ANDROID CORE COMPONENTS

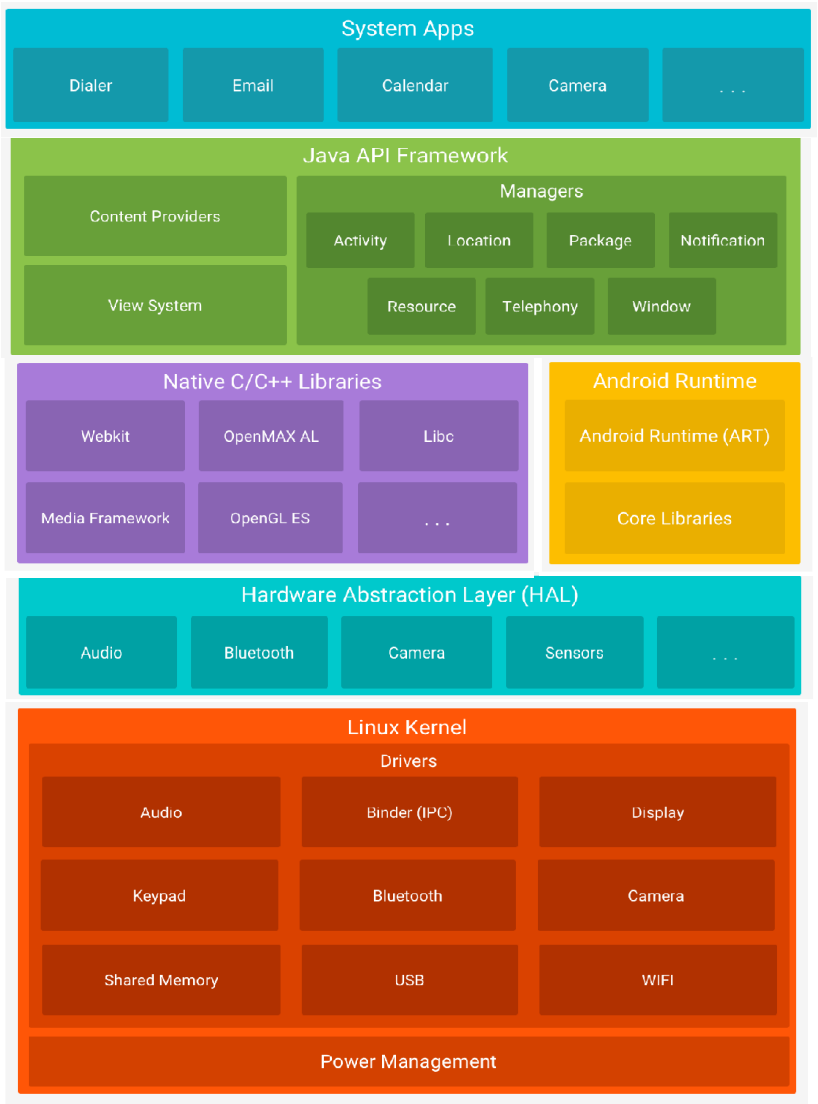
Roberto Beraldi

Android architecture

JAVA/KOTLIN

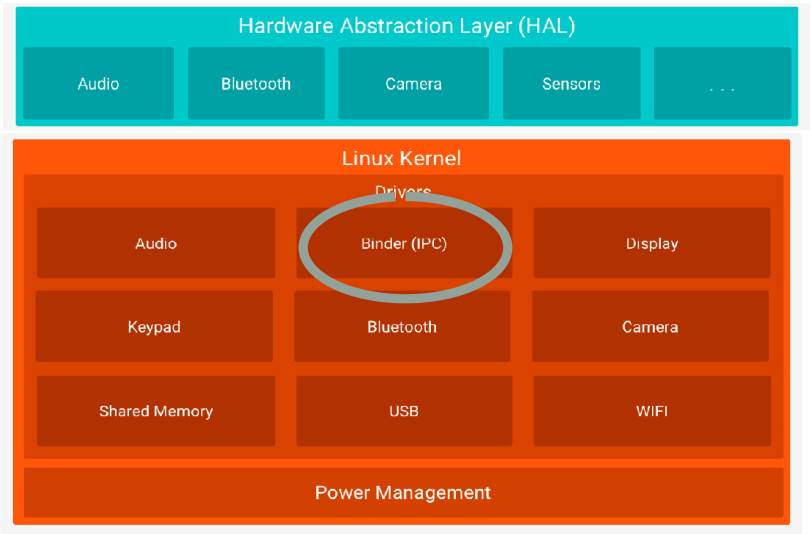
C/C++

KERNEL

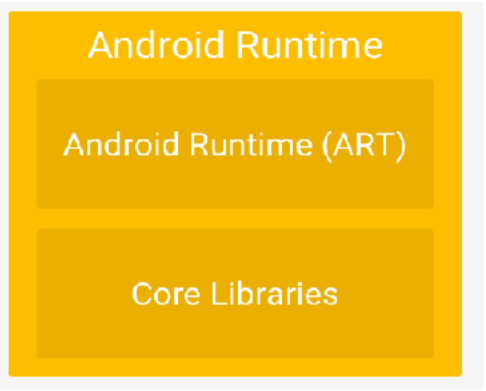


Android architecture (kernel)

- Provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc.
- Implements network stack, multitasking, etc..
- Binder is a special driver designed to provide secure communication between apps

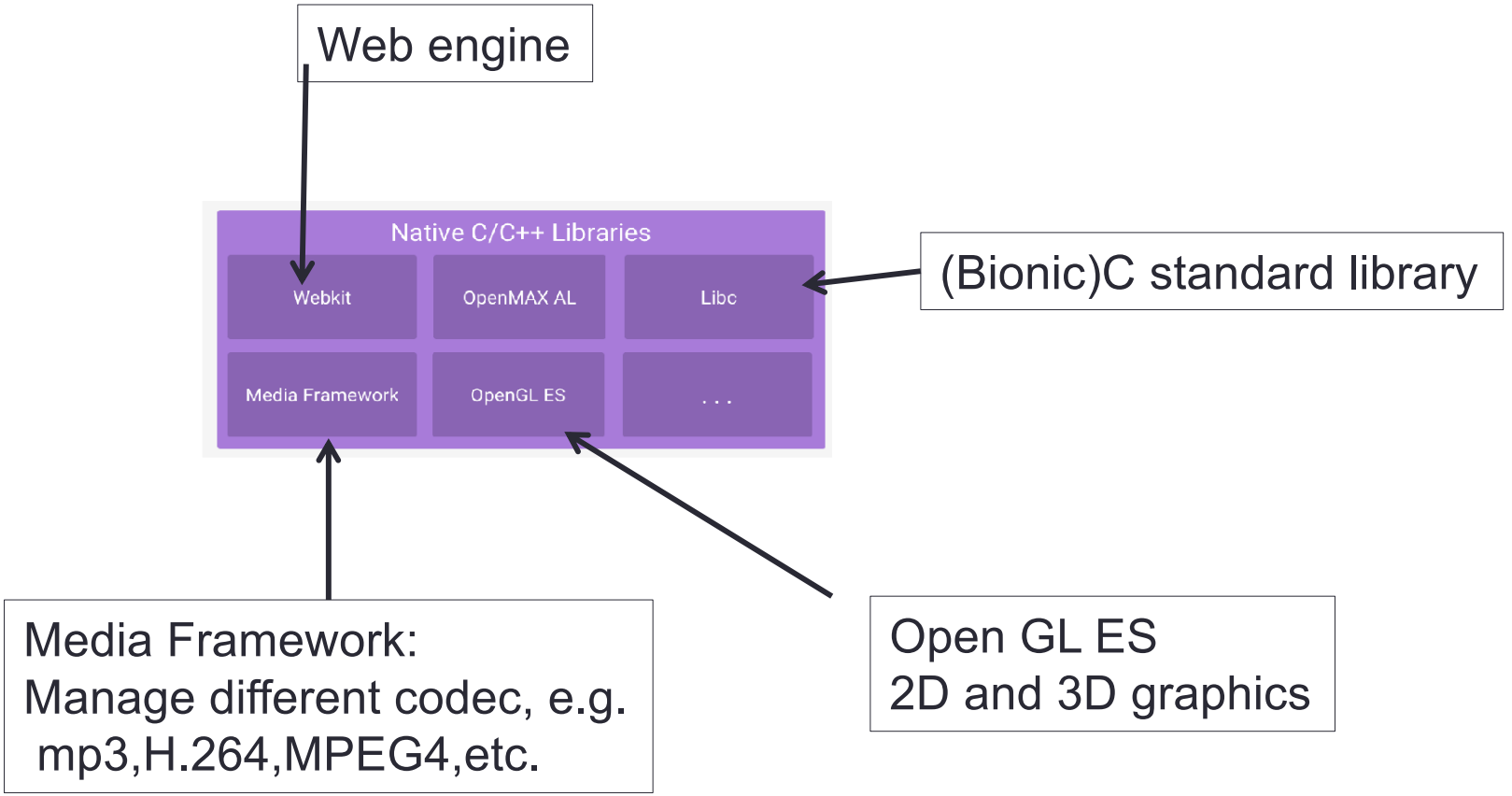


Android architecture: ART

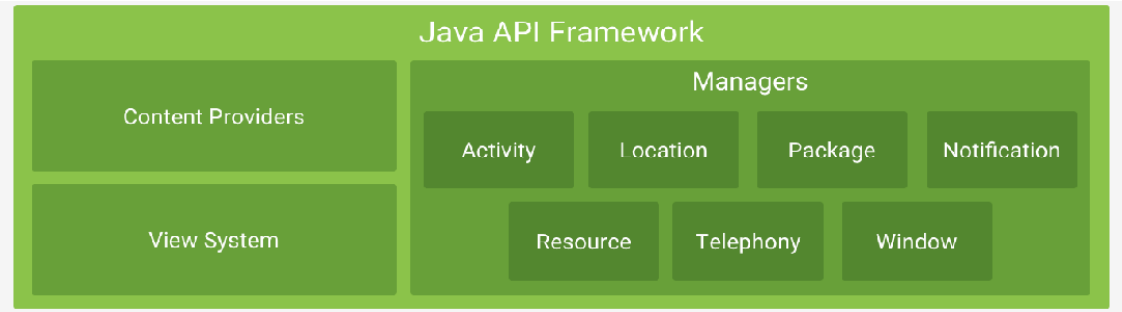


- Introduced with android 5 (before DVM)
- Register based architecture
- Code runs directly on hw (as opposed to DVM)
- Each app runs in its own process and with its own instance of the Android Runtime (ART)

Native SW libraries (C/C++)



Android framework

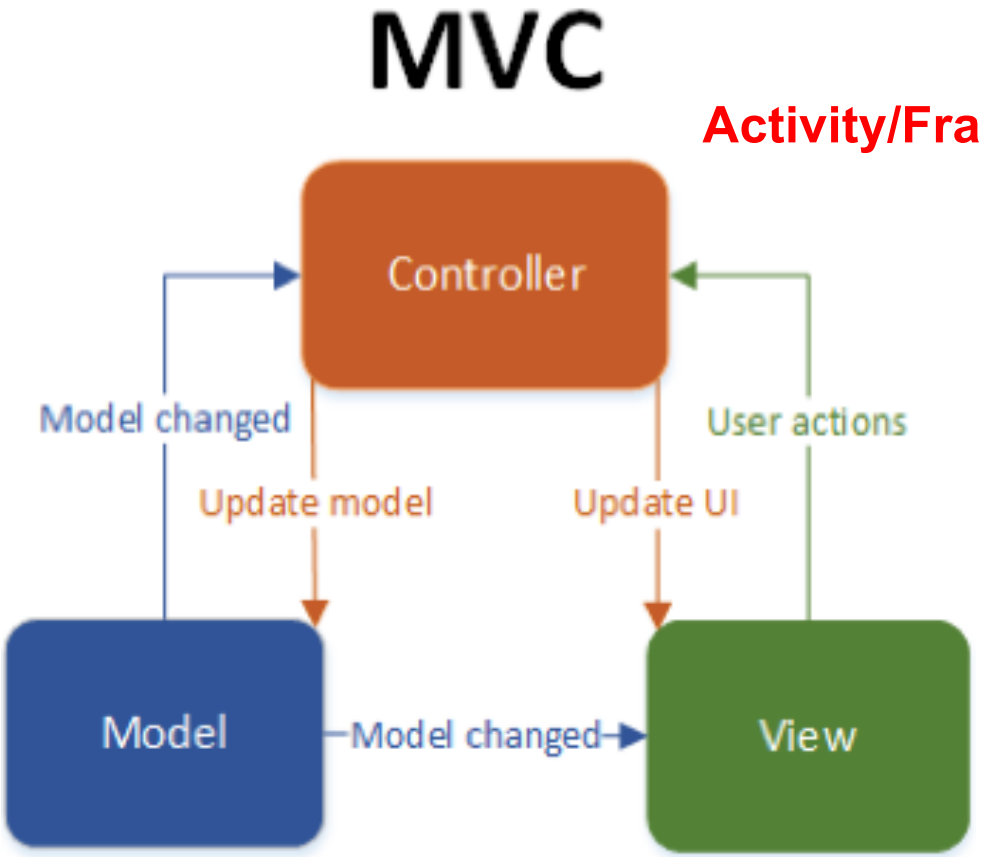


- The entire feature-set of the Android OS is available through Java packages.
- **android.app – Provides access to the application model and is the cornerstone of all Android applications.**
- **android.content – Facilitates content access, publishing and messaging between applications and application components.**
- **android.database – Used to access data published by content providers and includes SQLite database management classes.**
- **android.graphics – A low-level 2D graphics drawing API including colors, points, filters, rectangles and canvases.**
- **android.hardware – Presents an API providing access to hardware such as the accelerometer and light sensor.**
- **…**

