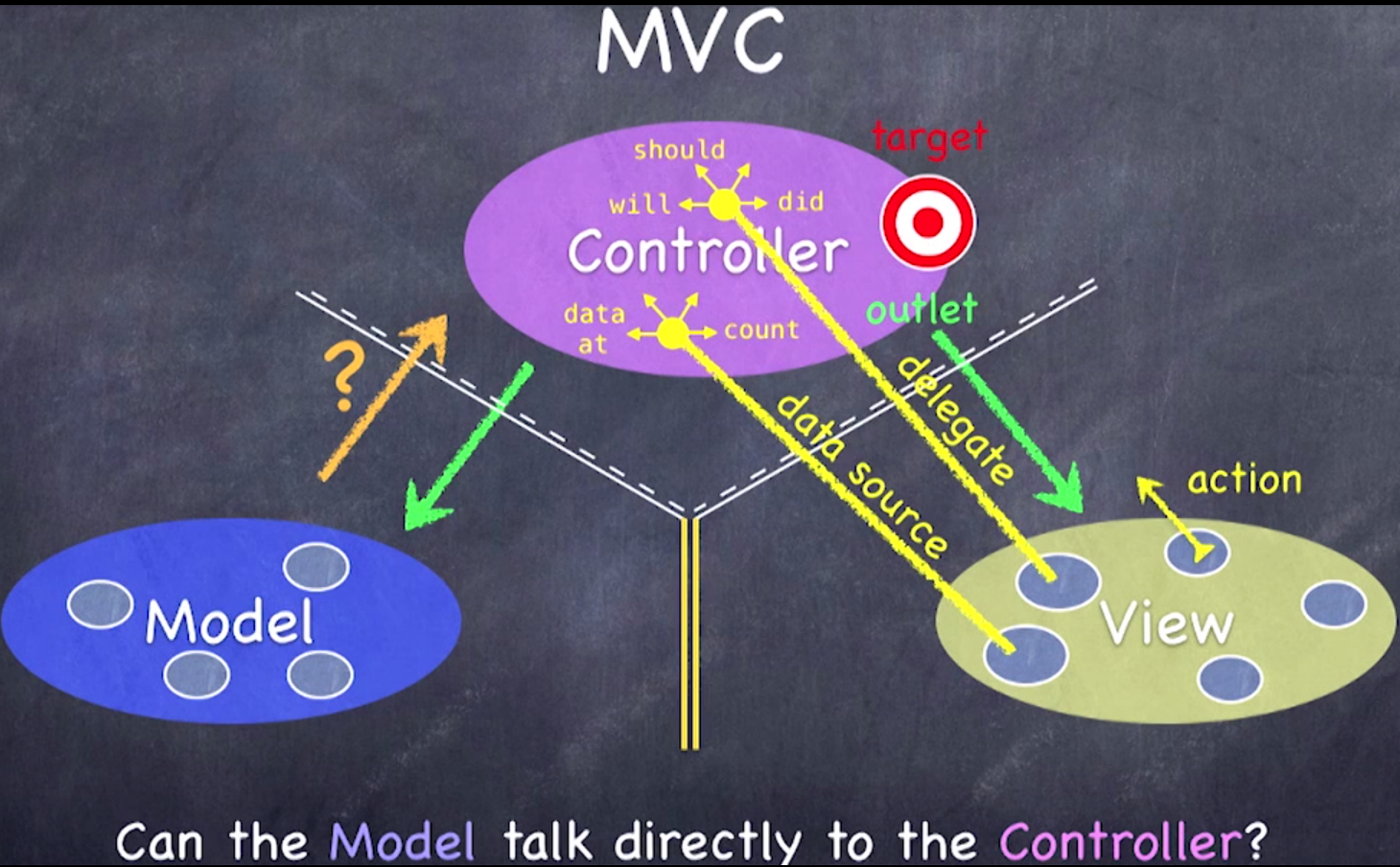
The Controller sets itself as the View's delegate.

