Andorid Programming

Week 10

Mina Jung

EECS, Syracuse University

Spring 2017

Part I

Drawable Resources

Outline I

- .png, .jpg, or .gif file in the res/drawable/ directory
- filename is used as the resource ID
- reference
  - In Java: R.drawable.filename
  - In XML: @[package:]drawable/filename

```
<ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/myimage" />
```

```
Resources res = getResources();
Drawable drawable = res.getDrawable(R.drawable.myimage);
```

- a resource defined in XML that points to a bitmap file
  - an alias for a raw bitmap file
  - can specify additional properties for the bitmap such as dithering and tiling
  - location: res/drawable/filename.xml
- filename is used as the resource ID
- reference
  - In Java: R.drawable.filename
  - In XML: @[package:]drawable/filename

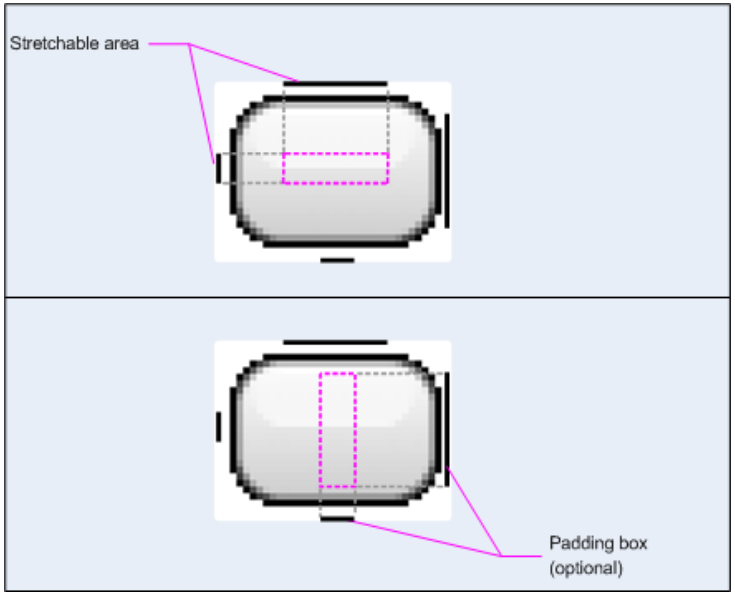
```
<?xml version="1.0" encoding="utf-8"?>
<bitmap
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@[package:]drawable/drawable_resource"
    android:antialias=["true" | "false"]
    android:dither=["true" | "false"]
    android:filter=["true" | "false"]
    android:gravity=["top" | "bottom" | "left" | "right" |
        "center_vertical" |
                    "fill_vertical" | "center_horizontal" |
                        "fill_horizontal" |
                    "center" | "fill" | "clip_vertical" |
                        "clip_horizontal"]
    android:mipMap=["true" | "false"]
    android:tileMode=["disabled" | "clamp" | "repeat" |
        "mirror"] />
```

Value	Description
`disabled`	Do not tile the bitmap. This is the default value.
`clamp`	Replicates the edge color if the shader draws outside of its original bounds
`repeat`	Repeats the shader's image horizontally and vertically.
`mirror`	Repeats the shader's image horizontally and vertically, alternating mirror images so that adjacent images always seam.

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/icon"
    android:tileMode="repeat" />
```

Value	Description
`top`	Put the object at the top of its container, not changing its size.
`bottom`	Put the object at the bottom of its container, not changing its size.
`left`	Put the object at the left edge of its container, not changing its size.
`right`	Put the object at the right edge of its container, not changing its size.
`center_vertical`	Place object in the vertical center of its container, not changing its size.
`fill_vertical`	Grow the vertical size of the object if needed so it completely fills its container.
`center_horizontal`	Place object in the horizontal center of its container, not changing its size.
`fill_horizontal`	Grow the horizontal size of the object if needed so it completely fills its container.
`center`	Place the object in the center of its container in both the vertical and horizontal axis, not changing its size.
`fill`	Grow the horizontal and vertical size of the object if needed so it completely fills its container. This is the default.
`clip_vertical`	Additional option that can be set to have the top and/or bottom edges of the child clipped to its container's bounds. The clip is based on the vertical gravity: a top gravity clips the bottom edge, a bottom gravity clips the top edge, and neither clips both edges.
`clip_horizontal`	Additional option that can be set to have the left and/or right edges of the child clipped to its container's bounds. The clip is based on the horizontal gravity: a left gravity clips the right edge, a right gravity clips the left edge, and neither clips both edges.

A NinePatch is a PNG image in which you can define stretchable regions that Android scales when content within the View exceeds the normal image bounds. You typically assign this type of image as the background of a View that has at least one dimension set to "wrap_content", and when the View grows to accomodate the content, the Nine-Patch image is also scaled to match the size of the View. An example use of a Nine-Patch image is the background used by Android's standard Button widget, which must stretch to accommodate the text (or image) inside the button.


Stretchable area
Padding box
(optional)

1. In Android Studio, right-click the PNG image you'd like to create a NinePatch image from, then click Create 9-patch file.

2. Type a file name for your NinePatch image, and click OK. Your image will be created with the .9.png file extension.

3. Double-click your new NinePatch file to open it in Android Studio. Your workspace will now open. The left pane is your drawing area, in which you can edit the lines for the stretchable patches and content area. The right pane is the preview area, where you can preview your graphic when stretched.

4. Click within the 1-pixel perimeter to draw the lines that define the stretchable patches and (optional) content area. Right-click (or hold Shift and click, on Mac) to erase previously drawn lines.

5. When done, click File > Save to save your changes

- res/drawable/filename.9.png

- reference
  - In Java: R.drawable.filename
  - In XML: @[package:]drawable/filename

```
<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:background="@drawable/myninepatch" />
```

- A LayerDrawable is a drawable object that manages an array of other drawables
  - each drawable in the list is drawn in the order of the list–the last drawable in the list is drawn on top
- Each drawable is represented by an <item> element inside a single <layer-list> element.

- syntax

```
<layer-list
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@[package:]drawable/drawable_resource"
        android:id="@[+][package:]id/resource_name"
        android:top="dimension"
        android:right="dimension"
        android:bottom="dimension"
        android:left="dimension" />
</layer-list>
```

- all drawable items are scaled to fit the size of the containing View, by default
  - To avoid scaling, the following example uses a <bitmap> element with centered gravity:

```
<item>
  <bitmap android:src="@drawable/image"
          android:gravity="center" />
</item>
```

- example: res/drawable/layers.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list
     xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
      <bitmap android:src="@drawable/android_red"
        android:gravity="center" />
    </item>
    <item android:top="10dp" android:left="10dp">
      <bitmap android:src="@drawable/android_green"
        android:gravity="center" />
    </item>
    <item android:top="20dp" android:left="20dp">
      <bitmap android:src="@drawable/android_blue"
        android:gravity="center" />
    </item>
</layer-list>
```

```
<ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/layers" />
```

- A StateListDrawable is a drawable object defined in XML that uses a several different images to represent the same graphic, depending on the state of the object
  - Button states (pressed, focused, or neither)
  - can provide a different background image for each state
- describe the state list in an XML file
  - each graphic is represented by an <item> element inside a single <selector> element
  - each <item> uses various attributes to describe the state
- during each state change, the state list is traversed top to bottom and the first item that meets the minimum criteria of the state is used

- syntax

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android"
    android:constantSize=["true" | "false"]
    android:dither=["true" | "false"]
    android:variablePadding=["true" | "false"] >
    <item
        android:drawable="@[package:]drawable/drawable_resource"
        android:state_pressed=["true" | "false"]
        android:state_focused=["true" | "false"]
        android:state_hovered=["true" | "false"]
        android:state_selected=["true" | "false"]
        android:state_checkable=["true" | "false"]
        android:state_checked=["true" | "false"]
        android:state_enabled=["true" | "false"]
        android:state_activated=["true" | "false"]
        android:state_window_focused=["true" | "false"] />
</selector>
```

- exampe: res/drawable/button.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<selector
     xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
          android:drawable="@drawable/button_pressed" /> <!--
               pressed -->
    <item android:state_focused="true"
          android:drawable="@drawable/button_focused" /> <!--
               focused -->
    <item android:state_hovered="true"
          android:drawable="@drawable/button_focused" /> <!--
               hovered -->
    <item android:drawable="@drawable/button_normal" /> <!--
         default -->
</selector>
```

```
<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:background="@drawable/button" />
```

- A Drawable that manages a number of alternate Drawables, each assigned a maximum numerical value
- syntax

```
<?xml version="1.0" encoding="utf-8"?>
<level-list
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@drawable/drawable_resource"
        android:maxLevel="integer"
        android:minLevel="integer" />
</level-list>
```

- example

```
<?xml version="1.0" encoding="utf-8"?>
<level-list
     xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@drawable/status_off"
        android:maxLevel="0" />
    <item
        android:drawable="@drawable/status_on"
        android:maxLevel="1" />
</level-list>
```

- a drawable object that can cross-fade between the two drawable resources
- each drawable is represented by an <item> element inside a single <transition> element
  - No more than two items are supported
  - To transition forward, call startTransition()
  - To transition backward, call reverseTransition()

- syntax

```
<?xml version="1.0" encoding="utf-8"?>
<transition
xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@[package:]drawable/drawable_resource"
        android:id="@[+][package:]id/resource_name"
        android:top="dimension"
        android:right="dimension"
        android:bottom="dimension"
        android:left="dimension" />
</transition>
```

- example: res/drawable/transition.xml:

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

```
<ImageButton
    android:id="@+id/button"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/transition" />
```

```
ImageButton button = (ImageButton) findViewById(R.id.button);
TransitionDrawable drawable = (TransitionDrawable)
    button.getDrawable();
drawable.startTransition(500);
```

- insets another drawable by a specified distance
- syntax

```
<?xml version="1.0" encoding="utf-8"?>
<inset
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/drawable_resource"
    android:insetTop="dimension"
    android:insetRight="dimension"
    android:insetBottom="dimension"
    android:insetLeft="dimension" />
```

- example:

```
<?xml version="1.0" encoding="utf-8"?>
<inset xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/background"
    android:insetTop="10dp"
    android:insetLeft="10dp" />
```

- clips another drawable based on this Drawable's current level
  - can control how much the child drawable gets clipped in width and height based on the level, as well as a gravity to control where it is placed in its overall container
  - often used to implement things like progress bars

- syntax

```
<?xml version="1.0" encoding="utf-8"?>
<clip
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/drawable_resource"
    android:clipOrientation=["horizontal" | "vertical"]
    android:gravity=["top" | "bottom" | "left" | "right" |
        "center_vertical" |
                    "fill_vertical" | "center_horizontal" |
                        "fill_horizontal" |
                    "center" | "fill" | "clip_vertical" |
                        "clip_horizontal"] />
```

- example: res/drawable/clip.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<clip xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/android"
    android:clipOrientation="horizontal"
    android:gravity="left" />
```

```
<ImageView
    android:id="@+id/image"
    android:background="@drawable/clip"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content" />
```

```
ImageView imageview = (ImageView) findViewById(R.id.image);
ClipDrawable drawable = (ClipDrawable) imageview.getBackground();
drawable.setLevel(drawable.getLevel() + 1000);
```

- changes the size of another drawable based on its current level
- syntax

```
<?xml version="1.0" encoding="utf-8"?>
<scale
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/drawable_resource"
    android:scaleGravity=["top" | "bottom" | "left" | "right" |
        "center_vertical" |
                         "fill_vertical" | "center_horizontal" |
                             "fill_horizontal" |
                         "center" | "fill" | "clip_vertical" |
                             "clip_horizontal"]
    android:scaleHeight="percentage"
    android:scaleWidth="percentage" />
```

- example:

```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/logo"
    android:scaleGravity="center_vertical|center_horizontal"
    android:scaleHeight="80%"
    android:scaleWidth="80%" />
```

- syntax

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape=["rectangle" | "oval" | "line" | "ring"] >
    <corners
        android:radius="integer"
        android:topLeftRadius="integer"
        android:topRightRadius="integer"
        android:bottomLeftRadius="integer"
        android:bottomRightRadius="integer" />
    <gradient
        android:angle="integer"
        android:centerX="float"
        android:centerY="float"
        android:centerColor="integer"
        android:endColor="color"
        android:gradientRadius="integer"
        android:startColor="color"
        android:type=["linear" | "radial" | "sweep"]
        android:useLevel=["true" | "false"] />
    <padding
        android:left="integer"
        android:top="integer"
```

```
        android:right="integer"
        android:bottom="integer" />
    <size
        android:width="integer"
        android:height="integer" />
    <solid
        android:color="color" />
    <stroke
        android:width="integer"
        android:color="color"
        android:dashWidth="integer"
        android:dashGap="integer" />
</shape>
```

