Project Phase -1
Team - 3

Presented By:

Lakshay Santosh Kucheriya
Nathan Pfau
Yutian Qin

Demo

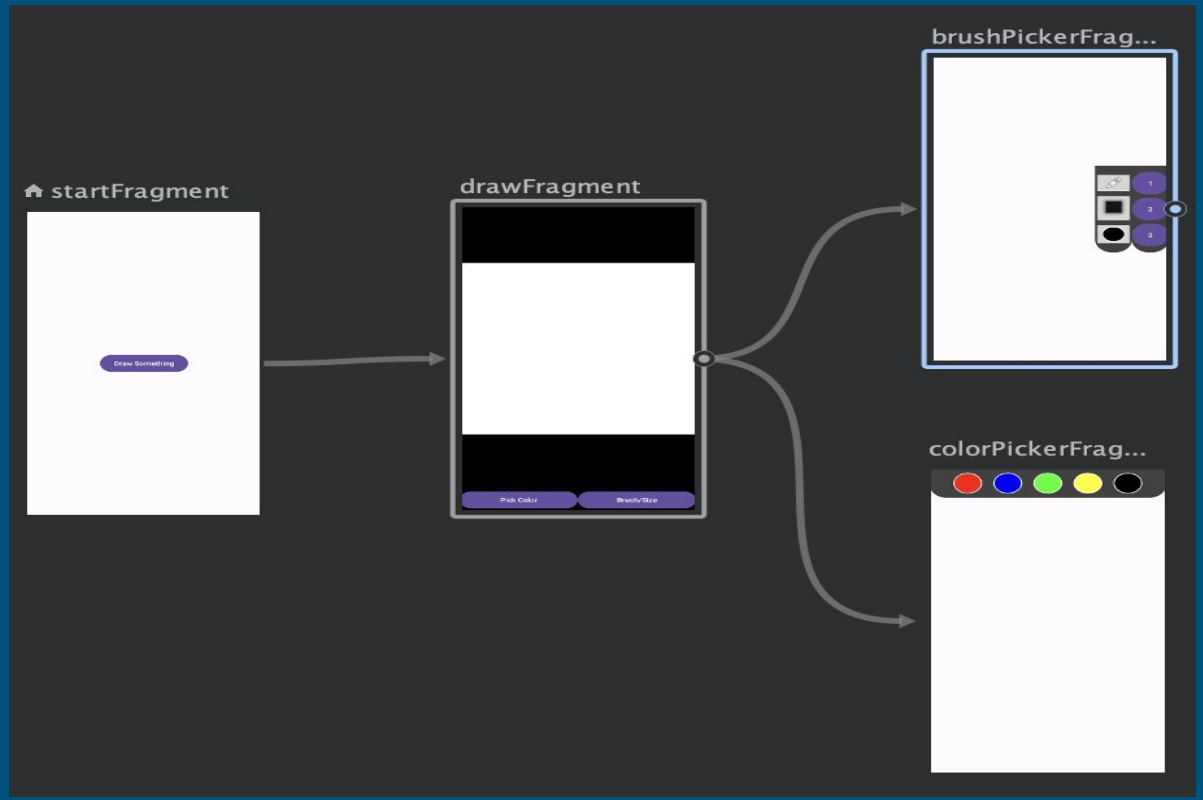
Visual design of our app

- Navigation Graph - A *navigation graph* is a resource file that contains all of your destinations and actions. The graph represents all of your app's navigation paths.
- Usually the "nodes" of the graph are Fragments which correspond to screen.
- The splash screen is its own activity and has a delay to move out to the main activity.
- The main activity moves between the start screen and the draw screen.
- The tool bars or pick color/brush/size are both their own fragment, used as pop up windows to display and go away.

