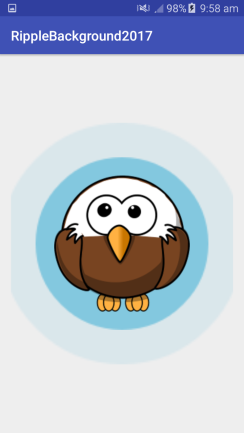
# IN721 2017 Practical 12.1 - Working with 3rd Party Libraries

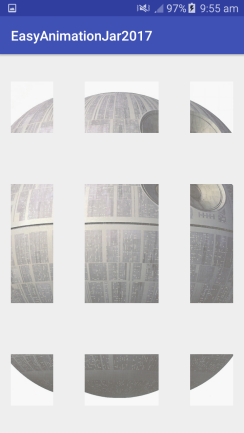
In this practical you will explore various protocols for using 3rd party libraries (i.e. those not part of the official Android classes) in your Android applications. The emphasis of this practical is on the exploration. You will not be told how to acquire, install or use these libraries -- you will get to figure that out on your own. This sort of task comprises much of the work of the IT professional. Ideally, you think it is fun.

Build the following apps using Android Studio and the specified 3rd party libraries. Note especially the different syntactic approaches (i.e. the specific method calls required) taken by the authors of these libraries.

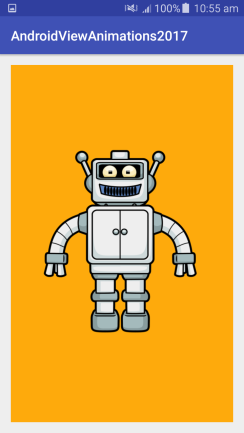
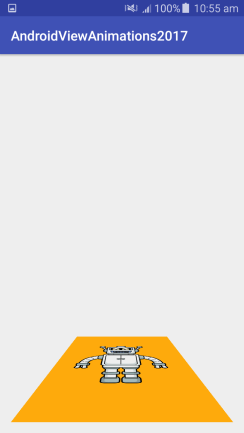
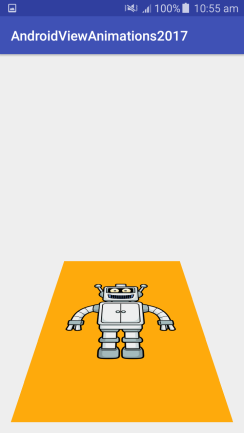
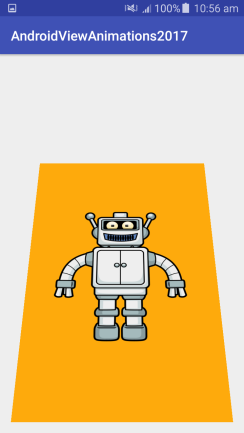
1. Use Android Ripple Background (android-arsenal.com/details/1/1107) to implement a button with a ripple animation (i.e. expanding rings like those that occur when a stone is dropped into a pond). The animation should turn (or toggle) on and off with a button click.

1. Use Easy Android Animations (android-arsenal.com/details/1/940) to implement an app that "explodes" an image when clicked. The image may be set at design time; it does not need to be changeable. Download and install the jar file for this library. You will need to look up how to add a jar to your Android Studio project

1. Use Android View Animations (android-arsenal.com/details/1/32) to implement an app that produces the "stand up" animation on an image when clicked. The image may again be set at design time.

Before During 1 During 2 During 3

1. Using Google Maps.Available3rd party libraries cover a wide range of complexity. At one end you have tools like Android Ripple Background that provide essentially a single method call. At the other, you have industrial-strength systems involving dozens of classes and hundreds of methods. Naturally, the more complex systems are more complicated to use, but support more extensive functionality.

Arguably the most extensive such system is Google Services, a composite API that provides access to a variety of Google products including Google Maps, Google+, Google Analytics and Google Cloud. Using the Google Services libraries requires registration, generation of a signed API key, installation of a separate SDK and mastery of many complex classes. If you can manage that, however, you can dramatically increase your app functionality with minimal programming effort. See <https://developers.google.com/maps/documentation/> for coverage of all the available functionality.

For this fourth task, use Google Services to gain access to the Google Maps API. Use Google Maps and the random latitude and longitude generation code from your Teleport Game to display a Google Map centred around a random location. Place a map marker at the teleport spot.

The Google Maps credentials model is much more elaborate than those we have used so far. To use Google Maps, you need a special API key, which you must bind to both a Gmail account, and the machine you are developing on. The easiest way to learn how to get your Google Maps credentials is to create an app in Android Studio with a Google Maps Activity (instead of an Empty Activity, as we have been doing) and follow the provided instructions. It is fine to start working with Google Maps this way, but you should then make sure you can add a map element manually to one of your own projects. Note also that to use Google Maps on a real device, you must have WiFi or mbile data enabled.

**Optional Extra**: To improve your app, incorporate GeoPlugin to determine whether you have teleported to a city. Add a title to your marker to distinguish between the two cases, as shown below.