Project 1 Team 1 Tutorial

Using a pop-up box to select between available Bluetooth devices

Design Goals for User Interfaces:

In general, user interfaces should aim to be as simple as possible, easy to navigate, and intuitive to use. They should attempt to maximize efficiency, e.g., minimizing the number of steps required to perform desired tasks. These steps could be any type of user input--for example, a swipe, voice command, typed text entry, or physical button press.

In our project, one of the interface challenges was how to choose between available Bluetooth devices when taking a sensor reading. In order to maximize the efficiency and ease of use, we first had to identify and analyze the different use cases. In the scenario when only one Bluetooth device is in range, a single button press connects the Android device to the sensor, and a second press takes the reading. However, in the case when multiple devices are in range, we chose to implement a pop-up box which listed all available Bluetooth devices and allowed the user to select the desired sensor. On selection, the device was automatically connected to the Android device. This feature was attached to the same button as in the first case. So, depending on available devices, the button chose the most efficient manner in which to behave.

This feature can be used whenever there are multiple cases associated with a single action, or when the user is required to choose from several different options. It provides a convenient way to keep the main UI free of clutter from excessive elements while also lending usability and visual appeal.

Why not voice control?

We use button instead of voice control due to three reasons. Firstly, we collect the sensor data outside room, where the background noise is big. And the voice control is not stable enough to identify user's voice with high background noise. Secondly, to use the voice control, we need push a button and then speak out the command. This is even more complicated than only push the button for command. Lastly, using voice control need connection to the Internet. In our case, it is not necessary to use voice control that requires the Internet connection.

Implementation:

Programmatic Steps for Starting a New Activity

- Create an intent to be used with startActivityForResult().
- Put pertinent information to be sent to the second activity into the intent as extras.
- Start activity using startActivityForResult() with an arbitrary, user-defined request code.
- Perform necessary processing in the new activity
- In the new activity, create an intent to return to the main activity, including any return data as extras in the intent.
- Call finish() to return to the main activity.
- Handle the return intent in onActivityResult() in the main activity, and retrieve the extras.

Using ListView for User Input

A separate XML layout must be created for the second activity. At minimum, this must include

the ListView widget (named myListViewWidget in the example below) to be used to get the user input. To connect this second MyInputActivity with the MainActivity, the manifest must be updated to include MyInputActivity. In Eclipse, this may be accomplished by opening the AndroidManifest.xml, navigating to “Application” tab, then selecting “Add” under “Application Nodes”. Specify the type of addition as an activity, and enter the activity name.

An adapter must be created in order to specify how the ListView will be filled with data. In the example, we use an array of strings with the simple_list_item_1 format for a simple display consisting of one string array element per line. The OnItemClickListener waits for the user click event for the associated ListView widget, and returns information regarding the position of the clicked item, e.g., clicking the topmost item in the list returns position equal to zero, while also returning an ID associated with the clicked row.

Main Activity

```
final static int PICK_ADDRESS_REQUEST = 1; //arbitrary integer
                void StartMyInputActivity(){
                                    Intent intent = new Intent(this, MyInputActivity.class);
               intent.putExtra("deviceList", deviceNames);
               startActivityForResult(intent, PICK_ITEM_REQUEST);
                }
@Override
       protected void onActivityResult(int requestCode, int resultCode, Intent data) {
              if (requestCode==PICK_ITEM_REQUEST & resultCode==RESULT_OK)
              {
                     /*
                Get return data from data.getExtra() or its variants, e.g.,
                //name of extra and default value if unavailable
                int index = data.getIntExtra(“chosenIndex”, 0);
                  Use the returned data
                  */
              }
       }
```

MyInputActivity

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
         ListView myListView = (ListView)findViewById(R.id.myListViewWidget);
         Intent intent = getIntent();
         String choiceList[] = intent.getStringArrayExtra("choiceList");
         myListView.setAdapter(new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, choiceList));
               // listening to single list item on click
               myListView.setOnItemClickListener(new OnItemClickListener()
               {
                        public void onItemClick(AdapterView<?> parent, View view, int
position, long id)
                        {
                            // Launching new Activity on selecting single List Item
                            Intent returnIntent = new Intent();
                            // sending data to new activity
                            returnIntent.putExtra("chosenIndex", position);
                            setResult(RESULT_OK, returnIntent);
                            finish();
                        }
               });
```