# Lecture 12:
# 2D Graphics with Drawable

Jianjun Chen (Jianjun.Chen@xjtlu.edu.cn)

# Drawables

- A drawable resource is a general concept for a graphic which can be drawn.

- E.g. It can be an image file that can be represented in Android via a `BitmapDrawable` class:
  - You can create a `BitmapDrawable` object that refers to a certain image.
  - There are a few functions that can be called to get information about this image, or modify it, or…

# Drawables

- There are many types of drawables:
  - `BitmapDrawable`: Images like PNG, JPEG.
  - `VectorDrawable`: Vector images defined through XMLs.
  - `ShapeDrawable`: Images that arranges predefined shapes.
  - `TransitionDrawable`: Images with transitions
  - …

- You can check this tutorial and API document.

# BitmapDrawable

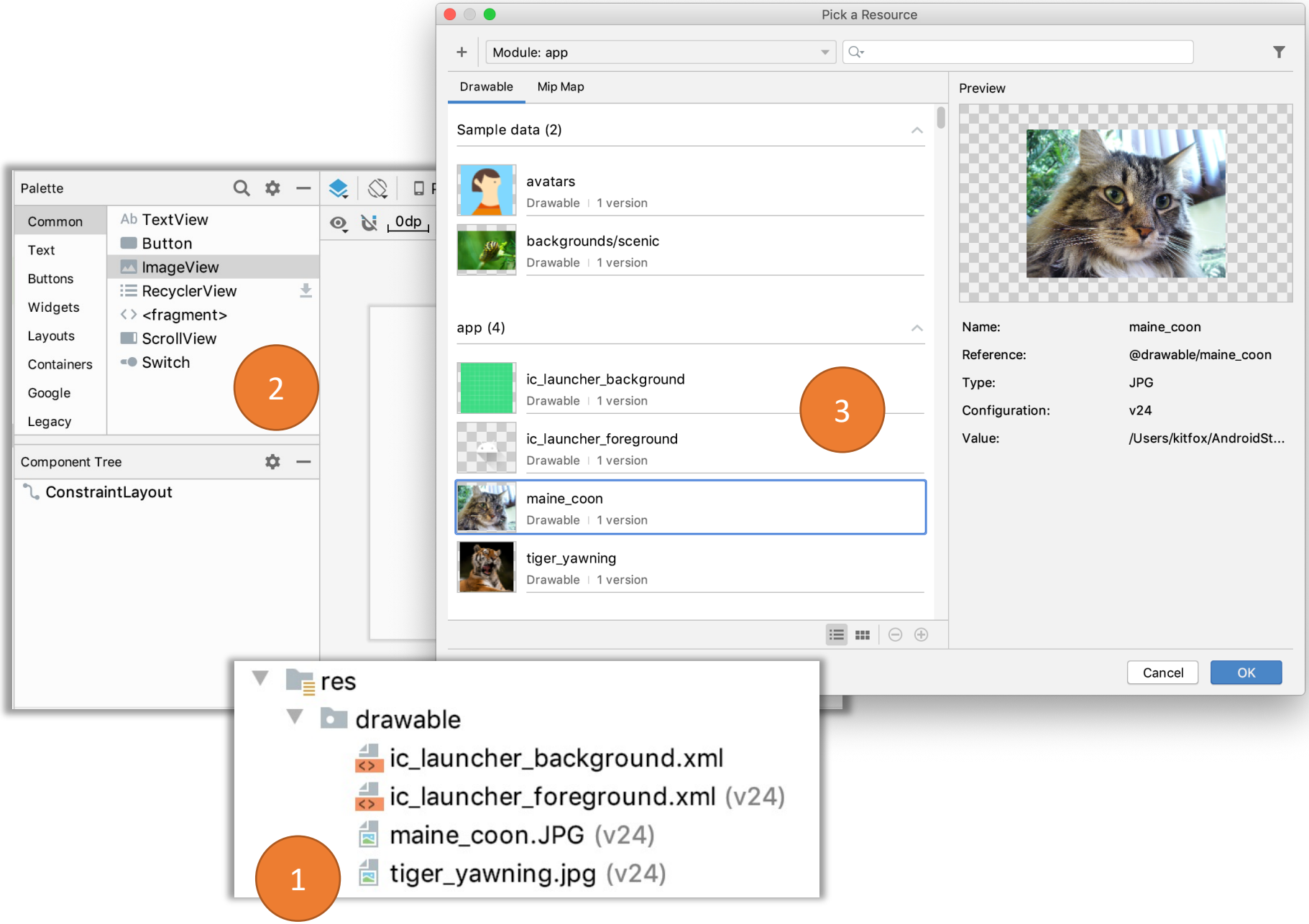
`ImageView`, the "/res/drawable" folder, the `Bitmap` class