Technical details :

Fragments used in the Navigation component -

- startFragment
  - Linear Layout
- drawFragment
  - Constraint Layout
  - Needed to be constraint for Custom View to be 1:1 ratio
- brushPickerFragment
  - Frame Layout
    - Linear Layouts
- colorPickerFragment
  - Frame Layout
    - Linear Layouts



Technical details (Continued) :

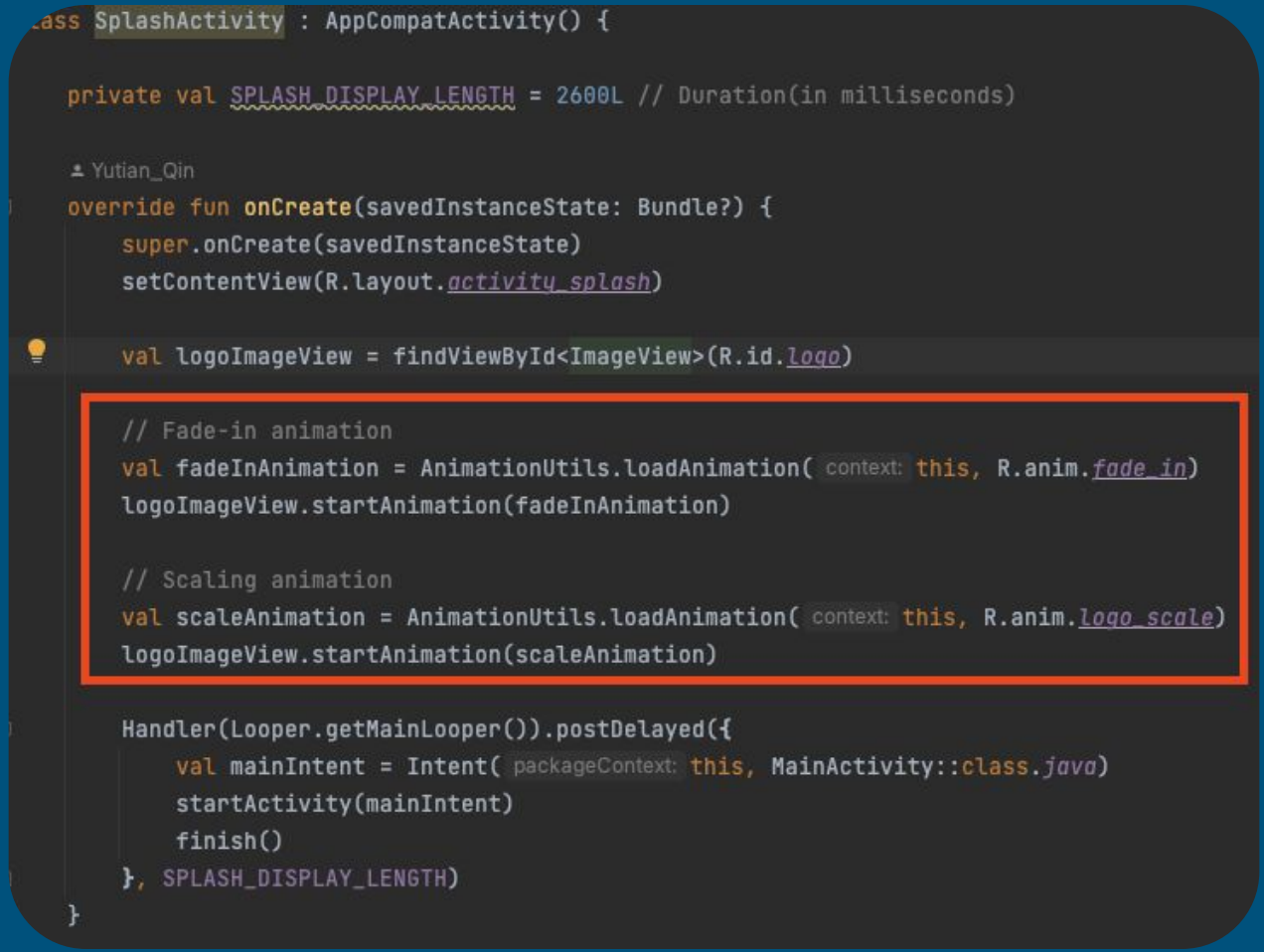
- Our view model contains the following :

```
val drawingBitmap: MutableLiveData<Bitmap> = MutableLiveData()

val currentColor: MutableLiveData<Int> = MutableLiveData(Color.BLACK)

var currentBrushType: MutableLiveData<BrushPickerFragment.BrushType> = MutableLiveData(BrushPickerFragment.BrushType.NORMAL)
var currentPenSize: MutableLiveData<Float> = MutableLiveData( value: 5f)
```

- It also has the functionality for updating the brush color, shape and size based on the user input.

