Mobile Monday Tech Lab 03

Android Development - Moving Images

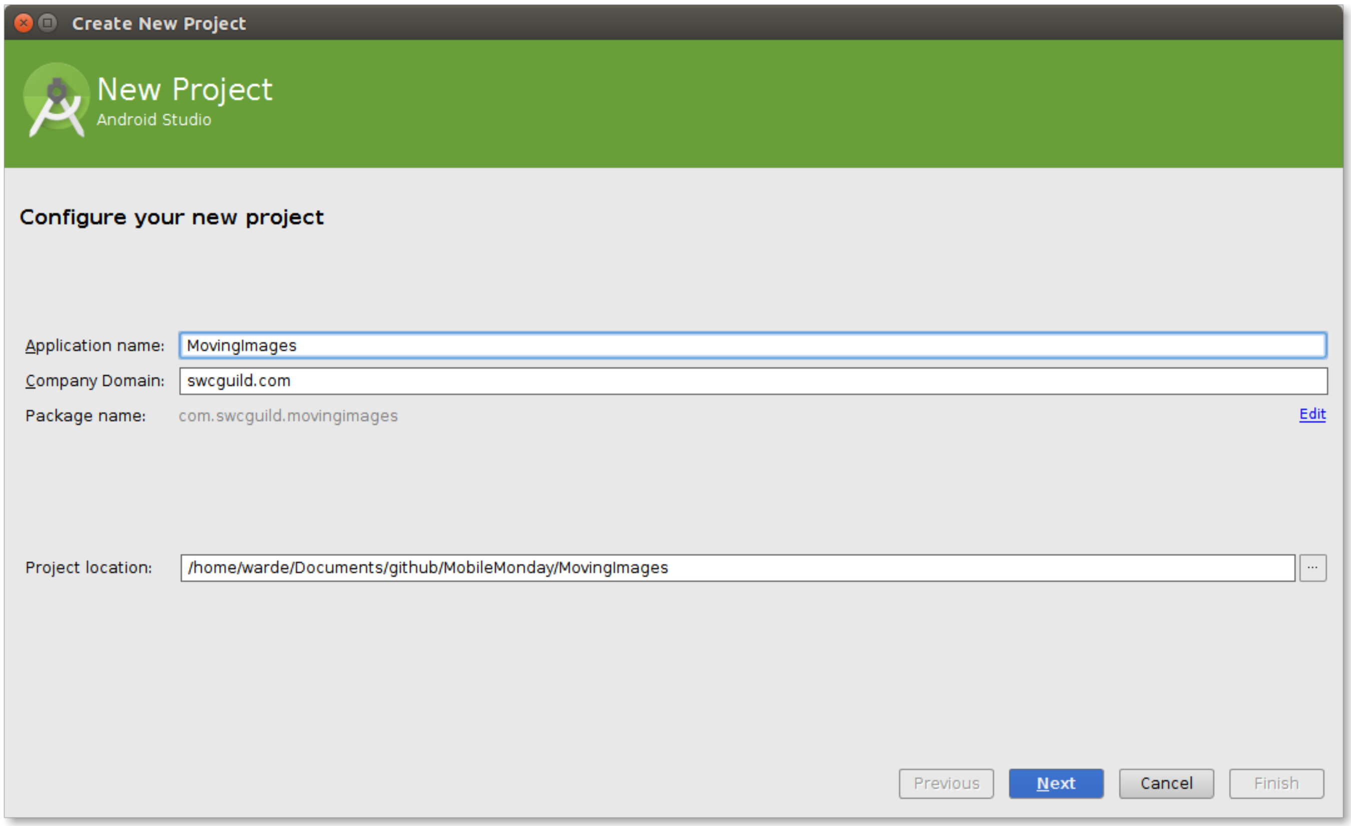
Overview

In this lab we are going to create an Android app that displays images, sets them in motion, and keeps them on the screen by reversing their direction when they hit the edge of the screen. As with all sessions in this series, the completed project can be found in GitHub @ https://github.com/swcguild/MobileMonday. Our objectives for this session are:

- Build a simple UI with a CustomView
- Learn how to import images into a project
- Create a custom class for images that move
- Create a class that manages the location and velocity of our moving images and draws them to the screen

Creating the Project

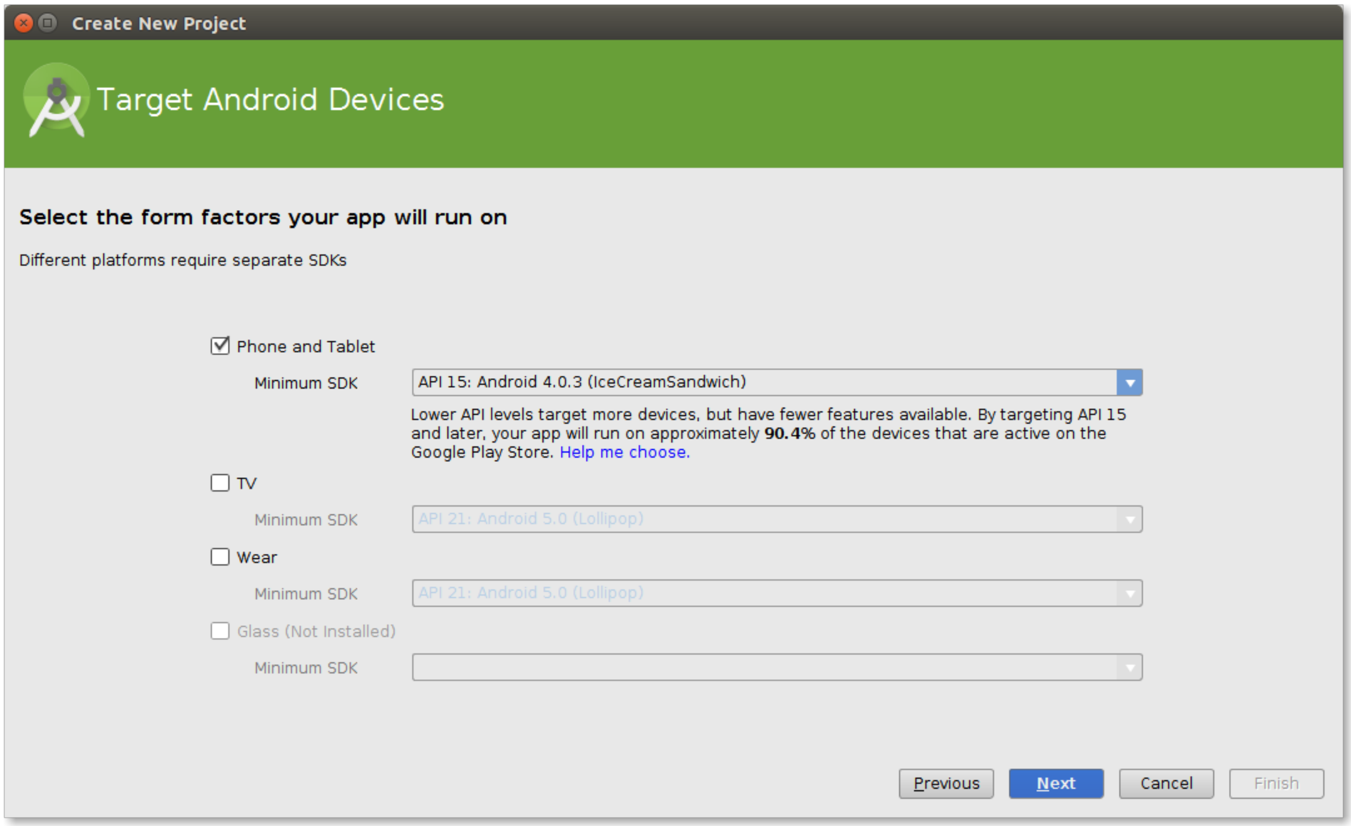
In this section we'll step through the process of creating the project. These steps will be very similar to those of the previous labs. Begin by creating a new project called **MovingImages**. Match the values shown below in the New Project Creation Wizard screenshots:



Configure your new project

Application name: MovingImages
Company Domain: swcguild.com
Package name: com.swcguild.movingimages
Project location: /home/warde/Documents/github/MobileMonday/MovingImages

Initial Creation Wizard Screen



Select the form factors your app will run on

Different platforms require separate SDKs

[✓] Phone and Tablet

Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available. By targeting API 15 and later, your app will run on approximately **90.4%** of the devices that are active on the Google Play Store. Help me choose.

[ ] TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

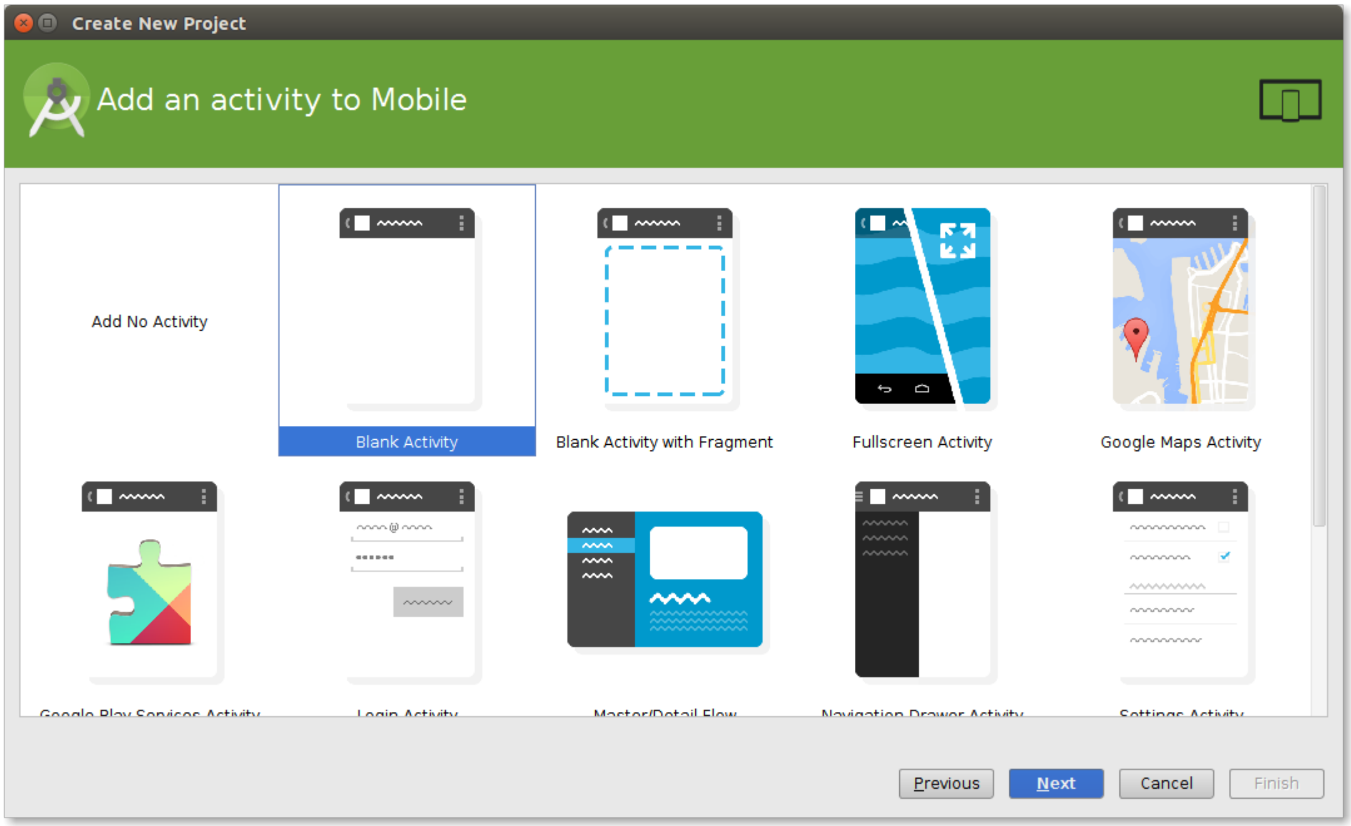
[ ] Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