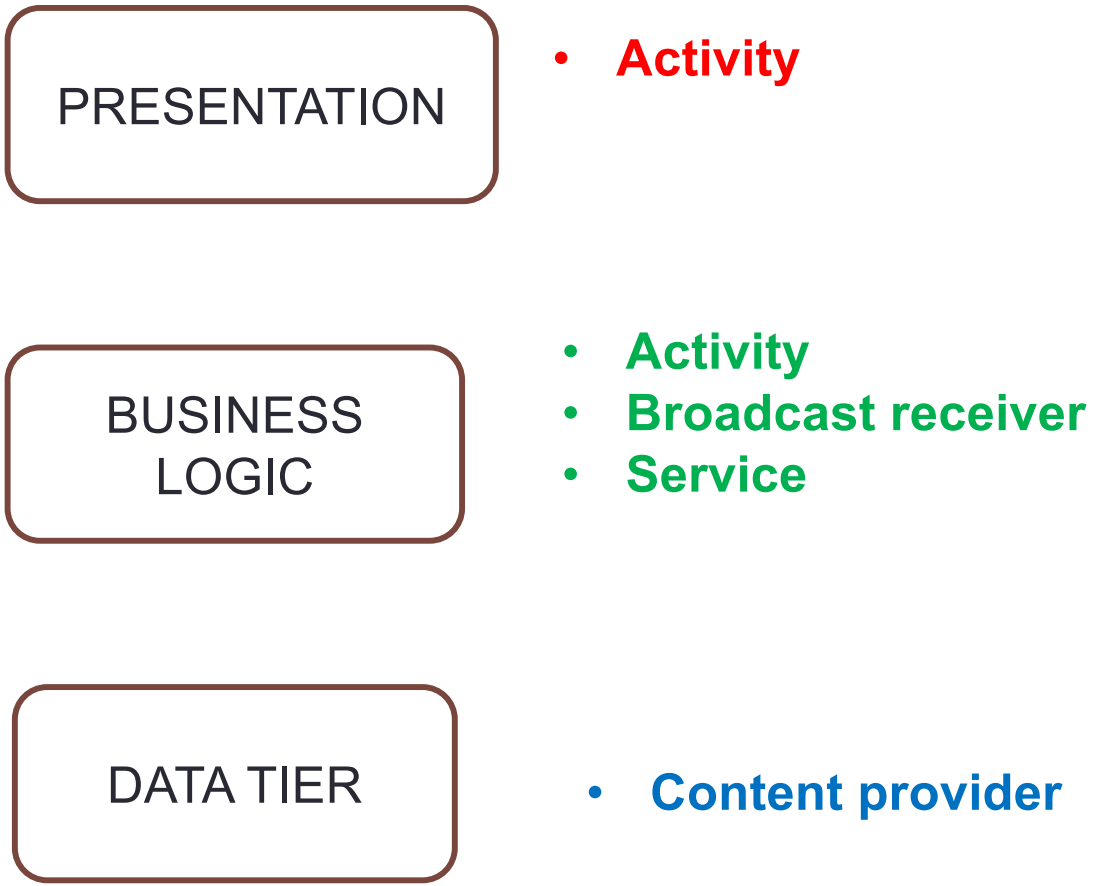
Core software components (android.*)

- Activity / Fragment
- Service
- Broadcast receiver
- Content provider

Mapping core components to MVC

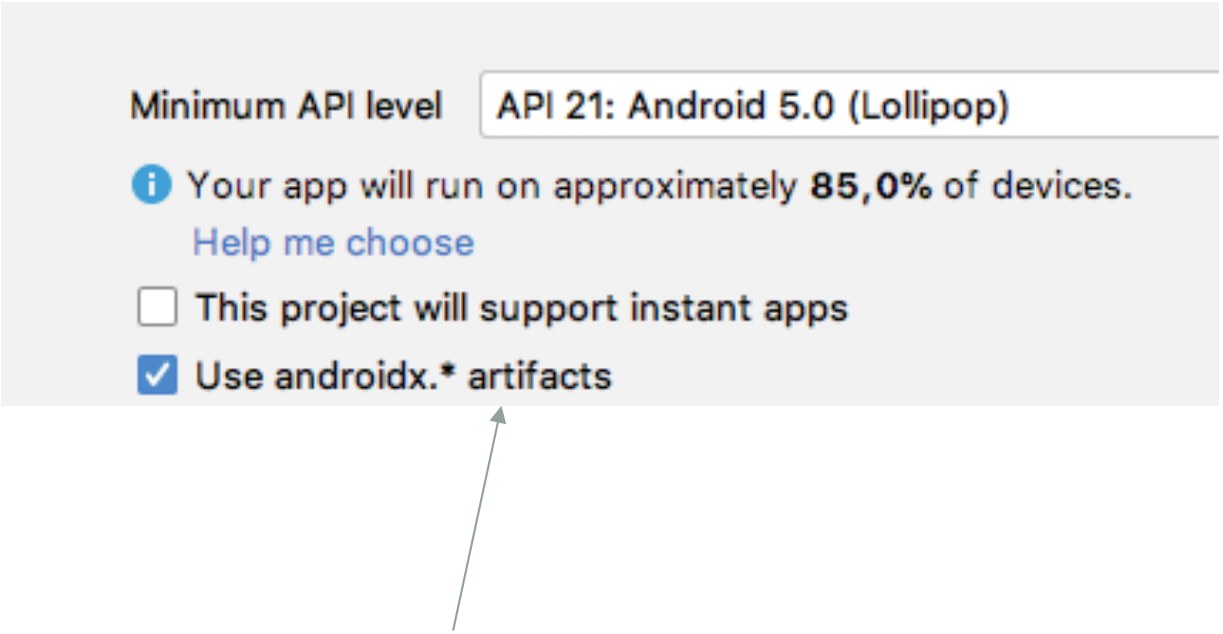


Mapping core components to the three-layers architecture



Jetpack components (**androidx.***)

- Data Binding
- Lifecycles
- LiveData
- Navigation
- Paging
- Room
- ViewModel
- WorkManager



Getting external libraries

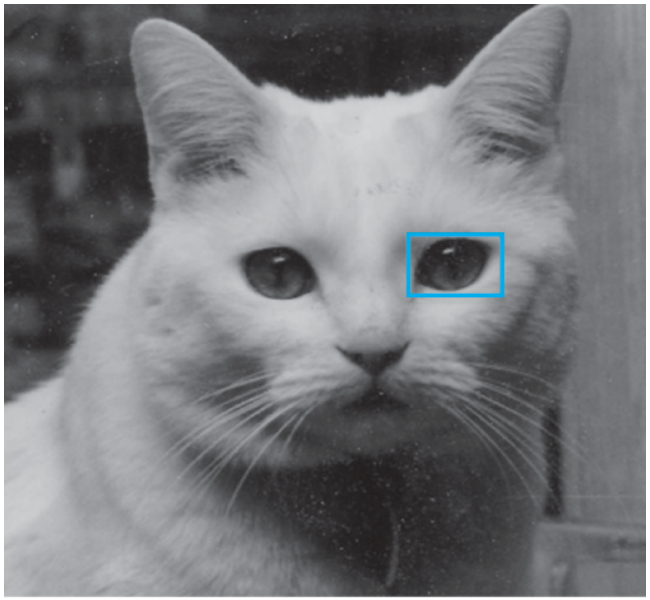
- Useful libraries can be easily imported. For example:
- dependencies {
  implementation 'org.tensorflow:tensorflow-lite:0.0.0-nightly'
  }

- Tensorflow light (ML)
- Picasso (Image downloading)
- ...

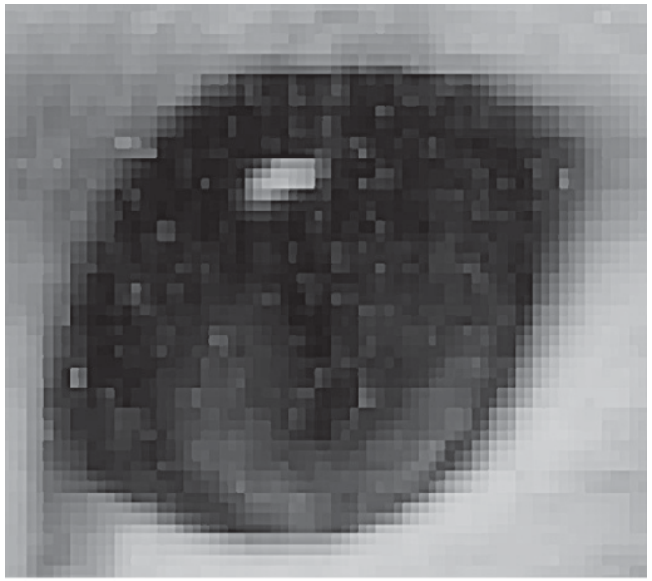
Digital images

- An image is a W x H matrix of pixels (Width, Height)
- **Pixel**: the smallest visual unit that can be controlled, e.g., its colors, and displayed (the atom of the display)
- **Size**: total number of pixels
  - e.g., 2048×1536, sometimes given as the result of the product (usually in MPixels)
- **Aspect Ratio** W/H
  - e.g., 4:3 or 16:9, or 1.77

Pixel



(a)



(b)

(a) Image of a cat

(b) Details of the image showing pixels

An example: grayscale image

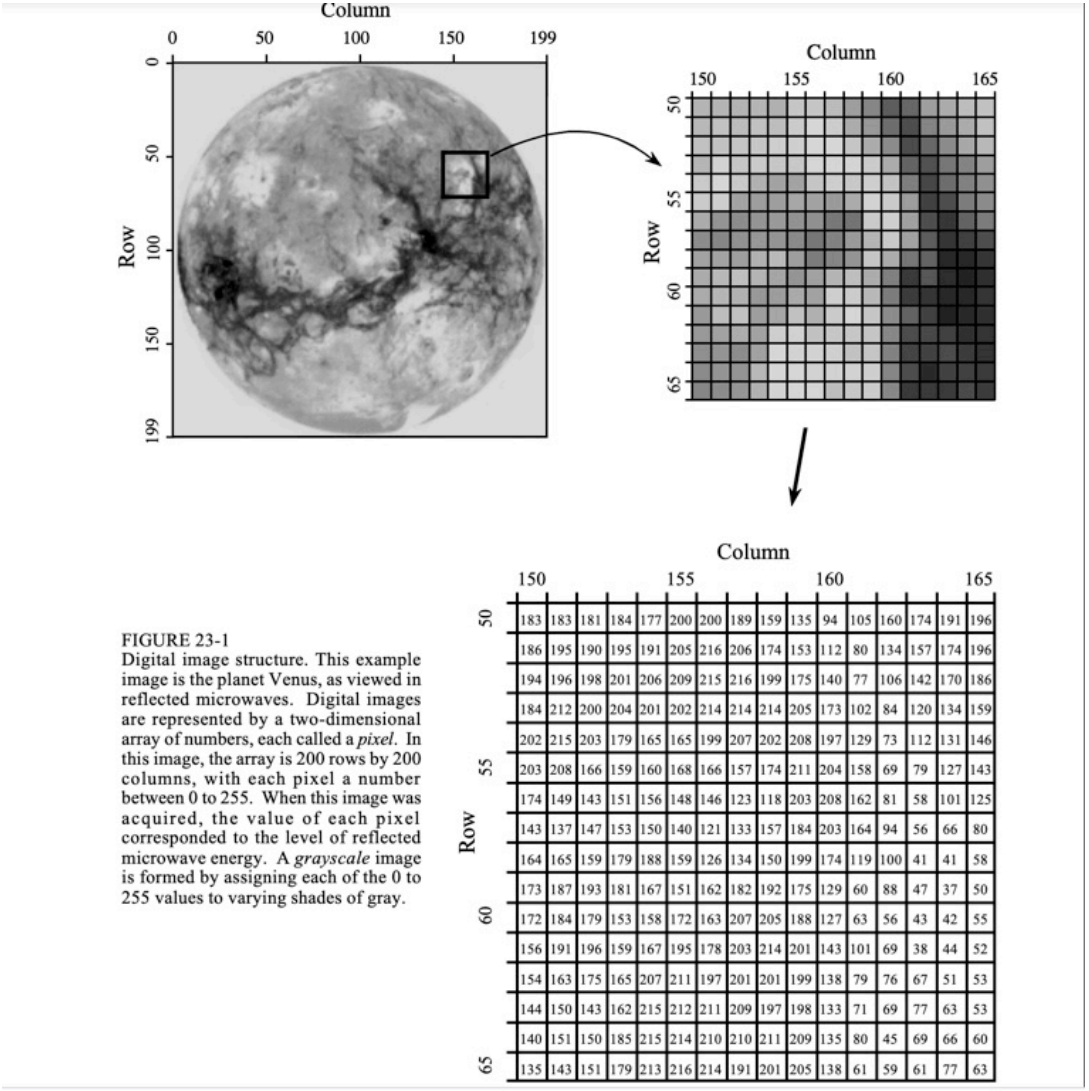
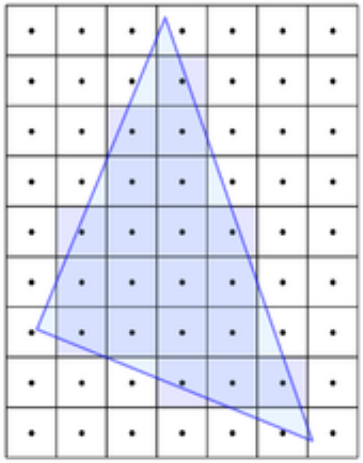


