Mobile 3D Graphics

Introduction to Android Drawables

What are Drawables?

- general concept for a graphic which can be drawn.
- The simplest case is a graphical file (bitmap), which would be represented in Android via **aBitmapDrawable** class.
- Every **Drawable** is stored as individual files in one of the *res/drawable* folders.
- Drawables can also be written in Java code.

Using drawables for views JAVA

- **In code** you can also assign ***drawables*** to views. Most views accept **an resource ID** as input parameter.
- **For example** the following code shows how to set a ***drawables*** as background to an **ImageView**.
-
```
ImageView imageView = (ImageView) findViewById(R.id.image);
imageView.setImageResource(R.drawable.hello);
```

Using drawables for views XML

- **Drawables** are referred to in **XML** via *@drawable/filename* whereby filename is the filename without the file **extension**.
- For example to access the *res/drawable/hello.png* Drawable, you would use @drawable/hello as demonstrated in the following snippet.

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textView1" android:layout_width="wrap content"
    android:layout_height="wrap_content"
    android:background="@drawable/hello"
    android:text="@string/hello_world" />
```

XML Drawables

1. **Shape Drawables** : are **XML** files which allow to define a geometric object with **colors, borders** and **gradients** which can get assigned to **Views**.
2. **State Drawables :** allow to define **states**. For each state a different **drawable** an get assigned to the View.
3. **Transition Drawables**: allow to define transitions which can be triggered in the coding.

XML Drawables
Shape Drawables

```
• <?xml version="1.0" encoding="UTF-8"?>
•  <shape
      xmlns:android=http://schemas.android.com/apk/res/android
      android:shape="rectangle">
      <stroke
            android:width="2dp"
            android:color="#FFFFFFFF" />
       <gradient
            android:endColor="#DDBBBBBB“
            android:startColor="#DD777777"
            android:angle="90" />
        <corners
            android:bottomRightRadius="7dp"
            android:bottomLeftRadius="7dp“
            android:topLeftRadius="7dp"
            android:topRightRadius="7dp" />
• </shape>
```

XML Drawables
State Drawables

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
<item android:drawable="@drawable/button_pressed"
    android:state_pressed="true" />
<item android:drawable="@drawable/button_checked"
    android:state_checked="true" />
<item android:drawable="@drawable/button_default" />
</selector>
```

XML Drawables

Transition Drawables

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

……………………………………………………………………………

```
final ImageView image = (ImageView)
findViewById(R.id.image); final ToggleButton button = (ToggleButton)
     findViewById(R.id.button);
button.setOnClickListener(new OnClickListener() {
       @Override
        public void onClick(final View v) {
        TransitionDrawable drawable = (TransitionDrawable) image.getDrawable();
        if (button.isChecked()) {
              drawable.startTransition(500);
         } else {
                  drawable.reverseTransition(500); } } });
```

Vector drawables

- As of **API level 21** you can use vector drawables in your Android application.
- These are similar to **svg files** but with a limited scope.
- Using **vector drawables** automatically scale to the density of the device.

Vector drawables

vectordrawable.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
     android:width="64dp"
     android:viewportHeight="600"
     android:viewportWidth="600" >
     <group
         android:name="rotationGroup"
         android:pivotX="300.0"
         android:pivotY="300.0"
         android:rotation="45.0" >
         <path
             android:name="v"
             android:fillColor="#000000"
             android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
      </group>
</vector>
```

Animation Drawables

- You can define an **animation drawables** and assign it to a **View** via the **setBackgroundResource()** method.
- <!-- Animation frames are phase*.png files inside the **res/drawable/ folder** -->