- example: res/drawable/gradient_box.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <gradient
        android:startColor="#FFFF0000"
        android:endColor="#80FF00FF"
        android:angle="45"/>
    <padding android:left="7dp"
        android:top="7dp"
        android:right="7dp"
        android:bottom="7dp" />
    <corners android:radius="8dp" />
</shape>
```

```
<TextView
    android:background="@drawable/gradient_box"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content" />
```

```
Resources res = getResources();
Drawable shape = res. getDrawable(R.drawable.gradient_box);

TextView tv = (TextView)findViewByID(R.id.textview);
tv.setBackground(shape);
```

- color is specified with an RGB value and alpha channel
  - #RGB
  - #ARGB
  - #RRGGBB
  - #AARRGGBB
- res/values/colors.xml
- syntax

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color
        name="color_name"
        >hex_color</color>
</resources>
```

- example:

```
Resources res = getResources();
int color = res.getColor(R.color.opaque_red);
```

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/translucent_red"
    android:text="Hello"/>
```

```
View experiment_9patch(int choice, LayoutInflater inflater,
     ViewGroup container){
     View rootView =
          inflater.inflate(R.layout.fragment_drawable_experiment,
          container, false);
     final TextView textView = (TextView)
          rootView.findViewById(R.id.textView);
     final ImageView imageView = (ImageView)
          rootView.findViewById(R.id.imageView);


     switch (choice) {
          case 0:
               imageView.setVisibility(View.INVISIBLE);
               textView.setVisibility(View.VISIBLE);
               textView.setBackgroundResource(R.drawable.fantasy_frame_s);
               break;
          case 1:
               imageView.setVisibility(View.INVISIBLE);
               textView.setVisibility(View.VISIBLE);
               textView.setBackgroundResource(R.drawable.fantasy_frame_s_n
               break;
          case 2:
               imageView.setVisibility(View.INVISIBLE);
               textView.setVisibility(View.VISIBLE);
               textView.setBackgroundResource(R.drawable.goldframe);
```

```
            break;
        case 3:
            imageView.setVisibility(View.INVISIBLE);
            textView.setVisibility(View.VISIBLE);
            textView.setBackgroundResource(R.drawable.goldframe_not9);
            break;
        default:
            break;
    }

    SeekBar widthSeekBar = (SeekBar)
         rootView.findViewById(R.id.seekBar1);
    widthSeekBar.setOnSeekBarChangeListener(new
         SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int
             progress, boolean fromUser) {
            ViewGroup.LayoutParams params =
                 textView.getLayoutParams();
            params.width = progress * 15;
            textView.setLayoutParams(params);
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
        }
```

```
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
    }
});

SeekBar heightSeekBar = (SeekBar)
    rootView.findViewById(R.id.seekBar2);
heightSeekBar.setOnSeekBarChangeListener(new
    SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int
        progress, boolean fromUser) {
        ViewGroup.LayoutParams params =
            textView.getLayoutParams();
        //params.width = progress * 5;
        params.height = progress * 15;
        textView.setLayoutParams(params);
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
```

```
            }
        });

        return rootView;
    }

    int left=0, bottom=0;
    View experiment_layerlist_drawable(int choice, LayoutInflater
         inflater, ViewGroup container){
        View rootView =
                inflater.inflate(R.layout.fragment_drawable_experiment,
                container, false);
        rootView.findViewById(R.id.textView).setVisibility(View.INVISIBLE);
        //rootView.findViewById(R.id.seekBar1).setVisibility(View.INVISIBLE
        //rootView.findViewById(R.id.seekBar2).setVisibility(View.INVISIBLE

        final ImageView imageView = (ImageView)
                rootView.findViewById(R.id.imageView);

        if (choice == 4) {
            rootView.findViewById(R.id.seekBar1).setVisibility(View.INVISIBI
            rootView.findViewById(R.id.seekBar2).setVisibility(View.INVISIBI
            imageView.setImageResource(R.drawable.layers);
            return rootView;
        }
```

```
imageView.setImageResource(R.drawable.shrek2);

SeekBar leftInsetSeekBar = (SeekBar)
     rootView.findViewById(R.id.seekBar1);
leftInsetSeekBar.setOnSeekBarChangeListener(new
     SeekBar.OnSeekBarChangeListener() {
     @Override
     public void onProgressChanged(SeekBar seekBar, int
          progress, boolean fromUser) {
          left = progress*3;
          Drawable[] layers = new Drawable[2];
          layers[0] =
               getResources().getDrawable(R.drawable.shrek2);
          layers[1] =
               getResources().getDrawable(R.drawable.annotation);
          final LayerDrawable layerDrawable = new
               LayerDrawable(layers);
          layerDrawable.setLayerInset(1, left, 0, 0, bottom);
          imageView.setImageDrawable(layerDrawable);
     }

     @Override
     public void onStartTrackingTouch(SeekBar seekBar) {
     }

     @Override
```

```
        public void onStopTrackingTouch(SeekBar seekBar) {
        }
});

SeekBar bottomInsetSeekBar = (SeekBar)
     rootView.findViewById(R.id.seekBar2);
bottomInsetSeekBar.setOnSeekBarChangeListener(new
     SeekBar.OnSeekBarChangeListener() {
     @Override
     public void onProgressChanged(SeekBar seekBar, int
          progress, boolean fromUser) {
          bottom = progress*3;
          Drawable[] layers = new Drawable[2];
          layers[0] =
               getResources().getDrawable(R.drawable.shrek2);
          layers[1] =
               getResources().getDrawable(R.drawable.annotation);
          final LayerDrawable layerDrawable = new
               LayerDrawable(layers);
          layerDrawable.setLayerInset(1, left, 0, 0, bottom);
          imageView.setImageDrawable(layerDrawable);
     }