# BitmapDrawable

- A `Drawable` that wraps a bitmap and can be tiled, stretched, or aligned.
- You can create a `BitmapDrawable` from:
  - An image file in `/res/drawable`
  - An input stream
  - Through XML inflation
  - From a `Bitmap` object.

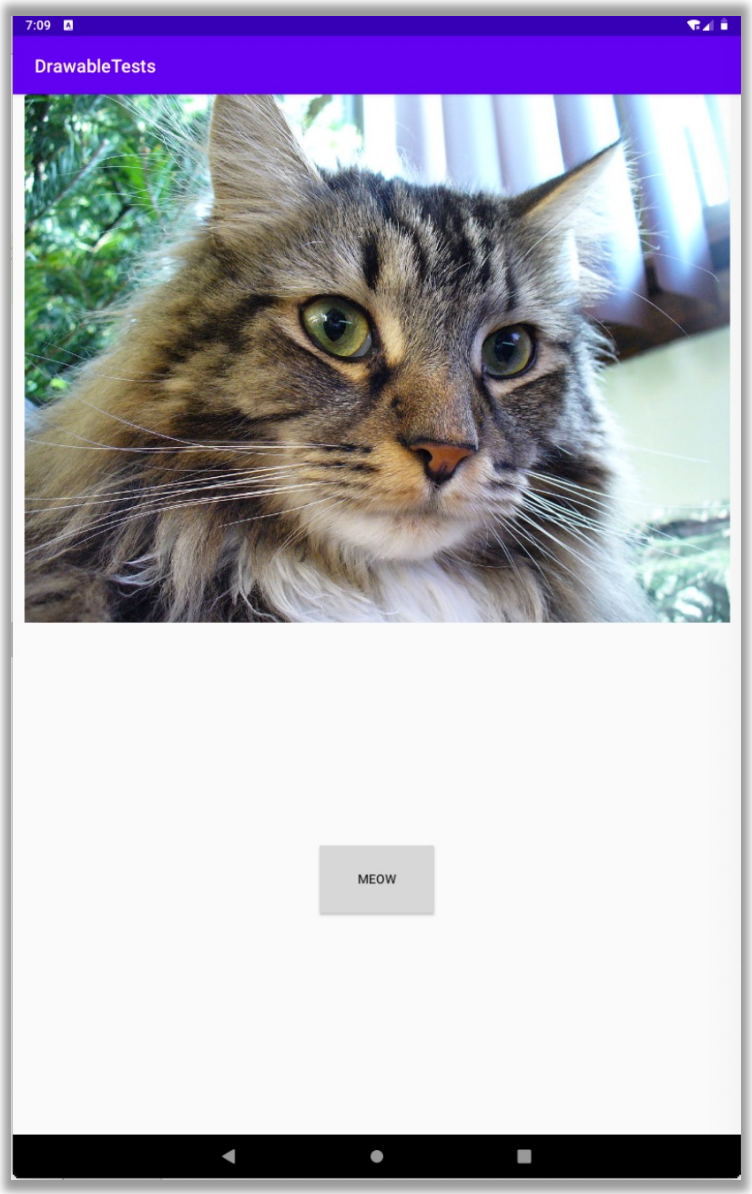
# Using an Image Resource

- To use an image resource, add your file to the "res/drawable/" directory of your project
  - Supported file types are PNG (preferred), JPG (acceptable), and GIF (discouraged)
- Then, use `ImageView` to load the image:
  - `ImageView.setImageResource()`
  - Or use the UI designer
- Put the `ImageView` into your activity