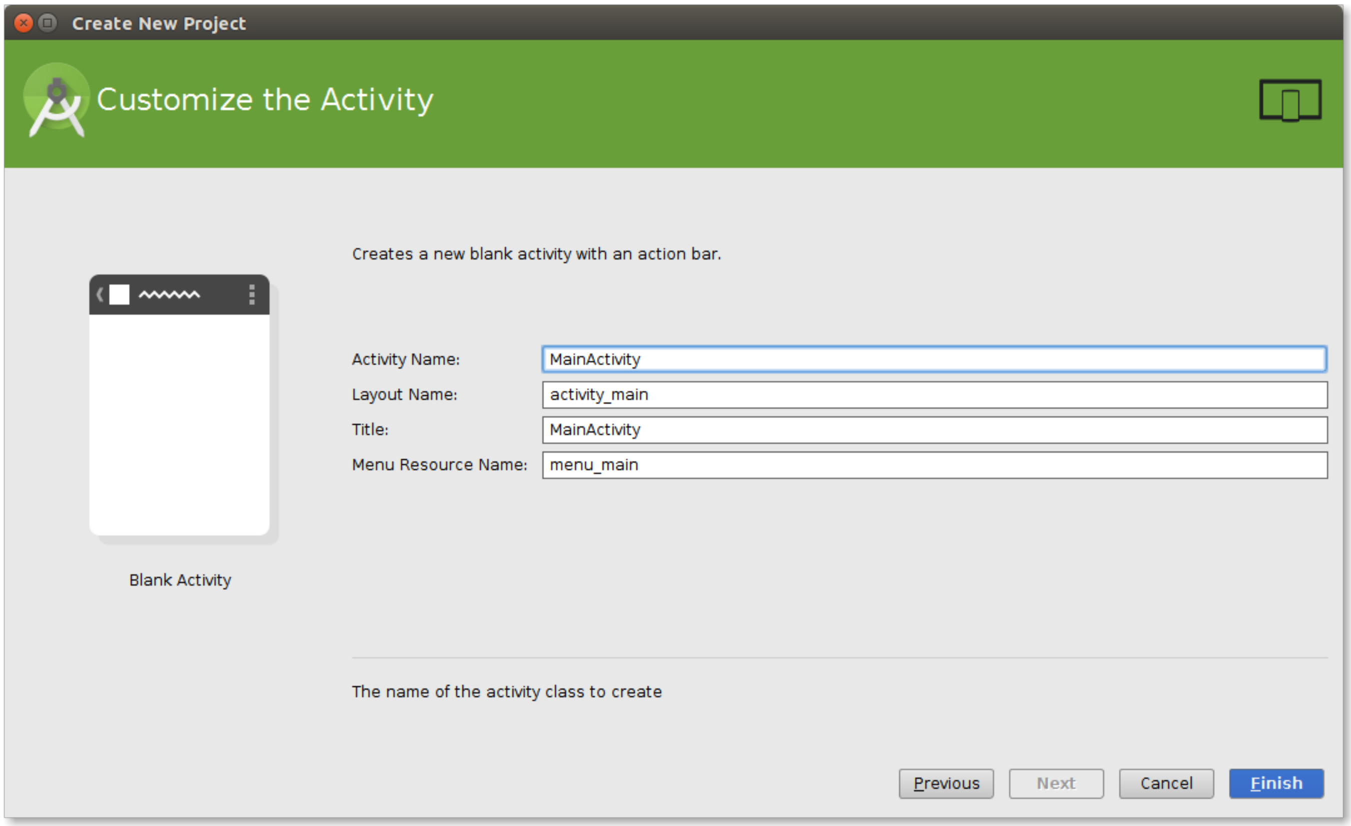
[ ] Glass (Not Installed)

