Flight Control Android Application

SUBMITTED BY:

YANA SIDANYCH

ITAY YAAKOV

About The Project

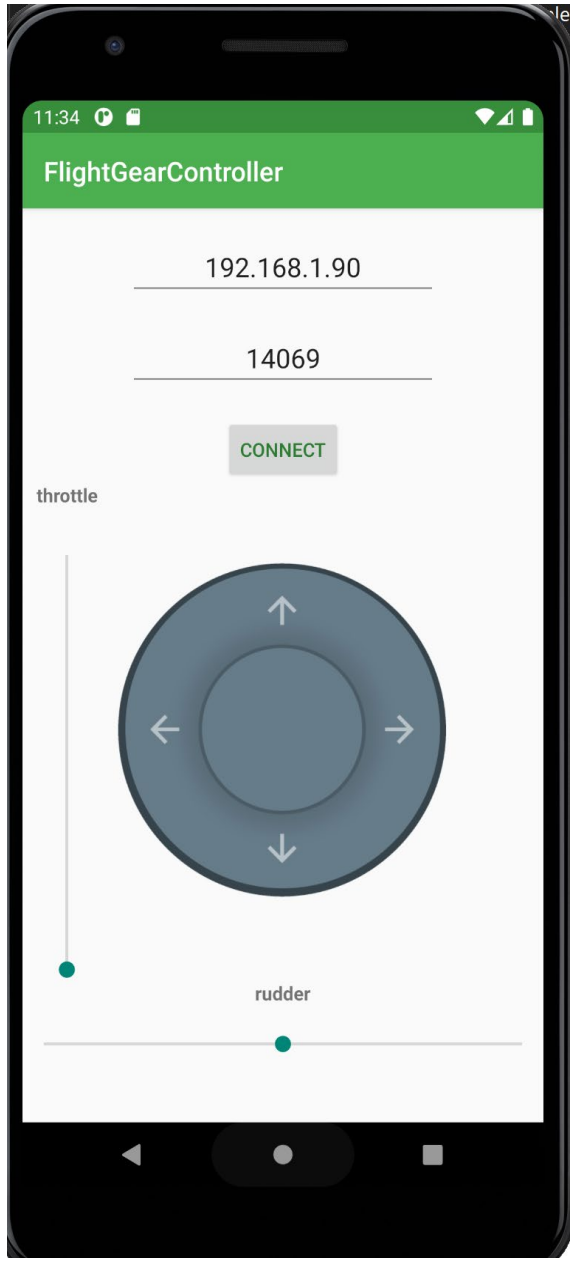
- We created a joystick app that allows the use to control a plane inside the flight gear simulator running on a different machine.
- The app was written for android devices using Kotlin.
- The app was created as a part of our Advanced Programming course:

  Course number: 89211

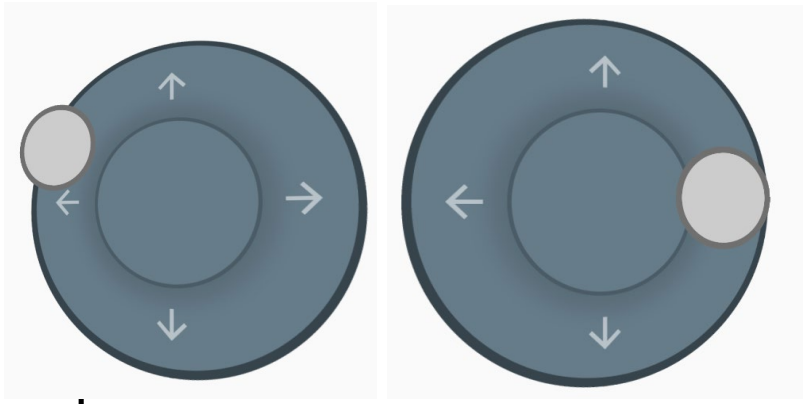
  Lecturer: Eliyahu Khalastchi

Our User Interface

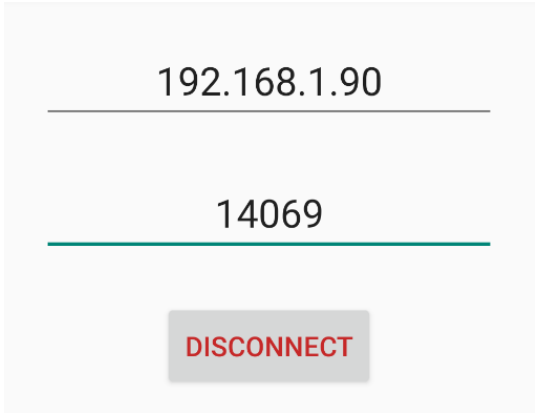
- Single screen – with 2 parts
  - **connection section** - let the user set flight gear connection settings and a button to connect/disconnect.
  - Joystick and sliders – control the plane

