

CSE4443 – Mobile User Interfaces

More About Views

Scott MacKenzie
York University

© Scott MacKenzie

Map

- Widgets
- Custom Components
- Drawing
- Image View
- Video View
- Web View
- Map View
- Surface View
- View Groups

© Scott MacKenzie

2

Map

- Widgets
- Custom Components
- Drawing
- Image View
- Video View
- Web View
- Map View
- Surface View
- View Groups

Review

Developers | Design | Develop | Distribute

Training | API Guides | Reference | Tools | Google Services | Samples

User Interface

- Overview
- Layouts
- Input Controls

UI Overview

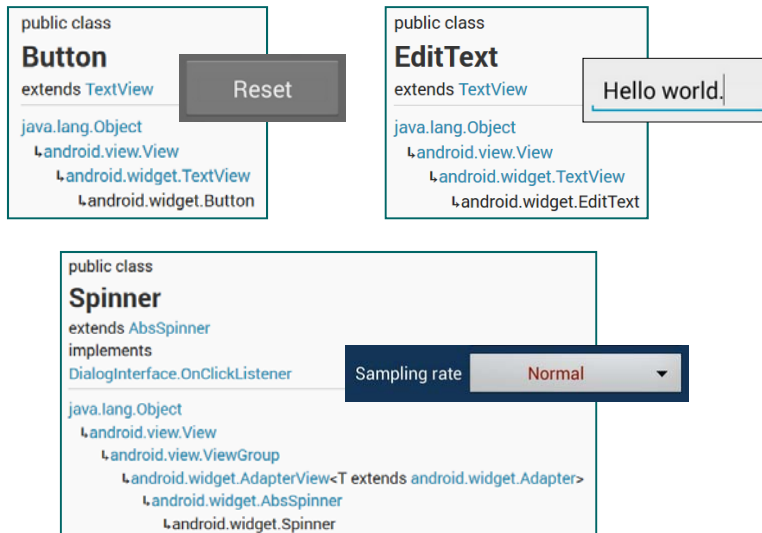
All user interface elements in an Android app are built using `View` and `ViewGroup` objects. A `View` is an object that draws something on the screen that the user can interact with. A `ViewGroup` is an object that holds other `View` (and `ViewGroup`) objects in order to define the layout of the interface.

(such as buttons and text)

... objects, as shown in figure 1. Each of these are widgets that draw some performance).

```
graph TD; ViewGroup[ViewGroup] --- View1[View]; ViewGroup --- View2[View];
```

Example Widgets



© Scott MacKenzie

5

Map

- Widgets
- Custom Components
- Drawing
- Image View
- Video View
- Web View
- Map View
- Surface View
- View Groups

© Scott MacKenzie

6

Custom Components

Custom Components

Android offers a sophisticated and powerful component model for building your UI, based on the fundamental layout classes: `View` and `ViewGroup`. Start with the platform includes a variety of pre-built `View` and `ViewGroup` subclasses, called widgets and layouts, respectively — that you can use to construct your UI.

A partial list of available widgets includes `Button`, `TextView`, `EditText`, `Listview`, `CheckBox`, `RadioButton`, `Spinner`, and the more special-purpose

IN THIS DOCUMENT

- [The Basic Approach](#)
- [Fully Customized Components](#)
- [Compound Controls](#)

The Basic Approach

Here is a high level overview of what you need to know to get started in creating your own `View` components:

1. Extend an existing `View` class or subclass with your own class.
2. Override some of the methods from the superclass. The superclass methods to override start with 'on', for example, `onDraw()`, `onMeasure()`, and `onKeyDown()`. This is similar to the `on...` events in `Activity` or `ListActivity` that you override for lifecycle and other functionality hooks.
3. Use your new extension class. Once completed, your new extension class can be used in place of the view upon which it was based.