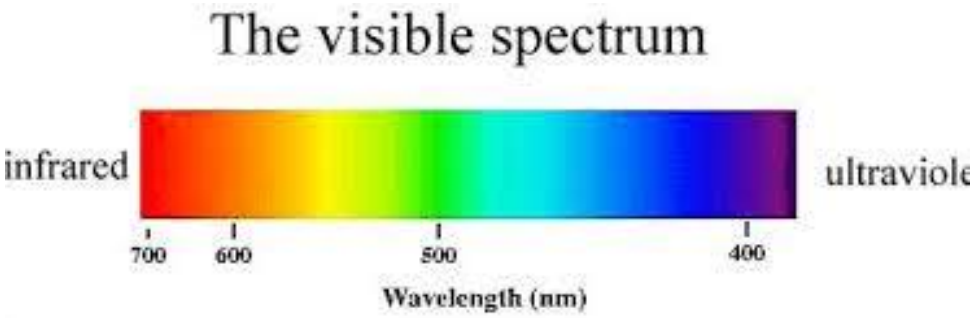
FIGURE 23-1
Digital image structure. This example image is the planet Venus, as viewed in reflected microwaves. Digital images are represented by a two-dimensional array of numbers, each called a *pixel*. In this image, the array is 200 rows by 200 columns, with each pixel a number between 0 to 255. When this image was acquired, the value of each pixel corresponded to the level of reflected microwave energy. A *grayscale* image is formed by assigning each of the 0 to 255 values to varying shades of gray.

Rasterization

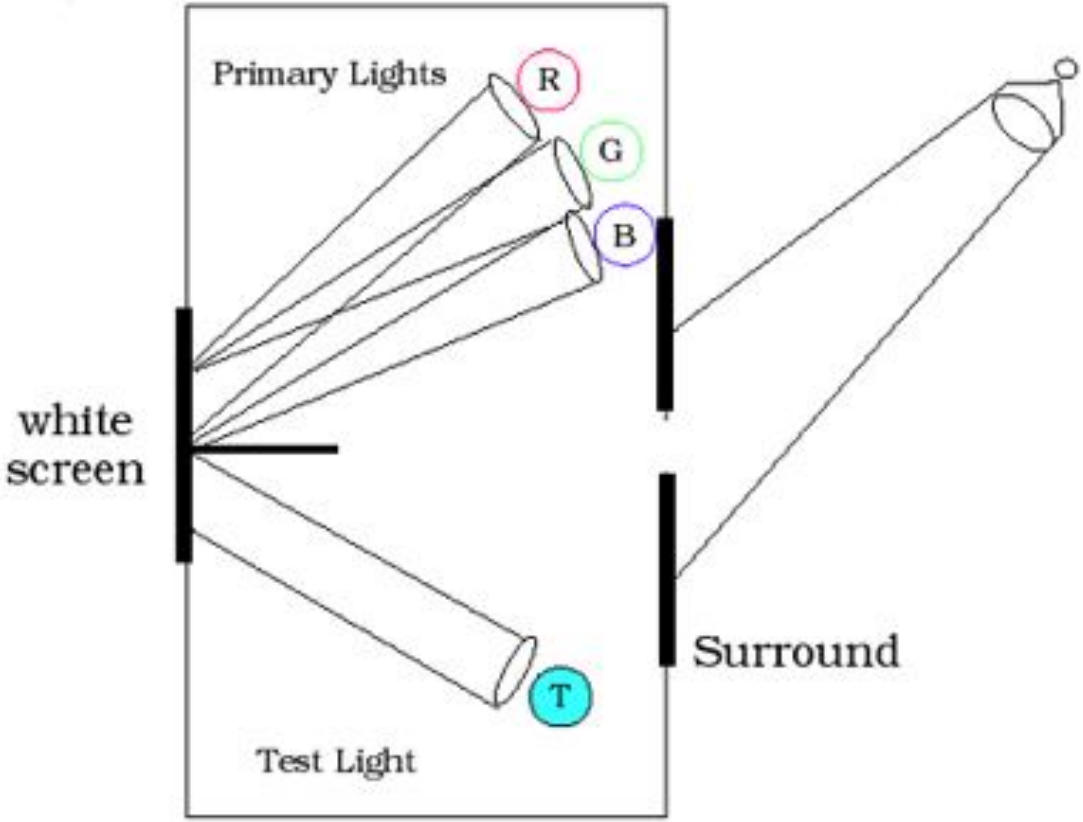


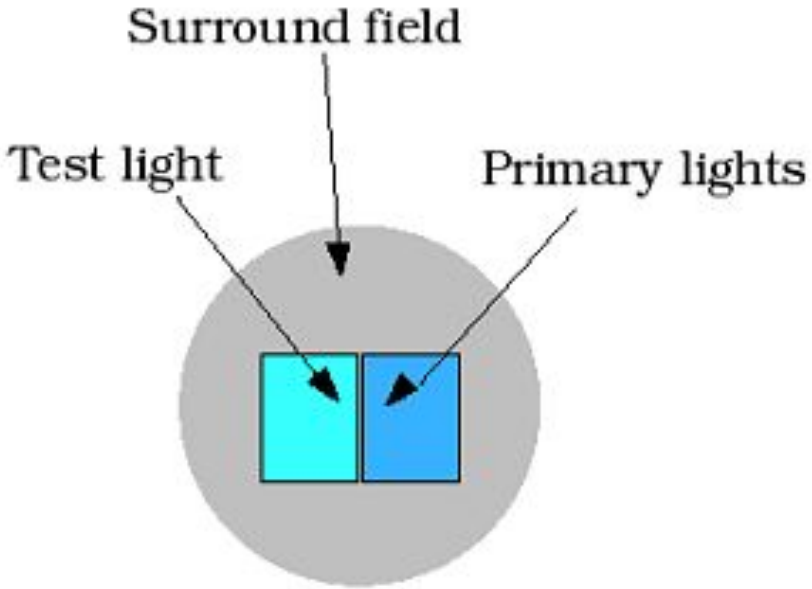
Colours

- The colour of the visible light is determined by its wavelength
- The same effect of a colour on the human eyes can be determined by summing **three** different colours of proper intensity, the **primary colours**
- This was established by an experiment in the 1930s by the Commission International on Illumination (CIE)
- Note:
  - the three wavelengths do not mix and generate a single wavelength
  - Humans do not distinguish among them, but the spectrum is different



Color matching experiment





Vector images

- Describe the image in terms of points, which are connected by lines and curves to form polygons and other shapes
- The most popular format are: PDF,SVG and EPS
- SVG (Scalable Vector Graphics) is a Web graphics language allowing for creating static or dynamic images


```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow
</svg>
```

Android >= 5

```
<shape android:shape="oval">
    <solid android:color="#e42828"/>
    <stroke android:color="#3b91d7" android:width="5dp"/>
    <!-- Set the same value for both width and height to get a circular shape -->
    <size android:width="250dp" android:height="250dp"/>
</shape>
```

iOS supports PDF

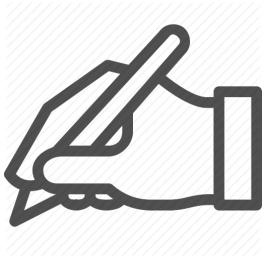
Device Coordinate Frame

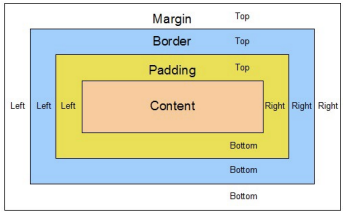
- Screen size, expressed in pixels
  - X_{max} = Width, e.g. 1440
  - Y_{max} =Height, e.g. 2960
- Origin at the left-upper corner
  - X increases left to right
  - Y increases downwards
  - Values are float (although pixels are discrete)
- Viewport: surface available for drawing (depends on the device)

How to draw on the screen?

- **Drawable**, “anything can be drawn”, e.g. bitmap, vector images, .png, 9patch,.. on the screen (only output)

- **View,** in addition to draw, it receives touch events





The View System

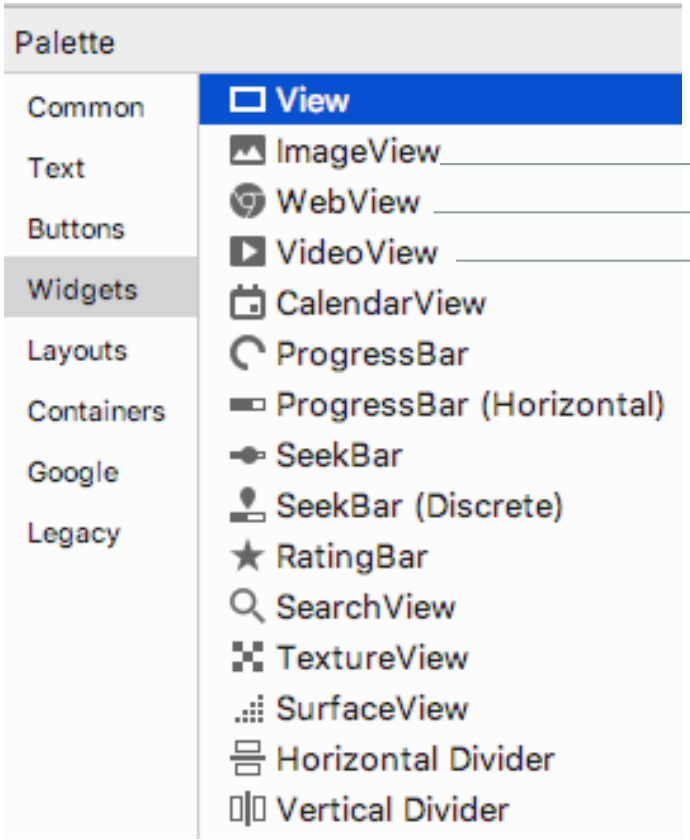
- A system for organizing GUI
- Screen = Tree of views
- View = rectangular shape on the screen that ‘knows how to draw itself’ wrt to the containing view
- ViewGroup: A view that contains other views