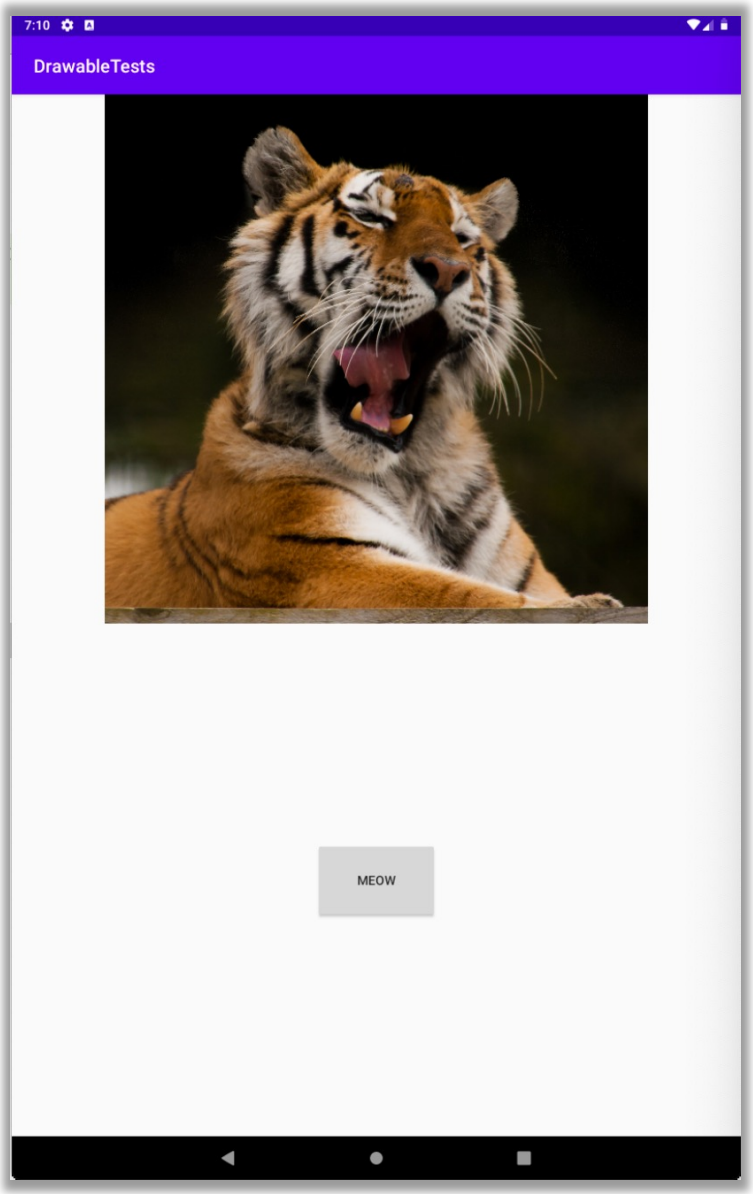

Pick a Resource
Module: app
Drawable
Mip Map
Sample data (2)
avatars
Drawable | 1 version
backgrounds/scenic
Drawable | 1 version
app (4)
ic_launcher_background
Drawable | 1 version
ic_launcher_foreground
Drawable | 1 version
maine_coon
Drawable | 1 version
tiger_yawning
Drawable | 1 version
Preview
Name: maine_coon
Reference: @drawable/maine_coon
Type: JPG
Configuration: v24
Value: /Users/kitfox/AndroidSt...
Cancel
OK
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Google
Legacy
TextView
Button
ImageView
RecyclerView
<fragment>
ScrollView
Switch
0dp
Component Tree
ConstraintLayout
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml (v24)
maine_coon.JPG (v24)
tiger_yawning.jpg (v24)
1
2
3