© Scott MacKenzie

7

Example - MyButton

Demo CustomButton

Tap the Buttons Below

B1 B2
Exit

```


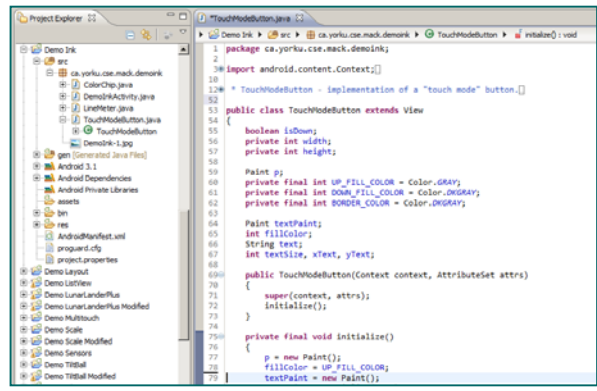

1 package ca.yorku.cse.mack.democustombutton;
2
3 import android.content.Context;
4
5 /**
6  * MyButton
7  * <code>
8  *
9  * Implementation of a custom button as a subclass of <code>View</code>.
10 *
11 * @author (c) Scott MacKenzie, 2011-2014
12 *
13 */
14 public class MyButton extends View
15 {
16     private final int UP_FILL_COLOR = Color.GRAY;
17     private final int DOWN_FILL_COLOR = 0xffff00ff;
18     private final int BORDER_COLOR = Color.DKGRAY;
19     private final int DEFAULT_WIDTH = 100;
20     private final int DEFAULT_HEIGHT = 100;
21
22     public int width;
23     public int height;
24     Paint linePaint, fillPaint, textPaint;
25     Path buttonPath;
26     String buttonText;
27     int textSize;
28     int xText, yText;
29     boolean isPressed;
30     ToneGenerator tg;
    
```

ESC

© Scott MacKenzie


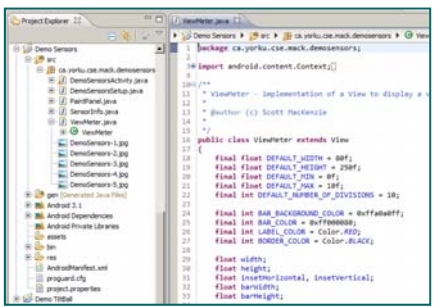
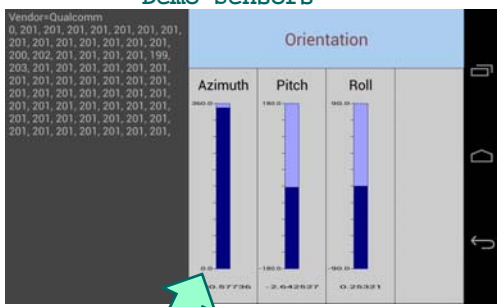
8

Example - TouchModeButton




© Scott MacKenzie 9

Example - ViewMeter




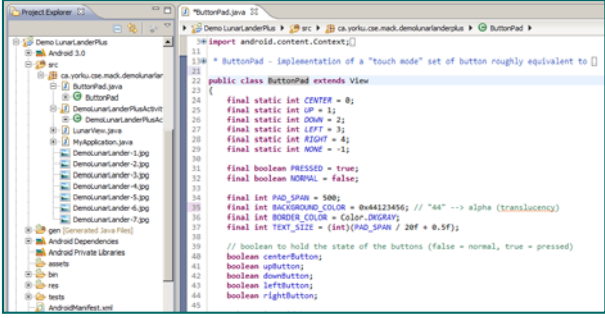
© Scott MacKenzie 10


Example - ButtonPad



Demo LunarLanderPlus








© Scott MacKenzie 11


Map

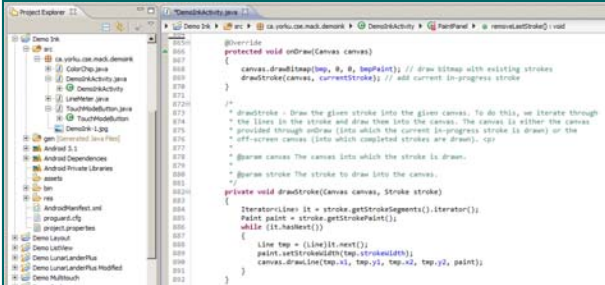
- Widgets
- Custom Components
- Drawing
- Image View
- Video View
- Web View
- Map View
- Surface View
- View Groups


