Algorithms

Get the sighting’s location

The location is needed for every sighting, at the creation of the sighting entry dialog fragment the following code is used:

**public** **void** onCreateGetLocation(**DialogFragment** dialog) {

**LocationManager** locationManager **=** (**LocationManager**)

getSystemService(**Context.LOCATION\_SERVICE**);

**Criteria** criteria **=** **new** **Criteria**();

**String** provider **=** locationManager**.**getBestProvider(criteria, **false**);

**Location** location **=** locationManager**.**getLastKnownLocation(provider);

**if** (location **!=** **null**) {

onLocationChanged(location);

**Toast.**makeText(this, "LAT: " **+** locLat **+** " LNG: " **+** locLng,

**Toast.LENGTH\_SHORT**)**.**show();

} **else** {

**Toast.**makeText(this, "Cant find location using " **+** provider,

**Toast.LENGTH\_SHORT**)**.**show();

}

}

A location manager is invoked from the Android system services. The location manager is asked for the best provider available, this can be GPS or network. The provider is saved in a string and used by the location manager to get the last known location.

**public** **void** onLocationChanged(**Location** location) {

locLat **=** location**.**getLatitude();

locLng **=** location**.**getLongitude();

}

If getting the location was successful its longitude and latitude values will be saved in the main activity for later use.

If the location cannot be accessed a toast is shown as feedback.

Opening a modal window

A number of modal windows is used in this application. They are created in the main activity. A listener is used to communicate between the window and its parent activity. The code for the creation of the sighting entry dialog fragment is shown below:

**public** **Dialog** onCreateDialog(**Bundle** savedInstanceState) {

**LayoutInflater** inflater **=** getActivity()**.**getLayoutInflater();

**View** view **=** inflater**.**inflate(**R.**layout**.**fragment\_sighting\_entry, **null**);

**AlertDialog.Builder** builder **=** **new** **AlertDialog**.**Builder**(getActivity());

builder

.setView(view)

.setTitle(**R.**string**.**sighting\_entry\_dialog)

.setPositiveButton(**R.**string**.**save,

**new DialogInterface**.**OnClickListener**() {

**@Override**

**public** **void** **onClick**(**DialogInterface** *dialog*, **int** *id*) {

fragmentListener

**.**onSightingEntryPositiveClick(**SightingEntryFragment.**this);

}

})

.setNeutralButton(**R.**string**.**cancel,

**new** **DialogInterface**.**OnClickListener**() {

**@Override**

**public** **void** **onClick**(**DialogInterface** *dialog*, **int** *id*) {

fragmentListener

**.**onSightingEntryNeutralClick(**SightingEntryFragment.**this);

}

});

fragmentListener**.**onCreateGetLocation(**SightingEntryFragment.**this);

fragmentListener**.**onCreateSightingSearch(view);

**ImageButton** specimenPictureButton **=** (**ImageButton**) view

**.**findViewById(**R.**id**.**captureSpecimenImage);

specimenPictureButton**.**setOnClickListener(**new** **OnClickListener**() {

**@Override**

**public** **void** **onClick**(**View** *view*) {

fragmentListener

**.**onCameraSpeciesClick(**SightingEntryFragment.**this);

}

});

**ImageButton** locationPictureButton **=** (**ImageButton**) view

**.**findViewById(**R.**id**.**captureLocationImage);

locationPictureButton**.**setOnClickListener(**new** **OnClickListener**() {

**@Override**

**public** **void** **onClick**(**View** *view*) {

fragmentListener

**.**onCameraLocationClick(**SightingEntryFragment.**this);

}

});

**return** builder**.**create();

}

The creation starts by making a view using a layout inflator with a provided XML layout. Than an alert dialog builder is used to build the dialog fragment and set some parameters, including its design (the view), the title and two buttons.

The buttons use a listener implemented by the parent activity to run methods when these are pressed. These methods are given the current dialog fragment, using this the parent activity can use find view by id to access the dialog’s elements.

The method onCreateSightingSearch(View view) requires to be given a view, as the autocomplete text field cannot be found using the dialog fragment.

On click listeners are used for the buttons to take photos. These also require a view to find the actual button on the modal window.

Adapter for the autocomplete text fields

The reserve and sighting modal windows contain text fields that autocomplete the reserve and species names respectively. The autocomplete text fields need an adapter that provides the layout and the data. The following code is used for the reserve name autocomplete text field:

**public** **void** onCreateReserveSearch(**View** view) {

**Resources** r **=** getResources();

**String** [] list **=** r**.**getStringArray(**R.**array**.**reserves\_names);

**ArrayAdapter<String>** adapter **=** **new** **ArrayAdapter<String>**(this,

**android.R.**layout**.**simple\_list\_item\_1, list);

**AutoCompleteTextView** textView **=** (**AutoCompleteTextView**)

view**.**findViewById(**R.**id**.**reserveName);

textView**.**setThreshold(1);

textView**.**setAdapter(adapter);

}

An array of strings is created with the resources stored in the array file of the application. A standard adapter is created for strings using a standard layout and the array of strings as data.

The autocomplete text view is located by its id and the created adapter is set as its adapter.