The X Course: Android

Session 5

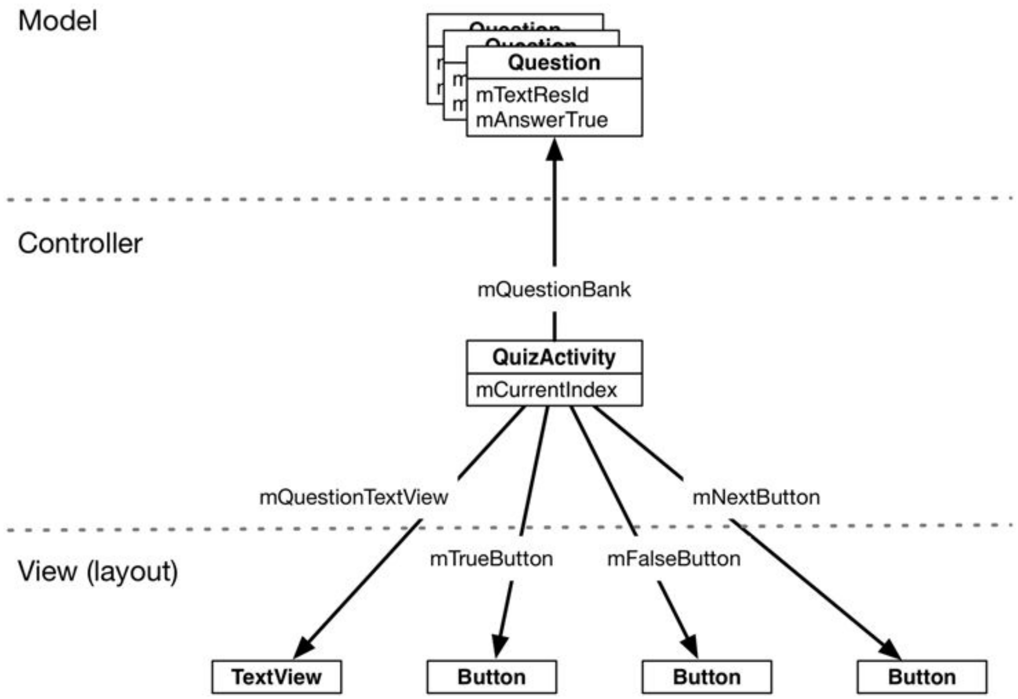
Agenda

- Android and Model-View-Controller Architecture.
- Applying MVC to GeoQuiz’s QuizActivity to have a **Multi-Question GeoQuiz.**

Android and Model-View-Controller

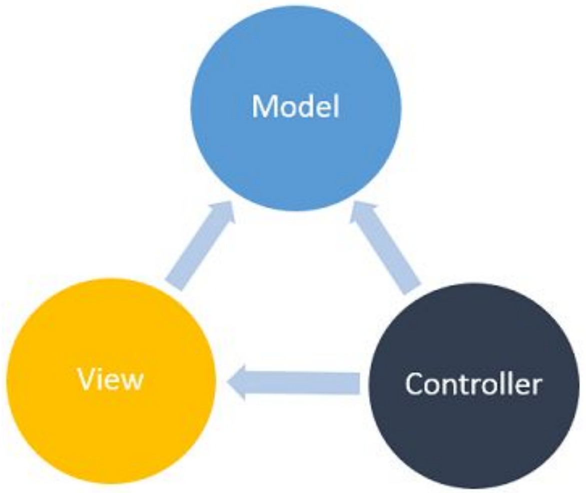
- **<u>Goal</u>**: Upgrade GeoQuiz to present more than one question.
- **<u>Steps:</u>**
    - Add a Question class (Model).
    - Create an array of Question objects for QuizActvity (Controller) to manage.
    - It will then interact with the TextView and the Buttons to display questions and provide feedback.

Android and Model-View-Controller



Android and Model-View-Controller

- Android applications may be designed around an architecture called Model-View-Controller.
- In MVC, all objects in your application must be a model object, a view object, or a controller object.
- This layers provide abstraction, where each layer is responsible for a part of the logic.



The Model Layer

- A **model** object holds the application’s data and “business logic.”
- **Model** classes are typically designed to model the things your app is concerned with.
- **Model** objects have no knowledge of the user interface; their sole purpose is holding and managing data.
- In Android applications, **model** classes are generally custom classes you create.

The Model layer: Question Class

- All of the model objects in your application compose its model layer.
- GeoQuiz’s model layer consists of the **Question class.**
- The **Question class** holds two pieces of data: the question text and the question answer (true or false).
- These variables need getter and setter methods.