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn = findViewById(R.id.catBtn);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ImageView img = findViewById(R.id.catView);
                img.setImageResource(R.drawable.tiger_yawning);
            }
        });
    }
}
```




Click “Meow”



# BitmapDrawable

- In the previous example, we didn't use `BitmapDrawable` at all.
- But for some other cases, we have to use this class.
  - Because some functions only accept objects of `Drawable`.
- If that is the case, you can use `getDrawable()` to create an object of `Drawable`.
  - See the next example.

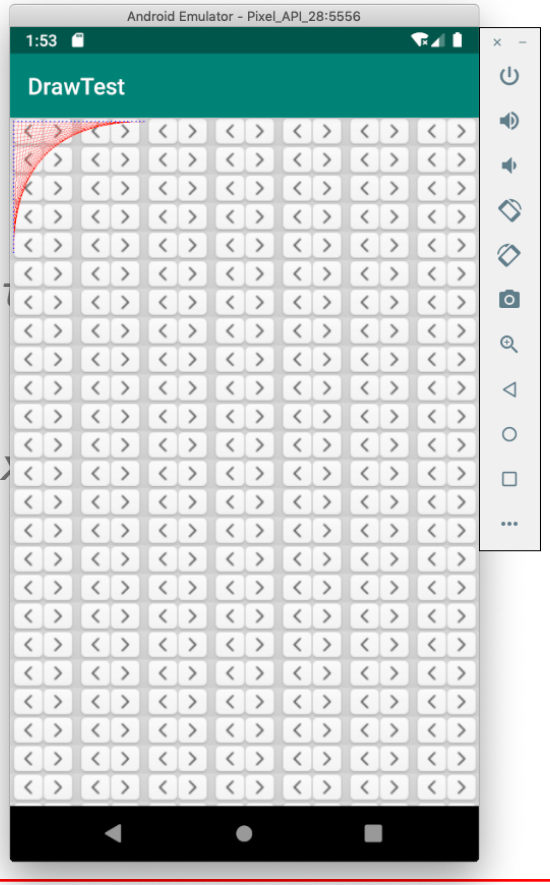
# BitmapDrawable: Example

- The example will try to set an ugly background for the `MyView` widget from the previous lecture:
  - Add a small icon into “res/drawable” and call it “icon”.
  - Add the follow lines inside the `onDraw()` method of `MyView`.
  - The `setBackground()` method of a `View` only accepts `drawables`.

```
Resources res = getResources();
BitmapDrawable bd =
        (BitmapDrawable) res.getDrawable(R.drawable.icon);
bd.setTileModeXY(Shader.TileMode.REPEAT, Shader.TileMode.REPEAT);
this.setBackground(bd);
```

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.translate(10, 10);
    //canvas.drawColor(Color.WHITE); // fill
    Paint p = new Paint();
    p.setColor(Color.RED);
    p.setStrokeWidth(c); // 0 means single pix

    canvas.drawLines(mPts, p);
    p.setColor(Color.BLUE);
    p.setStrokeWidth(3);
    canvas.drawPoints(mPts, p);
```




```
    Resources res = getResources();
    BitmapDrawable bd =
        (BitmapDrawable) res.getDrawable(R.drawable.icon);
    bd.setTileModeXY(Shader.TileMode.REPEAT, Shader.TileMode.REPEAT);
    this.setBackground(bd);
}
```

# Bitmap

A related class to BitmapDrawable

# Bitmap

- `Bitmap` is a class that stores bitmap image data.
  - You can load a JPEG/PNG file and create a `Bitmap` object based on it. Use `BitmapFactory`.

`decodeResource(Resources res, int id)`

Synonym for `decodeResource(android.content.res.Resources, int, android.graphics.BitmapFactory.Options)` with null Options.

`decodeFile(String pathName)`

Decode a file path into a bitmap.

  - You can also create an empty Bitmap object and draw on it using canvas.