Using View

- android.view.View

- android.opengl.GLSurfaceView

Example of views

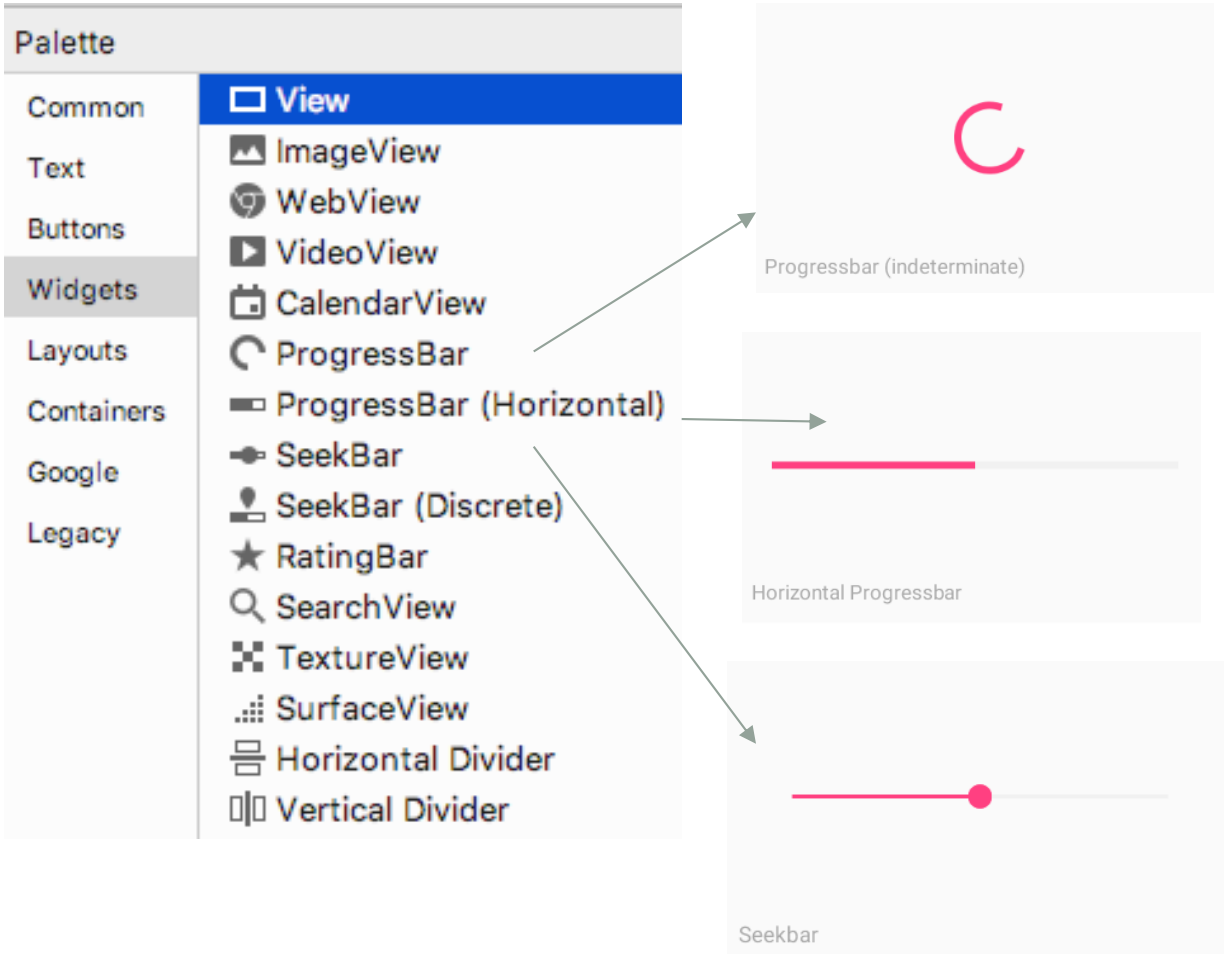


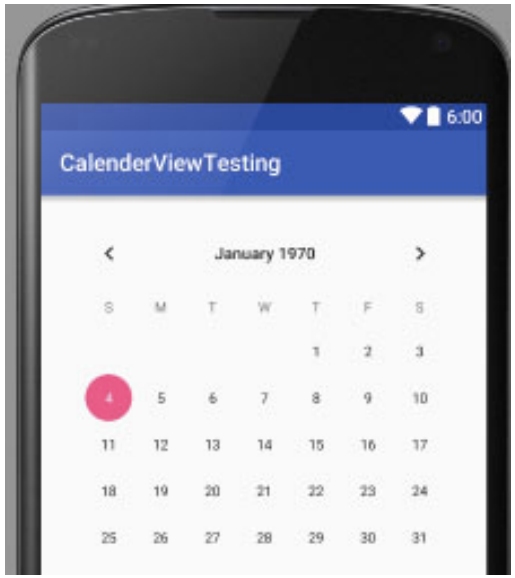
hosts a static image ..

a web page (used for mobile web app)

a video

Example of views

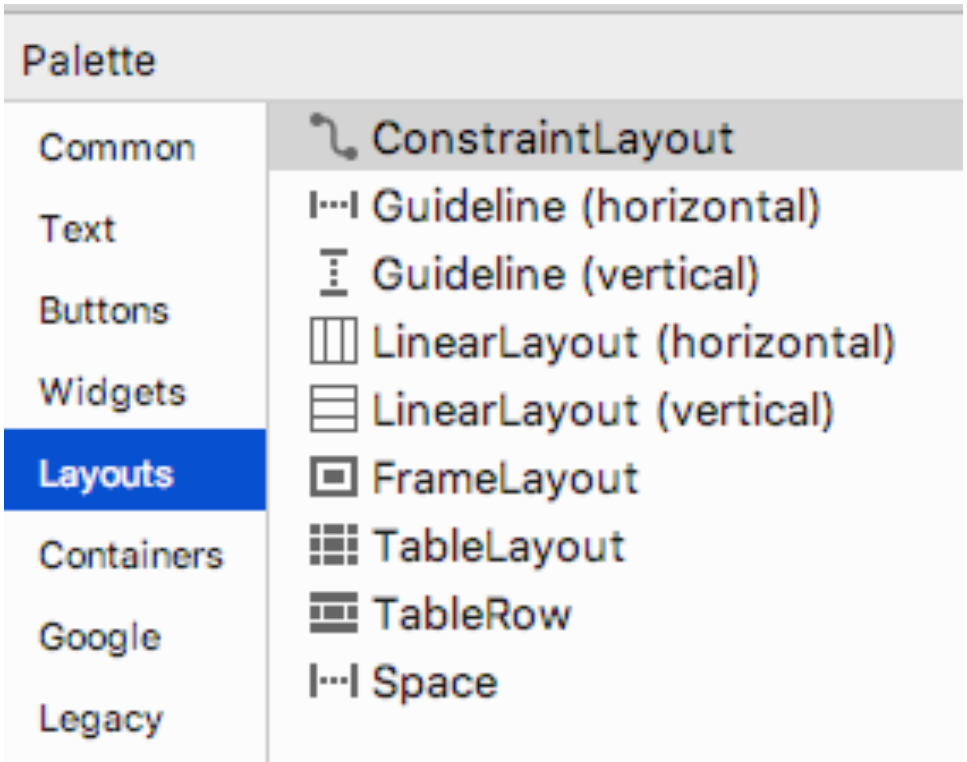




Calendar view

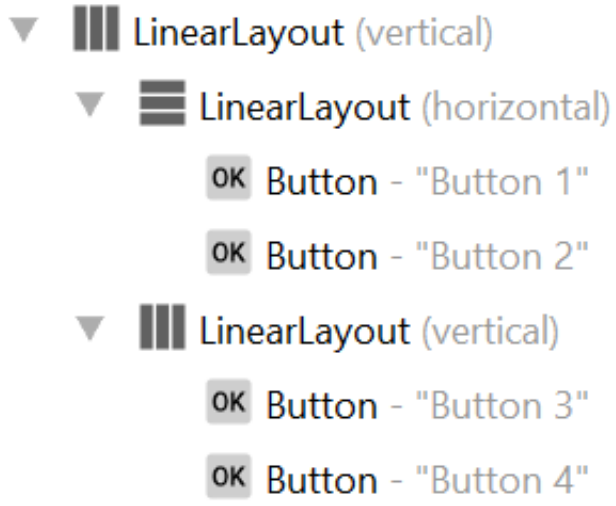
Layouts

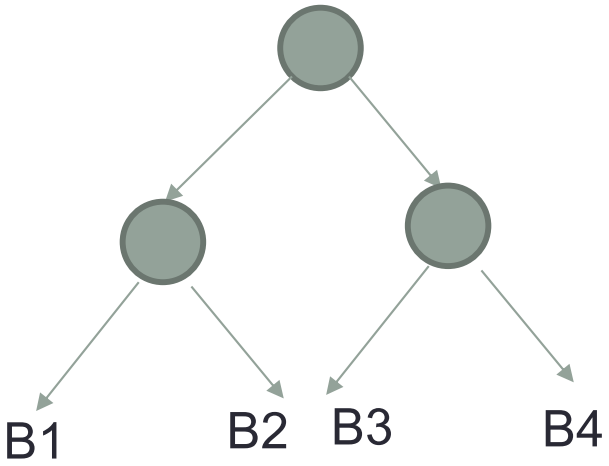
- Wiewgroups used to host other views

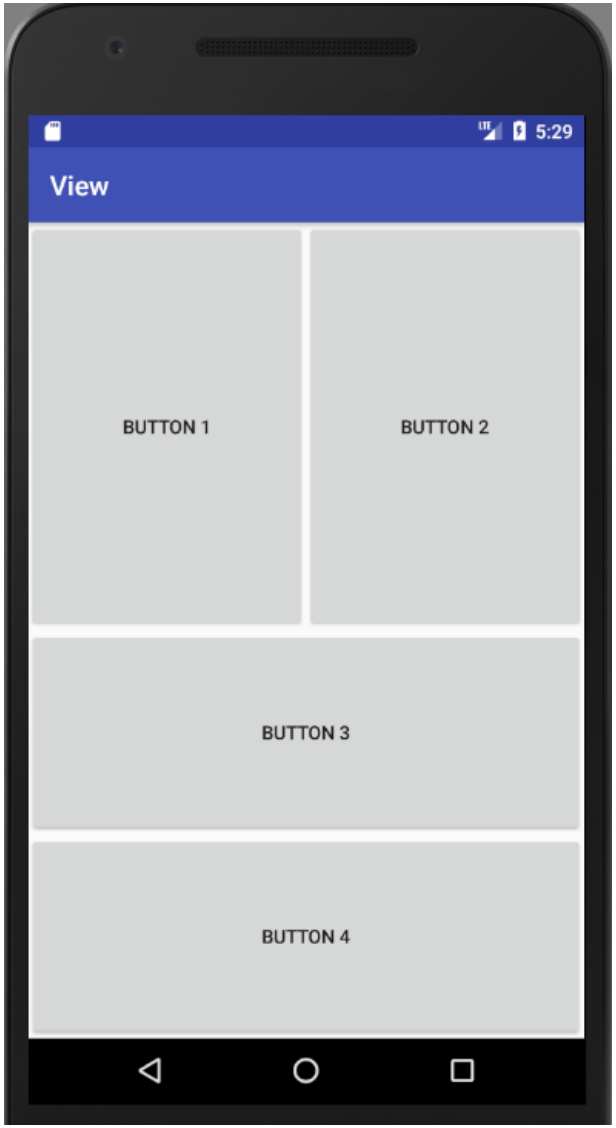


- Constrained and Linear are the most used ones

Layouts example



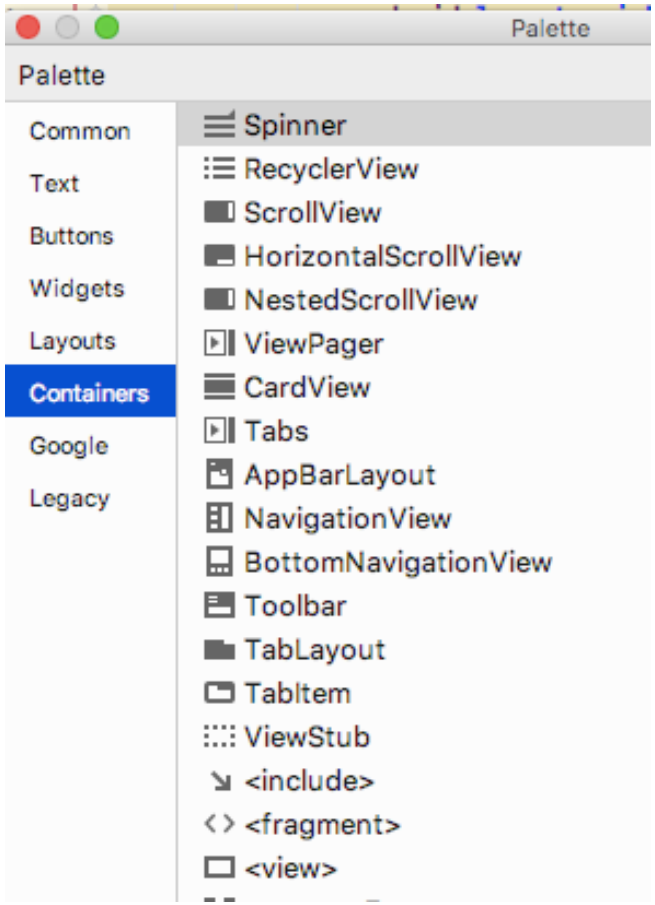


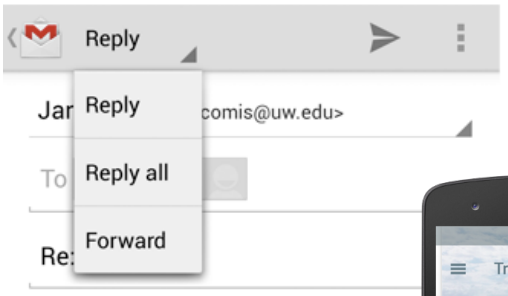


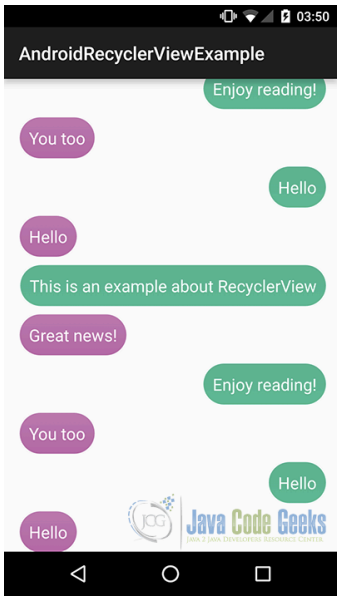
Each View:
H=matchParent
W=matchParent
Weigth=1

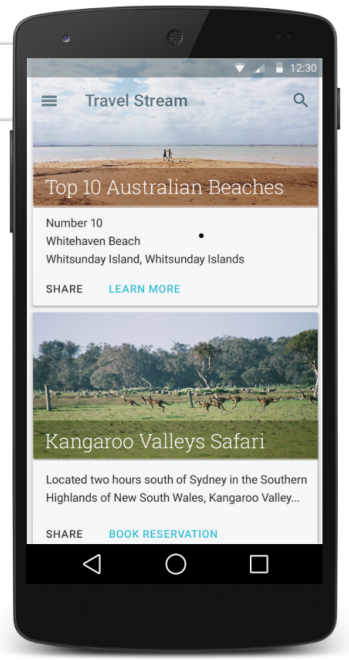
Containers

- Wiewgroups used to host dynamic content, or content bigger than the available screen size (support to scroll, etc)



