63. Android Picture-in-Picture Mode

When multi-tasking in Android was covered in earlier chapters, Picture-in-picture (PiP) mode was mentioned briefly but not covered in any detail. Intended primarily for video playback, PiP mode allows an activity screen to be reduced in size and positioned at any location on the screen. While in this state, the activity continues to run and the window remains visible regardless of any other activities running on the device. This allows the user to, for example, continue watching video playback while performing tasks such as checking email or working on a spreadsheet.

This chapter will provide an overview of Picture-in-Picture mode before Picture-in-Picture support is added to the VideoPlayer project in the next chapter.

63.1 Picture-in-Picture Features

As will be explained later in the chapter, and demonstrated in the next chapter, an activity is placed into PiP mode via an API call from within the running app. When placed into PiP mode, configuration options may be specified that control the aspect ratio of the PiP window and also to define the area of the activity screen that is to be included in the window. Figure 63-1, for example, shows a video playback activity in PiP mode:

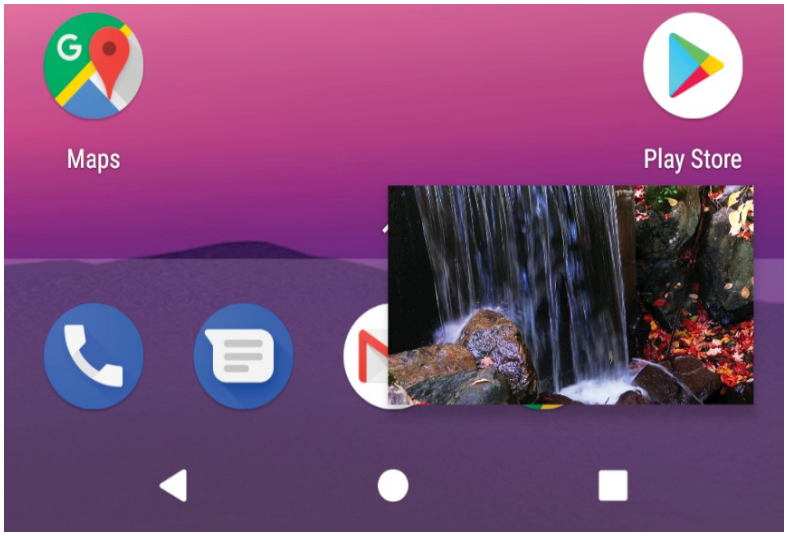

Figure 63-1

Figure 63-2 shows a PiP mode window after it has been tapped by the user. When in this mode, the window appears larger and includes a full screen

action in the center which, when tapped, restores the window to full screen mode and an exit button in the top right-hand corner to close the window and place the app in the background. Any custom actions added to the PiP window will also appear on the screen when it is displayed in this mode. In the case of Figure 63-2, the PiP window includes custom play and pause action buttons:

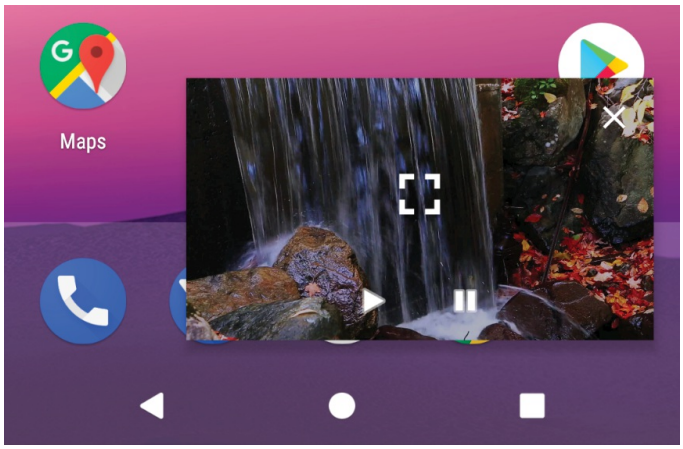

Figure 63-2

The remainder of this chapter will outline how PiP mode is enabled and managed from within an Android app.

63.2 Enabling Picture-in-Picture Mode

PiP mode is currently only supported on devices running Android 8.0 (API 26) or newer. The first step in implementing PiP mode is to enable it within the project's manifest file. PiP mode is configured on a per activity basis by adding the following lines to each activity element for which PiP support is required:

```
<activity android:name=".MyActivity"
    android:supportsPictureInPicture="true"
    android:configChanges=
       "screenSize|smallestScreenSize|screenLayout|orientation"
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

The *android:supportsPictureInPicture* entry enables PiP for the activity while the *android:configChanges* property notifies Android that the activity is able

to handle layout configuration changes. Without this setting, each time the activity moves in and out of PiP mode the activity will be restarted resulting in playback restarting from the beginning of the video during the transition.

63.3 Configuring Picture-in-Picture Parameters

PiP behavior is defined through the use of the PictureInPictureParams class, instances of which can be created using the Builder class as follows:

```
PictureInPictureParams params =
                new PictureInPictureParams.Builder().build();
```

The above code creates a default PictureInPictureParams instance with special parameters defined. The following optional method calls may also be used to customize the parameters:

- **setActions()** – Used to define actions that can be performed from within the PiP window while the activity is in PiP mode. Actions will be covered in more detail later in this chapter.
- **setAspectRatio()** – Declares the preferred aspect ratio for appearance of the PiP window. This method takes as an argument a Rational object containing the height width / height ratio.
- **setSourceRectHint()** – Takes as an argument a Rect object defining the area of the activity screen to be displayed within the PiP window.

The following code, for example, configures aspect ratio and action parameters within a PictureInPictureParams object. In the case of the aspect ratio, this is defined using the width and height dimensions of a VideoView instance:

```
Rational rational = new Rational(videoView.getWidth(),
                                 videoView.getHeight());

PictureInPictureParams params =
        new PictureInPictureParams.Builder()
                .setAspectRatio(rational)
                .setActions(actions)
                .build();
```

Once defined, PiP parameters may be set at any time using the *setPictureInPic*

```
setPictureInPictureParams(params);
```

Parameters may also be specified when entering PiP mode.

63.4 Entering Picture-in-Picture Mode

An activity is placed into Picture-in-Picture mode via a call to the *enterPictureInPictureMode()* method, passing through a PictureInPictureParams object:

```
enterPictureInPictureMode(params);
```

If no parameters are required, simply create a default PictureInPictureParams object as outlined in the previous section. If parameters have previously been set using the *setPictureInPictureParams()* method, these parameters are combined with those specified during the *enterPictureInPictureMode()* method call.

63.5 Detecting Picture-in-Picture Mode Changes

When an activity enters PiP mode, it is important to hide any unnecessary views so that only the video playback is visible within the PiP window. When the activity re-enters full screen mode, any hidden user interface components need to be re-instated. These and any other app specific tasks can be performed by overriding the *onPictureInPictureModeChanged()* method. When added to the activity, this method is called each time the activity transitions between PiP and full screen modes and is passed a Boolean value indicating whether the activity is currently in PiP mode:

```
@Override
public void onPictureInPictureModeChanged(
                    boolean isInPictureInPictureMode) {

    super.onPictureInPictureModeChanged(isInPictureInPictureMode);

    if (isInPictureInPictureMode) {
          // Acitivity entered Picture-in-Picture mode
    } else {
          // Activity entered full screen mode
    }
}
```

63.6 Adding Picture-in-Picture Actions

Picture-in-Picture actions appear as icons within the PiP window when it is tapped by the user. Implementation of PiP actions is a multi-step process that begins with implementing a way for the PiP window to notify the activity that

an action has been selected. This is achieved by setting up a broadcast receiver within the activity, and then creating a pending intent within the PiP action which, in turn, is configured to broadcast an intent for which the broadcast receiver is listening. When the broadcast receiver is triggered by the intent, the data stored in the intent can be used to identify the action performed and to take the necessary action within the activity.

PiP actions are declared using the RemoteAction instances which are initialized with an icon, a title, a description and the PendingIntent object. Once one or more actions have been created, they are added to an ArrayList and passed through to the *setActions()* method while building a PictureInPictureParams object.

The following code fragment demonstrates the creation of the Intent, PendingIntent and RemoteAction objects together with a PictureInPictureParams instance which is then applied to the activity's PiP settings:

```
final ArrayList<RemoteAction> actions = new ArrayList<>();

Intent actionIntent = new Intent("MY_PIP_ACTION");

final PendingIntent pendingIntent =
             PendingIntent.getBroadcast(MyActivity.this,
                           REQUEST_CODE, actionIntent, 0);

final Icon icon = Icon.createWithResource(MyActivity.this,
                           R.drawable.action_icon);

RemoteAction remoteAction = new RemoteAction(icon,
                           "My Action Title",
                           "My Action Description",
                           pendingIntent);

actions.add(remoteAction);

PictureInPictureParams params =
        new PictureInPictureParams.Builder()
                .setActions(actions)
                .build();

setPictureInPictureParams(params);
```

63.7 Summary

Picture-in-Picture mode is a multitasking feature introduced with Android 8.0 designed specifically to allow video playback to continue in a small window while the user performs tasks in other apps and activities. Before PiP mode can be used, it must first be enabled within the manifest file for those activities that require PiP support.

PiP mode behavior is configured using instances of the PictureInPictureParams class and initiated via a call to the *enterPictureInPictureMode()* method from within the activity. When in PiP mode, only the video playback should be visible, requiring that any other user interface elements be hidden until full screen mode is selected. These and other mode transition related tasks can be performed by overriding the *onPictureInPictureModeChanged()* method.

PiP actions appear as icons overlaid onto the PiP window when it is tapped by the user. When selected, these actions trigger behavior within the activity. The activity is notified of an action by the PiP window using broadcast receivers and pending intents.