Fragments in Android

Making Flexible Multi Pane GUIs to be Responsive to Screen Size

A.R. Kazemi

What is a Fragment?

- Modular GUI components which can be swapped in or out of activities
- Can be seen as sub (nested) activities

- Encapsulate the content of activity panels

- Fragments can be shown in two ways:
  - All fragments together
  - Replace one fragment by another

Building Flexible Responsive GUIs

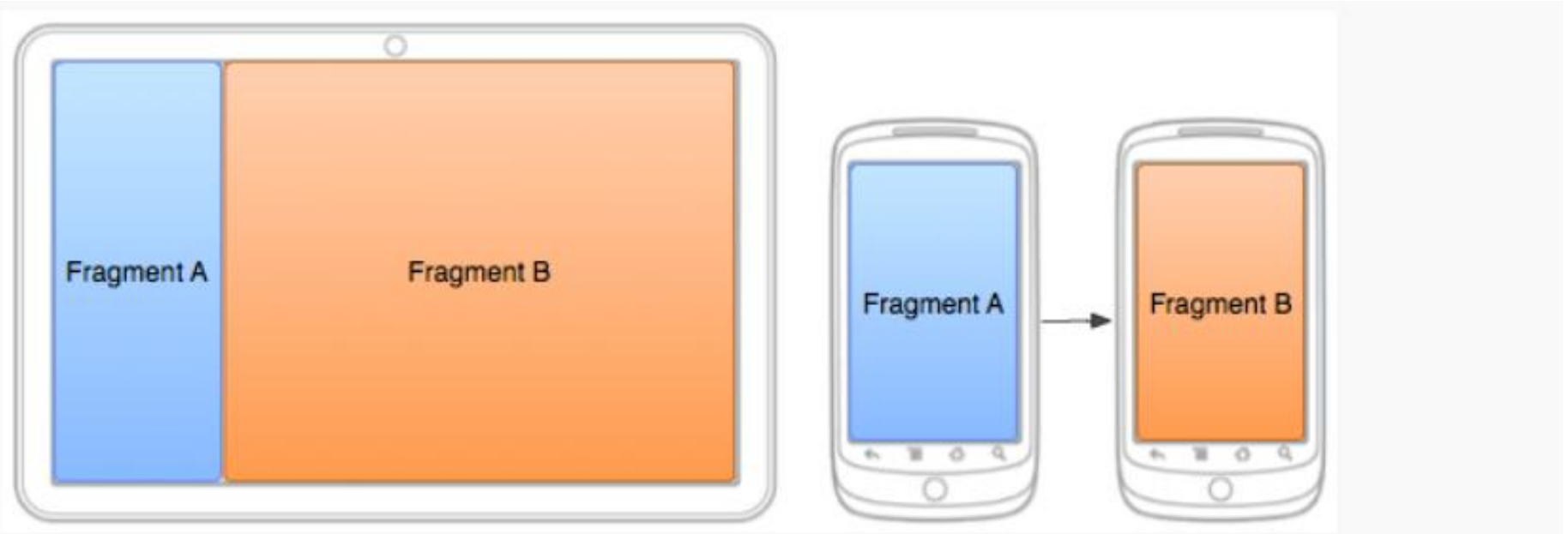

Figure 1. Two fragments, displayed in different configurations for the same activity on different screen sizes. On a large screen, both fragments fit side by side, but on a handset device, only one fragment fits at a time so the fragments must replace each other as the user navigates.

Creating a Fragment?

- Use support lib v4 for compatibility with Android 1.6+,
- Use support lib v7 for compatibility with Android 2.1+
- Use API v11+ framework's built in class for Android 3.0+

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.ViewGroup;

public class ArticleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.article_view, container, false);
    }
}
```

An activity with fragments (Layout)

- Notice the device specific suffix (-large) of the layout folder

res/layout-large/news_articles.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <fragment android:name="com.example.android.fragments.HeadlinesFragment"
              android:id="@+id/headlines_fragment"
              android:layout_weight="1"
              android:layout_width="0dp"
              android:layout_height="match_parent" />

    <fragment android:name="com.example.android.fragments.ArticleFragment"
              android:id="@+id/article_fragment"
              android:layout_weight="2"
              android:layout_width="0dp"
              android:layout_height="match_parent" />

</LinearLayout>
```

Support Libraries (what are them?)

