Android Programming
Lecture 21:
Dialog Boxes

11/30/2011
Interesting City App Issue

• Multiple jobs may be located at the same latitude/longitude

  – How is it handled?
    • By the ListView Activity
    • By the MapView Activity

  – Is it handled correctly/appropriately?
Interesting City App Issue

• A possible solution: Instead of mapping WorkOrder entries to Overlay entries,
  – Scan the parsed WorkOrders to determine all WorkOrderLocations (will be <= number of WorkOrders)
  – A WorkOrderLocation is associated with a latitude/longitude
  – A WorkOrderLocation is associated with (a container for) multiple WorkOrders at that location
    • How do you deal with “slightly-off” coordinates?
  – Build Overlays for each WorkOrderLocation
  – DetailView for a WorkOrderLocation should 1st be a ListView of WorkOrders at that location, which then allows selecting a single detail entry
Dialog Boxes

- Commonly used for providing immediate information to the user
- Oftentimes require acknowledgement or feedback
Dialog Boxes: Android

• In the Android system:
  – A dialog box is an Activity
  – Partially transparent, obscuring Activity behind
  – Floating

• Three types in Android
  – Dialog class
    • And its extensions (to support specific useful functionality)
  – Dialog themed Activities
  – Toasts
    • Already covered in class
Dialog Class: Key Subclasses

• A standard Dialog has a title and an arbitrarily defined content view (specified in XML)

• Subclasses include:
  – AlertDialog
    • Suggested as most appropriate to use for most scenarios
    • Text message + one to three feedback buttons
    • Checkbox/radiobutton
    • Arbitrary input (text)
  – {Date/Time}PickerDialog
  – ProgressDialog
    • Message and progress bar
Interacting With Your Activity

- Your Activity and the Dialog box need to work together
- Activity supports directly:
  - `onCreateDialog` function – actual initial creation of dialog
  - `onPrepareDialog` function [optional] – called each time the dialog is about to be shown to allow for real-time management of dialog
  - `showDialog` function – manages the process of showing a dialog (including managing `onCreateDialog` and `onPrepareDialog` for you)
  - `removeDialog` function – completely gets rid of dialog
  - These all take integers (you map integers to dialogs to be shown)
- Should consider implementing:
  - `DialogInterface.onDismissListener`
    - Catches closing of dialog
  - `DialogInterface.onCancelListener`
    - Catches cancel of dialog (a special case of closing)

Much of this is changing with Honeycomb
Using “Fragments” instead
Dialog Boxes

• Building a standard Dialog:
  – Construct the dialog, providing the current activity as a parameter to the constructor
  – Set the title to a string
  – Set the content to a view defined by an XML resource
  – Manipulate any View components defined in the dialog interface through the dialog
  – Show the dialog

See next slide for example
public class DialogBoxExamplesActivity extends Activity implements View.OnClickListener {
    int DIALOG_A = 1;
    int DIALOG_B = 2;
    Button buttonA;
    Button buttonB;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        buttonA = (Button) findViewById(R.id.buttonA);
        buttonB = (Button) findViewById(R.id.buttonB);
        buttonA.setOnClickListener(this);
        buttonB.setOnClickListener(this);
    }

    public void onClick(View arg0) {
        if (arg0.getId() == R.id.buttonA) { showDialog(DIALOG_A); } else if (arg0.getId() == R.id.buttonB) { showDialog(DIALOG_B); }
    }

    public Dialog onCreateDialog(int dialogID) {
        if (dialogID == DIALOG_A) {
            Dialog d = new Dialog(this);
            d.setTitle("This is the title for Dialog A");
            d.setContentView(R.layout.dialog_a_layout);
            return d;
        } else if (dialogID == DIALOG_B) {
            Dialog d = new Dialog(this);
            d.setTitle("This is the title for Dialog B");
            d.setContentView(R.layout.dialog_b_layout);
            return d;
        }
        return null;  // an error if get here - return null dialog
    }
}

dialog_b_layout.xml

This is the title for Dialog B
Dialog Boxes: Dismiss and Cancel

- `dismiss()` and `cancel()` methods can be called on dialog boxes to programmatically close the boxes.
- Dialog state is preserved even after these methods are called so can access information in dialog.
  - They are effectively *cached*.
- If call `removeDialog(int dialogID)`, it is dismissed AND removed from cache.

- Implement `DialogInterface.onDismissListener`:
  One method to implement: `public void onDismiss(DialogInterface dialog)`
  Hook to a dialog with `setOnDismissListener`.

- Implement `DialogInterface.onCancelListener`:
  One method: `public void onCancel(DialogInterface dialog)`
  Triggered by a `cancel()` call or hardware back button.
  Triggered in addition to `onDismiss`.
  Hook to a dialog with `setOnCancelListener`.