     @Override
     public void onStartTrackingTouch(SeekBar seekBar) {
     }
```

```
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
        }
    });
    return rootView;
}

View experiment_levellist_drawable(int choice, LayoutInflater
    inflater, ViewGroup container){
    View rootView =
        inflater.inflate(R.layout.fragment_drawable_experiment,
        container, false);
    rootView.findViewById(R.id.seekBar2).setVisibility(View.INVISIBLE);

    final TextView textView = (TextView)
        rootView.findViewById(R.id.textView);
    textView.setText("Default Level Value: 0");
    textView.setTextSize(30);
    final ImageView imageView = (ImageView)
        rootView.findViewById(R.id.imageView);
    imageView.setImageResource(R.drawable.levels);

    SeekBar levelSeekBar = (SeekBar)
        rootView.findViewById(R.id.seekBar1);
    levelSeekBar.setMax(200);
```

```
        levelSeekBar.setOnSeekBarChangeListener(new
             SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
                 progress, boolean fromUser) {
                textView.setText("Level: " +
                     Integer.toString(progress/100));
                imageView.setImageLevel(progress/100);
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });

        return rootView;
    }


    View experiment_transitionlist_drawable(int choice,
         LayoutInflater inflater, ViewGroup container){
```

```
View rootView =
     inflater.inflate(R.layout.fragment_drawable_experiment,
     container, false);
ViewGroup parent = (ViewGroup)
     rootView.findViewById(R.id.linearLayout);
parent.removeView(rootView.findViewById(R.id.textView));
parent.removeView(rootView.findViewById(R.id.seekBar1));
parent.removeView(rootView.findViewById(R.id.seekBar2));

final ImageView imageView = (ImageView)
     rootView.findViewById(R.id.imageView);
imageView.setImageResource(R.drawable.transitions);

Button button = new Button(getActivity());
button.setText("Start Transition");
button.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View view) {
          TransitionDrawable drawable = (TransitionDrawable)
               imageView.getDrawable();
          drawable.startTransition(500);
     }
});
parent.addView(button);

return rootView;
```

```
}

View experiment_clip_drawable(int choice, LayoutInflater
     inflater, ViewGroup container){
    View rootView =
         inflater.inflate(R.layout.fragment_drawable_experiment,
         container, false);
    rootView.findViewById(R.id.seekBar2).setVisibility(View.INVISIBLE);

    final TextView textView = (TextView)
         rootView.findViewById(R.id.textView);
    textView.setText("Clipping Value: 0");
    textView.setTextSize(30);
    final ImageView imageView = (ImageView)
         rootView.findViewById(R.id.imageView);
    imageView.setImageResource(R.drawable.clip);

    SeekBar levelSeekBar = (SeekBar)
         rootView.findViewById(R.id.seekBar1);
    levelSeekBar.setMax(10000);
    levelSeekBar.setOnSeekBarChangeListener(new
         SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int
             progress, boolean fromUser) {
```

```
                ClipDrawable drawable = (ClipDrawable)
                    imageView.getDrawable();
                drawable.setLevel(progress);
                textView.setText("Clipping Value: " +
                    Integer.toString(progress));

            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });

        return rootView;
    }

    View experiment_shape_drawable(int choice, LayoutInflater
        inflater, ViewGroup container){
        View rootView =
            inflater.inflate(R.layout.fragment_drawable_experiment,
            container, false);
```

```
ViewGroup parent = (ViewGroup)
    rootView.findViewById(R.id.linearLayout);
parent.removeView(rootView.findViewById(R.id.textView));
parent.removeView(rootView.findViewById(R.id.seekBar1));
parent.removeView(rootView.findViewById(R.id.seekBar2));
parent.removeView(rootView.findViewById(R.id.imageView));

LinearLayout.LayoutParams lp = new
    LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTEN
    LinearLayout.LayoutParams.WRAP_CONTENT);
lp.setMargins(0, 50, 0, 0);

TextView textView1 = (TextView) new TextView(getActivity());
textView1.setText("TextView with a light green background
    and a dark green boarder");
textView1.setTextSize(25);
textView1.setBackgroundResource(R.drawable.shape_green_rectangle);
textView1.setLayoutParams(lp);
parent.addView(textView1);

TextView textView2 = (TextView) new TextView(getActivity());
textView2.setText("TextView with a rounded blue rectangle");
textView2.setTextSize(25);
textView2.setBackgroundResource(R.drawable.shape_rounded_blue_recta
textView2.setLayoutParams(lp);
parent.addView(textView2);
```

```
        TextView textView3 = (TextView) new TextView(getActivity());
        textView3.setText("TextView with an oval shape");
        textView3.setTextSize(25);
        textView3.setBackgroundResource(R.drawable.shape_oval);
        textView3.setLayoutParams(lp);
        parent.addView(textView3);

        return rootView;
    }


    View experiment_animation_drawable(int choice, LayoutInflater
         inflater, ViewGroup container){
        View rootView =
             inflater.inflate(R.layout.fragment_drawable_experiment,
             container, false);
        ViewGroup parent = (ViewGroup)
             rootView.findViewById(R.id.linearLayout);

        parent.removeView(rootView.findViewById(R.id.textView));
        parent.removeView(rootView.findViewById(R.id.seekBar1));
        parent.removeView(rootView.findViewById(R.id.seekBar2));
```

```
        final ImageView imageView = (ImageView)
                rootView.findViewById(R.id.imageView);
        imageView.setImageResource(R.drawable.animation);

        Button button = new Button(getActivity());
        button.setText("Start Animation");
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                AnimationDrawable frameAnimation =
                        (AnimationDrawable) imageView.getDrawable();
                frameAnimation.start();
            }
        });
        parent.addView(button);

        return rootView;
    }

    View experiment_bitmap_drawable(int choice, LayoutInflater
            inflater, ViewGroup container){
        View rootView =
                inflater.inflate(R.layout.fragment_drawable_experiment,
                container, false);
        ViewGroup parent = (ViewGroup)
                rootView.findViewById(R.id.linearLayout);
```

```
rootView.findViewById(R.id.textView).setVisibility(View.INVISIBLE);
rootView.findViewById(R.id.seekBar1).setVisibility(View.INVISIBLE);
rootView.findViewById(R.id.seekBar2).setVisibility(View.INVISIBLE);

final ImageView imageView = (ImageView)
    rootView.findViewById(R.id.imageView);