MVC



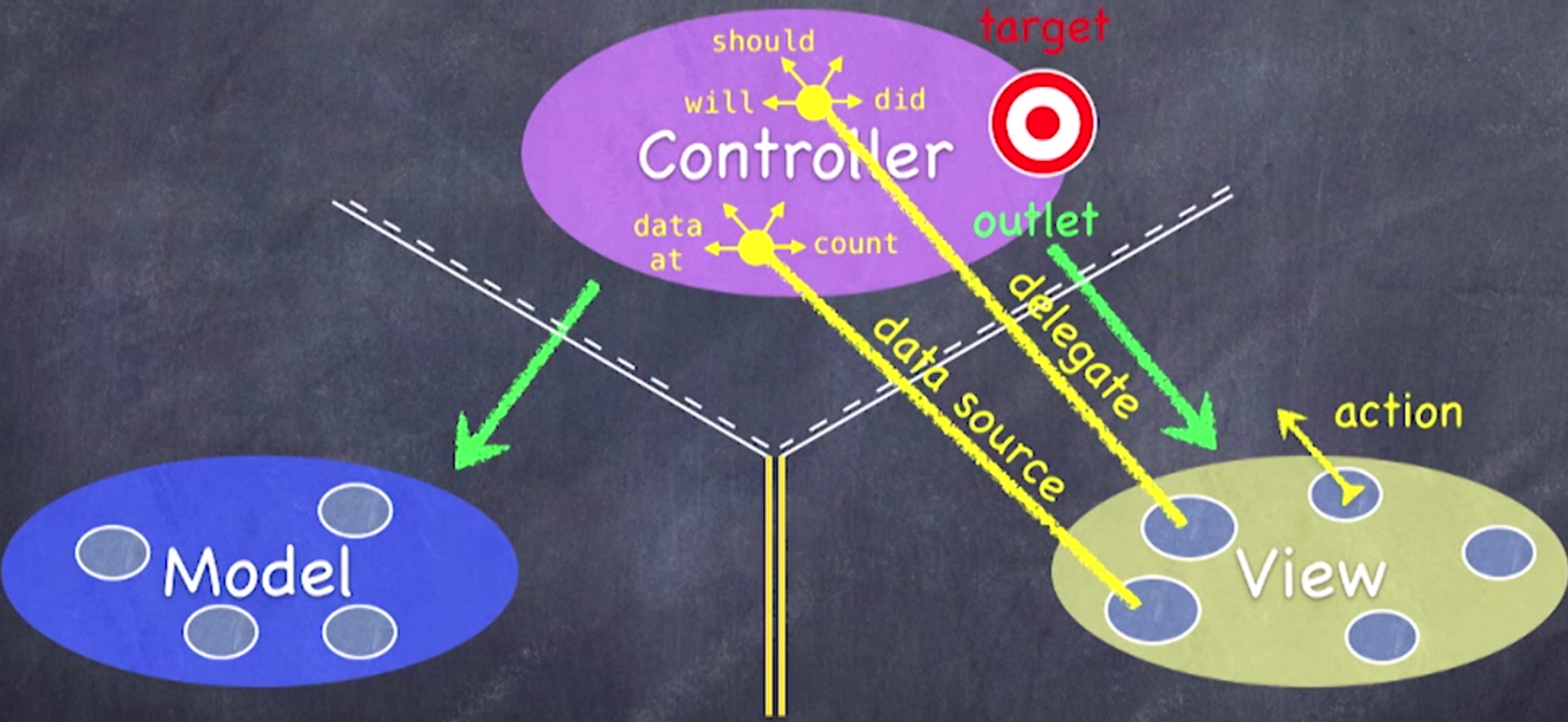
Controllers interpret/format Model information for the View.