Example - PaintPanel



Demo Ink





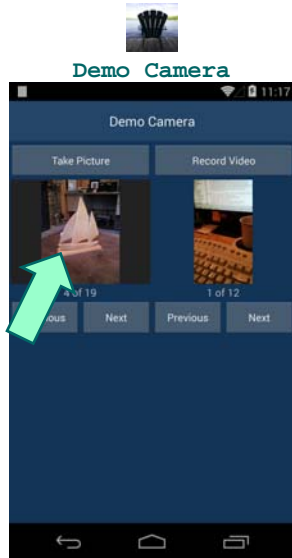


© Scott MacKenzie

Map

- Widgets
- Custom Components
- Drawing
- Image View
- Video View
- Web View
- Map View
- Surface View
- View Groups

Example



```
Project Explorer | Demo CameraActivity.java | Demo Camera | DemoCameraActivity | DemoCameraActivity | ImageView | ImageView  
1 package ca.yorku.cse.mack.democamera;  
2  
3 import java.io.File[];  
4  
5 * Demo Camera - demo application that uses an Android device's built-in camera |  
6  
7 public class DemoCameraActivity extends Activity implements OnClickListener, OnTouchListener  
8 {  
9     private static final String TAG = "DEMO_CAMERA"; // for Log.i messages  
10  
11     private static final int IMAGE_SIZE = 100;  
12     private static final int VIDEO_SIZE = 200;  
13     private static final int IMAGE_WIDTH = 300;  
14     private static final int VIDEO_WIDTH = 400;  
15  
16     public static final int MEDIA_TYPE_IMAGE = 1;  
17     public static final int MEDIA_TYPE_VIDEO = 2;  
18     public static final String STORAGE_DIRECTORY = "CameraStuff";  
19  
20     Uri fileUri;  
21     Button imageCameraButton, videoCameraButton;  
22     Button imagePreviewButton, imageNextButton, videoPreviewButton, videoNextButton;  
23     ImageView imageView;  
24     VideoView videoView;  
25     File mediaStorageDirectory;
```



© Scott MacKenzie

15

Example



```
Project Explorer | DemoWebDownloadActivity.java | Demo WebDownload | DemoWebDownloadActivity | DemoWebDownloadActivity | TextView | TextView  
1 package ca.yorku.cse.mack.demowebdownload;  
2  
3 import android.os.Bundle; |  
4  
5 * DemoWebDownload - demonstrate downloading an image from an internet web site and displaying the |  
6  
7 public class DemoWebDownloadActivity extends Activity |  
8 { |  
9     final String TEST_URL = "http://www.yorku.ca/mack/nordich2012-15.jpg"; |  
10     TextView textView; |  
11     ImageView imageView; |  
12     ImageDownloader imageDownloader; |  
13  
14     @Override |  
15     protected void onCreate(Bundle savedInstanceState) |  
16     { |  
17         super.onCreate(savedInstanceState); |  
18         setContentView(R.layout.main); |  
19         textView = (TextView)findViewById(R.id.textView); |  
20         imageView = (ImageView)findViewById(R.id.imageView); |  
21  
22         imageDownloader = new ImageDownloader(); |  
23         imageDownloader.download(TEST_URL, imageView); |  
24  
25         textView.setText("Image from " + TEST_URL); |  
26     } |  
27 } |
```



© Scott MacKenzie

16

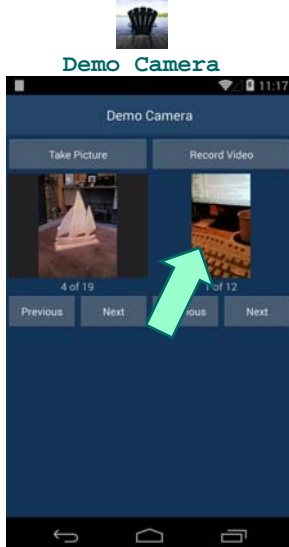
Map

- Widgets
- Custom Components
- Drawing
- Image View
- Video View
- Web View
- Map View
- Surface View
- View Groups