Bitmap bitmap = Bitmap.createBitmap(300, 300,
    Bitmap.Config.ARGB_8888);
Canvas canvas = new Canvas(bitmap);

Paint paint = new Paint();
paint.setColor(Color.RED);
paint.setStyle(Paint.Style.FILL);
canvas.drawCircle(150, 150, 100, paint);

paint.setColor(Color.BLUE);
paint.setStrokeWidth(10);
paint.setStyle(Paint.Style.STROKE);
canvas.drawCircle(150, 150, 100, paint);


paint.setColor(Color.WHITE);
paint.setStrokeWidth(4);
paint.setTextSize(60);
canvas.drawText("4.5", 110, 165, paint);
```

```
        imageView.setImageBitmap(bitmap);


        return rootView;
    }
```

Part II

Drag and Drop

Outline I

- With the Android drag/drop framework, you can allow users to move data from one View to another View in the current layout using a graphical drag and drop gesture
  - drag event class
  - drag listeners
  - helper methods and classes
- create a drag event listener object ("listeners") from a class that implements View.OnDragListener
  - set the drag event listener object for a View with the View object's setOnDragListener() method
  - each View object also has a onDragEvent() callback method

1. Started
   - in response to the user's gesture to begin a drag, startDrag() tells the system to start a drag
   - startDrag() provides the data to be dragged, metadata for this data, and a callback for drawing the drag shadow
2. Continuing
   - the user continues the drag
   - as the drag shadow intersects the bounding box of a View object, the system sends one or more drag events to the View object's drag event listener
   - the listener may choose to alter its View object's appearance in response to the event
3. Dropped
   - the user releases the drag shadow within the bounding box of a View that can accept the data

4. Ended
    - indicate that the drag operation is over

getAction() value	Meaning
`ACTION_DRAG_STARTED`	A View object's drag event listener receives this event action type just after the application calls `startDrag()` and gets a drag shadow.
`ACTION_DRAG_ENTERED`	A View object's drag event listener receives this event action type when the drag shadow has just entered the bounding box of the View. This is the first event action type the listener receives when the drag shadow enters the bounding box. If the listener wants to continue receiving drag events for this operation, it must return boolean `true` to the system.
`ACTION_DRAG_LOCATION`	A View object's drag event listener receives this event action type after it receives a `ACTION_DRAG_ENTERED` event while the drag shadow is still within the bounding box of the View.
`ACTION_DRAG_EXITED`	A View object's drag event listener receives this event action type after it receives a `ACTION_DRAG_ENTERED` and at least one `ACTION_DRAG_LOCATION` event, and after the user has moved the drag shadow outside the bounding box of the View.
`ACTION_DROP`	A View object's drag event listener receives this event action type when the user releases the drag shadow over the View object. This action type is only sent to a View object's listener if the listener returned boolean `true` in response to the `ACTION_DRAG_STARTED` drag event. This action type is not sent if the user releases the drag shadow on a View whose listener is not registered, or if the user releases the drag shadow on anything that is not part of the current layout. The listener is expected to return boolean `true` if it successfully processes the drop. Otherwise, it should return `false`.
`ACTION_DRAG_ENDED`	A View object's drag event listener receives this event action type when the system is ending the drag operation. This action type is not necessarily preceded by an `ACTION_DROP` event. If the system sent a `ACTION_DROP`, receiving the `ACTION_DRAG_ENDED` action type does not imply that the drop operation succeeded. The listener must call `getResult()` to get the value that was returned in response to `ACTION_DROP`. If an `ACTION_DROP` event was not sent, then `getResult()` returns `false`.

getAction() value	getClipDescription() value	getLocalState() value	getX() value	getY() value	getClipData() value	getResult() value
ACTION_DRAG_STARTED	X	X	X			
ACTION_DRAG_ENTERED	X	X	X	X		
ACTION_DRAG_LOCATION	X	X	X	X		
ACTION_DRAG_EXITED	X	X				
ACTION_DROP	X	X	X	X	X	
ACTION_DRAG_ENDED	X	X				X

- During a drag and drop operation, the system displays a image that the user drags
- For data movement, this image represents the data being dragged
- For other operations, the image represents some aspect of the drag operation

```
// Create a string for the ImageView label
private static final String IMAGEVIEW_TAG = "icon bitmap"

// Creates a new ImageView
ImageView imageView = new ImageView(this);

// Sets the bitmap for the ImageView from an icon bit map (defined
    elsewhere)
imageView.setImageBitmap(mIconBitmap);

// Sets the tag
imageView.setTag(IMAGEVIEW_TAG);

    ...

// Sets a long click listener for the ImageView using an anonymous
    listener object that
// implements the OnLongClickListener interface
imageView.setOnLongClickListener(new View.OnLongClickListener() {

  // Defines the one method for the interface, which is called when the
      View is long-clicked
  public boolean onLongClick(View v) {
    // Create a new ClipData.
    // This is done in two steps to provide clarity. The convenience
        method
```

```
// ClipData.newPlainText() can create a plain text ClipData in one
     step.

// Create a new ClipData.Item from the ImageView object's tag
ClipData.Item item = new ClipData.Item(v.getTag());

// Create a new ClipData using the tag as a label, the plain text
     MIME type, and
// the already-created item. This will create a new ClipDescription
     object within the
// ClipData, and set its MIME type entry to "text/plain"
ClipData dragData = new
     ClipData(v.getTag(),ClipData.MIMETYPE_TEXT_PLAIN,item);

// Instantiates the drag shadow builder.
View.DragShadowBuilder myShadow = new
     MyDragShadowBuilder(imageView);

// Starts the drag

v.startDrag(dragData,  // the data to be dragged
  myShadow,  // the drag shadow builder
  null,      // no need to use local data
  0          // flags (not currently used, set to 0)
);
```

```
    }
}
```

```
private static class MyDragShadowBuilder extends View.DragShadowBuilder
    {

  // The drag shadow image, defined as a drawable thing
  private static Drawable shadow;

  // Defines the constructor for myDragShadowBuilder
  public MyDragShadowBuilder(View v) {

    // Stores the View parameter passed to myDragShadowBuilder.
    super(v);