MVC



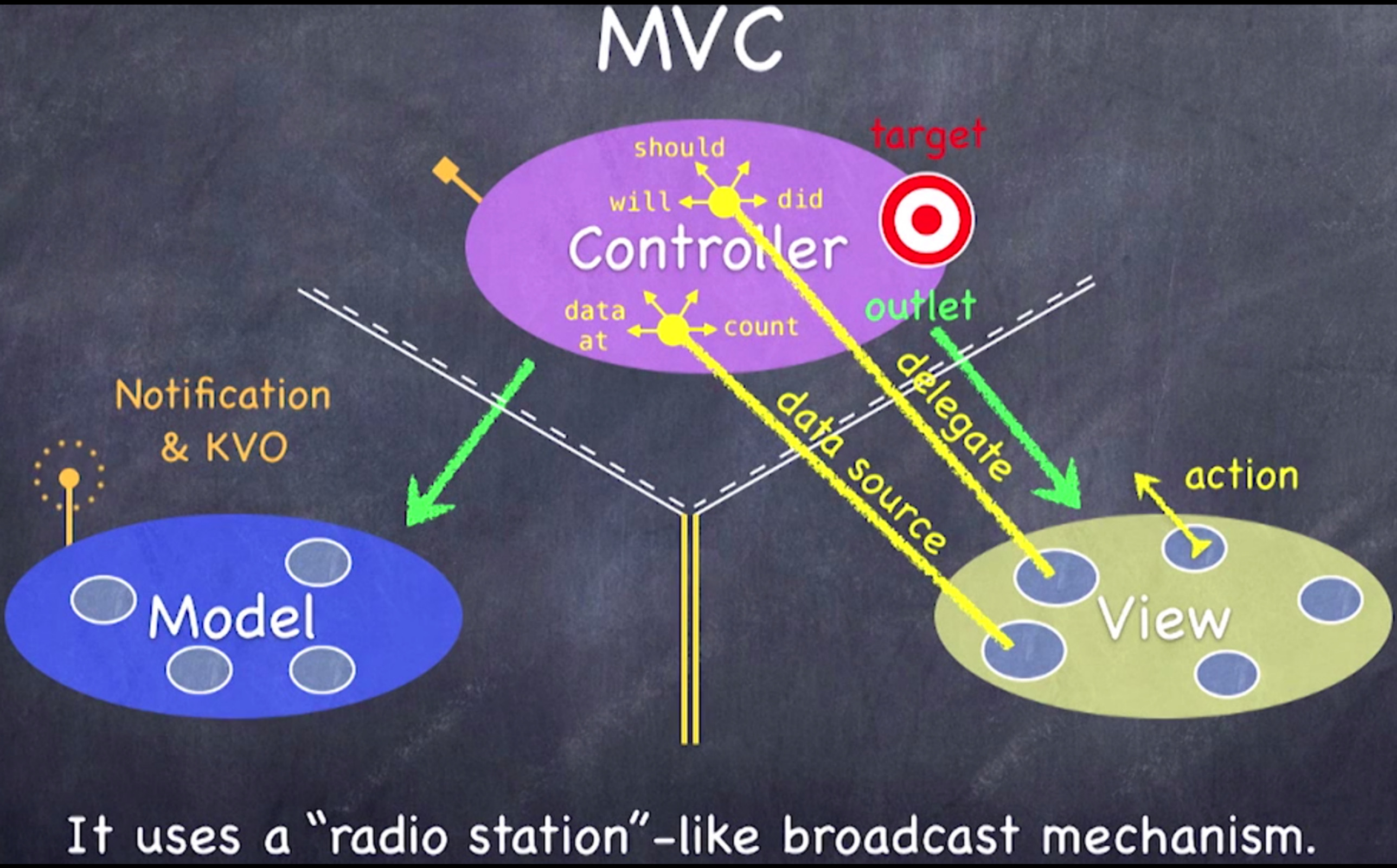
Can the Model talk directly to the Controller?