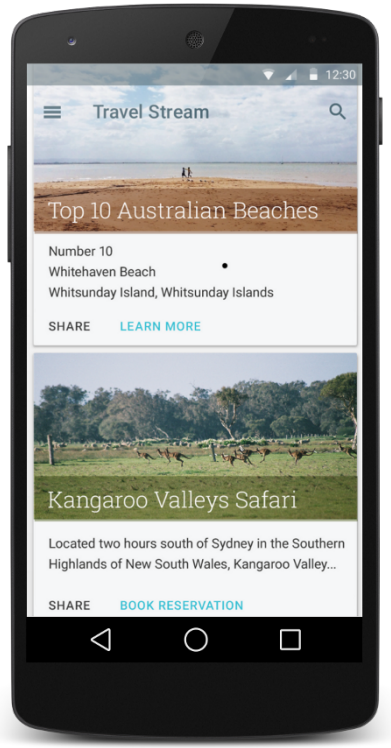




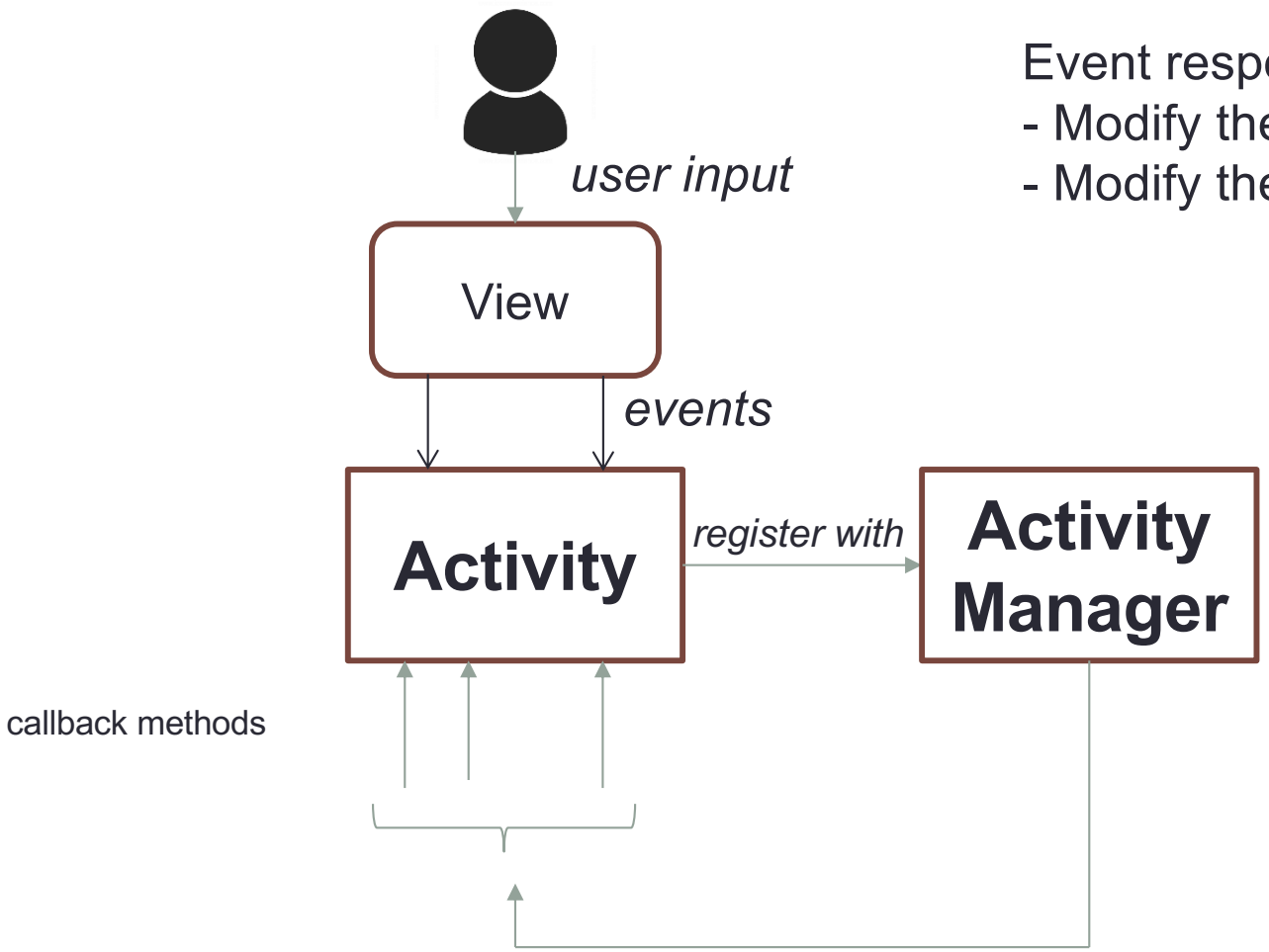


Containers and dynamic views

- Dynamic content management requires special care to optimize performance
  - content should in general be retrieved from slow source (internet)
  - Use a separate thread to execute content download
  - View elements must be added and the view re-rendered
- Very common pattern is a scrollable list of items
- For this there are special views (RecyclerView)



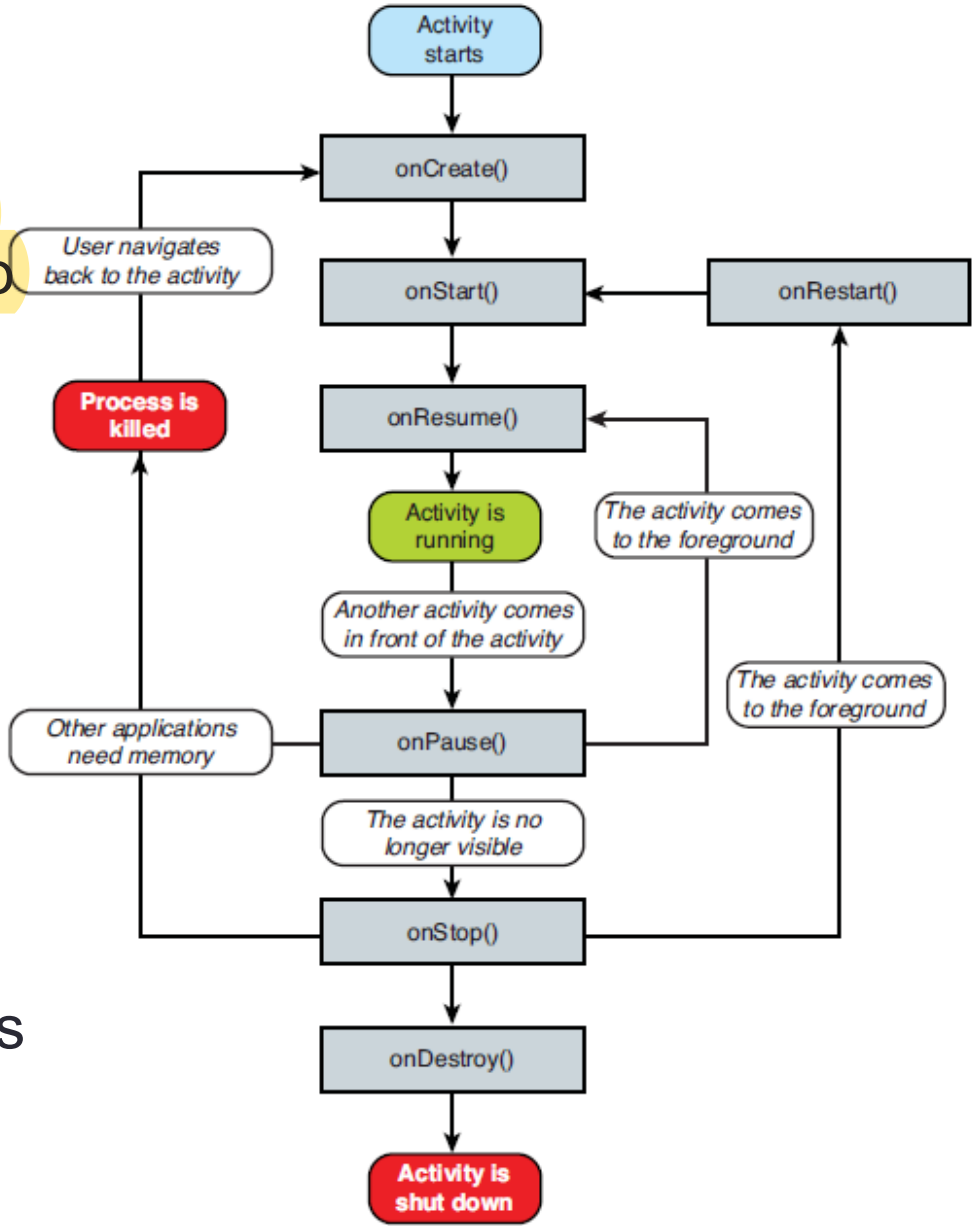
Activity and activity manager



Event response:
- Modify the data (model)
- Modify the view

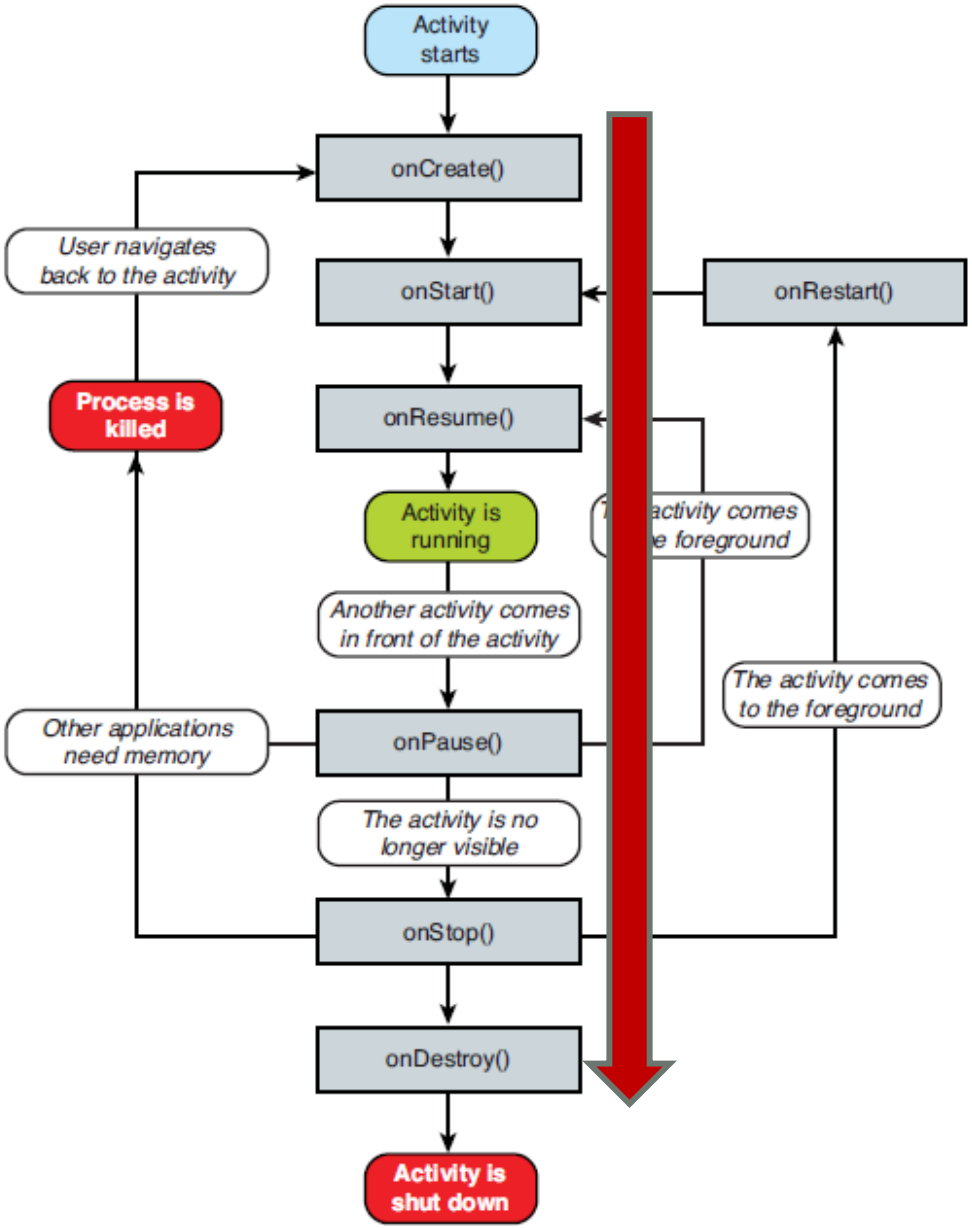
Activity lifecycle

- Each activity in an application goes through its own **lifecycle**: it responds to a set of methods called by "android"
- When an activity is created, is the *onCreate(…)* method executed
- the *onDestroy()* kills the activity executed
- In between, various methods are called allowing the activity to be managed



