CMSC 436 Lab 11

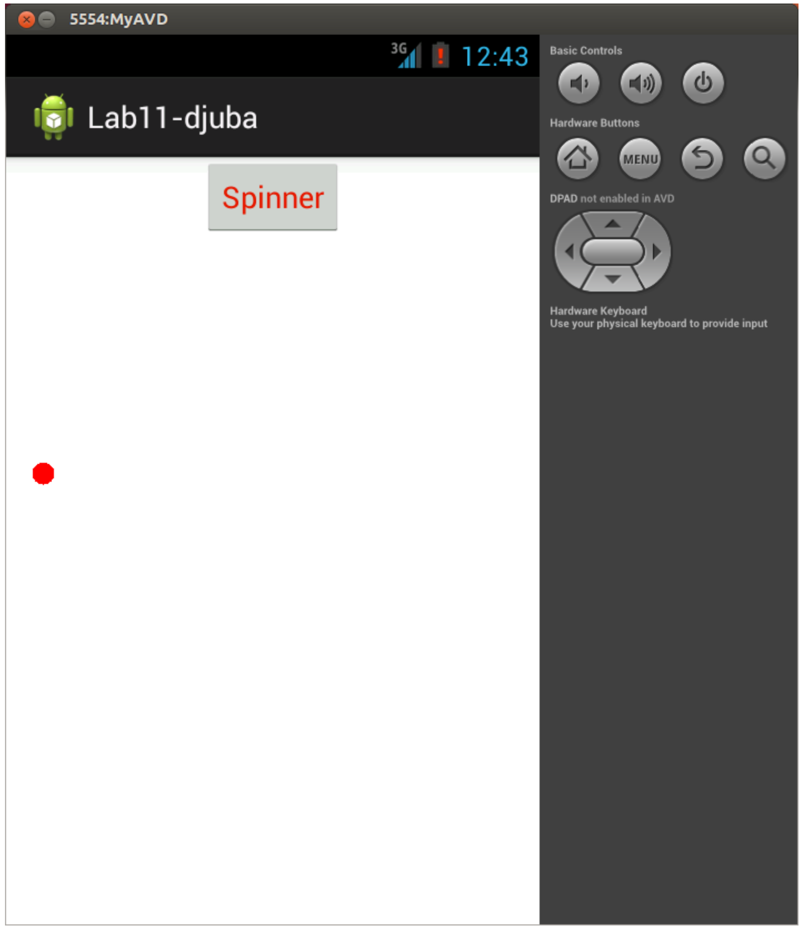
Animation and Graphics

Overview

- For this lab you will use Property Animation to animate the color of the text on a button and cause the button to spin when clicked
- You will also animate a bouncing ball on a SurfaceView using the techniques you would use when developing a 2D real-time game