**Public constructors**

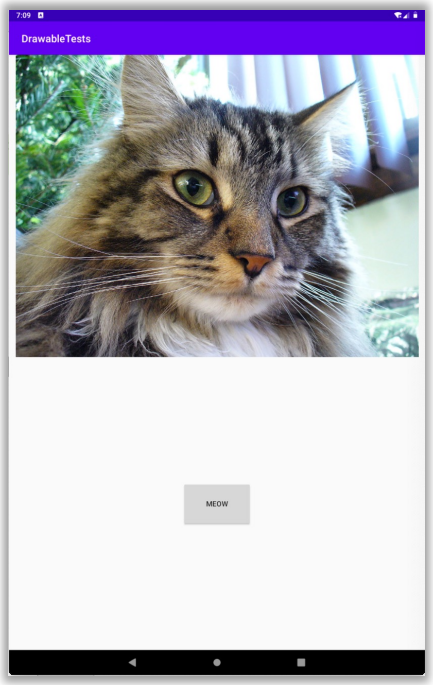
`Canvas()`

Construct an empty raster canvas.

`Canvas(Bitmap bitmap)`

Construct a canvas with the specified bitmap to draw into.

# Loading an Image File



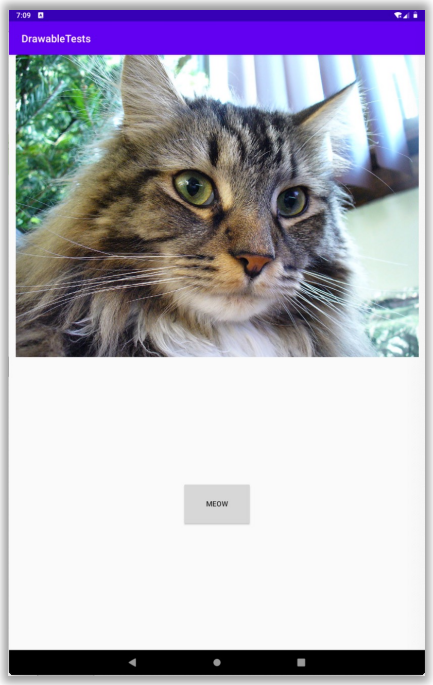

We will use the same example as before:

- From “res/drawable”:

```
public void onClick(View v) {
    ImageView img = findViewById(R.id.catView);
    Bitmap bmp = BitmapFactory.decodeResource(
            MainActivity.this.getResources(),
            R.drawable.tiger_yawning);
    img.setImageBitmap(bmp);
}
```

- From the external storage:

# Loading an Image File



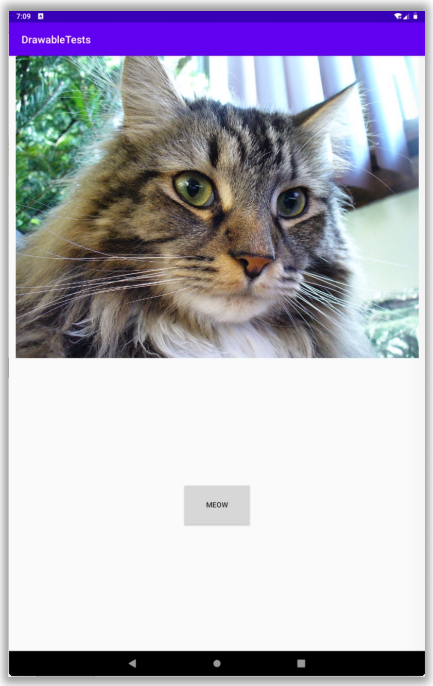

We will use the same example as before:

- From “res/drawable”:
- From the external storage:

```
public void onClick(View v) {
    ImageView img = findViewById(R.id.catView);
    img.setImageBitmap(BitmapFactory.decodeFile("/storage/0FEC-1113/Tiger_Yawning.jpg"));
}
```