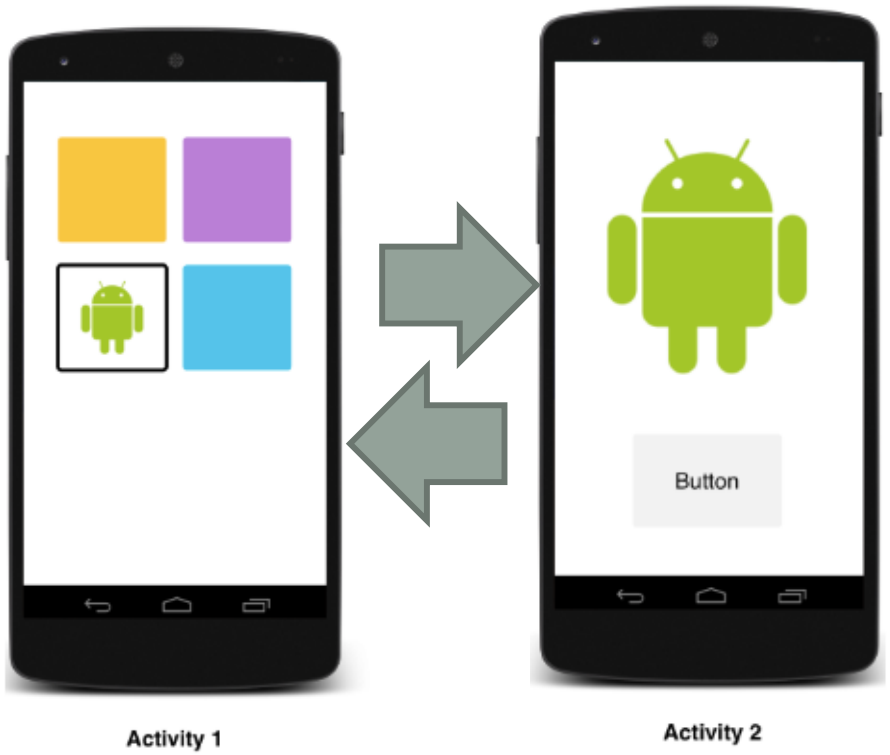
Activity lifecycle

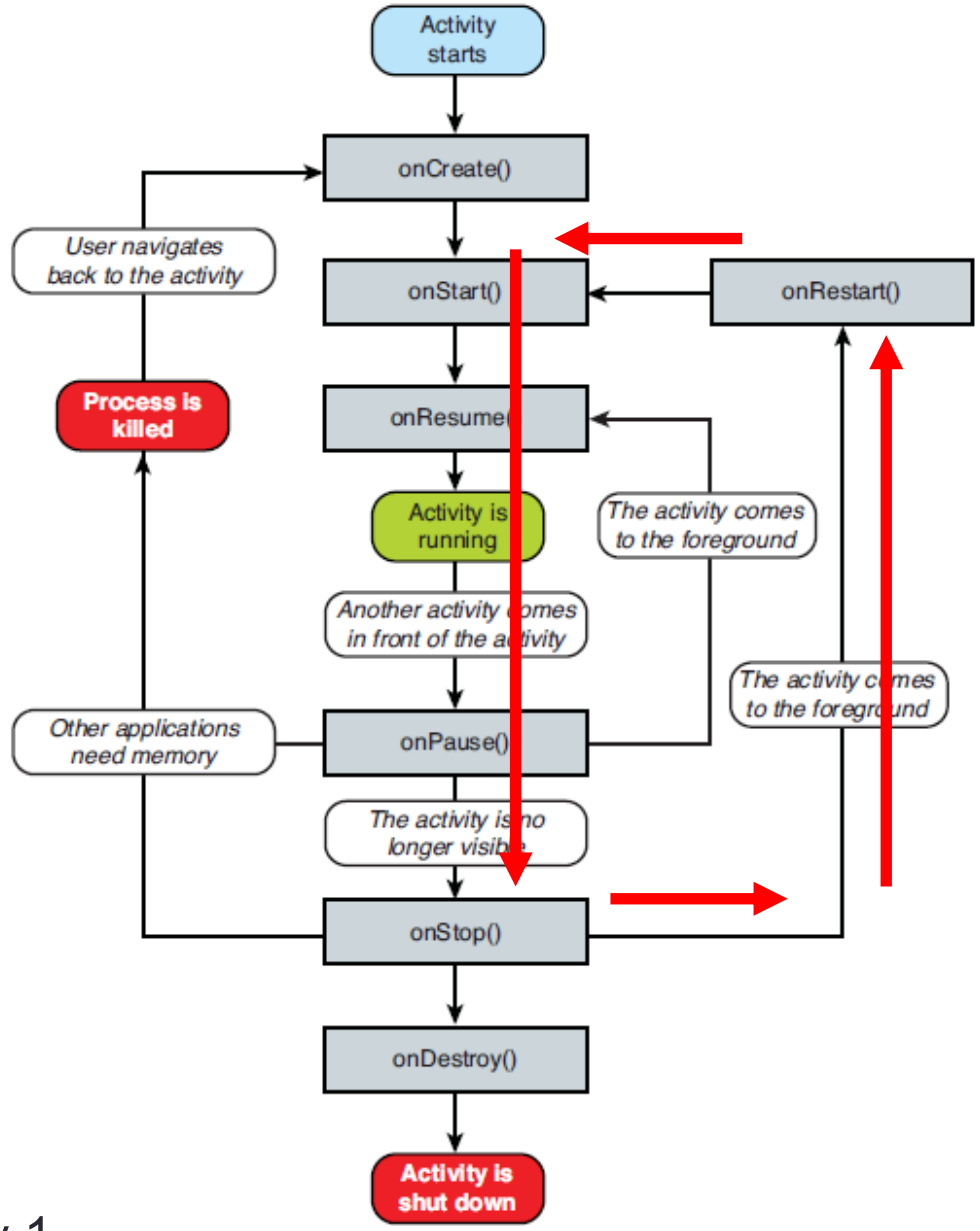
- Create-destroy path



Activity lifecycle

- Visible-Unvisible-Visible

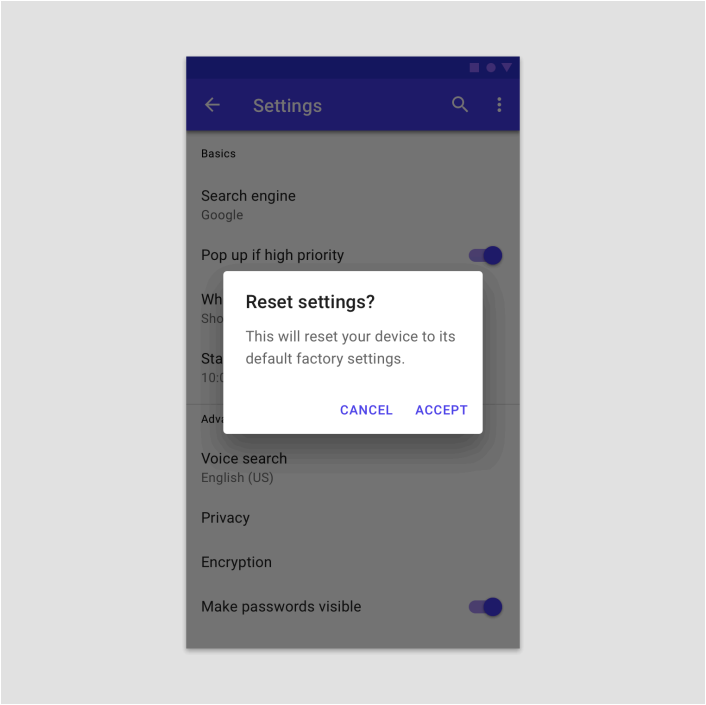


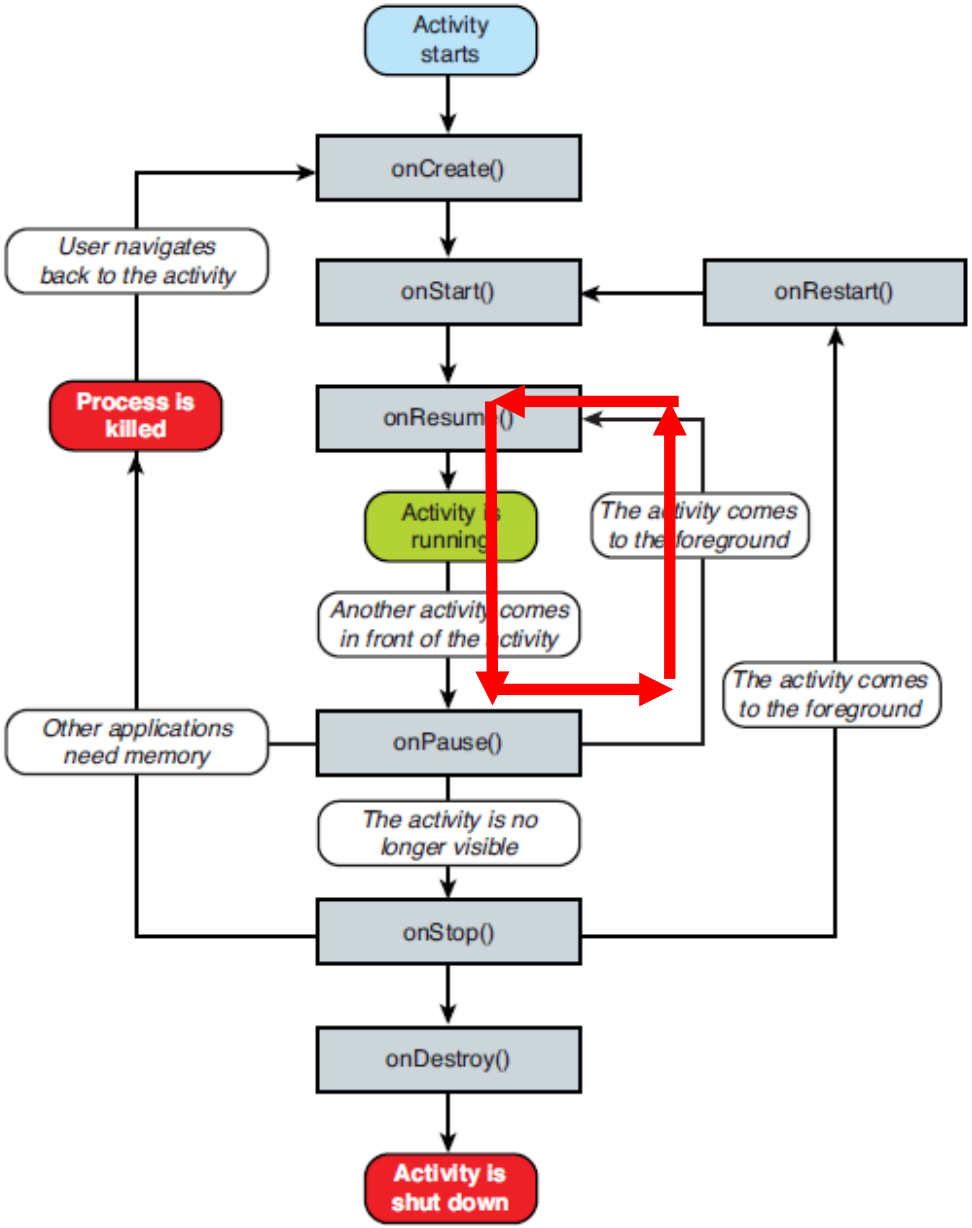


The arrow is the path of Activity 1

Activity lifecycle

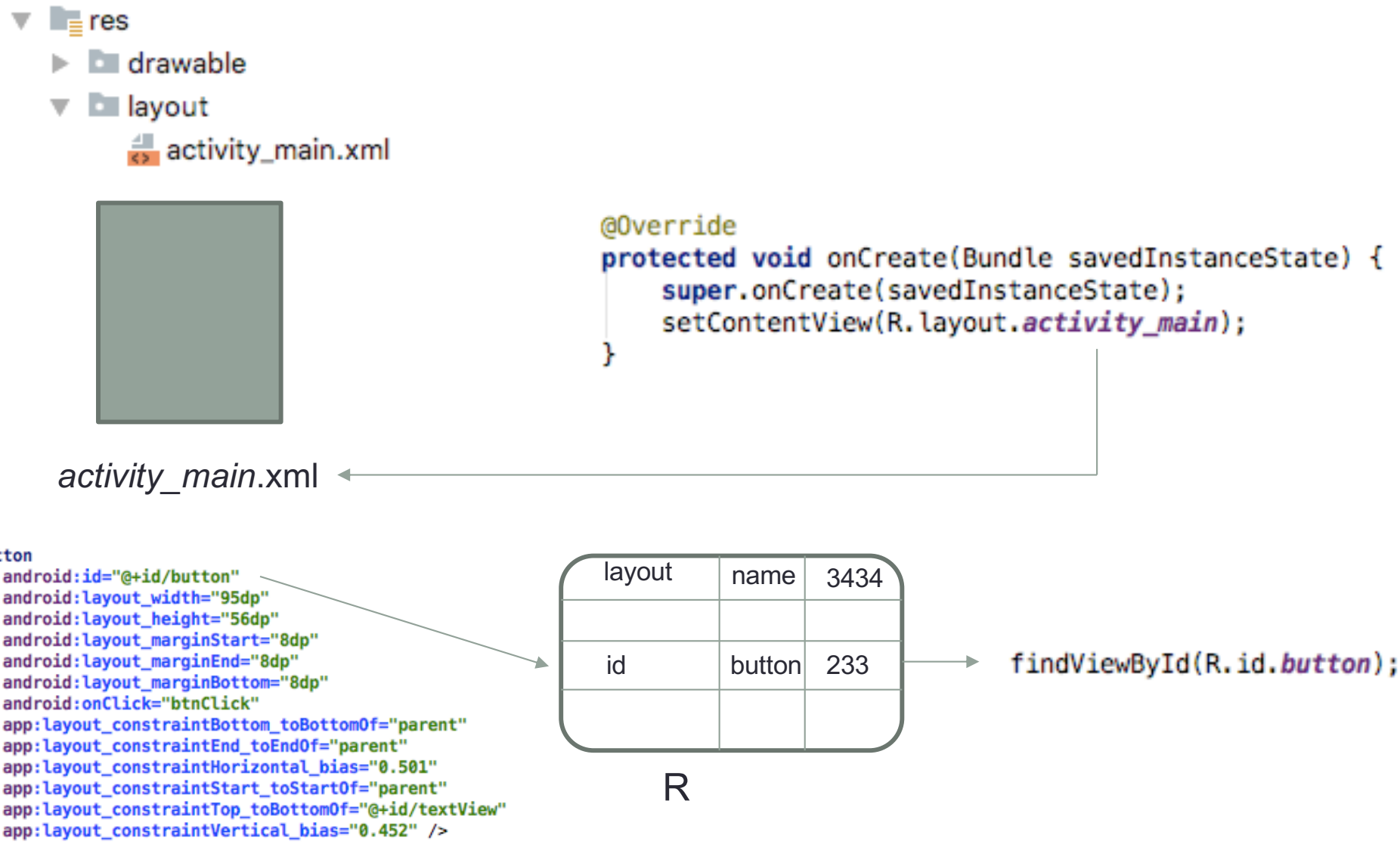
- Visible-Partially_visible-Visible path





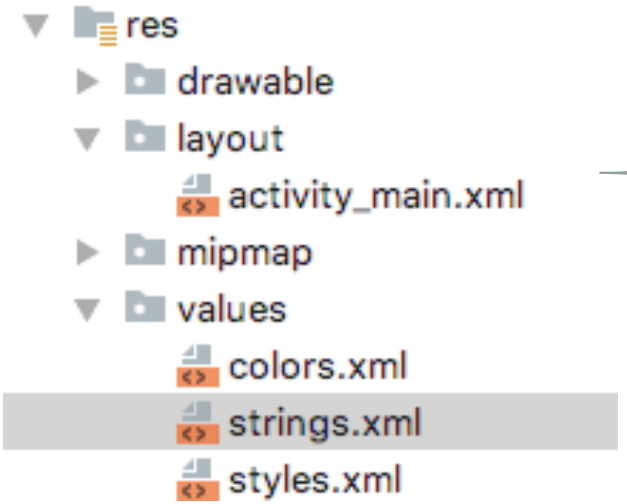
Hello World example

Object view representation



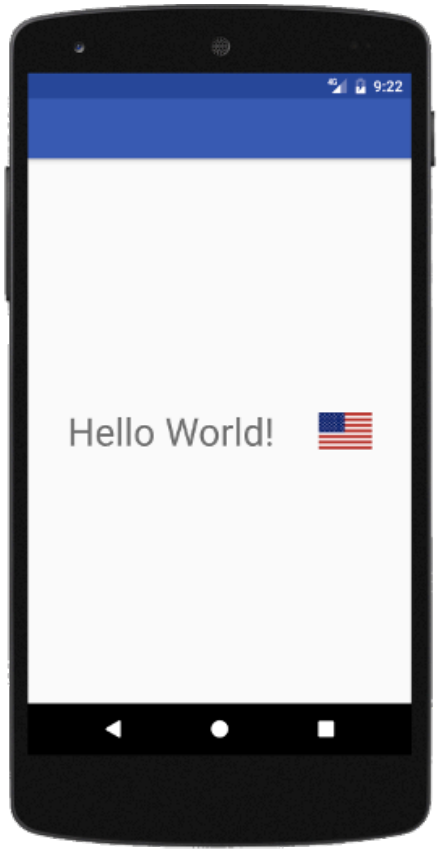
Responsive layout

- The same name is repeated inside different folders, with name **layout-*qualifier***



layout-**sw800dp**
layout-**land**
..

Multilanguage support




```
res/
    values/
        strings.xml
    values-b+es/
        strings.xml
    mipmap/
        country_flag.png
    mipmap-b+es+ES/
        country_flag.png
```

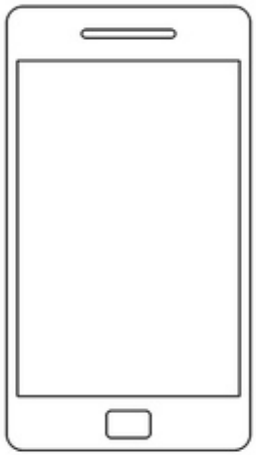
```xml
<resources>
    <string name="hello_world">Hello World!</string>
</resources>
```

```xml
<resources>
    <string name="hello_world">¡Hola Mundo!</string>
</resources>
```

Recap

- `res/drawable/` (required directory holding at least one graphic file, for the app's icon on Google Play)