MVC



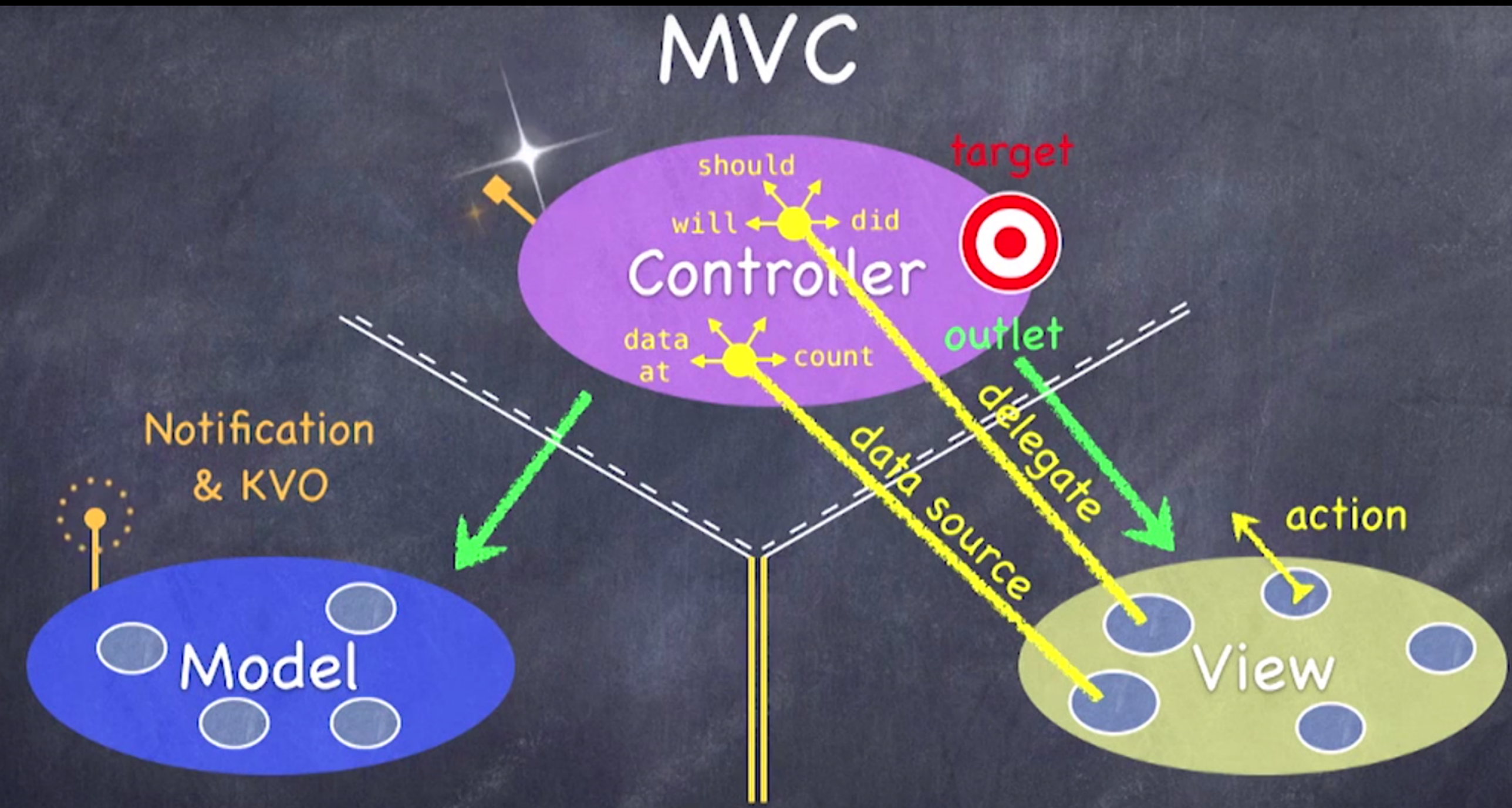
No. The Model is (should be) UI independent.