Minimum SDK:

Platform and SDK Options Screen



Initial Activity Selection - Blank Activity Selected

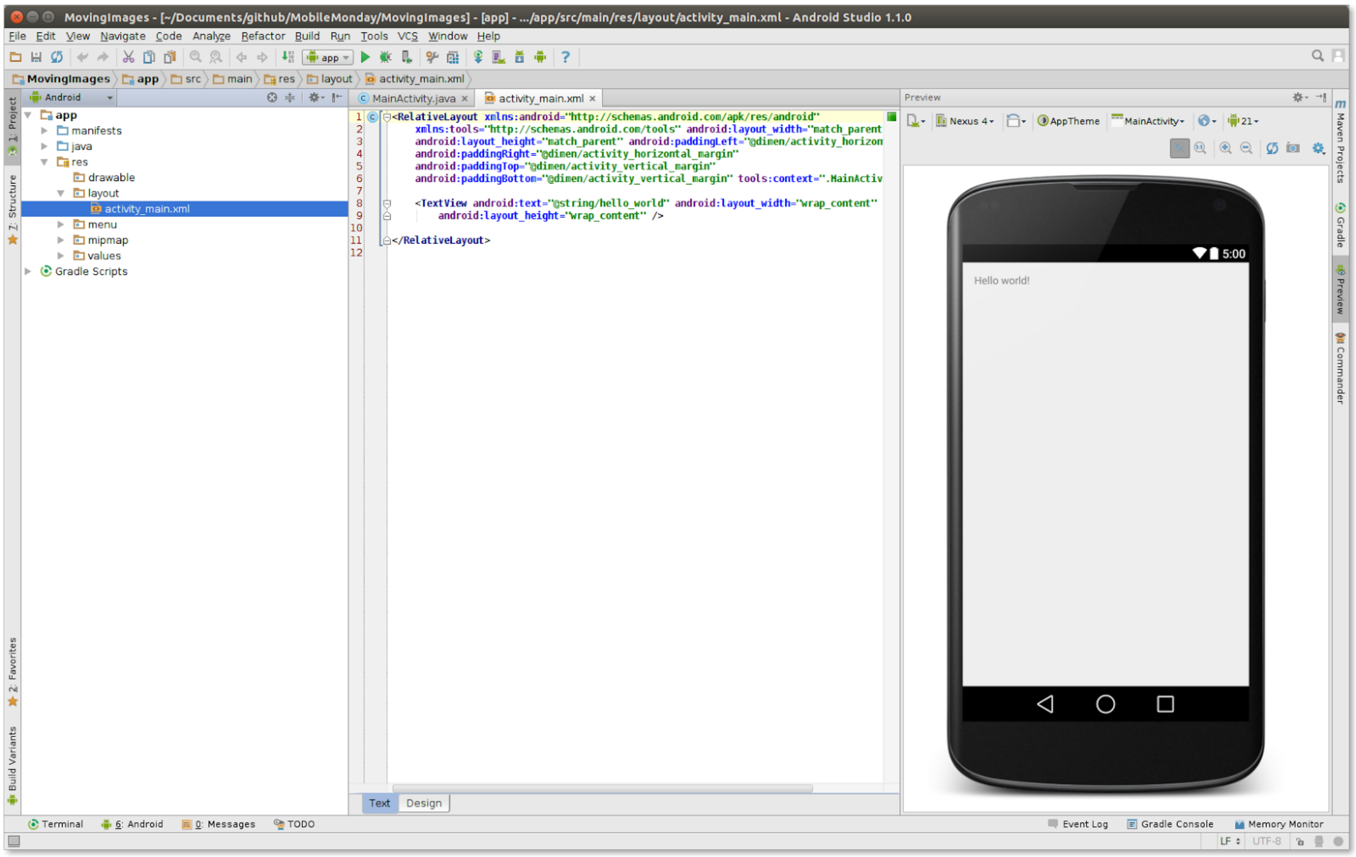


Creates a new blank activity with an action bar.

