UNIVERSITY of WASHINGTON

Android 250 - Lecture 7

Animation

Margaret Maynard-Reid

May 11, 2015

Agenda

- Animation
  - Frame Animation
  - View Animation
  - Property Animation
- Material Design
- Sample Code
  - SampleAnimation
  - SampleViewFlipper
  - CardView
  - FloatingActionButtonBasic
  - RevealEffectBasic

Android Stories

- You Can Now Use Shazam, Instacart, And Other Android Apps With “Okay Google” Commands
- Google will intro Voice Access service at I/O for controlling apps without touching your phone
- Android may soon offer more granular privacy control

Review from Last Week

- Can you create a custom TextView?
- Which MotionEvent action indicates a touch has started?
- What are some common gestures in Android?
- What do you use to store/retrieve custom gestures?

Why Use Animation?

Animation is a fundamental building block for interactive applications

- Transitions (Maintaining Context, Swiping, etc.)
- Spatial awareness (Homescreen, Boundaries, etc.)
- Noting effects (Deletions, Updates, etc.)

Animation Key Classes

- Animation
  - Defined in Android as an abstract class
  - "Rotate 10 degrees"
- AnimationSet
  - A group of related or dependent animations
  - "Scale 50%" and "Rotate 10 degrees" at the same time
- Interpolators
  - Control the rate of change in an animation

Animation Types

- Drawable Animation
  - sequenced animations using drawables
- View Animation
  - older system used to animate Views
- Property Animation
  - introduced in API 11
  - can animate object properties

Animation

Drawable Animation

Drawable Animation

- Also called Frame-by-Frame animation
- Use a AnimationDrawable:
    - Define a set of images as the frames of animation
    - Define how long each image stays on screen
    - Define whether the animation should loop

AnimationDrawable

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">

    <item android:drawable="@drawable/frame1" android:duration="300" />

    <item android:drawable="@drawable/frame2" android:duration="300" />

    <item android:drawable="@drawable/frame3" android:duration="300" />

</animation-list>
```

Sample Code

- Walk through SampleAnimation
  - Take a look at an example of using Drawable Animation

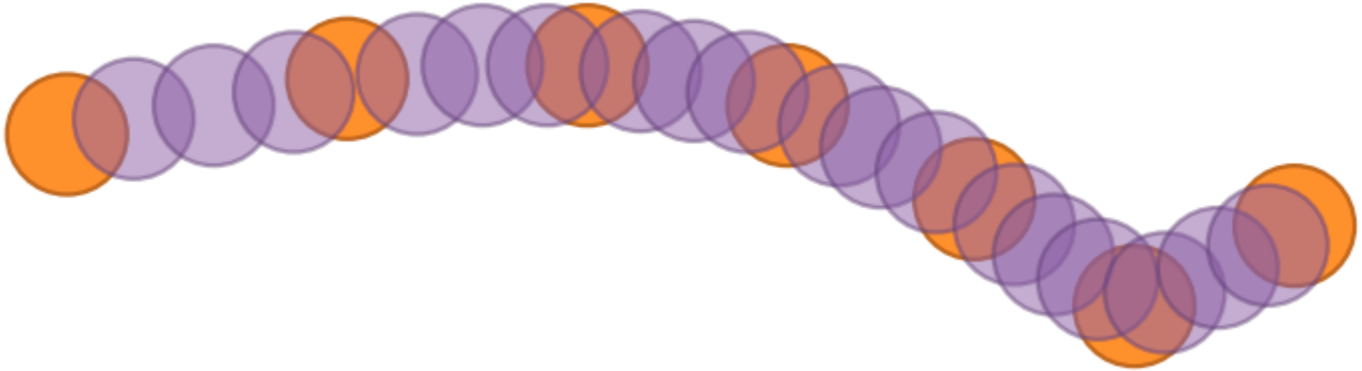
Break

Animation

View Animation

View Animation

- Tweened animation
  - Animation through the creation of inbetween frames given key frames



View Animation

Animations:

- AlphaAnimation - controls the alpha level of an object
- RotateAnimation - controls the rotation of an object
- ScaleAnimation - controls the scale of an object
- TranslateAnimation - controls the position of an object

View Animation - Interpolators

An interpolator defines the rate of change of an animation:

- **AccelerateDecelerateInterpolator** - the rate of change starts and ends slowly but accelerates through the middle
- **AccelerateInterpolator** - the rate of change starts out slowly and then accelerates
- **AnticipateInterpolator** - the change starts backward then flings forward
- **AnticipateOvershootInterpolator -** where the change starts backward then flings forward and overshoots the target value and finally goes back to the
- **BounceInterpolator** - the change bounces at the end
- **CycleInterpolator** - repeats the animation for a specified number of cycles
- **DecelerateInterpolator** - the rate of change starts out quickly and then decelerates
- **LinearInterpolator** - the rate of change is constant
- **OvershootInterpolator** - the change flings forward and overshoots the last value then comes back final value

Creating View Animations

Call startAnimation() or setAnimation() with a start time set in the Animation

Animation spinAnimation = AnimationUtils.loadAnimation(this, R.anim.spin);

someView.startAnimation(spinAnimation);

Animation

Property Animation

Property Animation

Used to animate object properties

- Any numeric property or field can be animated
  - X and Y positions
  - Alpha
  - Height and Width
  - TextSize
  - TextColor
  - ...

Creating Property Animations

- Create a ValueAnimator and deal with updates