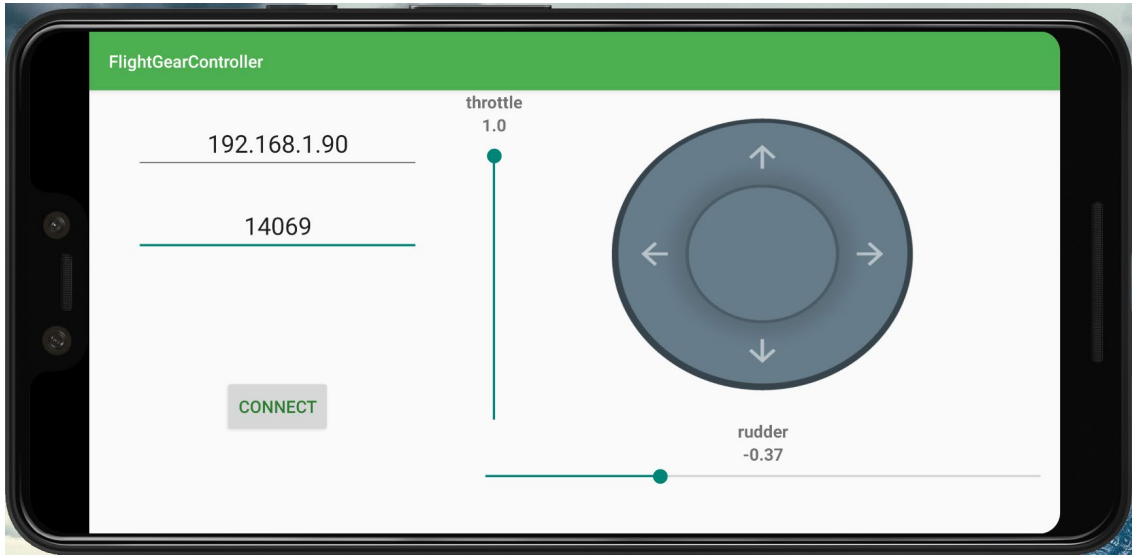


Nice to have features

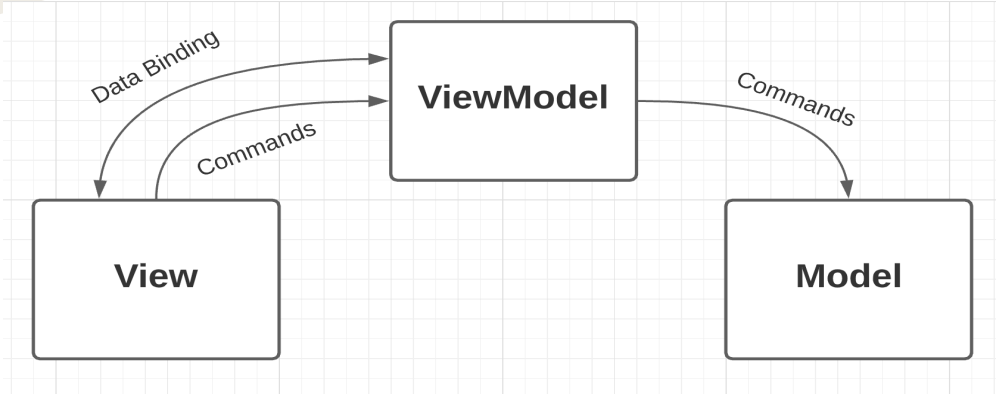


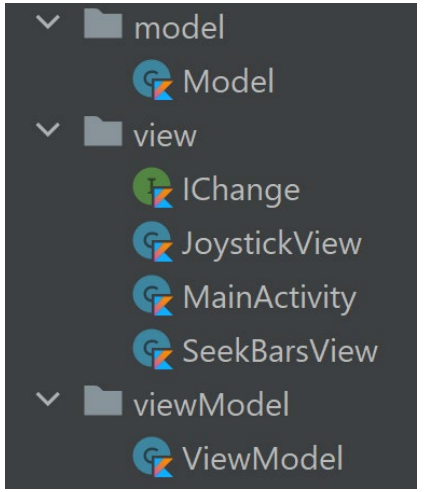
- 3D joystick
- Connect and disconnect (can change ip and port any time)
- Landscape support





The MVVM Architecture





- The code is organized in three packages: model, view and viewModel.
- **View** - contains the three main views: joystick, throttle and rudder sliders and mainActivity. They are responsible for showing and managing all the GUI components. We use binding between the ui and the variables.
- **model** - is responsible for connecting to the simulator and sending the flight commands
- **viewModel** – get notify of events and activate the model methods.