Activity Name:	MainActivity
Layout Name:	activity_main
Title:	MainActivity
Menu Resource Name:	menu_main

The name of the activity class to create

MainActivity Options - Defaults Chosen


MovingImages - [~/Documents/github/MobileMonday/MovingImages] - [app] - .../app/src/main/res/layout/activity_main.xml - Android Studio 1.1.0

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizon
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActiv

    <TextView android:text="@string/hello_world" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

Newly Created Project

Import an Image

Importing an image into an Android Studio project is easy. Simply find an image between 50x50 and 100x100 px and save it in the **res/drawable** folder of your project. For this lab, a png with a transparent background will look the best. This is the image used in the example:

Create Moving Image Class

In this section we will create a class to hold the location (x/y coordinates), velocity, and bit map for an image. This will be a very simple, stripped down class containing just public member variables. Create a new class called **MovingImage** in the **com.swcguild.movingimages package** and enter the following code:

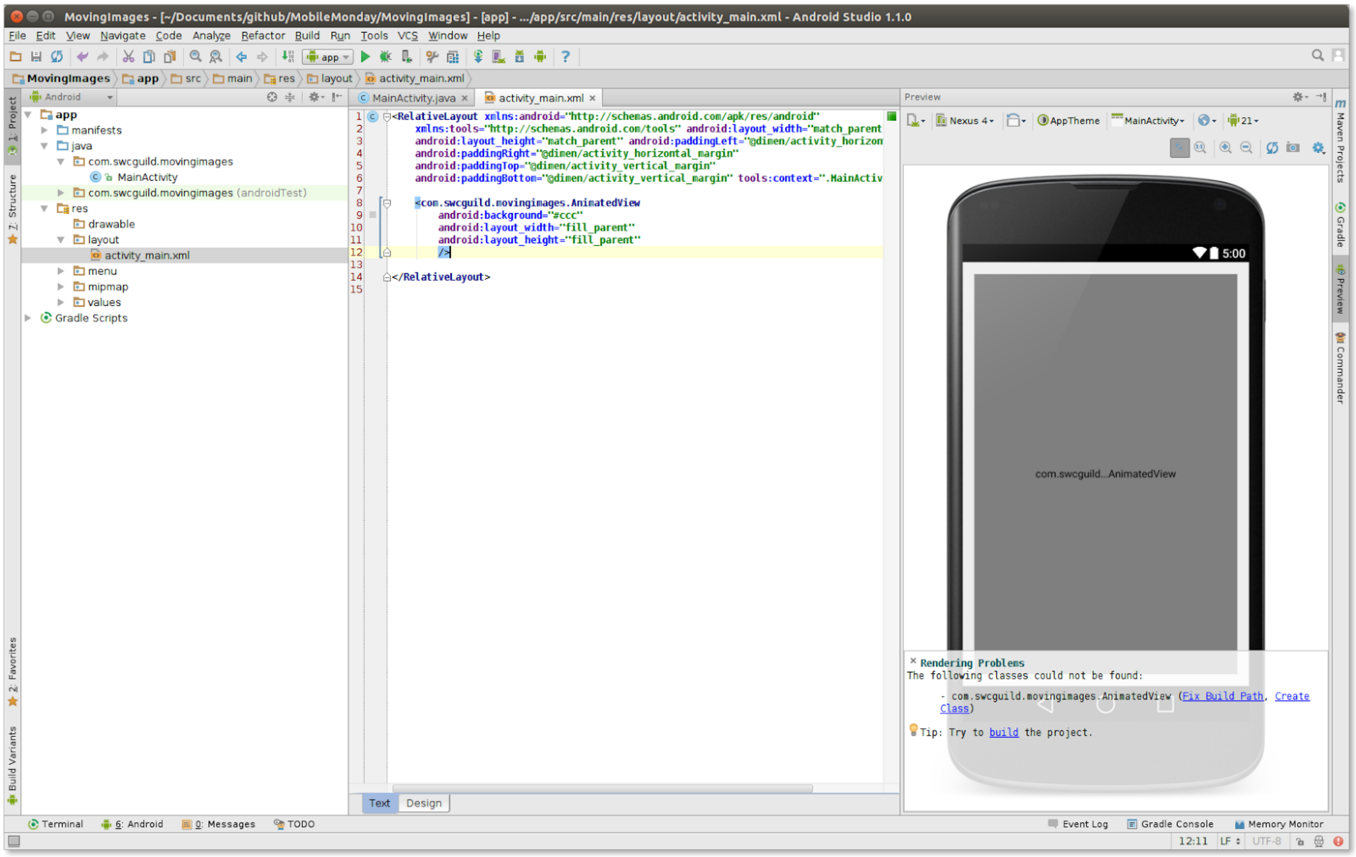
```
public class MovingImage {
    public int x;
    public int y;
    public int xVelocity;
    public int yVelocity;
    public BitmapDrawable image;
}
```

