User Guide - COMPASS Application

Compass is an instrument that is used for navigation. It shows the geographical direction which is one amongst North (N), East (E), West (W), South (S) – depending upon which side the user is pointing. Generally, North is represented as 0° which increases as a clockwise direction to 90° as East, 180° as South and 270° as West.

Advantages of using a compass app-

- For outdoor activities like trekking, hiking, to know the exact direction can be a matter of life and death.
- Compass apps are usually lightweight (use less memory space) and free (compare to traditional compasses which are costly).

To load and install the application follow the steps below:

1. Use https://www.dropbox.com/s/dhd1n8va636tytj/app-debug.apk?dl=0 link to download and load the application on your smartphone.
2. Click on the .apk file. It will prompt for a user permission to Install or cancel. Click on install app and wait for the installation to complete (1-3 seconds).
3. Click on done or open. It will prompt for the user permission to access the location of the device. Enable the location services for the app.
4. Re-launch the app.
5. The app is ready to use.

In this application, the developer has chosen figure 1 which not only shows the 4 directions but also shows the combination of directions.

Here

- N is North
- S is the South
- E is the East
- W is the West
- NE is North-East
- SE / ES is the South-East
- SW is South-West
- WN / NW is North-West.

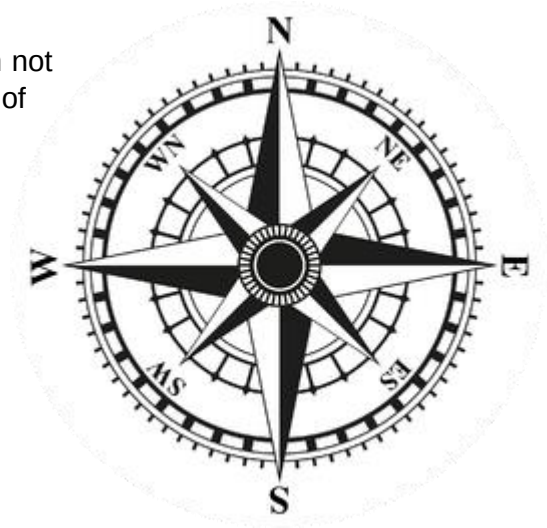

Figure 1: Compass

This application works on two sensors – magnetic field sensors and accelerometer sensors to determine the direction of the compass. Both the sensors are the basic sensors of a smartphone.

In accordance with the two sensors, the developer has added a feature of location (GPS) to determine the actual location(Co-ordinates) of the user - latitude and longitude of the user at that time.

By- Rahul Thakur s1776200
The University of Edinburgh

Some benefits of using this application:

- Digital Compass
- Ad Free
- Easy calibration
- Clean Design
- Show angles in Degree with Direction
- Small size
- Tells the Position of the user (Co-ordinates)
- Simple to use, use it like a real compass
- Magnetic and true north are available, the app automatically takes care of variation
- No internet connection required
- Rotating orienting image
- Easy Readability (white background with black text)
- Fast
- Free

Figure 2: Screenshot of Application

The app can run on devices with OS 4- Ice Cream Sandwich and above and API level 14
For this particular app, it has been tested on OS 7- Nougat with the API level 27.
The minimum SDK level the app can run smoothly is **14** and the target/compiled SDK version is **27**.
The app is really easy to use, just click on the app icon, calibrate it if it's not calibrated perfectly and now you are good to go. It will tell you the exact coordinates of the user.

*For better performance by the app, place it on a flat surface and check for the direction.

Requirements

A smartphone should have android version 4 or above with Accelerometer and Magnetometer sensors.
Smartphones without accelerometer or magnetometer can still download the app and can enjoy the user's location.

Permissions

GPS and network location:
Needed by the compass to calculate the true north as well as location.

Prevent phone from sleeping:
Needed by the compass to keep the display on while using the compass.

Note

** Keep the device away from metal objects, machinery and where there are chances of a high magnetic field to avoid false results
** Smartphones use magnetic and orientation sensor(Accelerometer) in order to compute the direction of the magnetic North. The sensor might be in an unknown(sleep) state when the application is started. Sensors require a lot of values to reach optimal accuracy and precision. To do so, move your phone in space in a ∞ figure pattern until the accuracy turns to high.
The app might work differently on different devices depending upon the quality of sensors used by the device

*** Aap might crash for the first time after installation on different devices, before asking permissions. Once permission is enables, re launch the app, it will work again.

By- Rahul Thakur s1776200
The University of Edinburgh

Programmer Guide - COMPASS Application

The input for the compass app is from two sensors – magnetometer sensor and accelerometer sensors. Another input is also there with is coming from the location services ie. GPS.

The output of this application is the logical combination of both the sensors to show the degrees and the user's current location.