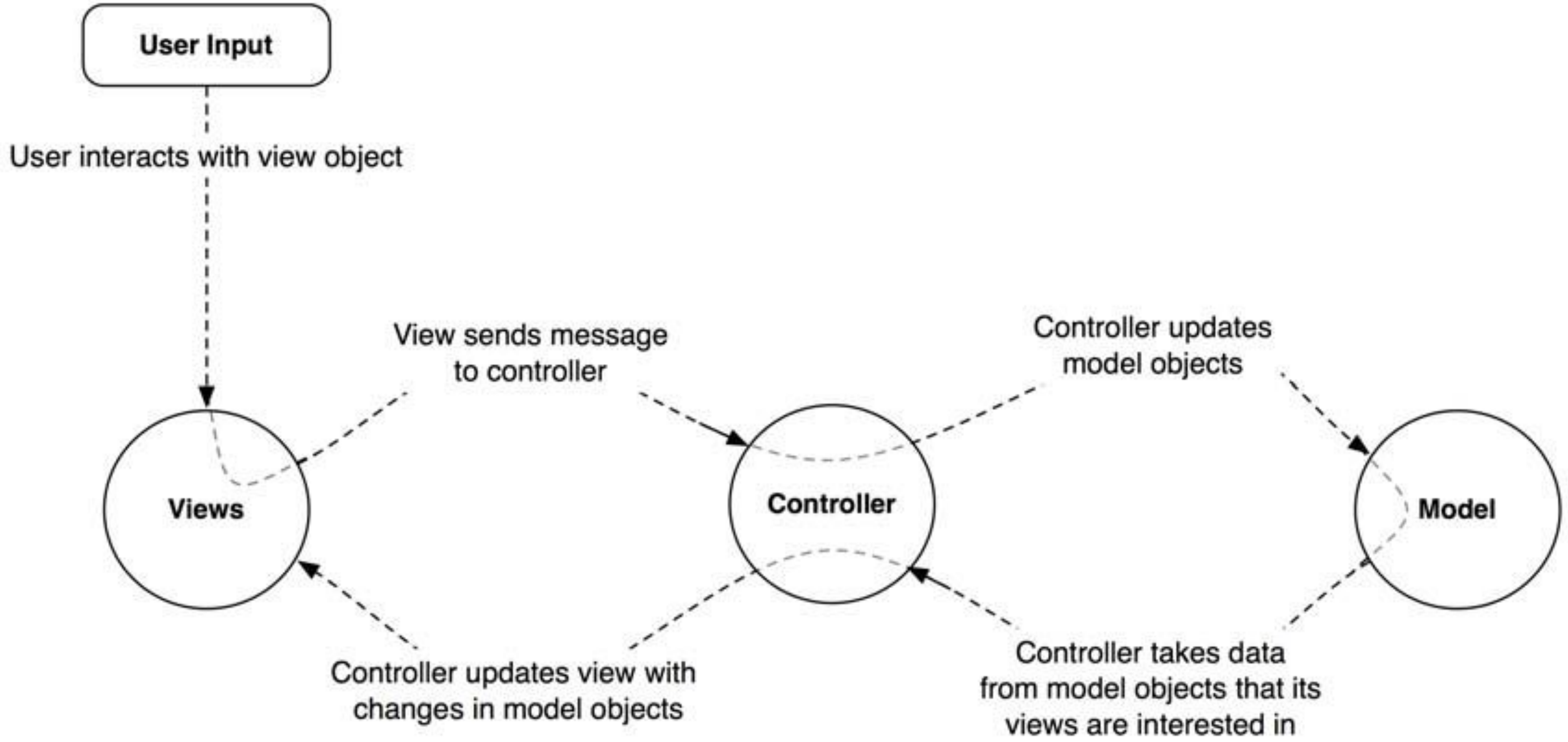
The View Layer

- **View** objects know how to draw themselves on the screen and how to respond to user input, like touches.
- *Rule of thumb:* if you can see it on screen, then it is a **view.**
- An application’s **view** objects make up its view layer.
- GeoQuiz’s **view layer** consists of the widgets that are inflated from activity_quiz.xml and activity_cheat.xml

The Controller Layer

- **Controller** objects tie the view and model objects together. They contain “application logic.”
- **Controllers** respond to various events triggered by view objects
- **Controllers** manage the flow of data to and from model objects and the view layer.
- GeoQuiz’s controller layer, at present, consists of **QuizActivity** .

MVC Flow Example


User Input
User interacts with view object
View sends message to controller
Controller updates model objects
Views
Controller
Model
Controller updates view with changes in model objects
Controller takes data from model objects that its views are interested in

But Why MVC ? Separation

- **<u>Separating code</u>** into classes helps you design and understand the application as a whole
- You can think in terms of classes instead of individual variables and methods.
- **<u>Separating classes</u>** into model, view, and controller layers helps you design and understand an application; you can think in terms of layers instead of individual classes.

But Why MVC ? Reduce Complication

- An application can accumulate features until it is too complicated to understand.
- Although GeoQuiz is not a complicated app, you can still see the benefits of keeping layers separate.

But Why MVC ? Reusability

- Classes have restricted responsibilities.
- For instance, your model class, Question , knows nothing about the widgets used to display a true-false question.
- This makes it easy to use Question throughout your app for different purposes. For example, if you wanted to display a list of all the questions at once, you could use the same object that you use here to display just one question at a time.

Let’s add MVC Architecture to GeoQuiz

- Create Question Class. (Model)
- Add Next Button in Layout activity_quiz.xml (View)
- Update the QuizActivity to have an array of Questions and handle them appropriately.

MVC Architecture: Further Readings

- https://medium.com/upday-devs/android-architecture-patterns-part-1-model-view-controller-3baecef5f2b6
- https://openclassrooms.com/en/courses/4661936-develop-your-first-android-application/4679186-learn-the-model-view-controller-pattern
- https://androvaid.com/android-mvc-example/