Modify activity_main.xml

Your **activity_main.xml** should have a root element of **RelativeLayout** that contains one **TextView** element. Remove the **TextView** element and replace it with the following (this new element refers to the custom view class that we will create in the next step):

```
    <com.swcguild.movingimages.AnimatedView
        android:background="#ccc"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
    />
```

Your editor should now look like this:


MovingImages - [~/Documents/github/MobileMonday/MovingImages] - [app] - .../app/src/main/res/layout/activity_main.xml - Android Studio 1.1.0
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MovingImages app src main res layout activity_main.xml
app
manifests
java
com.swcguild.movingimages
MainActivity
com.swcguild.movingimages (androidTest)
res
drawable
layout
activity_main.xml
menu
mipmap
values
Gradle Scripts
MainActivity.java
activity_main.xml
Preview
Nexus 4
AppTheme
MainActivity
21
5:00
com.swcguild...AnimatedView
Text
Design
Terminal
6: Android
0: Messages
TODO
Event Log
Gradle Console
Memory Monitor
12:11
LF
UTF-8

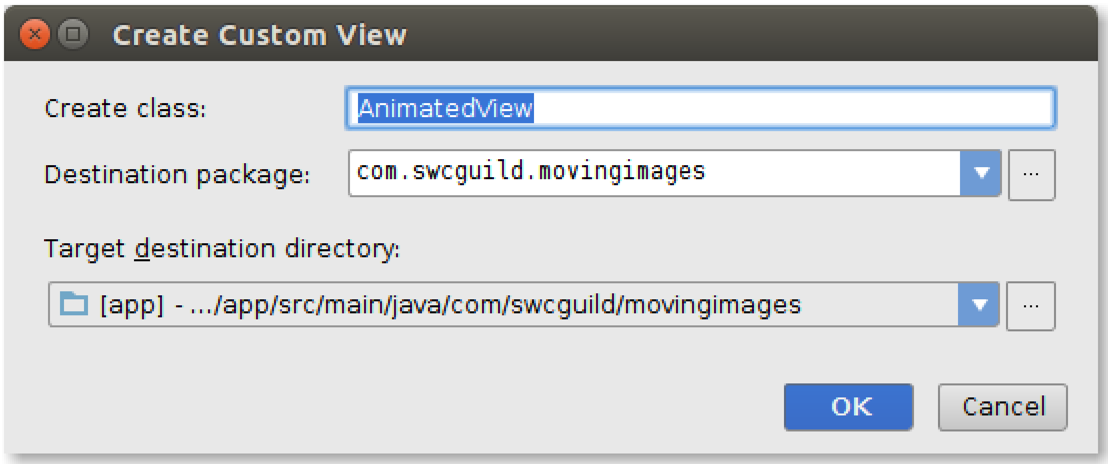
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizon
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActiv

    <com.swcguild.movingimages.AnimatedView
        android:background="#ccc"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        />

</RelativeLayout>
```

```
Rendering Problems
The following classes could not be found:

    - com.swcguild.movingimages.AnimatedView (Fix Build Path, Create
    Class)

Tip: Try to build the project.
```

Changes to activity_main.xml

Create Animated View Class

The class referred to in the XML element we created in the previous step does not yet exist. You'll notice that Android Studio is complaining about this fact in the semi transparent Rendering Problems dialog box in the lower right corner of the image above. Click on the **Create Class** link in the dialog box. You should now see the **Create Custom View** dialog. Fill it in with the following values and click OK:



Create Custom View

Create class: AnimatedView

Destination package: com.swcguild.movingimages

Target destination directory: [app] - .../app/src/main/java/com/swcguild/movingimages

Create Custom View Dialog

The editor should automatically display the new class.

Modify Animated View Class

In this section we'll implement the bulk of our app. the **AnimatedView** class contains all the logic that manages, moves, and draws the images to the screen.

Extend ImageView

The first thing we need to do is have **AnimatedView** extend **ImageView** rather than **View**. Do that now.

Create Member Variables

Now we will create the required member variables for the class. Add the following code to the top of AnimatedView:

```
    // the Handler allows us to send messages to the thread that will run our
    // animation
    private Handler h;
    private final int FRAME_RATE = 30;
    // holds all the images we want to display
    List<MovingImage> images = new ArrayList<>();
    // thread on which our animation will run
    // invalidate is a method on View (ImageView extends View) - it invalidates the
    // entire view in preparation for it being drawn again
    private Runnable r = new Runnable() {
        public void run() {
            invalidate();
        }
    };
```

Modify Constructor

Next we'll add code to the constructor to initialize our Handler and create a list of moving images. Modify your constructor so it looks like this (note that we are using the 2 parameter version of the constructor):

```
public AnimatedView(Context context, AttributeSet attrs) {
    super(context, attrs);

    // Create a Handler so we can send messages to our thread
    h = new Handler();

    // Construct the MovingImage objects that we want to show on the screen
    MovingImage img = new MovingImage();
    img.x = 10;
    img.y = 10;
    img.xVelocity = 5;
    img.yVelocity = 5;
    img.image =
        (BitmapDrawable) context.getResources().getDrawable(R.drawable.android);
    images.add(img);
    img = new MovingImage();
    img.x = 20;
    img.y = 25;
    img.xVelocity = 2;
    img.yVelocity = 7;
    img.image =
        (BitmapDrawable) context.getResources().getDrawable(R.drawable.android);
    images.add(img);
}
```

Implement onDraw Method

Our final coding task is to implement the onDraw method. This method updates the location of each of our images (based on where they currently are and their x/y velocities), detects collision with the screen boundaries, reversing course if needed, and redraws the images to the screen. Add the following method to your class:

```
    @Override
    public void onDraw(Canvas c) {

        // go through each MovingImage in images and update its position, change direction if
        // needed and redraw to the screen
        for (MovingImage img : images) {
            // Move the image
            img.x += img.xVelocity;
            img.y += img.yVelocity;

            // Detect collisions with the edge of the screen - reverse direction if needed
            if ((img.x > this.getWidth() - img.image.getBitmap().getWidth()) || (img.x < 0)) {
                img.xVelocity *= -1;
            }

            if ((img.y > this.getHeight() - img.image.getBitmap().getHeight()) || (img.y < 0)) {
                img.yVelocity *= -1;
            }

            // Draw this image
            c.drawBitmap(img.image.getBitmap(), img.x, img.y, null);
            // place our thread on the queue to be run after a delay
            h.postDelayed(r, FRAME_RATE);
        }
    }
```

Build, Deploy, and Run

Finally, build and run the app. You can run this app on an emulator but it will perform much better on an actual device.