```
ValueAnimator animator = ValueAnimator.ofInt(30);
animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
        @Override
        public void onAnimationUpdate(ValueAnimator animation) {
                float value = ((Float) (animation.getAnimatedValue())).floatValue();
                someView.setTranslationY(value));
}

animator.setDuration(30000);
animator.start();
```

ViewPropertyAnimator

- Introduced in Android 3.1
- A **simple** way to animate several properties of a View **simultaneously**
- Better performance
- Auto-start: no need to call start() in order to start animation
- The call to View.animate() returns an instance of ViewPropertyAnimator

ViewPropertyAnimator

Can perform these animations:

1. Fade in/out ← alter alpha channel values
2. Move ← alter X and Y values
3. Rotate ← rotate around X, Y or Z
4. Grow/Shrink ← alter the scale

Define Animation in XML

- <animator> ← ValueAnimator
- <objectAnimator> ← ObjectAnimator
- <set> ← AnimatorSet

Animation

Comparison

View vs Property Animation

View Animation	Property Animation
Pros: ● Easier to set up Cons: ● Can only animate the Views (where they are drawn)	Pros: ● Can animation any property of the objects Cons: ● More code to set up

/anim vs /animator

/res/anim	/res/animator
● Points to Animation: android.view.animation.Animation ● Used for View animation	● Points to Animator or its subclasses: android.animation.Animator ● Used for Object & Property animation

Animating Activity Transitions

- Starting Activity
    - Call overridePendingTransition() after startActivity()
- Ending Activity
    - Override finish() and set your overridePendingTransition() with your specified animations

Animating Fragment Transitions

- Use the setCustomAnimations() methods as a part of the FragmentTransaction

Animating Layout Changes

Use the LayoutTransition class

– Allows you to apply a default animation to a ViewGroup

http://developer.android.com/reference/android/animation/LayoutTransition.html

Sample Code

- Walk through SampleAnimation
  - View Animation
  - Property Animation
- Walk through SampleViewFlipper

Break

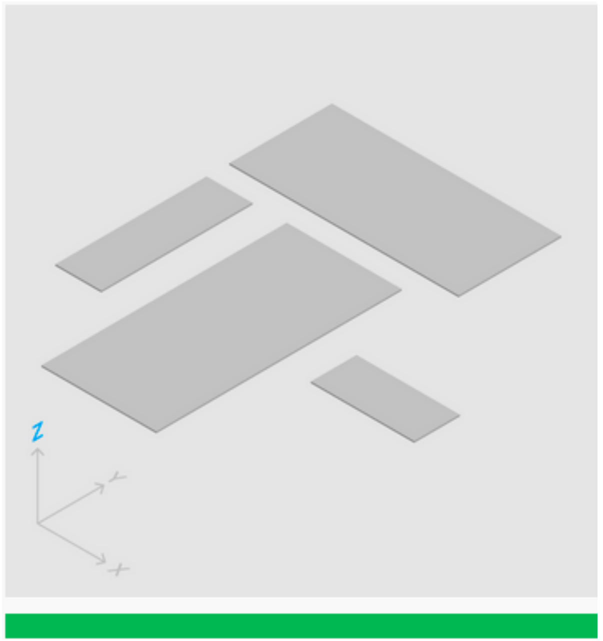
Material Design

- Unify the experience for the end users, across all Google platforms.
- Grounded in the physics of real world, use paper and ink

http://www.google.com/design/spec/material-design/introduction.html

What is Material?

- A metaphor of tactile
