CSc 317
Animation

Benjamin Dicken

Announcements

- The Schedule
- Final Project Documents

DVD Screensaver

Move the DVD logo

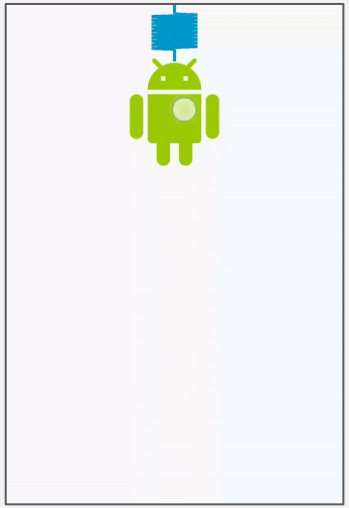
- Use an **ObjectAnimator** to animate the logo
- The value itself doesn't particularly matter, but set the animation to last a very long time
- Use an **onAnimationUpdate** to check if the logo should change course, and use **setX** and **setY** to update the location of the image

```
final ImageView iv = findViewById(R.id.dvd);
final ValueAnimator dvdAnimator = ...
dvdAnimator.setDuration(10000000L);
dvdAnimator.start();
dvdAnimator.addUpdateListener(new ObjectAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
        iv.setX(dvdX);  iv.setY(dvdY);
    }});
```

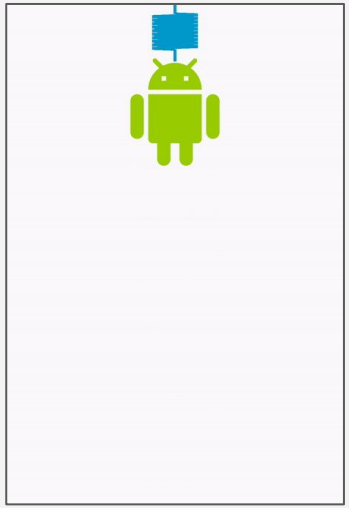
```
final ImageView iv = findViewById(R.id.dvd);
final ValueAnimator dvdAnimator = ...
dvdAnimator.setDuration(10000000L);
dvdAnimator.start();
dvdAnimator.addUpdateListener(new
ObjectAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator
animation) {
        iv.setX(dvdX);  iv.setY(dvdY);
    }});
```

Physics-Based Animation

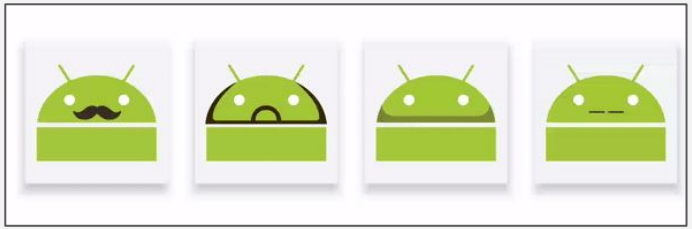
- Physics-based animation with **Spring** and **Fling** animation
  - **Spring** - Movement with spring-like characteristics, two-drections
  - **Fling** force-based, slows in one direction



Bounce



Stiffness



Fling

Spring animation

```
ImageView iv = findViewById(R.id.???);

SpringAnimation animation = new SpringAnimation(iv, DynamicAnimation.X);
animation.setStartValue(0);

// Define the spring force and stiffness
SpringForce force = new SpringForce();
Force.setDampingRatio(SpringForce.DAMPING_RATIO_HIGH_BOUNCY);
Force.setStiffness(SpringForce.STIFFNESS_LOW);
animation.setSpring(force);

// Go from start location (0) to 400
animation.animateToFinalPosition(400);
```

One image, multiple values

```
ImageView iv = findViewById(R.id.???);

SpringAnimation aX = new SpringAnimation(iv, DynamicAnimation.X);
SpringAnimation aY = new SpringAnimation(iv, DynamicAnimation.Y);
aX.setStartValue(0);
aY.setStartValue(0);

// Set start, force, stiffness . . .

// Animate from start locations
aX.animateToFinalPosition(400);
aY.animateToFinalPosition(700);
```

Move the Android

Use the starter code **PhysicsBasedAnimation**

Whenever the Android is clicked, move to a random location

```
ImageView iv = findViewById(R.id.???);

SpringAnimation animation = new SpringAnimation(iv, DynamicAnimation.X);
animation.setStartValue(0);

// Define the spring force and stiffness
SpringForce force = new SpringForce();
Force.setDampingRatio(SpringForce.DAMPING_RATIO_HIGH_BOUNCY)
        .setStiffness(SpringForce.STIFFNESS_LOW);
animation.setSpring(force);

// Go from start location (0) to 400
animation.animateToFinalPosition(400);
```

View.OnTouchListener

- Can respond to touch events, but in a more robust way than just setting an onClick in XML
  - Can detect ***press*** vs ***release*** vs ***move***
- Useful for drag-and-drop

View.OnTouchListener

```
public class ??? implements View.OnTouchListener {
    @Override
    public boolean onTouch(View v, MotionEvent event){
        // Or ACTION_DOWN, ACTION_UP
        if (event.getAction() == MotionEvent.ACTION_MOVE) {
            // Can call v.setX(), v.setY(), event.getRawX(),
            // event.getRawY(), and more . . .
        }
        return true;
    }
}
```

Drag-and-drop images

- Use a **View.OnTouchListener** to make the three images drag-and-droppable
- Un-comment the three **ImageView**s in the XML

```
public class ??? implements View.OnTouchListener {
    @Override
    public boolean onTouch(View v, MotionEvent event){
        // Or ACTION_DOWN, ACTION_UP
        if (event.getAction() == MotionEvent.ACTION_MOVE) {
            // Can call v.setX(), v.setY(), event.getRawX(),
            // event.getRawY(), and more . . .
        }
        return true;
    }
}
```

Requesting a Permission

```
public void checkCameraPermission(View v) {
    if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
        // You can use the API that requires the permission.
        dispatchTakePictureIntent();
    } else if (/* Should we educate user why permission is needed? */) {
        // Explain why your app need the permission
    } else {
        // Ask for the permission
        requestPermissions(new String[] { Manifest.permission.CAMERA }, 100);
    }
}
```

Respond to Request

```
@Override
public void onRequestPermissionsResult(
        int requestCode, String[] permissions, int[] grantResults) {
   super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if(requestCode == 100) {
        // If request is cancelled, the result arrays are empty.
        if (grantResults.length > 0 &&
            grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            // Permission granted
            dispatchTakePictureIntent();
        }  else {
            // Permission denied
            System.out.println("Why u no gib permission?");
        }
    }
}
```

Requeset Permission

- Go to one of your previous apps that used the camera permission (CollageMaker, Tasks)
- Add a permission request popup
- Test it out!