- `res/layout/` (required directory holding an XML file that defines the default layout)
- `res/anim/` (required if you have any `res/anim-<qualifiers>` folders)
- `res/xml/` (required if you have any `res/xml-<qualifiers>` folders)
- `res/raw/` (required if you have any `res/raw-<qualifiers>` folders)

1. App refers to a symbol (eg. main)
2. xml refer to symbol (eg, @/string/hello)

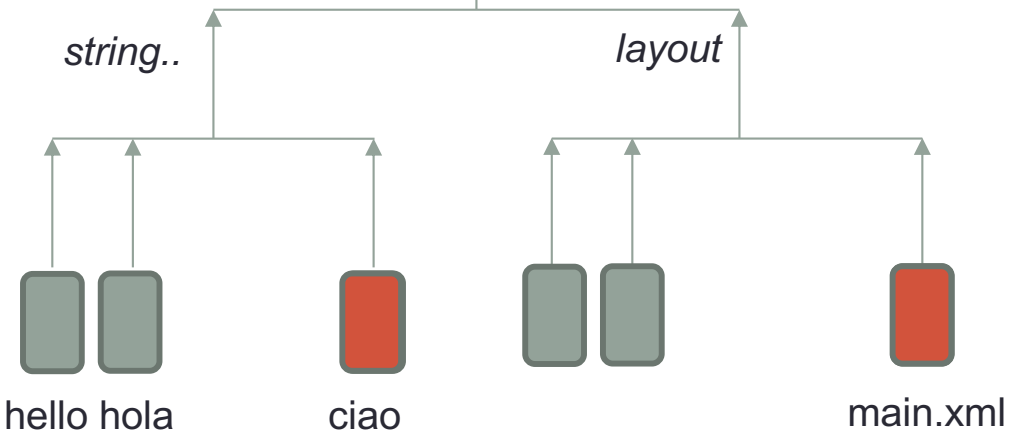


3. Mapping depends on locale setting
7.0+ improved symbol resolution

@string/hello → string-eng/hello

3. Mapping depends on screen properties
(dpi,viewport,orientation,..)

main → /layout-sw800dp/main

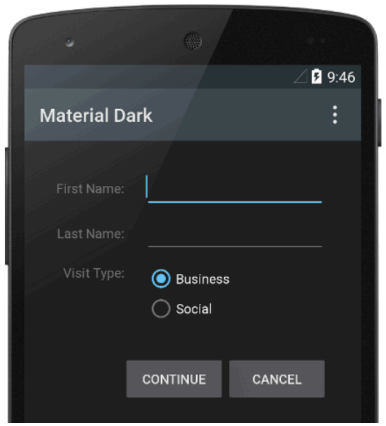


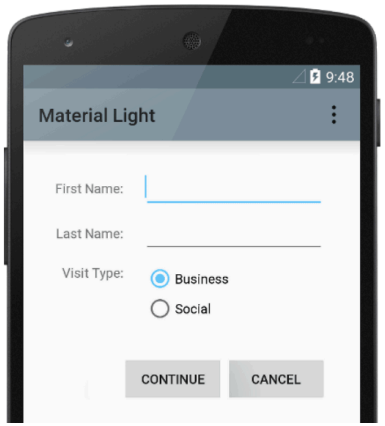
Style/theme

- «Styles and themes on Android allow you to separate the details of your app design from the UI structure and behavior, similar to stylesheets in web design»

- «A *style* is a collection of attributes that specify the appearance for a single View. A style can specify attributes such as font color, font size, background color, and much more»

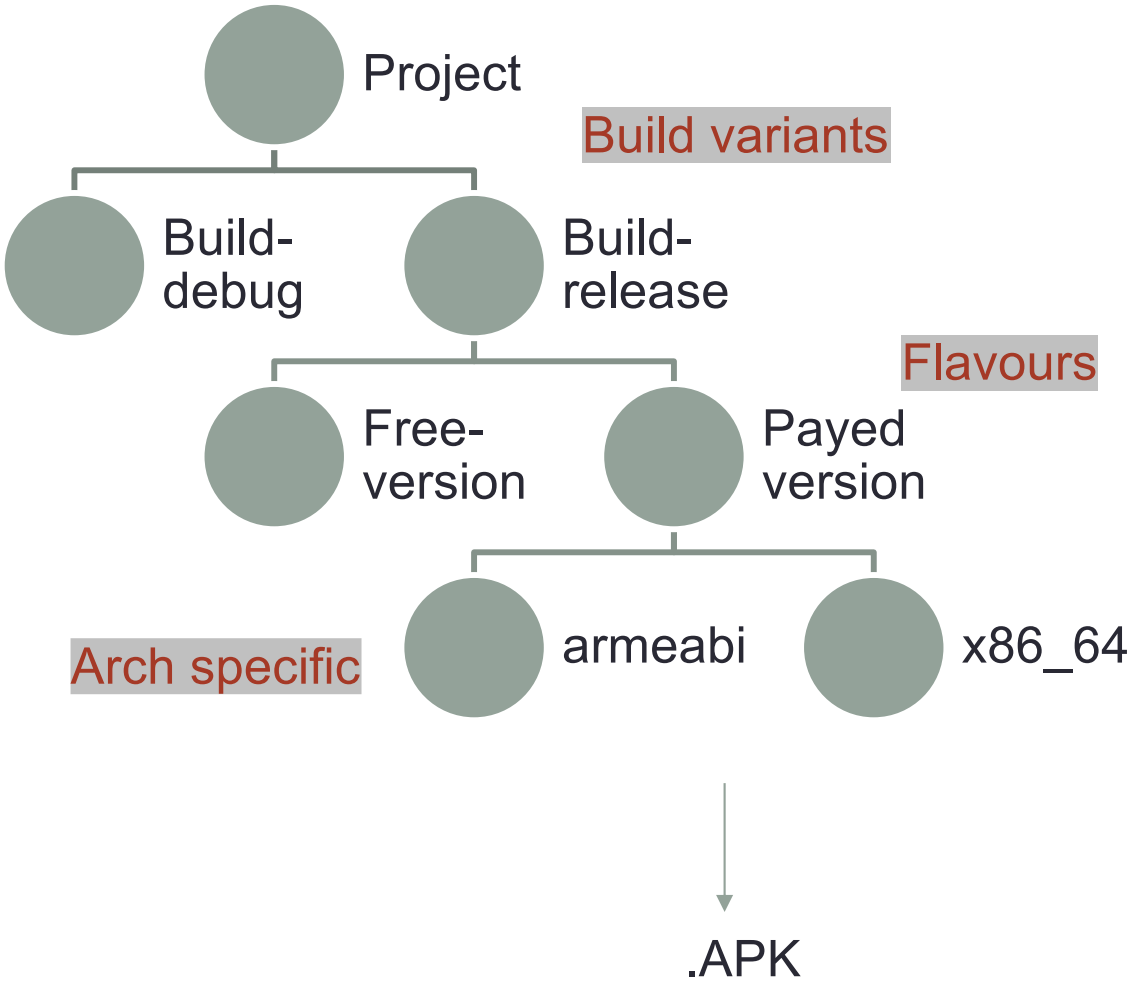
- «A *theme* is a type of style that's applied to an entire app, activity, or view hierarchy—not just an individual view»