MVC



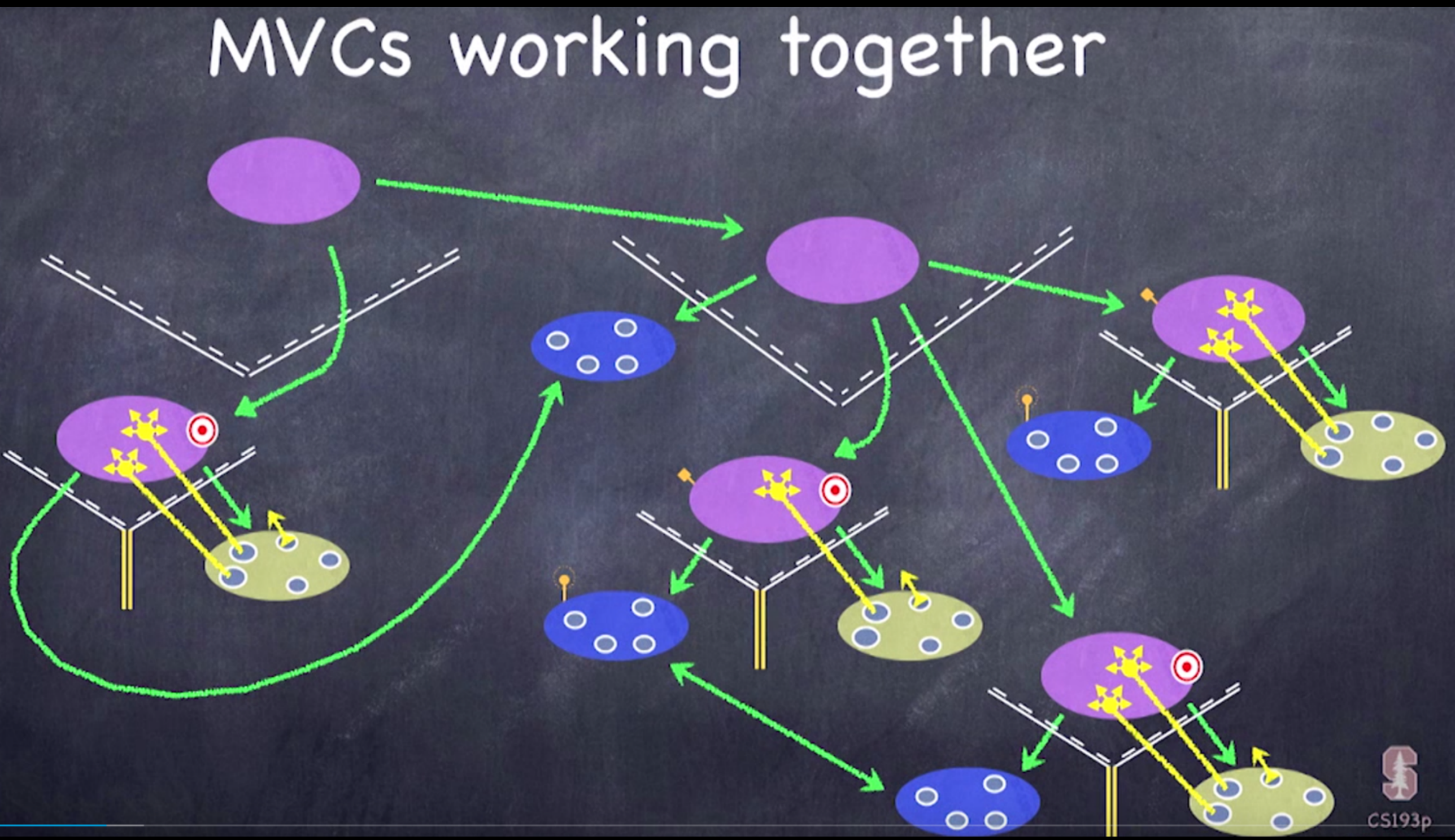
It uses a "radio station"-like broadcast mechanism.

