**Week 12: Animation**

**Part 1: Creating Project**

1. Create an app that uses the **Empty Views Activity template (minimum API level 21)**. Named the project as Animation Property Demo.

1. Create the following colours in the colors.xml resource file:

<color name="bright\_sun">#fcfcb7</color>  
<color name="blue\_sky">#1e7ac7</color>  
<color name="sunset\_sky">#ec8100</color>  
<color name="night\_sky">#05192e</color>  
<color name="sea">#224869</color>

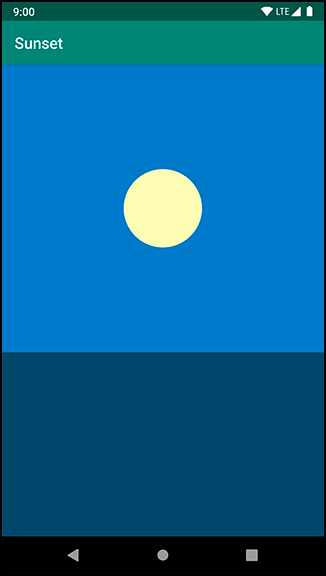
1. Add an oval shape drawable for a circular sun called sun.xml:

*<?*xml version="1.0" encoding="utf-8"*?>*<shape xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="oval">  
 <solid android:color="@color/bright\_sun" />  
</shape>

1. Next, build the entire scene out in a layout file. Open res/layout/activity\_main.xml, delete the current contents, and add the following:

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/scene"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <FrameLayout  
 android:id="@+id/sky"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="0.61"  
 android:background="@color/blue\_sky">  
 <ImageView  
 android:id="@+id/sun"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:layout\_gravity="center"  
 android:src="@drawable/sun" />  
 </FrameLayout>  
 <View  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="0.39"  
 android:background="@color/sea" />  
</LinearLayout>

1. Check out the preview. You should see a daytime scene of the sun in a blue sky over a dark blue sea.



1. Set the viewbinding in the gradle file.
2. Declare the following attributes in the MainActivity class:

private lateinit var binding: ActivityMainBinding  
  
private val blueSkyColor: Int by *lazy* **{** ContextCompat.getColor(this, R.color.*blue\_sky*)  
**}**private val sunsetSkyColor: Int by *lazy* **{** ContextCompat.getColor(this, R.color.*sunset\_sky*)  
**}**private val nightSkyColor: Int by *lazy* **{** ContextCompat.getColor(this, R.color.*night\_sky*)  
**}**

1. Inside of onCreate(…), write the following code:

binding = ActivityMainBinding.inflate(*layoutInflater*)  
setContentView(binding.*root*)

1. Add the onClickListener for the scene view object:

binding.scene.setOnClickListener **{** startAnimation()  
**}**

1. Write this first step in a new function called startAnimation():

private fun startAnimation() {}

1. Add the following code in the startAnimation to define the sun location:

val sunYStart = binding.sun.*top*.toFloat()  
val sunYEnd = binding.sky.*height*.toFloat()

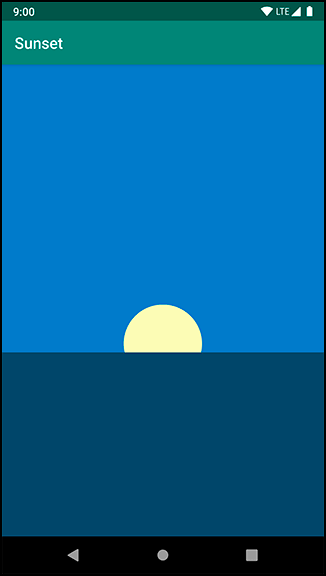
Log.*d*("Test", "sunYStart:" + sunYStart )  
Log.*d*("Test", "sunYEnd:" + sunYEnd )

1. Create the first object animator to start with the top of the view at its current location. It needs to end with the top at the bottom of sunView’s parent, skyView:

val heightAnimator = ObjectAnimator  
 .ofFloat(binding.sun, "y", sunYStart, sunYEnd)  
 .setDuration(3000)

heightAnimator.start()

1. Now that you know where the animation should start and end, create and run an ObjectAnimator to perform it. Run Sunset and press anywhere on the scene to run the animation.



1. To add this sensation of acceleration, all you need to do is use an **AccelerateInterpolator**. Place the following code before the heightAnimator.start()

heightAnimator.*interpolator* = AccelerateInterpolator()

1. Run Sunset one more time and press to see your animation. Your sun should now start moving slowly and accelerate to a quicker pace as it moves toward the horizon.

1. Now that your sun is animating down, let’s animate the sky to a sunset-appropriate color.

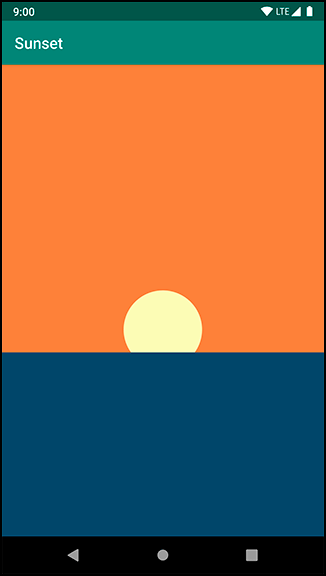
val sunsetSkyAnimator = ObjectAnimator  
 .ofInt(binding.sky, "backgroundColor", blueSkyColor, sunsetSkyColor)  
 .setDuration(3000)

sunsetSkyAnimator.start()

1. If you run it you will see that something is amiss.
2. A TypeEvaluator is an object that tells ObjectAnimator what value is, say, a quarter of the way between a start value and end value. Add the following code before the sunsetSkyAnimator.start().

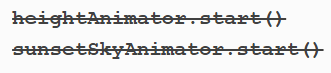
sunsetSkyAnimator.setEvaluator(ArgbEvaluator())

1. Run your animation once again, and you should see the sky fade to a beautiful orange colour:



1. Repeat the step 16 and create another object animator called **nightSkyAnimator**. It should change from sunset sky color to night sky color.

1. It is much easier to use an AnimatorSet to start the animation with sequence. Comment out the two start statements.



1. Create the AnimatorSet as follows:

val animatorSet = AnimatorSet()  
animatorSet.play(heightAnimator)  
 .with(sunsetSkyAnimator)  
 .before(nightSkyAnimator)  
animatorSet.start()

1. Run your app one more time and savor the soothing sunset you have created. Magic.

1. Challenges: Add the ability to reverse the sunset after it is completed, so your user can press for a sunset, and then press a second time to get a sunrise. You will need to build another AnimatorSet to do this – AnimatorSets cannot be run in reverse