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

# Loading an Image File

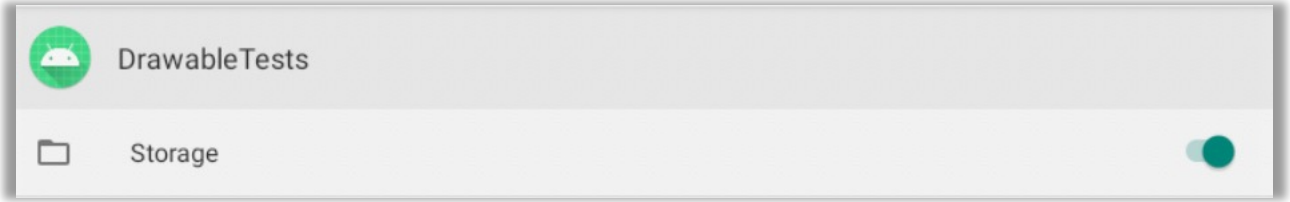



We will use the same example as before:

- From “res/drawable”:
- From the external storage:
  - You can drag files directly to AVD
    - Saved in “Downloads”
  - Remember to add permissions

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

  - And grant permissions manually



# Drawing on Empty Bitmaps

```
Bitmap myBitmap = Bitmap.createBitmap
        (BITMAP_WIDTH, BITMAP_HEIGHT, Bitmap.Config.ARGB_8888);
Canvas canvas = new Canvas(myBitmap);
canvas.drawColor(Color.WHITE); // background
Paint p = new Paint();
```

8 bit precision for Alpha (transparency), Red, Green and Blue

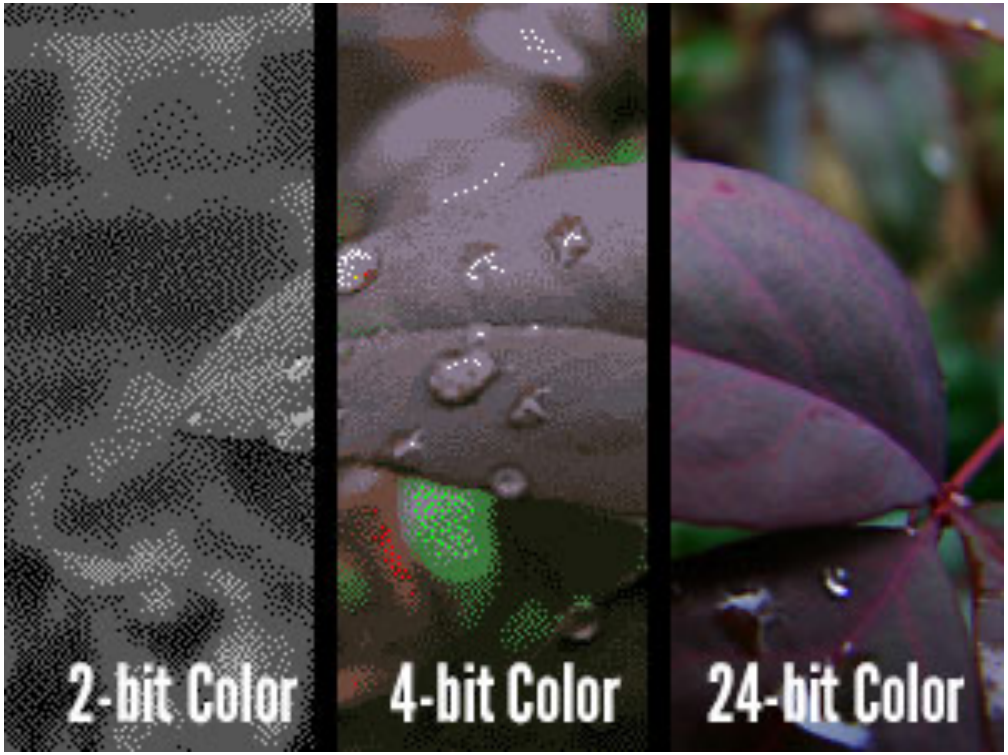
```
p.setStyle(Paint.Style.STROKE);
p.setColor(Color.RED);
p.setStrokeWidth(0);
canvas.drawCircle(100, 100, 20, p);

p.setStyle(Paint.Style.FILL_AND_STROKE);
p.setColor(Color.BLUE);
canvas.drawCircle(100, 200, 20, p);
```

Whether to fill the circle or not

(x, y, radius, painter)

# Colour Depth




Also read https://www.webopedia.com/TERM/A/alpha_channel.html

# Reasons to Use Bitmap

- You can send this Bitmap object to another activity via an Intent.
- You can easily combine this Bitmap with others.
- Supports scaling, rotating and other operations.
  - See different versions of `Bitmap.createBitmap()`.