MVC

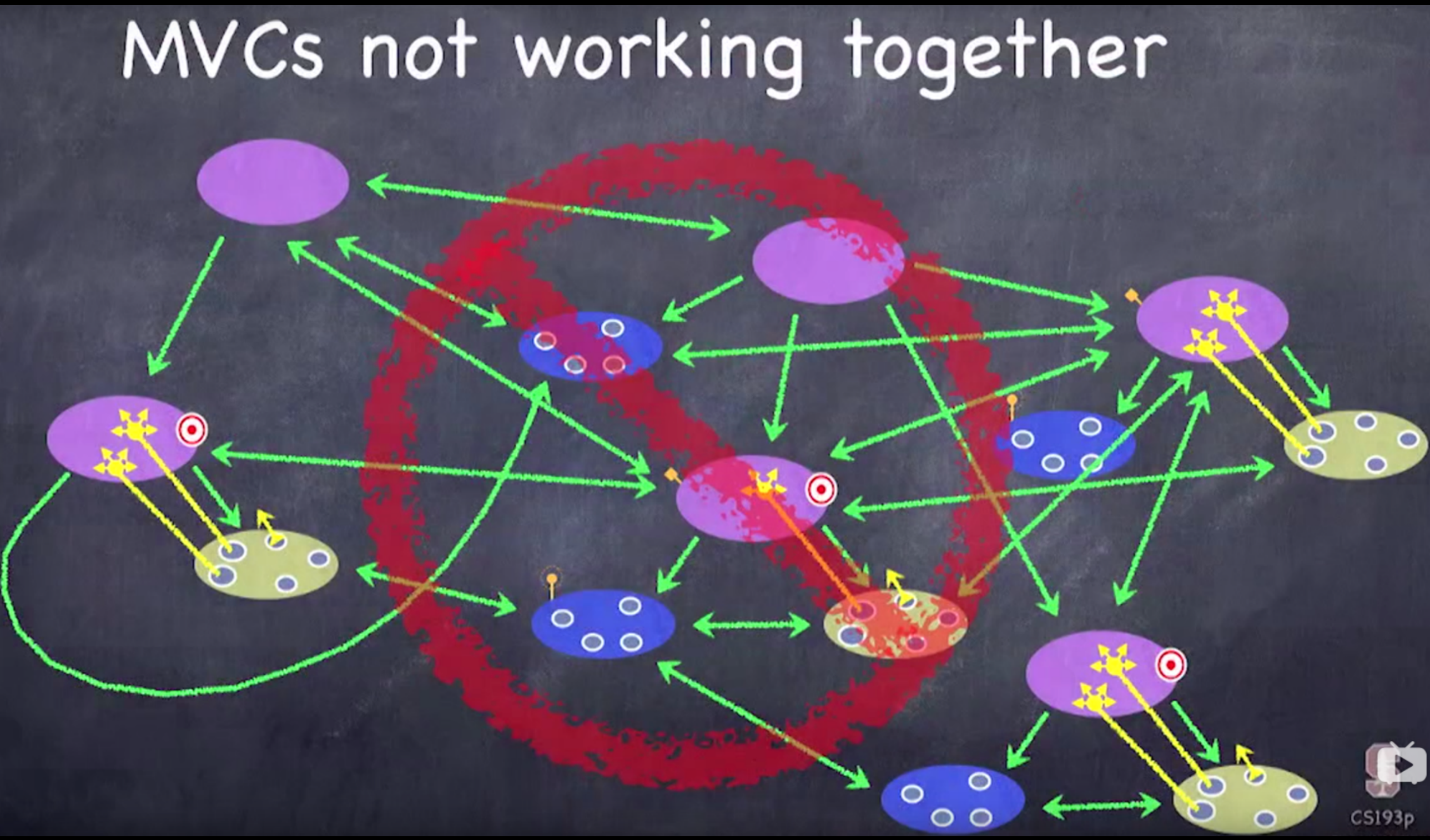


Controllers (or other Model) "tune in" to interesting stuff.

MVCs working together



MVCs not working together



2. CocoaPods

(1). What

CocoaPods 是一个负责管理 iOS 项目中第三方开源库的工具。CocoaPods 的项目源码在 GitHub（ https://github.com/CocoaPods ）上管理。

开发 iOS 项目不可避免地要使用第三方开源库，在使用第三方库时，除了需要导入源码，集成这些依赖库还需要我们手动去配置，还有当这些第三方库发生了更新时，还需要手动去更新项目，这就显得非常麻烦。

而 CocoaPods 的出现使得我们可以节省设置和更新第三方开源库的时间，通过 CocoaPods，我们可以将第三方的依赖库统一管理起来，配置和更新只需要通过简单的几行命令即可完成。

(2). Why

在使用 CocoaPods 之前，开发项目需要用到第三方开源库的时候，我们需要：

1. 把开源库的源代码复制到项目中

2. 添加一些依赖框架和动态库

3. 管理它们的更新

在使用 CocoaPods 之后，我们只需要把用到的开源库放到一个名为 Podfile 的文件中，然后执行 pod update 就可以了，CocoaPods 就会自动将这些第三方开源库的源码下载下来，并且为我们的工程设置好相应的系统依赖和编译参数。

(3). How

Let’s try!

Q&A