The MVVM Architecture – Code Examples

- Let's demonstrate the data binding between the ip and port GUI EditText elements in the view, to the ip and port variables in the viewModel:

```
android:text="@={viewModel.ip}"
```

```
android:text="@={viewModel.port}"
```

From: activity_main.xml

- In these two lines of code, you can see how the text atrribute of the GUI EditText elements is bound to the ip and port variables that exists in the viewModel.kt file.
- viewModel and model are defined in **activity_main.xml** for data binding:

```
<variable
    name="model"
    type="com.example.flightgearcontroller.model.Model" />

<variable
    name="viewModel"
    type="com.example.flightgearcontroller.viewModel.ViewModel" />
```

- Here the button text and color are changed due to the **binding of isConnected** var from model.kt

```
android:text="@{model.isConnected() ? @string/button_text_off : @string/button_text_on}"
android:textColor="@{model.isConnected() ? @color/md_red_800 : @color/md_green_800}"
```

The MVVM Architecture – Code Examples

- mainActivity set the settings for Joystick and SeekBar views.
- Each of this views contains an object from type IChange interface.
  In this interface there is only one function, **onChange**.
  This is strategy design pattern
- Every view has a public variable **sendUpdateEvent**, and the view calls the onChange method which isn't implmeneted by the view.
- When these two GUI components are created inside the mainActivity.kt file, the onChange function is implemented and is set to trigger the viewModel sendCommands function.
- Now the lambda function in the mainActivity activates the send command method on the viewModel.

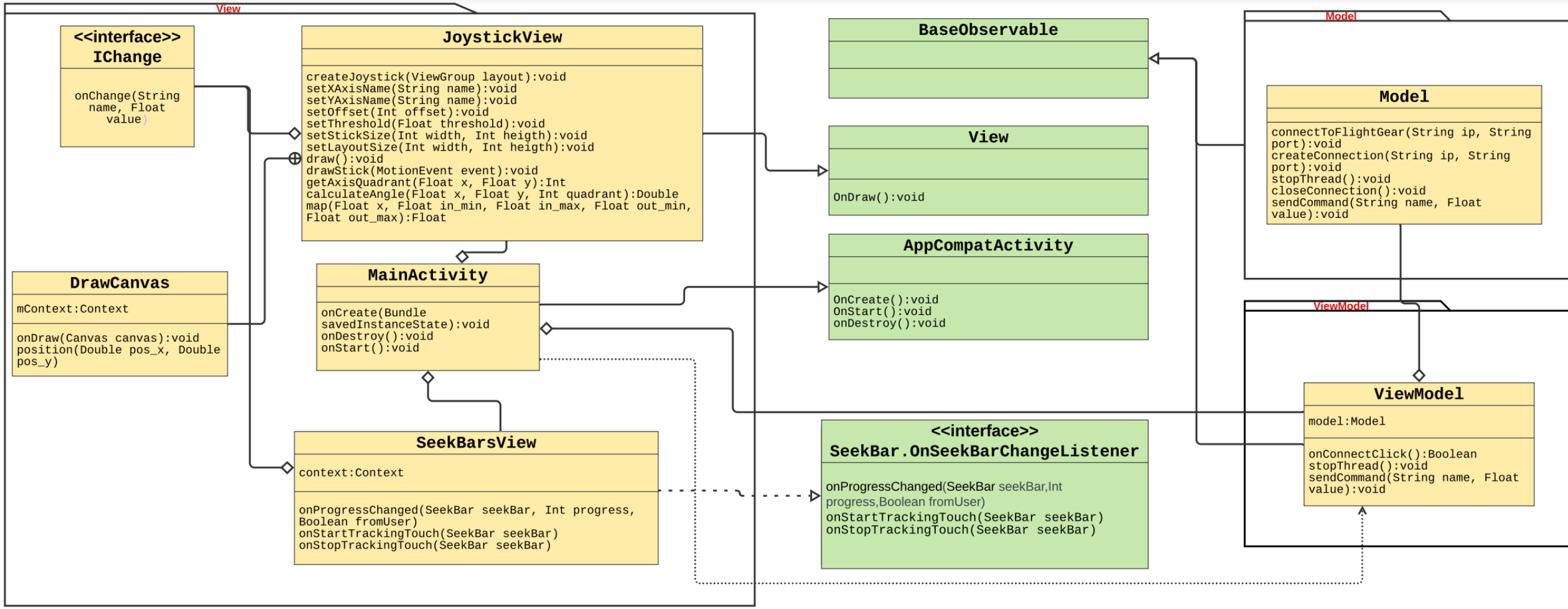
```
public var sendUpdateEvent: IChange? = null
```

```
// Interface for strategy design pattern
fun interface IChange {
    fun onChange(name: String, value: Float)
}
```

```
// create sliders object
seekBars = SeekBarsView(this)
// listen to values change of sliders
seekBars!!.sendUpdateEvent = IChange { name, value ->
    viewModel!!.sendCommand(name, value)
}
```

```
when (seekBar) {
    throttleSeekBar -> sendUpdateEvent?.onChange("throttle", progressF)
    rudderSeekBar -> sendUpdateEvent?.onChange("rudder", progressF)
}
```

UML Class Diagram



Clarification:

Green Class/Interface - Kotlin provided class

Dotted line - data binding

Thank you for listening :)

ENJOY THE FLIGHT !