Embedded Systems: The Social Compass

Andre Bododea
s1350924

Introduction

A compass application was requested by the customer. The main feature required was that the application includes a visual compass which rotates depending on the direction the phone is facing. It must also display the direction from 0° to 360°. Anything past these two main features is left to the discretion of the engineer. This guide will begin be organized by first explaining the control flow, then it will go on to discuss the main compass feature, and finally the extra features will be discussed.

Control Flow

The user is presented with a start when he first opens up the application (**fig 1**). This gives him the option to choose either the compass feature or the camera feature. If he chooses the compass feature (**fig 2**), he has access to the rotating compass and the direction in degrees relative to 0° (North).

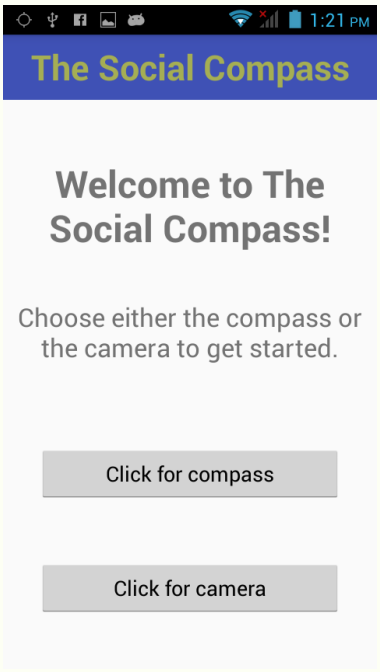

Figure 1: Start screen

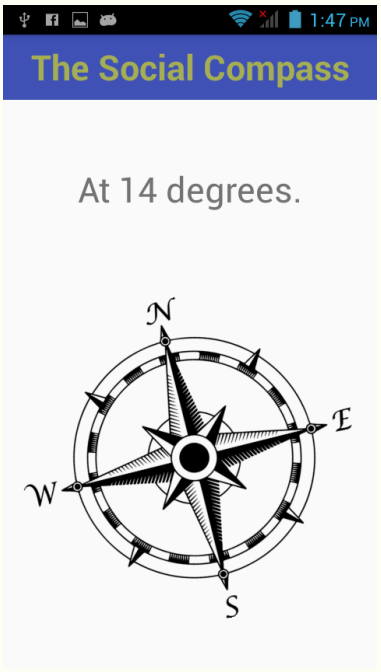

Figure 2: Compass screen

Figure 3: Camera intro

If the user chooses the camera (**fig 3**), he is presented with instructions. The user can then click the camera button and proceed to the camera app (**fig 4**). Once the user takes a photo, the photo is then saved and the user is presented with the option to share the photo they just took (**fig 5**). The user may then choose to share the photo, which is now tagged with their latitude, longitude, and current direction (**fig 6**). Sharing via Facebook, text, email, and any other installed app is supported.

Compass Feature

CompassActivity

The sensors used in the compass (**fig 2**) are the magnetometer and accelerometers. They are implemented via two listeners, one for each. Any time the values are changed, the onSensorChanged() function is called and the sensor values are passed to arrays. These values are then passed to the calculateOrientation() method which was designed by me. The built-in SensorManager.getRotationMatrix() function is used with the accelerometer and magnetometer values, then the SensorManager.getOrientation() function to convert to degrees between -180 and $+180$. That value is then transformed to a range between 0 and 360 to calculate the degrees respective to 0 (magnetic North).

The visual compass is an image that is rotated via the RotateAnimation() function. This simply takes an image and rotates it based on the number of degrees calculated as discussed previously. In order to prevent erratic and shaky movements, a low-pass filter is applied in order to smooth the motion of the image. The function is written as such: $\texttt{magnetic\_field\_x} = \alpha \times \texttt{previous\_magnetic\_field\_x} + (1-\alpha) \times \texttt{current\_magnetic\_field\_x}$ where $\alpha = 0.7$ for this application.

Extra Features

CameraActivity

The extra feature implemented is a camera with tagging and sharing features. An onClickListener() is used to wait for the button press in (**fig 3**). When this method is called, it starts a camera intent (**fig 4**), and creates a URI where the produced photo is saved. Immediately thereafter, a message is displayed that acknowledges the success of the photo taking. Then the custom function overlayPhoto() is called, which transforms the photo at the given URI into a bitmap.

It then puts that bitmap into a Canvas object and uses the Paint object to write the coordinates and the direction over the bitmap. This bitmap is then saved as a JPEG file at the same URI where it was read from and shared via startActivity(Intent.createChooser(shareIntent, "Share images to..")); (**fig 5**). The user can choose an app to share with and share the tagged photo. (**fig 6**)

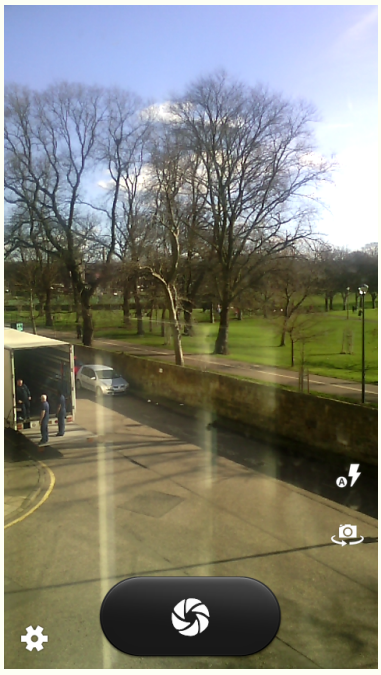

Figure 4: Photo capture

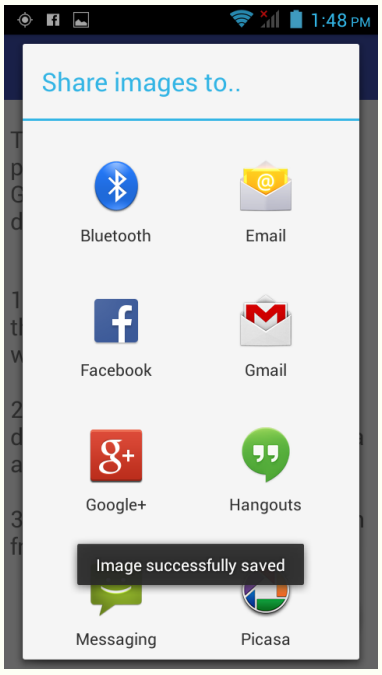

Figure 5: Share photo

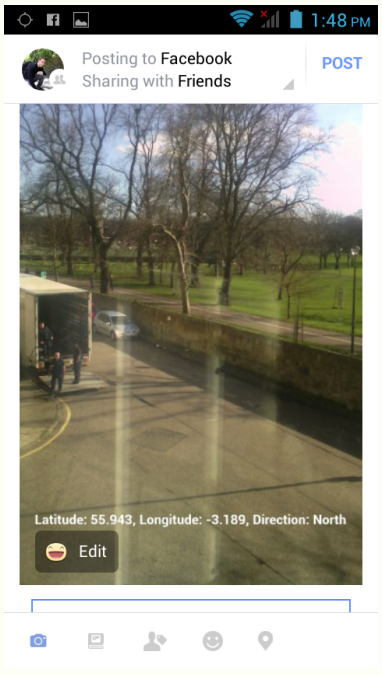

Figure 6: Facebook sharing