© Scott MacKenzie

17

Example



The screenshot shows the 'Demo Camera' app interface. At the top, there are two buttons: 'Take Picture' and 'Record Video'. Below these, there is a gallery of images. The first image is a sailboat, and the second is a wooden structure. A green arrow points from the 'Record Video' button to the second image in the gallery. Below the gallery, there are navigation buttons: 'Previous', 'Next', 'Previous', and 'Next'. The status bar at the top shows the time as 11:17.

```

package ca.yorku.cs.sc.mack.democamera;

import java.io.File;

...

public class DemoCameraActivity extends Activity implements OnClickListener, OnTouchListener {
    private static final String HDEBUG = "HDEBUG"; // for Log.i messages
    private static final int IMAGE_SIZE = 300;
    private static final int VIDEO_SIZE = 300;
    private static final int IMAGE_WIDTH = 300;
    private static final int VIDEO_WIDTH = 400;
    public static final int MEDIA_TYPE_IMAGE = 1;
    public static final int MEDIA_TYPE_VIDEO = 2;
    public static final String WORKING_DIRECTORY = "CameraStuff";
    Uri fileUri;
    Button imageCameraButton, videoCameraButton;
    Button imagePreviewButton, imageCameraButton, videoPreviewButton, videoNextButton;
    ImageView imagePreview;
    VideoView videoView;
    File mediaStorageDirectory;
  
```

© Scott MacKenzie

18


Map

- Widgets
- Custom Components
- Drawing
- Image View
- Video View
- **Web View**
- Map View
- Surface View
- View Groups


© Scott Mackenzie

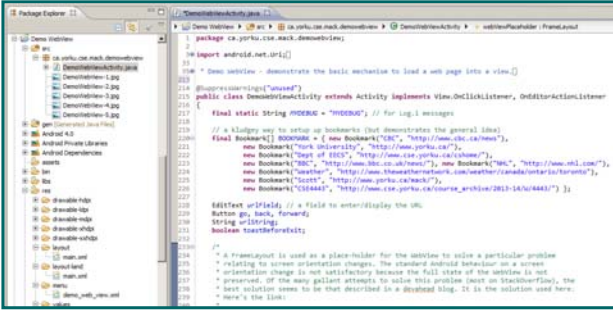
19


Example



Demo WebView







© Scott Mackenzie

20


Map

- Widgets
- Custom Components
- Drawing
- Image View
- Video View
- Web View
- **Map View**
- Surface View
- View Groups


© Scott MacKenzie

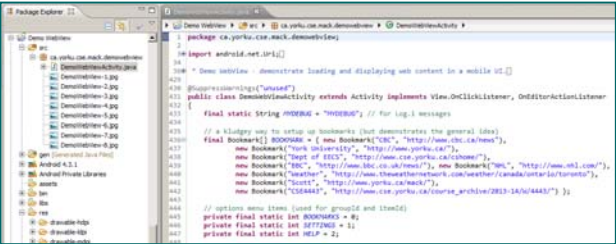
21


Example



Demo MapApp







© Scott MacKenzie

22

Map

- Widgets
- Custom Components
- Drawing
- Image View
- Video View
- Web View
- Map View
- Surface View
- View Groups

The screenshot shows the Android Studio API Guides for the 'Canvas and Drawables' section. The page title is 'Canvas and Drawables'. The main content area contains the following text:

The Android framework APIs provides a set 2D drawing APIs that allow you to render your own custom graphics onto a canvas or to modify existing Views to customize their look and feel. When drawing 2D graphics, you'll typically do so in one of two ways:

- Draw your graphics or animations into a View object from your layout. In this manner, the drawing of your graphics is handled by

The 'IN THIS DOCUMENT' sidebar lists the following sections:

- Draw with a Canvas
- On a View
- On a SurfaceView
- resource images
- resource XML