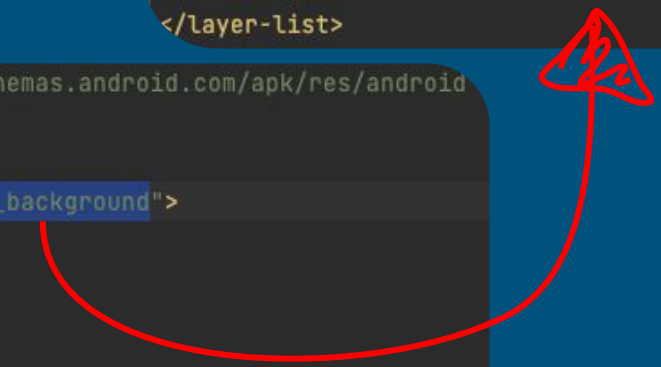
Interesting feature (Splash Screen)

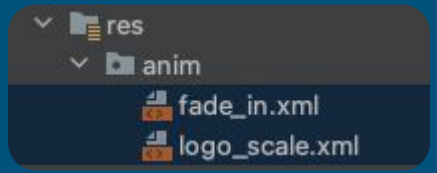
```
<activity android:name=".SplashActivity"
    android:theme="@style/SplashTheme"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name=".MainActivity"></activity>
```


lass SplashActivity : AppCompatActivity() {
private val SPLASH_DISPLAY_LENGTH = 2600L // Duration(in milliseconds)
Yutian_Qin
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_splash)
val logoImageView = findViewById<ImageView>(R.id.logo)
// Fade-in animation
val fadeInAnimation = AnimationUtils.loadAnimation(context: this, R.anim.fade_in)
logoImageView.startAnimation(fadeInAnimation)
// Scaling animation
val scaleAnimation = AnimationUtils.loadAnimation(context: this, R.anim.logo_scale)
logoImageView.startAnimation(scaleAnimation)
Handler(Looper.getMainLooper()).postDelayed({
val mainIntent = Intent(packageContext: this, MainActivity::class.java)
startActivity(mainIntent)
finish()
}, SPLASH_DISPLAY_LENGTH)
}

```xml
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape>
            <gradient
                android:angle="45"
                android:startColor="#00FFFF"
                android:endColor="#00008B"
                android:type="linear" />
        </shape>
    </item>
</layer-list>
```

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/splash_background">

    <ImageView
        android:id="@+id/logo"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_centerInParent="true"
        android:src="@mipmap/logo" />

</RelativeLayout>
```






```xml
<alpha
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="2000"
    android:fromAlpha="0.0"
    android:toAlpha="1.0" />
```

```xml
<scale
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="1200"
    android:fromXScale="0.1"
    android:fromYScale="0.1"
    android:pivotX="50%"
    android:pivotY="50%"
    android:toXScale="1.0"
    android:toYScale="1.0" />
```

Testing

- Manual Testing
- Unit Tests
  - Checking View Model has correct initialized values
  - Future goal add testing to update color/size/type

```
@Test
fun testDefaultValues() {
    assertNull(viewModel.drawingBitmap.value)
    assertEquals(Color.BLACK, viewModel.currentColor.value)
    assertEquals(BrushPickerFragment.BrushType.NORMAL, viewModel.currentBrushType.value)
    assertEquals( expected: 5f, viewModel.currentPenSize.value)
}
```

Testing

- Instrument Tests
  - Navigate through the screens
  - Once on fragments check if buttons work/displayed

```
@Test
fun clickColorPickerButton_shouldShowPopup() {
    // Start the Fragment
    launchFragmentInContainer<DrawFragment>()
    // Click on the button
    onView(withId(R.id.colorPickerButton)).perform(click())
    // Check that the buttons are present
    onView(withId(R.id.redButton)).check(matches(isDisplayed()))
    onView(withId(R.id.blackButton)).check(matches(isDisplayed()))
    onView(withId(R.id.yellowButton)).check(matches(isDisplayed()))
    onView(withId(R.id.blueButton)).check(matches(isDisplayed()))
    onView(withId(R.id.greenButton)).check(matches(isDisplayed()))
}
```

Thank you