# VectorDrawable

Vector image

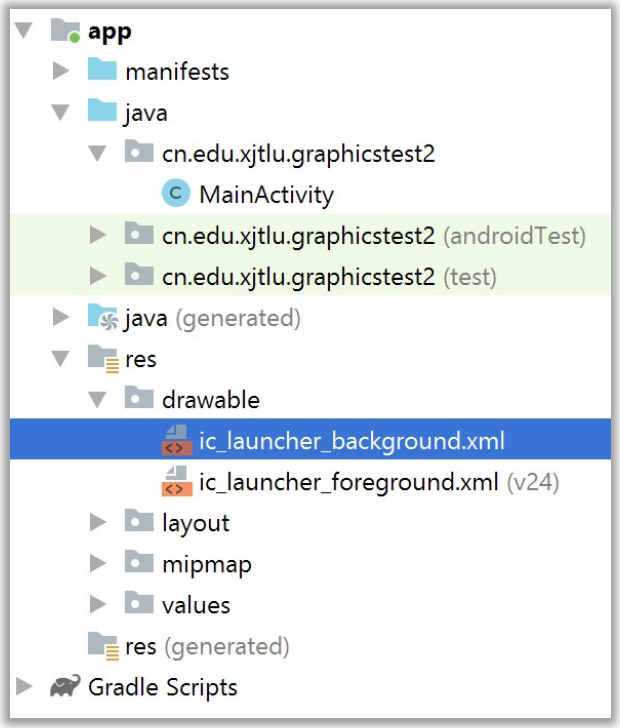
https://developer.android.google.cn/reference/android/graphics/drawable/VectorDrawable

# Vector Images VS Bitmaps

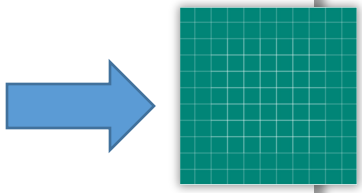
- A vector image stores its line strokes, colours and other elements as “commands”.
  - E.g. draw a line from (0% width, 0% height) to (59% width, 62% height).
- Every time it is opened, the system will re-interpret it by running these commands again.
  - The relative values and positions allow vector images to adapt to different images sizes without losing image quality.
- While bitmaps (bmp, jpg, png) are represented by pixels. Using a non-integer scaling leads to a blurry image (e.g. resize a bitmap to 171% of its original size)

# Vector Drawables

- In the android system, vector images are described by XML files.
  - i.e. Vector Drawables.




```
<?xml version="1.0" encoding="utf-8"?>
<vector
xmlns:android="http://schemas.android.com/apk/res/an
droid"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path
        android:fillColor="#008577"
        android:pathData="M0,0h108v108h-108z" />
    <path
        android:fillColor="#00000000"
        android:pathData="M9,0L9,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
……
```