- Depth & Shadow
- Use z value to indicate depth of the View
- x & y can vary but z should be 1dp



Material Design

- Tangible Surfaces
- A Bold, Print-Like Aesthetic
- Animation:
  - Authentic motion
  - Responsive interaction
  - Meaningful transition
  - Delightful details

Tangible Surfaces

- Floating App Bar ActionBar
- Floating Action Button
- CardView

A Bold, Print-Like Aesthetic

- Primary and accent color
- Coloring tinting
- Status bar can be colored or translucent
- Pallete class - helper class used to extract colors from the images in your app.

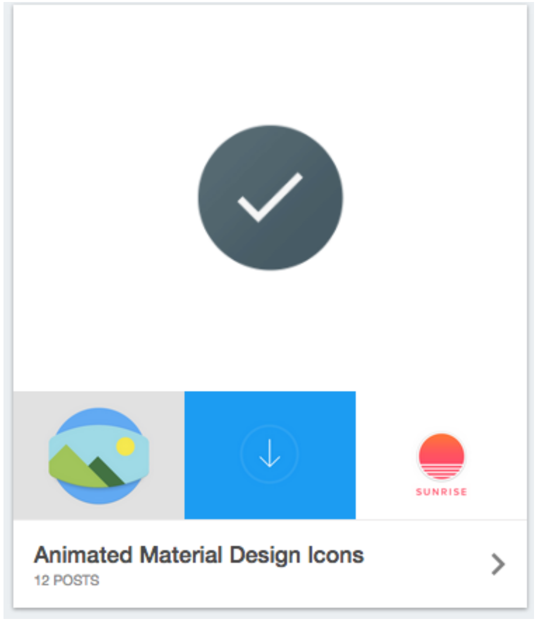
https://developer.android.com/reference/android/support/v7/graphics/Palette.html

Material Design Animation

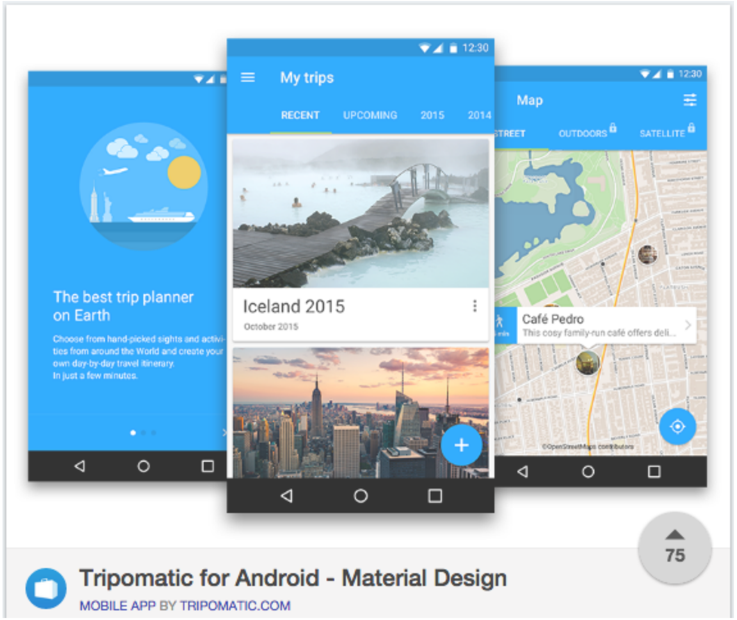
A few examples:

- Touch feedback: Ripple Effect
- Circular Reveal
- Curved Motion
- Animated Activity Transitions

Showcase Material Design



Animated Material Design Icons
12 POSTS



Choose from hand-picked sights and activities from around the World and create your own day-by-day travel itinerary. In just a few minutes.

Tripomatic for Android - Material Design
MOBILE APP BY TRIPOMATIC.COM

http://www.materialup.com/

Material Theme

Not all material design features and themes are backwards compatible. Use alternative resources:

- /values-v11, define a styles.xml:

```
<style name="Theme.Base" parent="android:Theme.Holo.Light" />
```

- Under values-v21, define a separate styles.xml file

```
<style name="Theme.Base" parent="android:Theme.Material.Light">
```

CardView

- android.support.v7.widget.CardView
- Subclass of FrameLayout
- You can set elevation
- You can set corner radius

Circular Reveal

Use ViewAnimationUtils.createCircularReveal() method:

```
View shape = rootView.findViewById(R.id.circle);
Animator animator = ViewAnimationUtils.createCircularReveal(
        shape,
        0,
        0,
        0,
        (float) Math.hypot(shape.getWidth(), shape.getHeight()));
animator.setInterpolator(new AccelerateDecelerateInterpolator());
animator.start();
```

Floating Action Button

- Use a ImageView
- Set the background ImageView
- Clip it to a circle shape
- Use StateListDrawable to give shadow animation

Material Design Sample Code

- Walk through a few Material Design samples:
    - Material theme
    - CardView
    - Circular Reveal
    - Floating Action Button

Next week

- Homework 2 due on 5/18/2015
- We will cover Notifications