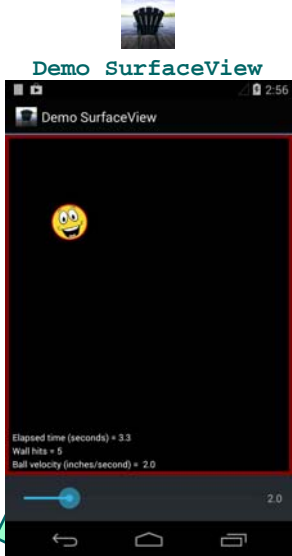
The callout box highlights the 'On a SurfaceView' section, which contains the following text:

On a SurfaceView

The `SurfaceView` is a special subclass of View that offers a dedicated drawing surface within the View hierarchy. The aim is to offer this drawing surface to an application's secondary thread, so that the application isn't required to wait until the system's View hierarchy is ready to draw. Instead, a secondary thread that has reference to a `SurfaceView` can draw to its own Canvas at its own pace.

To begin, you need to create a new class that extends `SurfaceView`. The class should also implement `SurfaceHolder.Callback`. This subclass is an interface that will notify you with information about the underlying `Surface`, such as when it is created, changed, or destroyed. These events are important so that you know when you can start drawing, whether you need to make adjustments based on new surface properties, and when to stop drawing and potentially kill some tasks. Inside your `SurfaceView` class is also a good place to define your secondary Thread class, which will perform all the drawing procedures to your Canvas.


Example



Demo SurfaceView

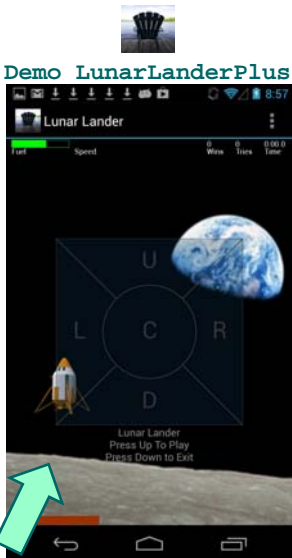
Elapsed time (seconds) = 3.3
Wall hits = 5
Ball velocity (inches/second) = 2.0

```
1 package ca.yorku.cse.mack.demosurfaceview;  
2  
3 import ca.yorku.cse.mack.demosurfaceview.MySurfaceView.MyThread;  
4  
12 * Demo SurfaceView - simple demo app that uses the <code>SurfaceView</code> class and a secondary[]  
256 public class DemoSurfaceViewActivity extends Activity implements SeekBar.OnSeekBarChangeListener  
257 {  
258     final static String #DEBUG = "#DEBUG"; // for Log.i messages  
259  
260     final static String ELAPSED_TIME_KEY = "elapsed_time";  
261     final static String WALL_HITS_KEY = "wall_hits";  
262     final static String VELOCITY_KEY = "velocity";  
263  
264     MySurfaceView mySurfaceView;  
265     SeekBar seekBar;  
266     TextView seekBarValue;  
267     CountdownTimer timer;  
268     float velocity;  
269 }
```



© Scott MacKenzie 25


Example



Demo LunarLanderPlus

Lunar Lander
Press Up To Play
Press Down to Exit

```
1 package ca.yorku.cse.mack.demolunarlanderplus;  
2  
3 import android.app.ActionBar[];  
4  
12 * Demo LunarLanderPlus - demo of the Lunar Lander arcade game[]  
256 public class DemoLunarLanderPlusActivity extends Activity implements View.OnTouchListener  
257 {  
258     final static String #DEBUG = "#DEBUG"; // for Log.i messages  
259     final int MAX_TOUCH_POINTS = 50;  
260     final int INVALID = -1;  
261  
262     private static final int MENU_EASY = 1;  
263     private static final int MENU_HARD = 2;  
264     private static final int MENU_MEDIUM = 3;  
265     private static final int MENU_PAUSE = 4;  
266     private static final int MENU_RESUME = 5;  
267     private static final int MENU_START = 6;  
268     private static final int MENU_STOP = 7;  
269     private static final int MENU_EXIT = 8;  
270  
271     /** A handle to the thread that's actually running the animation. */  
272     private LunarThread lunarThread;  
273  
274     /** A handle to the view in which the game is running. */  
275     private LunarView lunarView;  
276 }
```