Key variables used in this application are

- **asensor** – sensor type for accelerometer sensor
- **msensor** – sensor type for magnetic field sensor
- **sm** – type sensor manager to initialize android device sensor capabilities
- **accelerometerValues** - type float, taking data from an accelerometer sensor
- **magneticFieldValues -** type float, taking data from magnetometer sensor
- **Lat , Long** – type double, taking data from the GPS
- **CurrentDegree** – type float, storing the value of current degrees by calculating it from the data collected from the two sensors
- **mFusedLocationClient**- type FusedLocationProviderClient, a google API client to calculate the location services

Key functions used in the application are

- **onCreate(BundlesavedInstanceState)** - Overriding this function to initialize or set all the text view, colour to the text, etc here.
  The other important thing here is calling initialize() function here.
- **Initialize()** – As the name suggests it initializes both the sensors.
  It also checks and requests permission for the location services. If the location permission is given then it will calculate the user co-ordinates using the Google API and stores it into the two location variables that are being displayed inside initialize function using latLong textview.
  Here the data of long variable is trim to two decimal places using String.*format*("%.2f",Long).
- **onResume()** – registering both the sensors
- **onPause()** – unregistering both the sensors to save the battery life
- **onSensorChanged()** – this is the second most important function after onCreate().
  It performs the calculations on the changing values of both the sensors. It is storing the values of the both the sensors to their variables. These values are then stored in a martrix R from where current degree is calculated using
  float degre = (float) Math.*toDegrees*(newvalue[0]);
  degre = (degre + 360)% 360;
  After calculating the degrees its calling a self created function calculateOrientation().
- **calculateOrientation()** – it has a parameter of float degree and its computing the calculations of East, West, North, South, NE, NW, SE, SW by using this logic-
  if (degree > = 22 && degree <= 67) then print NorthEast
  if (degree > = 68 && degree <= 111) then print East
  if (degree > = 112 && degree <= 156) then print SouthEast
  if (degree > = 157 && degree <= 202) then print South
  if (degree > = 203 && degree <= 247) then print SouthWest
  if (degree > = 248 && degree <= 292) then print West
  if (degree > = 293 && degree <= 336) then print NorthWest
  else print North

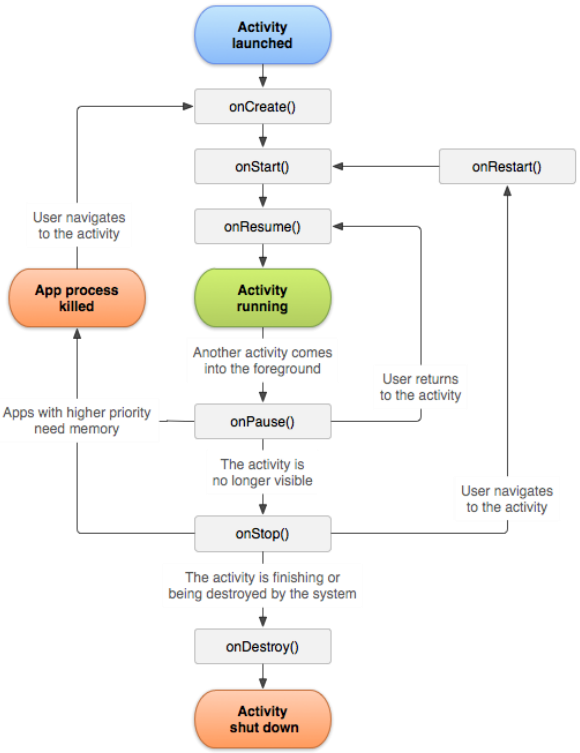

Figure 3: Activity Cycle

By- Rahul Thakur s1776200
The University of Edinburgh

After calculating and displaying the exact degrees its calling the animation function – rotate animation with image.startRotation().

Designing

In this app, the developer has used 4 text views

1) Heading as Compass in Magenta colour.
2) Readings from the app – current degree
3) Location – to tell that the numbers below this are the location
4) Latitude-longitude – current co-ordinates in Grey colour

It has one Image view also which is also shown in figure 1.
The same image is also used by the developer as the app icon for all the mipmap depending upon the resolution.

Tools used:

Android Studio 3.0.1
JRE 1.8.0_152-release-915-b01 amd64
Device used : Lenovo k33a42 with OS – 7
The app can run on devices with OS 4- Ice Cream Sandwich and above and API level 14
For this particular app, its have been tested on OS 7- Nougat with the API level 27.
The minimum SDK level the app can run smoothly is **14** and the target/compiled SDK version is **27**.

Extra features:

Implemented the GPS with a compass to let the user know his coordinates.

Runtime permissions – with 6.0 Android Marshmallow, Google has made it compulsory to tell the users what all special permissions are requesting by the app rather than blindly accepting all the permissions at install time. Now the user is prompted to accept the permissions which are required during the app.

Following are my manifest.xml snippet for requesting the permission.

```
<uses-feature android:name="android.hardware.sensor.accelerometer" android:required="true" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-feature android:name="android.hardware.location.gps" />
```

To reduce the App size –

1) Shrink the image.png.
2) The following is the snippet in the build.gradle that have been used to shrink the app size.

```
buildTypes {
    release {
        minifyEnabled true
        shrinkResources true
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
aaptOptions
        {
            cruncherEnabled = false
        }
```

- minify is an Android tool that helps to decrease the size of the application when we build it. It's a powerful code/statement which means smaller apk files. It detects the unused code and libraries and ignores them while building the project.
- Shrinkresourses also helps in shrinking the resources which your application is not using.
- Cruncher Enabled crunches the images up to 25% and helps is shrinking the app size.

*This app can be integrated with google maps as well as camera application for geotagging in the images, which can be available in the further updates/versions.