```
<animation-list android:id="@+id/selected" android:oneshot="false">
<item android:drawable="@drawable/phase1" android:duration="400" />
<item android:drawable="@drawable/phase2" android:duration="400" />
<item android:drawable="@drawable/phase3" android:duration="400" />
</animation-list>
```

,,,

- ImageView img = (ImageView)findViewById(R.id.yourid);
  img.**setBackgroundResource**(R.drawable.your_animation_file);

// Get the AnimationDrawable object.

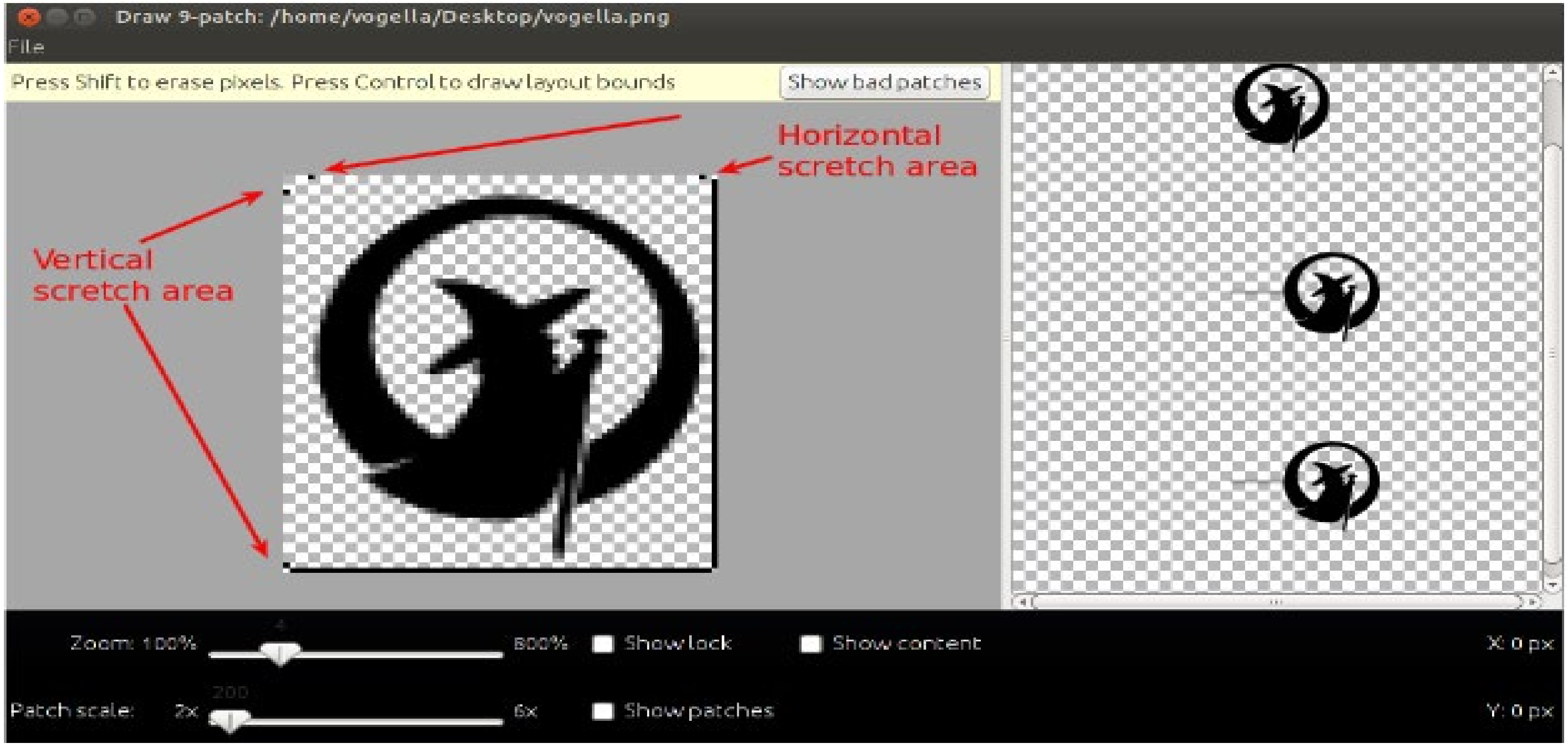
    AnimationDrawable frameAnimation = (AnimationDrawable) img.getBackground();

// Start the animation (looped playback by default).

    frameAnimation.start();

9 Patch Drawables

- ***9 Patch drawables*** are ***Drawables*** which have a **one pixel** additional border. On the top and left you define the area which should be scaled if the ***Drawable*** is to small for the view. This is the stretch area.



Custom Drawables

- You can also create ***custom Drawable***, which can use the **Canvas API** for their display.
- For these **drawables** you can use the full **Canvas API** to design them to your need.

Exercise: Create Custom rounded corner drawable

References

- **Android Drawables resources**

PathMorphing with AnimatedVectorDrawables in Android

- https://lewismcgeary.github.io/posts/animated-vector-drawable-pathMorphing/

See Blog post with examples for animated vector graphics == vogella training and consulting support

- http://blog.sqisland.com/2014/10/first-look-at-animated-vector-drawable.html