© Scott MacKenzie 26

Map

- Widgets
- Custom Components
- Drawing
- Image View
- Video View
- Web View
- Map View
- Surface View
- View Groups

© Scott MacKenzie

27

Developers ▾ Design Develop Distribute

Training API Guides Reference Tools Google Services Samples

Android APIs API level: 19

android android.accessibilityservice android.accounts android.animation android.app android.app.admin android.app.backup android.appwidget android.bluetooth android.content android.content.pm android.content.res android.database.sqlite android.drm

Interfaces
ActionMode.Callback
ActionProvider.VisibilityListener
Choreographer.FrameCallback
CollapsibleActionView

public abstract class **ViewGroup**
extends View
implements ViewManager, ViewParent

java.lang.Object
↳ android.view.View
↳ android.view.ViewGroup

Known Direct Subclasses
AbsoluteLayout, AdapterView.T extends Adapter, DrawerLayout, FragmentBreadCrumbs, FrameLayout, GridLayout, LinearLayout, PagerTitleStrip, RelativeLayout, SlidingDrawer, SlidingPanelLayout, ViewPager

Known Indirect Subclasses
AbsListView, AbsSpinner, AdapterViewAnimator, AdapterViewFlipper, AppWidgetHostView, CalendarView, DatePicker, DatePickerFilter, ExpandableListView, and 21 others.

Class Overview

A ViewGroup is a special view that can contain other views (called children.) The view group is the base class for layouts and views containers. This class also defines the `ViewGroup.LayoutParams` class which serves as the base class for layouts parameters.

Also see `ViewGroup.LayoutParams` for layout attributes.

Developer Guides
For more information about creating user interface layouts, read the [XML Layouts developer guide](#).

Use Tree Navigation

A ViewGroup is a special view that can contain other views (called children.) The view group is the base class for layouts and views containers. This class also defines the `ViewGroup.LayoutParams` class which serves as the base class for layouts parameters.

© Scott MacKenzie

28

Developers | Design | Develop | Distribute

Training | API Guides | Reference | Tools | Google Services | Samples

Android APIs | API level: 19+

public abstract class **ViewGroup**
 extends View
 implements ViewGroup.LayoutParams

Known Direct Subclasses: AbsoluteLayout, AdapterView, DrawerLayout, FragmentBreadCrums, FrameLayout, GridLayout, **LinearLayout**, PagerTitleStrip, **RelativeLayout**, SlidingDrawer, SlidingPanelLayout, ViewPager

Known Indirect Subclasses: AbsListView, AbsSpinner, AdapterViewAnimator, AdapterViewFlipper, AdapterViewHostView, CalendarView, DatePicker, DatePickerDialog, Ex

Class Overview

see [ViewGroup.LayoutParams](#) for layout attributes.

also see [ViewGroup.LayoutParams](#) class for layout attributes.

Developer Guides

For more information about creating user interface layouts, read the [XML Layouts developer guide](#).

read the [XML Layouts developer guide](#).

ESC

© Scott MacKenzie 29

Postscript

- Display properties include
 - Size (inches, cm)
 - Resolution
 - Pixel density
 - Orientation (natural, actual)

Demo Display

Display Properties

Pixel density: 2.0 (relative to 160 dpi)

Natural orientation: PORTRAIT

Current display size: w = 768, h = 1184

Current orientation: PORTRAIT

Current rotation: 0 degrees

One-inch button: Button

Half-inch circle: Circle

ESC

© Scott MacKenzie 30

Homework

- Download all the “demo” programs from earlier slides
- Import into Android projects
- Connect your device and click “Run”
- Observe UI behaviour
- Ready the APIs (and links within)
- Study source code

© Scott MacKenzie

31

Thank You

© Scott MacKenzie

32