    // Creates a draggable image that will fill the Canvas provided by
        the system.
    shadow = new ColorDrawable(Color.LTGRAY);
  }

  // Defines a callback that sends the drag shadow dimensions and touch
      point back to the
  // system.
  @Override
  public void onProvideShadowMetrics (Point size, Point touch) {
    // Defines local variables
    private int width, height;
```

```
        // Sets the width of the shadow to half the width of the original
            View
        width = getView().getWidth() / 2;

        // Sets the height of the shadow to half the height of the original
            View
        height = getView().getHeight() / 2;

        // The drag shadow is a ColorDrawable. This sets its dimensions to
            be the same as the
        // Canvas that the system will provide. As a result, the drag
            shadow will fill the
        // Canvas.
        shadow.setBounds(0, 0, width, height);

        // Sets the size parameter's width and height values. These get
            back to the system
        // through the size parameter.
        size.set(width, height);

        // Sets the touch point's position to be in the middle of the drag
            shadow
        touch.set(width / 2, height / 2);
    }
```

```
    // Defines a callback that draws the drag shadow in a Canvas that the
        system constructs
    // from the dimensions passed in onProvideShadowMetrics().
    @Override
    public void onDrawShadow(Canvas canvas) {

      // Draws the ColorDrawable in the Canvas passed in from the system.
      shadow.draw(canvas);
    }
}
```

```
// Creates a new drag event listener
mDragListen = new myDragEventListener();

View imageView = new ImageView(this);

// Sets the drag event listener for the View
imageView.setOnDragListener(mDragListen);

...

protected class myDragEventListener implements View.OnDragListener {

  // This is the method that the system calls when it dispatches a drag
       event to the
  // listener.
  public boolean onDrag(View v, DragEvent event) {

    // Defines a variable to store the action type for the incoming
         event
    final int action = event.getAction();

    // Handles each of the expected events
    switch(action) {

    case DragEvent.ACTION_DRAG_STARTED:
```

```
        // Determines if this View can accept the dragged data
        if(
             event.getClipDescription().hasMimeType(ClipDescription.MIMETYPE_T
             ) {

          // As an example of what your application might do,
          // applies a blue color tint to the View to indicate that it
               can accept
          // data.
          v.setColorFilter(Color.BLUE);

          // Invalidate the view to force a redraw in the new tint
          v.invalidate();

          // returns true to indicate that the View can accept the
               dragged data.
          return true;

        }

        // Returns false. During the current drag and drop operation,
             this View will
        // not receive events again until ACTION_DRAG_ENDED is sent.
        return false;

      case DragEvent.ACTION_DRAG_ENTERED:
```

```
        // Applies a green tint to the View. Return true; the return
            value is ignored.
        v.setColorFilter(Color.GREEN);

        // Invalidate the view to force a redraw in the new tint
        v.invalidate();

        return true;

      case DragEvent.ACTION_DRAG_LOCATION:

        // Ignore the event
        return true;

      case DragEvent.ACTION_DRAG_EXITED:

        // Re-sets the color tint to blue. Returns true; the return value
            is ignored.
        v.setColorFilter(Color.BLUE);

        // Invalidate the view to force a redraw in the new tint
        v.invalidate();

        return true;
```

```
case DragEvent.ACTION_DROP:

  // Gets the item containing the dragged data
  ClipData.Item item = event.getClipData().getItemAt(0);

  // Gets the text data from the item.
  dragData = item.getText();

  // Displays a message containing the dragged data.
  Toast.makeText(this, "Dragged data is " + dragData,
       Toast.LENGTH_LONG);

  // Turns off any color tints
  v.clearColorFilter();

  // Invalidates the view to force a redraw
  v.invalidate();

  // Returns true. DragEvent.getResult() will return true.
  return true;

case DragEvent.ACTION_DRAG_ENDED:

  // Turns off any color tinting
  v.clearColorFilter();
```

```
        // Invalidates the view to force a redraw
        v.invalidate();

        // Does a getResult(), and displays what happened.
        if (event.getResult()) {
          Toast.makeText(this, "The drop was handled.",
              Toast.LENGTH_LONG);

        } else {
          Toast.makeText(this, "The drop didn't work.",
              Toast.LENGTH_LONG);
        }

        // returns true; the value is ignored.
        return true;

        // An unknown action type was received.
      default:
        Log.e("DragDrop Example","Unknown action type received by
            OnDragListener.");
        break;
      }

      return false;
    }
};
```

```
// Set Long Click Listener => Start Drag!
imageView.setOnLongClickListener(new MyLongClickListener());

// Set Drag Event Listener
imageView.setOnDragListener(new MyDragListener());

private final class MyLongClickListener implements
     View.OnLongClickListener {
  @Override
  public boolean onLongClick(View view){
    ClipData data = ClipData.newPlainText("", "");
    View.DragShadowBuilder shadowBuilder = new
         View.DragShadowBuilder(view);
    //MyDragShadowBuilder shadowBuilder = new
         MyDragShadowBuilder(view);
    view.startDrag(data, shadowBuilder, view, 0);
    return true;
  }
}
```

```
private static class MyDragShadowBuilder extends
    View.DragShadowBuilder {
  private static Drawable shadow;

  public MyDragShadowBuilder(View v) {
    super(v);
    shadow = new ColorDrawable(Color.RED);
  }

  @Override
  public void onProvideShadowMetrics (Point size, Point touch) {
    int width, height;

    // Sets the width/height of the shadow to half the width of the
        original View
    width = getView().getWidth() / 2;
    height = getView().getHeight() / 2;
    shadow.setBounds(0, 0, width, height);

    // Sets the size parameter's width and height values. These get
        back to the system
    // through the size parameter.
    size.set(width, height);

    // Sets the touch point's position
    touch.set(0, 0);
```

```
    }

    // Defines a callback that draws the drag shadow in a Canvas that
         the system constructs
    // from the dimensions passed in onProvideShadowMetrics().
    @Override
    public void onDrawShadow(Canvas canvas) {
      // Draws the ColorDrawable in the Canvas passed in from the
           system.
      shadow.draw(canvas);
    }
}
```