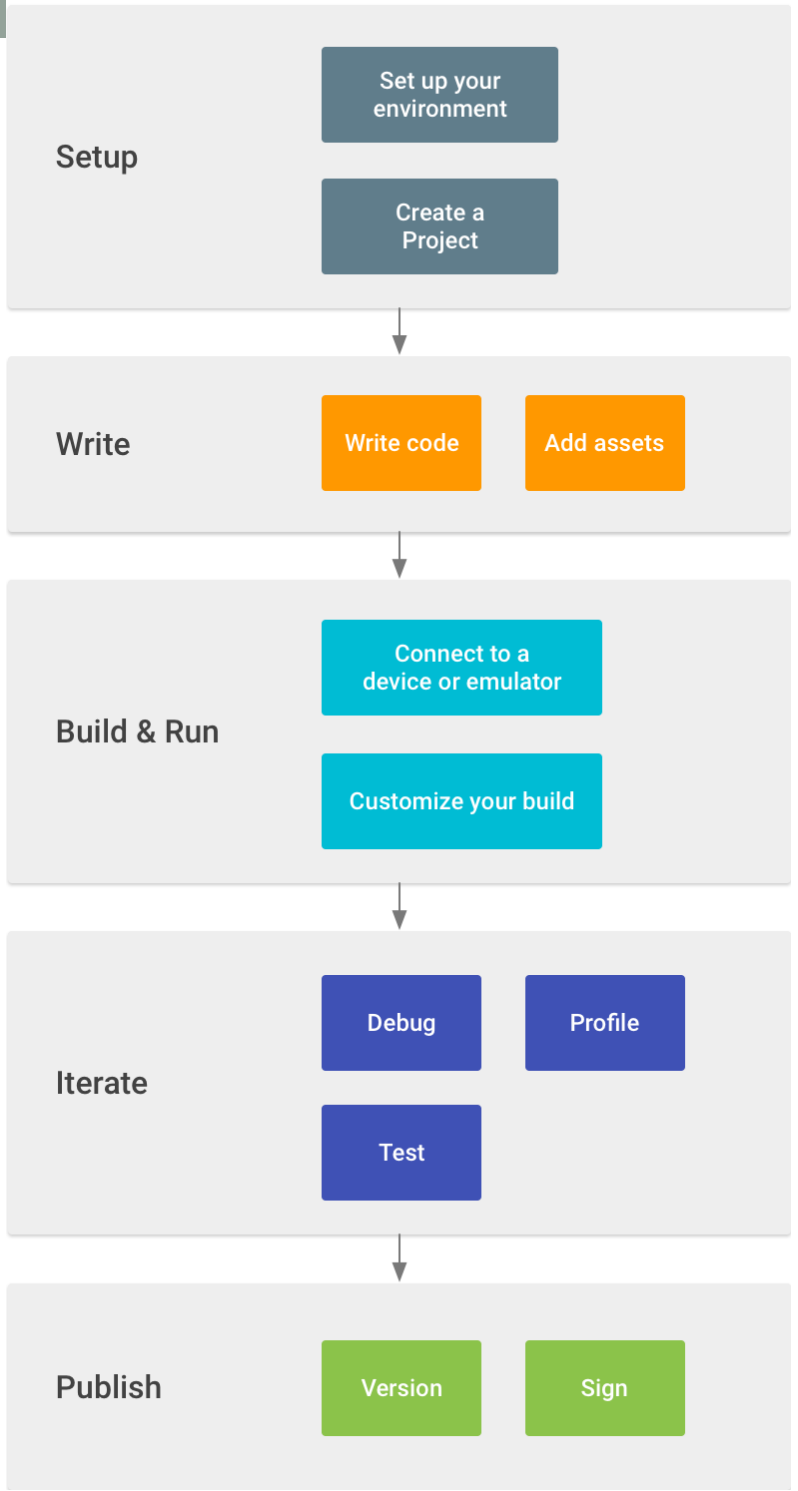




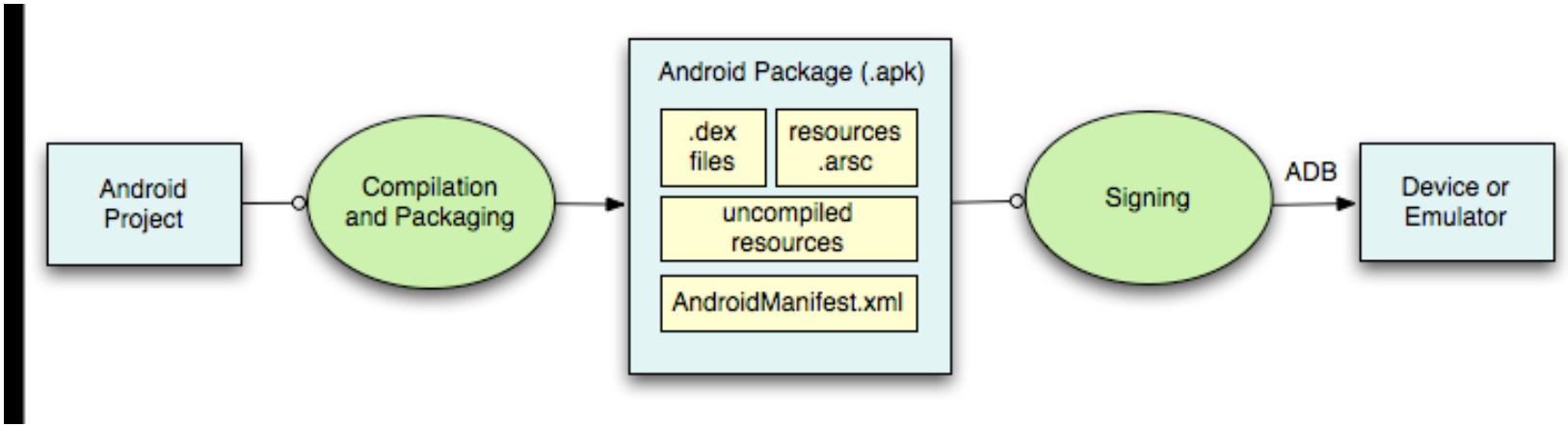

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="GreenText" parent="TextAppearance.AppCompat">
        <item name="android:textColor">#00FF00</item>
    </style>
</resources>
```

Workflow

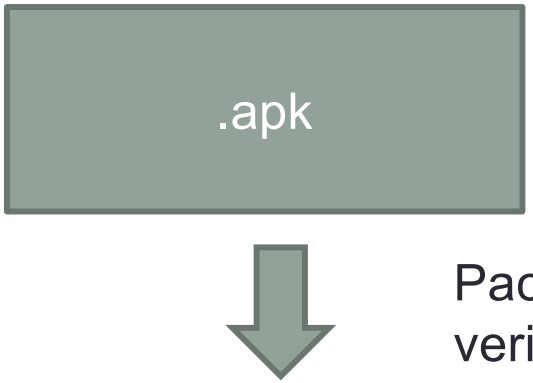

Project
Build variants
Build-
debug
Build-
release
Flavours
Free-
version
Payed
version
Arch specific
armeabi
x86_64
.APK


Setup
Set up your environment
Create a Project
Write
Write code
Add assets
Build & Run
Connect to a device or emulator
Customize your build
Iterate
Debug
Profile
Test
Publish
Version
Sign

What an application is composed of?



Android package



Package Manager
verifies that everything is consistent

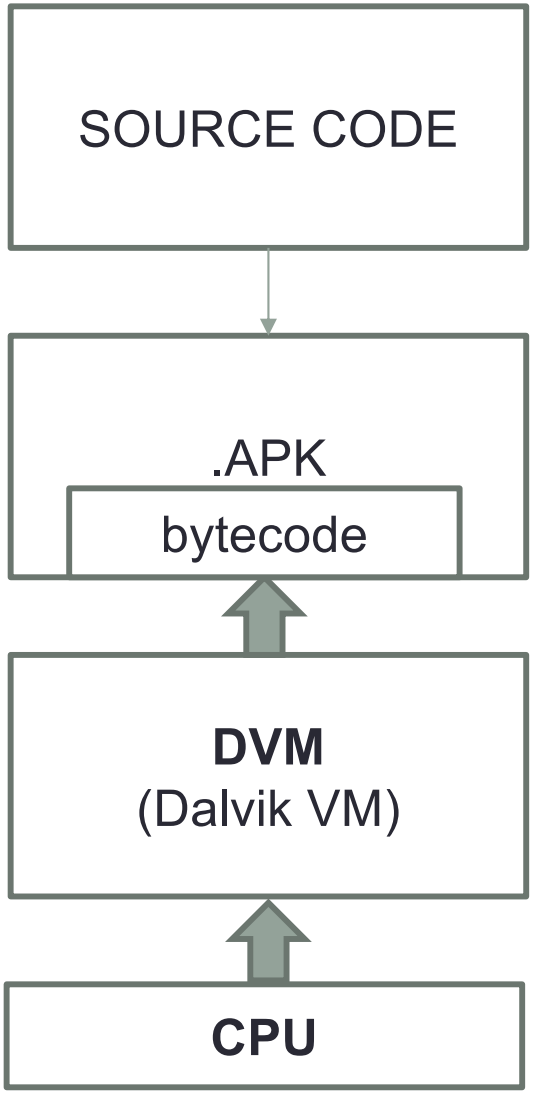
com.example.android.tiltspot (Version Name: 1.0, Version Code: 1)

APK size: **1,5 MB**, Download Size: **1,3 MB**

Compare with previous APK...

ile	Raw File Size	Download Size	% of Total Downl...
classes.dex	990,7 KB	909,2 KB	74,7%
res	238,4 KB	230,6 KB	19%
resources.arsc	227 KB	51,2 KB	4,2%
META-INF	27,3 KB	24,6 KB	2%
CERT.SF	13,4 KB	12,1 KB	1%
MANIFEST.MF	13,3 KB	12 KB	1%
CERT.RSA	597 B	595 B	0%
AndroidManifest.xml	817 B	817 B	0,1%

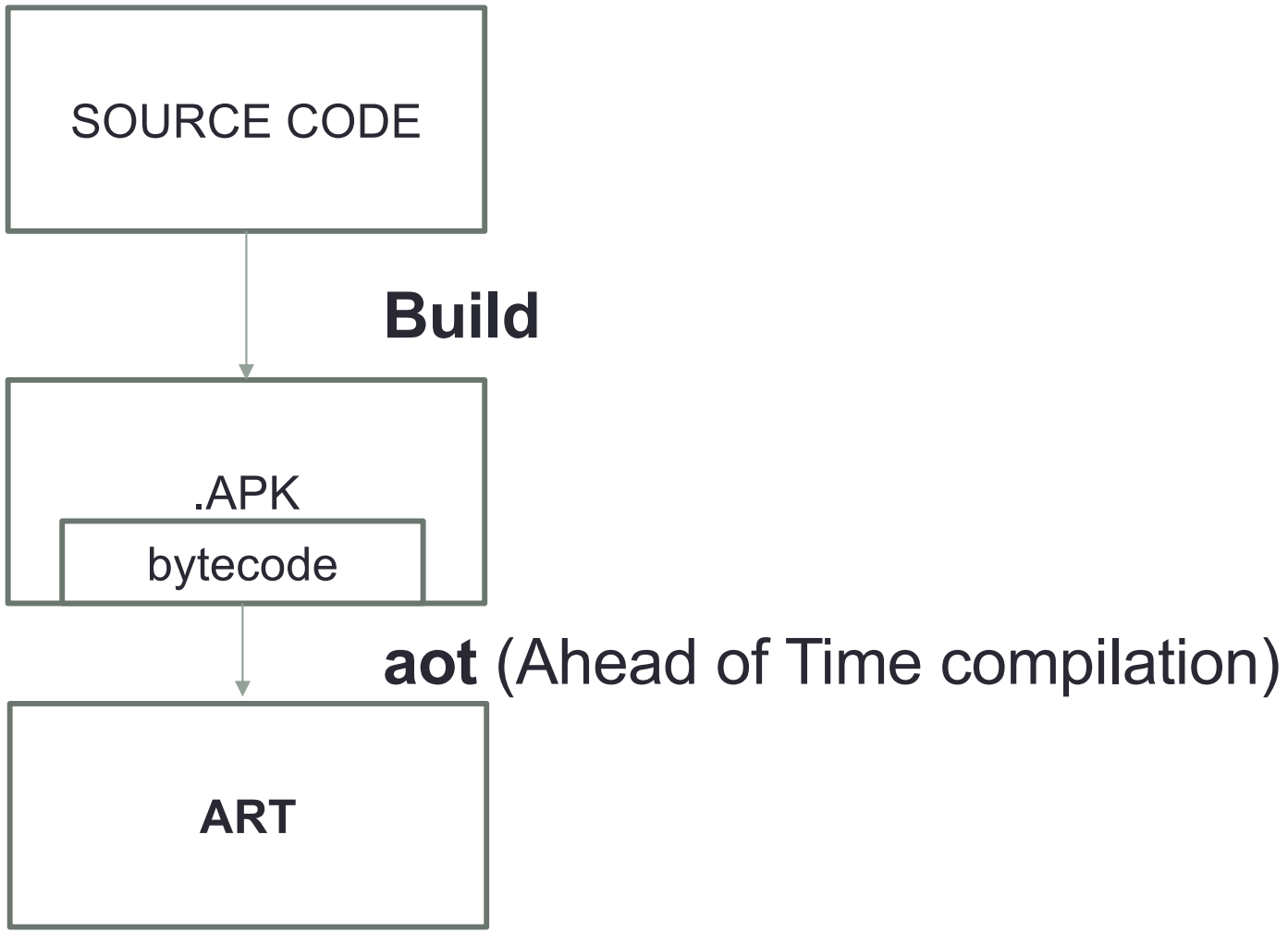
DVM



Build

bytecode is hw-arch agnostic

AOT


SOURCE CODE
Build
.APK
bytecode
aot (Ahead of Time compilation)
ART