- Libraries which provide new features from higher Android API levels to devices with lower API level/Android version
  - Example new features:
    - fragments

- Currently there are two support libraries:
  - Support lib v4
  - Support lib v7

An activity with fragments (Class)

- Notice: **FragmentActivity** is a special activity class to enable using fragments using support library v4
- When using Fragments, you must use it or one of its sub-classes as base class of your activity

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);
    }
}
```

Add or Remove a Fragment to an Activity at Runtime

- Using **FragmentManager** class allows for dynamic swapping in/out fragments to/from an activity
- For that you should avoid adding fragments in activity layout
  - Consider this empty activity layout:

res/layout/news_articles.xml:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

- Use **FragmentManager** to create a FragmentTransaction which provides api to add/remove/replace and other fragment transactions

Adding a fragment

- Consider the two layouts shown in previous slides
- If the empty layout is loaded, add a fragment dynamically
- If we are recovering from a previous state do not add again
- Create a fragment object and add it using transactions via FragmentManager

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);

        // Check that the activity is using the layout version with
        // the fragment_container FrameLayout
        if (findViewById(R.id.fragment_container) != null) {

            // However, if we're being restored from a previous state,
            // then we don't need to do anything and should return or else
            // we could end up with overlapping fragments.
            if (savedInstanceState != null) {
                return;
            }

            // Create a new Fragment to be placed in the activity layout
            HeadlinesFragment firstFragment = new HeadlinesFragment();

            // In case this activity was started with special instructions from an
            // Intent, pass the Intent's extras to the fragment as arguments
            firstFragment.setArguments(getIntent().getExtras());

            // Add the fragment to the 'fragment_container' FrameLayout
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.fragment_container, firstFragment).commit();
        }
    }
}
```

Replace a Fragment with Another

```
// Create fragment and give it an argument specifying the article it should show
ArticleFragment newFragment = new ArticleFragment();
Bundle args = new Bundle();
args.putInt(ArticleFragment.ARG_POSITION, position);
newFragment.setArguments(args);

FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack so the user can navigate back
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();
```

The `addToBackStack()` method takes an optional string parameter that specifies a unique name for the transaction. The name isn't needed unless you plan to perform advanced fragment operations using the `FragmentManager.BackStackEntry` APIs.

Communicating with Other Fragments

- Fragments can not directly communicate
- Fragments communicate via their attached activity which has implemented a listening interface of them
  - Define an interface in the fragment
  - Implement it in the attaching (parent) activity
  - Send fragment's messages to parent activity by calling proper methods of the defined interface

Define an Interface in Fragment

```
public class HeadlinesFragment extends ListFragment {
    OnHeadlineSelectedListener mCallback;

    // Container Activity must implement this interface
    public interface OnHeadlineSelectedListener {
        public void onArticleSelected(int position);
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);

        // This makes sure that the container activity has implemented
        // the callback interface. If not, it throws an exception
        try {
            mCallback = (OnHeadlineSelectedListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnHeadlineSelectedListener");
        }
    }

    ...
}
```

Implement it in the attaching (parent) activity

```
public static class MainActivity extends Activity
        implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article
    }
}
```

Send a Message From Fragment to the Parent (attached) Activity

- In the fragment class call a method of the fragment-activity communication interface which is implemented by the activity:

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    // Send the event to the host activity
    mCallback.onArticleSelected(position);
}
```

- This can be done on any event such as on clicking a list item component which is within the fragment

Deliver the Message to other fragment

```
public static class MainActivity extends Activity
        implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article

        ArticleFragment articleFrag = (ArticleFragment)
                getSupportFragmentManager().findFragmentById(R.id.article_fragment);

        if (articleFrag != null) {
            // If article frag is available, we're in two-pane layout...

            // Call a method in the ArticleFragment to update its content
            articleFrag.updateArticleView(position);
        } else {
            // Otherwise, we're in the one-pane layout and must swap frags...

            // Create fragment and give it an argument for the selected article
            ArticleFragment newFragment = new ArticleFragment();
            Bundle args = new Bundle();
            args.putInt(ArticleFragment.ARG_POSITION, position);
            newFragment.setArguments(args);

            FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();

            // Replace whatever is in the fragment_container view with this fragment,
            // and add the transaction to the back stack so the user can navigate back
            transaction.replace(R.id.fragment_container, newFragment);
            transaction.addToBackStack(null);

            // Commit the transaction
            transaction.commit();
        }
    }
}
```