```
private final class MyDragListener implements View.OnDragListener
{
  @Override
  public boolean onDrag(View v, DragEvent event) {
    int id = (Integer) v.getId();
    Map<String,?> movie = moviesList.get(id);
    int icon = (Integer) movie.get("image");

    switch (event.getAction()) {
    case DragEvent.ACTION_DRAG_STARTED:
      Log.d("onDrag", "DRAG_STARTED");
      break;
    case DragEvent.ACTION_DRAG_ENTERED:
      Log.d("onDrag", "DRAG_ENTERED");
      Drawable[] layers = new Drawable[2];
      layers[0] = getResources().getDrawable(icon);
      layers[1] =
           getResources().getDrawable(R.drawable.shape_droptarget);
      LayerDrawable layerDrawable = new LayerDrawable(layers);
      if (v instanceof ImageView) {
        ImageView imageView = (ImageView) v;
        imageView.setImageDrawable(layerDrawable);
      }
      break;
    case DragEvent.ACTION_DRAG_EXITED:
      Log.d("onDrag", "DRAG_EXITED");
```

```
        if (v instanceof ImageView) {
          ImageView imageView = (ImageView) v;
          imageView.setImageResource(icon);
        }
        break;
      case DragEvent.ACTION_DROP:
        String s = Integer.toString(v.getId());
        if (v instanceof ImageView) {
          ImageView imageView = (ImageView) v;
          imageView.setImageResource(icon);
        }
        View view = (View) event.getLocalState();
        ViewGroup owner = (ViewGroup) view.getParent();

        if (choice == 0) {
          // Swap the positions of the two views
          int indexFrom = owner.indexOfChild(view);
          int indexTo = owner.indexOfChild(v);
          owner.removeView(view);
          owner.addView(view, indexTo);
          owner.removeView(v);
          owner.addView(v, indexFrom);
        } else if (choice ==1) {
          float x1 = view.getX();
          float y1 = view.getY();
          float x2 = v.getX();
```

```
            float y2 = v.getY();
            v.animate().setDuration(1000)
              .x(x1)
              .y(y1)
              .rotationYBy(720)
              .scaleX(1.0F).scaleY(1.0F);

            view.animate().setDuration(1000)
              .x(x2)
              .y(y2)
              .rotationYBy(720)
              .scaleX(1.0F).scaleY(1.0F);
          }
          break;
        case DragEvent.ACTION_DRAG_ENDED:
          break;
        default:
          break;
        }
        return true;
    }
}
```

Part III

Animation Resources

Outline I

- Property Animation
  - creates an animation by modifying an object's property values over a set period of time with an Animator
- View Animation
  - Tween animation: creates an animation by performing a series of transformations on a single image with an Animation
  - Frame animation: creates an animation by showing a sequence of images in order with an AnimationDrawable

- modifies properties of the target object, such as background color or alpha value, over a set amount of time
- res/animator/filename.xml
- reference
  - In Java: R.animator.filename
  - In XML: @[package:]animator/filename
- syntax

```
<set
  android:ordering=["together" | "sequentially"]>

    <objectAnimator
        android:propertyName="string"
        android:duration="int"
        android:valueFrom="float | int | color"
        android:valueTo="float | int | color"
        android:startOffset="int"
        android:repeatCount="int"
        android:repeatMode=["repeat" | "reverse"]
        android:valueType=["intType" | "floatType"]/>

    <animator
        android:duration="int"
        android:valueFrom="float | int | color"
        android:valueTo="float | int | color"
        android:startOffset="int"
        android:repeatCount="int"
        android:repeatMode=["repeat" | "reverse"]
        android:valueType=["intType" | "floatType"]/>

    <set>
        ...
    </set>
</set>
```

- example: res/animator/property_animator.xml:

```
<set android:ordering="sequentially">
    <set>
        <objectAnimator
            android:propertyName="x"
            android:duration="500"
            android:valueTo="400"
            android:valueType="intType"/>
        <objectAnimator
            android:propertyName="y"
            android:duration="500"
            android:valueTo="300"
            android:valueType="intType"/>
    </set>
    <objectAnimator
        android:propertyName="alpha"
        android:duration="500"
        android:valueTo="1f"/>
</set>
```

```
AnimatorSet set = (AnimatorSet)
     AnimatorInflater.loadAnimator(myContext,
    R.anim.property_animator);
set.setTarget(myObject);
set.start();
```

- performs transitions such as rotating, fading, moving, and stretching on a graphic
- res/anim/filename.xml
- reference
  - In Java: R.anim.filename
  - In XML: @[package:]anim/filename
- syntax

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@[package:]anim/interpolator_resource"
    android:shareInterpolator=["true" | "false"] >
    <alpha
        android:fromAlpha="float"
        android:toAlpha="float" />
    <scale
        android:fromXScale="float"
        android:toXScale="float"
        android:fromYScale="float"
        android:toYScale="float"
        android:pivotX="float"
        android:pivotY="float" />
    <translate
        android:fromXDelta="float"
        android:toXDelta="float"
        android:fromYDelta="float"
        android:toYDelta="float" />
    <rotate
        android:fromDegrees="float"
        android:toDegrees="float"
        android:pivotX="float"
        android:pivotY="float" />
    <set>
        ...
```

```
    </set>
</set>
```

- example: res/anim/hyperspace_jump.xml:

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <scale
        android:interpolator="@android:anim/accelerate_decelerate_inte
        android:fromXScale="1.0"
        android:toXScale="1.4"
        android:fromYScale="1.0"
        android:toYScale="0.6"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fillAfter="false"
        android:duration="700" />
    <set
        android:interpolator="@android:anim/accelerate_interpolator"
        android:startOffset="700">
        <scale
```

```
            android:fromXScale="1.4"
            android:toXScale="0.0"
            android:fromYScale="0.6"
            android:toYScale="0.0"
            android:pivotX="50%"
            android:pivotY="50%"
            android:duration="400" />
        <rotate
            android:fromDegrees="0"
            android:toDegrees="-45"
            android:toYScale="0.0"
            android:pivotX="50%"
            android:pivotY="50%"
            android:duration="400" />
    </set>
</set>
```

```
ImageView image = (ImageView) findViewById(R.id.image);
Animation hyperspaceJump = AnimationUtils.loadAnimation(this,
    R.anim.hyperspace_jump);
image.startAnimation(hyperspaceJump);
```

- shows a sequence of images in order (like a film)
- res/drawable/filename.xml
- reference
  - In Java: R.drawable.filename
  - In XML: @[package:]drawable.filename
- syntax

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list
     xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot=["true" | "false"] >
    <item
        android:drawable="@[package:]drawable/drawable_resource_name"
        android:duration="integer" />
</animation-list>
```