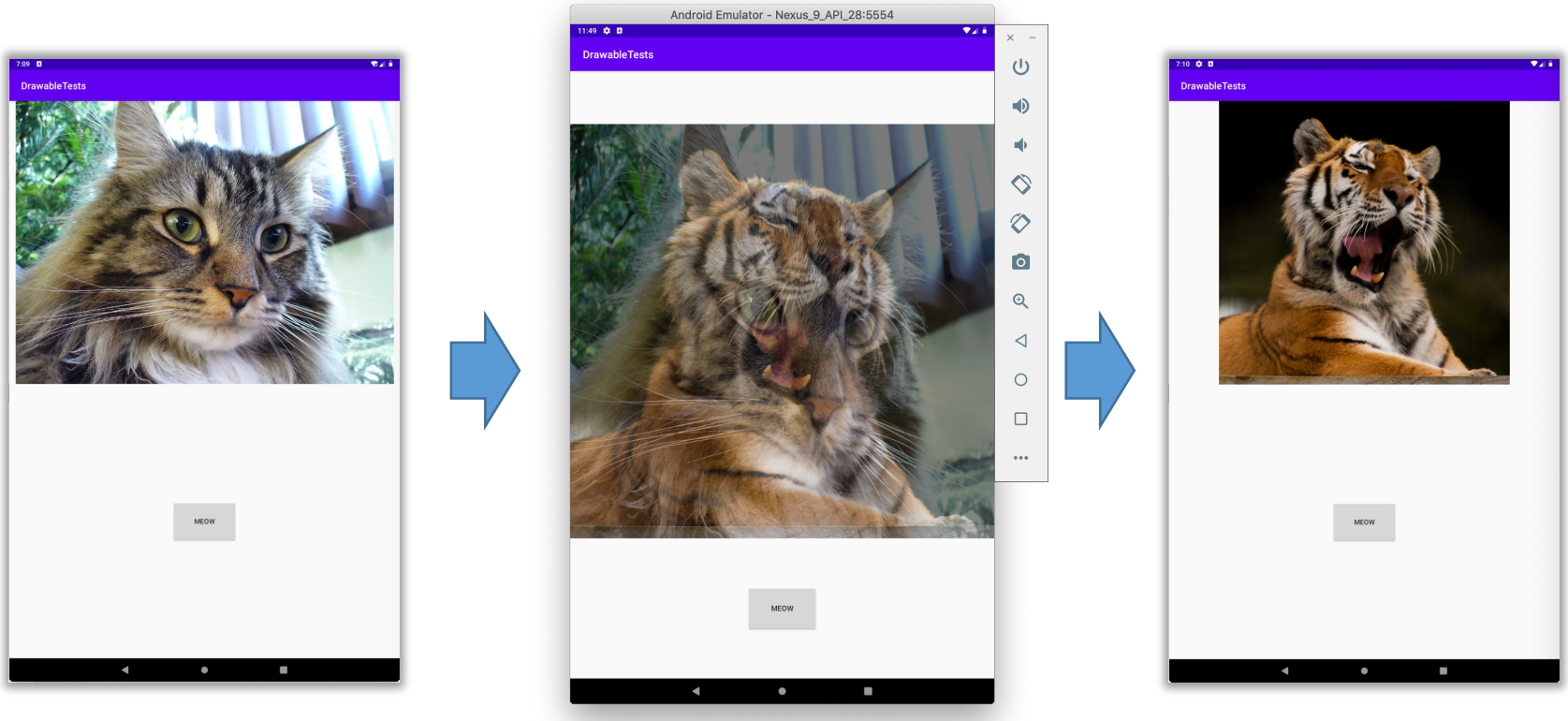


# TransitionDrawable

Image with transitions

# TransitionDrawable

- This drawable class allows you to apply transitions to a displayed image



# TransitionDrawable

- The drawable resource must be created first.
  - Using XML in “/res/drawable”

```
<?xml version="1.0" encoding="utf-8"?>
<transition
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/maine_coon"/>
    <item android:drawable="@drawable/tiger_yawning"/>
</transition>
```

  - Using Java code

# TransitionDrawable

- The drawable resource must be created first.
  - Using XML in “/res/drawable”
  - Using Java code

```
Drawable images[] = {getDrawable(R.drawable.maine_coon),
getDrawable(R.drawable.tiger_yawning)};
TransitionDrawable transition = new TransitionDrawable(images);
img.setImageDrawable(transition);
transition.startTransition(1000);
```

# TransitionDrawable

- Once created, load it into ImageView.
- Later you will be able to start the transition.

```
ImageView img = findViewById(R.id.catView);
Drawable images[] = {getDrawable(R.drawable.maine_coon),
getDrawable(R.drawable.tiger_yawning)};
TransitionDrawable transition = new TransitionDrawable(images);
img.setImageDrawable(transition);
transition.startTransition(1000);
```

# Lab

Try to recreate every example in this lecture.

But firstly, make sure you can work out the lab task of Lecture 11.