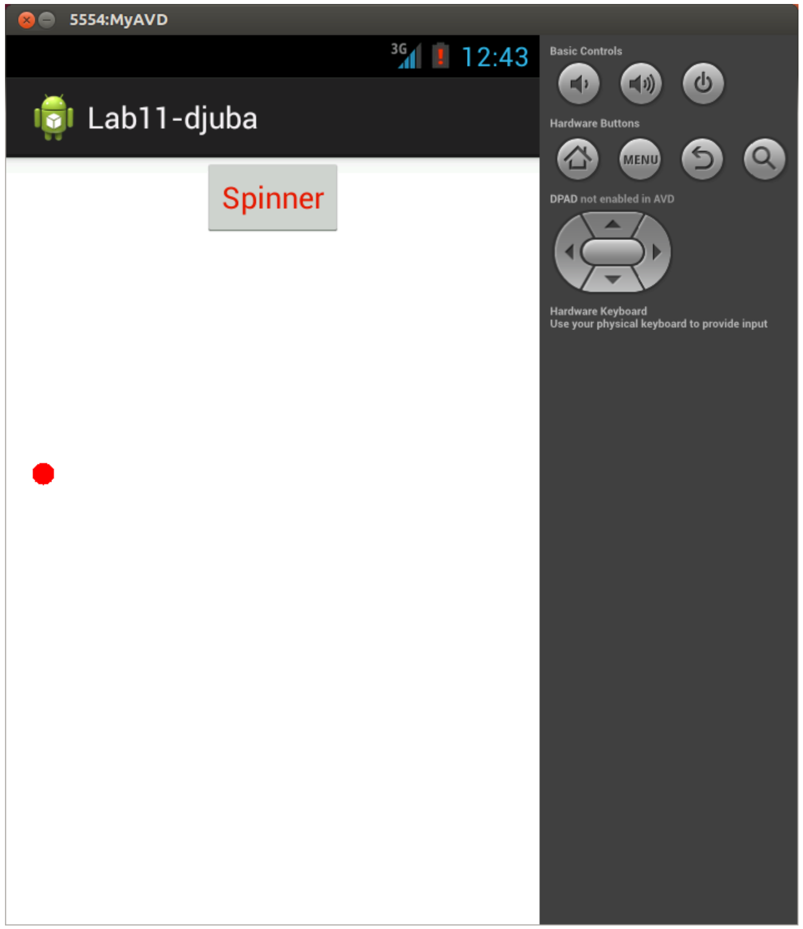
Interface

- The Spinner button should be centered at the top of the screen
- The color of the text on the button should continuously change
- Clicking the button should cause it to rotate 360°



Interface

- The red ball should continuously move in a straight line
- Hitting the edge of the screen should cause it to bounce and change direction



Property Animation

- The Android Property Animation system allows you to animate any property of an object over time (even properties that do not have a graphical effect)
- Information about the Property Animation system can be found at http://developer.android.com/guide/topics/graphics/prop-animation.html

ViewPropertyAnimator

- If the property you wish to animate is a property of the View class, you can animate it very easily using a ViewPropertyAnimator
- See the section “Animating with ViewPropertyAnimator”
- Use a ViewPropertyAnimator to make the button rotate 360° over 1 second when it is clicked (this should take one line of code)

ObjectAnimator

- Unfortunately the ViewPropertyAnimator cannot animate properties, such as text color, that are not properties of the View class
- For these properties you can use an ObjectAnimator- see the section “Animating with ObjectAnimator”

ObjectAnimator

- Use an ObjectAnimator to change the color of the text on the button from red, to green, to blue, and then back to red
- The animation should start immediately, cycle every 3 seconds, and repeat forever

ObjectAnimator

- Colors are represented as integers, but due to the way the RGB channels are packed into the integer, simply interpolating between the colors as if they were regular numbers will not cause a smooth change in the color
- To fix this, you will need to set the ObjectAnimator to use an ArgbEvaluator, which will correctly interpolate the individual color channels

SurfaceView Animations

- To animate the bouncing ball, you will draw onto a SurfaceView using a second thread
- This is a good method for applications that require real-time 2D rendering, since they don't have to wait for the View hierarchy to be redrawn to update their animation
- Information about SurfaceView animations can be found in the “On a SurfaceView” section at http://developer.android.com/guide/topics/graphics/2d-graphics.html

SurfaceView Animations

- The ball should be colored red and have a radius of 10 pixels
- It should start in the center of the screen with an (x, y) velocity vector of (100, 100)
- When the ball reaches the edge of the screen, either its x or y velocity component should be reversed to cause it to bounce
- To move the ball, compute the time elapsed since the last frame, multiply this by the (x, y) velocity, and add the result to the previous position

SurfaceView Animations

- To see how to set up SurfaceView animations, take a look at the example code in <your-sdk-directory>/samples/LunarLander/
- LunarView.java should show you what needs to go in the SurfaceView callbacks, how to set up the threading, etc.
- Note that the Lunar Lander application is far more complex than your lab needs to be
- You do not have to support destroying and recreating the Activity, or pausing and resuming the animation

SurfaceView Animations

- You should also look at lunar_layout.xml to see how to add your SurfaceView class to your Activity's layout
- You can center the button using the android:layout_centerHorizontal attribute