- example: res/anim/rocket.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list
     xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@drawable/rocket_thrust1"
         android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust2"
         android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust3"
         android:duration="200" />
</animation-list>
```

```
ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
rocketImage.setBackgroundResource(R.drawable.rocket_thrust);

rocketAnimation = (AnimationDrawable) rocketImage.getBackground();
rocketAnimation.start();
```

```
// Fragment_Animation.java

public void onClick(View view) {
  switch (view.getId()) {
  case R.id.button1:  // move and rotate
    imageView.animate().setDuration(1000);
    imageView.animate().x(500).y(800)
      .rotationYBy(720)
      .scaleX(0.4F).scaleY(0.4F);
    break;
  case R.id.button2:  // move back
    imageView.animate().setDuration(1000)
      .x(imageView.getLeft())
      .y(imageView.getTop())
      .rotationYBy(720)
      .scaleX(1.0F).scaleY(1.0F);
    viewGroup.animate().setDuration(1000);
    viewGroup.animate().x(viewGroup.getLeft()).y(viewGroup.getTop())
      .rotationYBy(900)
      .scaleX(1.0F).scaleY(1.0F);
    break;
  case R.id.button3:  // fade out -- change transparency
    imageView.animate().setDuration(1000)
      .alpha(0f);
    break;
  case R.id.button4:  // fade in
```

```
        imageView.animate().setDuration(1000)
          .alpha(1f);
        break;
      case R.id.button5:  // use animator from XML
        AnimatorSet set = (AnimatorSet)
            AnimatorInflater.loadAnimator(getActivity(),
            R.animator.fancy_animation);
        set.setTarget(imageView);
        set.start();
      break;
      case R.id.button6: // view group animation
        viewGroup.animate().setDuration(1000);
        viewGroup.animate().x(480).y(920)
          .rotationYBy(900)
          .scaleX(1.8F).scaleY(1.8F);
        break;
      default:
        break;
      }
  }
```

- /res/animator/fancy_animation

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:duration="1000"
        android:valueFrom="800"
        android:valueTo="0"
        android:propertyName="x"
        android:valueType="floatType"/>

    <objectAnimator
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:duration="1000"
        android:valueFrom="1500"
        android:valueTo="0"
        android:propertyName="y"
        android:valueType="floatType"/>

    <objectAnimator
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:duration="1000"
        android:interpolator="@android:interpolator/accelerate_decelerate"
        android:propertyName="rotationY"
        android:valueFrom="0"
```

```
        android:valueTo="720" />

    <objectAnimator
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:duration="1000"
        android:interpolator="@android:interpolator/accelerate_decelerate"
        android:propertyName="rotationX"
        android:valueFrom="0"
        android:valueTo="720" />
</set>
```

```
// Activity_Animation.java

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        Fragment_Animation fragment =
             Fragment_Animation.newInstance(0);
        switch (id){
            case R.id.action_from_left:
                fragment.setEnterTransition(new
                     Slide(Gravity.LEFT));
                getSupportFragmentManager().beginTransaction()
                        .replace(R.id.container, fragment)
                        .commit();
                break;
            case R.id.action_in_and_out:
                fragment.setEnterTransition(new
                     Slide(Gravity.RIGHT));
                fragment.setExitTransition(new
                     Slide(Gravity.BOTTOM));
                getSupportFragmentManager().beginTransaction()
                        .replace(R.id.container, fragment)
                        .commit();
                break;
            case R.id.action_combined:
```

```
                fragment.setEnterTransition(new
                    Slide(Gravity.BOTTOM));
                fragment.setExitTransition(new Slide(Gravity.LEFT));
                getSupportFragmentManager().beginTransaction()
                        .replace(R.id.container, fragment)
                        .commit();
                break;
            case R.id.action_others:
            default:
                getSupportFragmentManager().beginTransaction()
                        .replace(R.id.container, fragment)
                        .commit();
                break;

        }
        return true;

    }
```

```
// Prepare the shared element in Activity_DragandDrop.java

imageView.setTransitionName(name); // for animation
imageView.setOnClickListener(new MyClickListener());



// Event Listener

private final class MyClickListener implements View.OnClickListener
    {
  @Override
  public void onClick(View view){
    int id = (Integer) view.getId();
    HashMap<String,?> movie = (HashMap<String,?>)
        moviesList.get(id);
    mListener.onItemSelected(movie, view);
  }
}
```

```
// Load Detail Fragment

    @Override
    public void onItemSelected(HashMap<String, ?> movie, View
        sharedImage) {
        Fragment_DetailView details =
            Fragment_DetailView.newInstance(movie);
        details.setSharedElementEnterTransition(new
            DetailsTransition());
        details.setEnterTransition(new Fade());
        //details.setEnterTransition(new Slide());
        details.setExitTransition(new Fade());
        details.setSharedElementReturnTransition(new
            DetailsTransition());

        getSupportFragmentManager().beginTransaction()
                        .addSharedElement(sharedImage,
                            sharedImage.getTransitionName())
                        .replace(R.id.container, details)
                        .addToBackStack(null)
                        .commit();
    }
```

```
// DetailTransition.java

public class DetailsTransition extends TransitionSet {
    public DetailsTransition() {
        setOrdering(ORDERING_TOGETHER);
        addTransition(new ChangeBounds())
                .addTransition(new ChangeTransform())
                .addTransition(new ChangeImageTransform());
    }
}
```

```
    <Button
        android:id="@+id/button3"
        android:text="Drag and Drop Experiment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:transitionName="testAnimation"/>
```

```
<FrameLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:ignore="MergeRootFrame"
    android:transitionName="testAnimation"/>
```

```
case R.id.button3:
  intent = new Intent(getActivity(), Activity_DragandDrop.class);
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
    ActivityOptionsCompat options = ActivityOptionsCompat.
      makeSceneTransitionAnimation(getActivity(), view,
          "testAnimation");
    getActivity().startActivity(intent, options.toBundle());
  }
  else {
    startActivity(intent);
  }
  break;
```