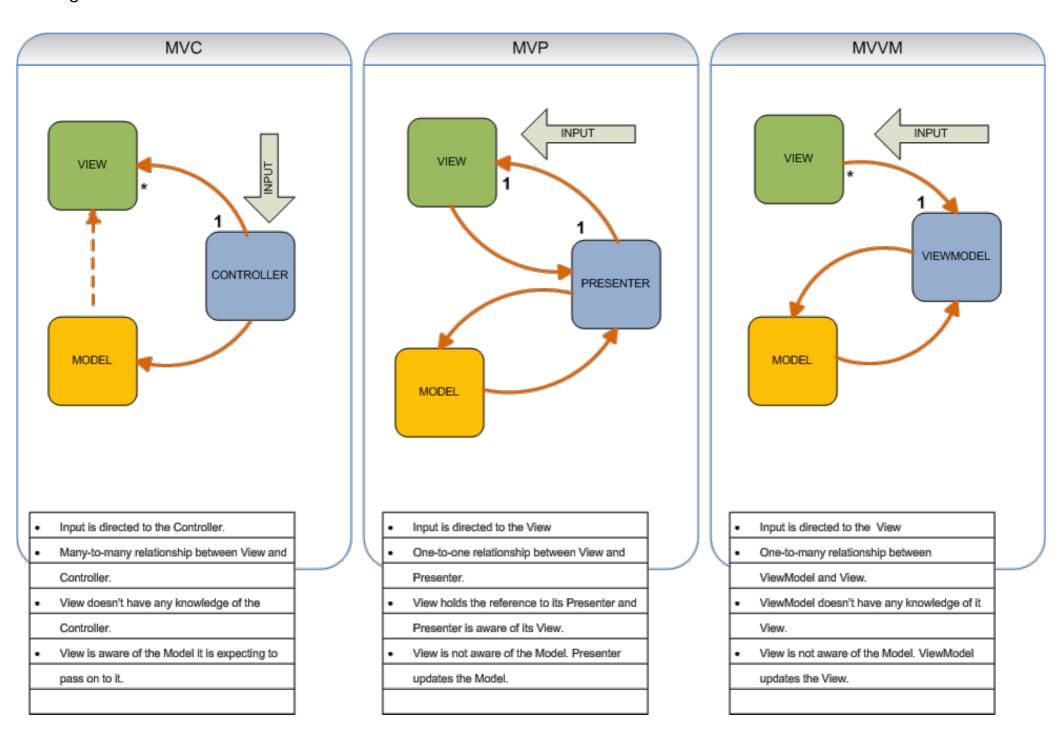
## MVC, MVP, MVVM

- 1. 위 디자인 패턴을 자신이 이해한 대로 서술하고 왜 이렇게 다양한 패턴이 탄생했을까 의견을 제시
- MVC : Model View Controller의 약자로 에플리케이션을 세가지의 역할로 구분한 개발 방법론이다.

사용자가 Controller를 조작하면 Controller는 Model을 통해서 데이터를 가져오고 그 정보를 바탕으로 시각적인 표현을 담당하는 View를 제어해서 사용자에게 전달하게 된다.

- MVP: MVC 패턴과 다르게 입력이 View에서 처리된다. Presenter는 View의 인스턴스를 갖고 있으며 View와 1대1 관계이고, 그에 해당하는 Model의 인스턴스 또한 갖고 있기때문에 View와 Model 사이에서 다리와 같은 역할을 한다. View에서 이벤트가 발생하면 Presenter에게 전달해주고 Presenter는 해당 이벤트에 따른 Model을 조작하고 그 결과를 바인딩을 통해 View에게 통보를 하여 View를 업데이트 시켜준다.
- MVVM: ViewModel 뷰모델 말그대로 View를 나타내주기 위한 Model이라고 생각하면 된다. View보다는 Model과 유사하게 디자인 되며, View의 바인딩 될 때 가장 강력하다. MVP와 같이 View에서 입력이 처리된다.

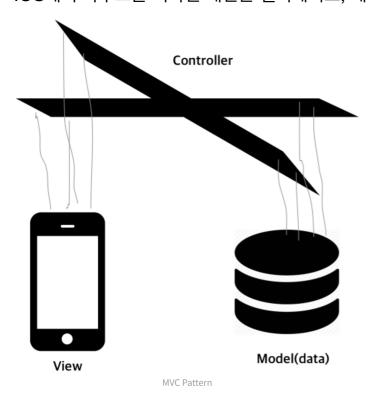
Command를 통하여 Behavior를 View의 특정한 ViewAction(Event)와 연결할 수 있으며, ViewModel의 속성과 특정 View의 속성을 Binding 시켜 줌으로써 ViewModel 속성이 변경 될때마다 View를 업데이트 시켜줄 수 있다.



- 2. 각 디자인 패턴의 장단점을 분석하고 자신은 어떤 패턴을 사용할 때 이상적일지 의견을 제시
- MVC : 장점은 동일한 model로 부터 view들을 표시할수있고, view의 동기화, controller를 플러그처럼 장착할수있다. 프레임워크로 확장할수있다. 단점으로는 복잡도가 증가하고, 최신 사용자 인터페이스툴에 mvc를 적용하기 어렵고, 너무 많은 업데이트들이 동 시에 진행될 수 있다.
- MVP : MVC 패턴과는 다르게 Presenter를 통해 Model과 View를 완벽히 분리해 주기 때문에 View는 Model을 따로 알고 있지 않아도 된다는 장점이 있다. 단점으로는 View와 1대1 관계이기 때문에 View와의 의존성이 매우 강하다.
- MVVM : 가장 큰 장점이라 함은 Command와 Data Binding으로 MVP 패턴과 달리 View와의 의존성을 완벽히 분리 할 수 있다는 장점이 있다.

개인적인 생각으로 MVC패턴이 제일 동선이 간편하고, 하나의 컨트롤러에서 다수의 뷰를 제어할수있기 때문에 그리고 나머지 패턴은 복잡하게 느껴진다.

- 3. 실제로 애플이 사용하고 있는 MVC 패턴은 어떤 패턴에 가장 가까울까? 자신의 생각과 그 이유를 제시
- IOS에서 자주쓰는 디자인 패턴은 델리게이트, 데이터소스, MVC, Target-Action Pair



- MVC는 Cocoa 애플리케이션을위한 뛰어난 설계의 중심입니다. 이 패턴을 사용하는 장점은 많이 있습니다. 이러한 용도의 많은 개체가 더 재사용 가능하다 경향이 있으며, 그 인터페이스는 더 나은 정의되는 경향이 있습니다. MVC 디자인이있다 어플리케이션도 쉽게 확장 다른 응용 프로그램보다. 또한 많은 Cocoa 기술과 아키텍처는 MVC를 기반으로하며, 사용자 정의 개체는 MVC 역할 중 수행하는 것이 필요합니다.
- 코코아 터치는\_아이폰 및 아이팟 터치의 소프트웨어 계층 중 가장 상위 계층이다. 개발자로 하여금, 더 아래 계층을 자세히 알 수고를 덜어준다. 코어 오에스(Core OS) 계층, 코어 서비시즈(Core Services) 계층, 미디어(Media) 계층은\_OSX에도 있으나, 코코아 터치 계층은 아이폰 및 아이팟 터치를 위해 아이폰OS에 변형되어 새로이 도입되었다.