LE/EECS 4443: Mobile User Interfaces (LAB)

Week 9: Advanced Activities & Animations

Shogo Toyonaga[1]

[1]Lassonde School of Engineering
York University

February 9, 2025

Table of Contents

Introduction

By the end of this tutorial, you will be able to...

1. Save & restore custom views or objects
2. Use Intents to handle communication between multiple Activities
3. Implement basic support for the action bar / fragment(s)
4. Implement basic animation support for your views

Saving & Restoring States

- In general, simple Strings & primitive data types can be saved and restored through onSaveInstanceState() $\wedge$ onRestoreInstanceState().
- Custom Views or Objects must implement the Serializable Interface in order to be saved and restored using Bundle.getSerializable() & Bundle.putSerializable()

Note: "The serialization interface has no methods or fields and serves only to identify the semantics of being serializable. "

Controlling Orientation

- The orientation of your application can be forcibly constrained by setting android:screenOrientation to portrait or landscape in your activity's Manifest. In this way, you don't necessarily need to save & restore states by overriding the Android Activity Lifecycle.
- Alternatively, you can set android:configChanges to orientation in the Activity manifest. This will prevent the system from restarting the application across portrait & landscape changes.
- Lastly, you can also set the orientation of your application at run-time in your Controller by using:

  setRequestedOrientation (int requestedOrientation)

Controlling Orientation: Example

Note: “Some configuration changes always cause the activity to restart. You can’t disable them. For example, you can’t disable the dynamic colors change introduced in Android 12L (API level 32).”

```
<activity
    ...
    android:name=".MyActivity"
    android:configChanges="orientation|screenSize|screenLayout|keyboardHidden"
    android:label="@string/app_name">
    ...
```

“The following manifest code disables Activity recreation for MyActivity when the screen orientation and keyboard availability changes.”

Activity-to-Activity Transitions I

- Recall: "An Intent is a messaging object you can use to request an action from another app component."
- We can use an Intent with:
  1. startActivity($\cdots$)
  2. startActivityForResult($\cdots$) $\wedge$ onActivityResult($\cdots$).
     - If the requestCode ≥ 0 in startActivityForResult, this code will be returned when the activity exits.
     - onActivityResult($\cdots$) is used to perform actions after the activity you launched is terminated, returning data.
     - The resultCode is used to determine whether the app crashed or didn't return any data.
     - The data is what we obtain from the Activity that we launched. We can use it in many ways such as loading an image selected from a gallery.

Activity-to-Activity Transitions II

- You can use setResult(int resultCode, Intent data) in the child activity to forcibly set the results and data that you want to return to the parent activity after completing an action.

Case Study: Demo_GeoQuiz I

QuizActivity.java

```
    @Override
    public void onClick(View v) {
        if (v == mTrueButton) {
            checkAnswer(true);
        } else if (v == mFalseButton) {
            checkAnswer(false);
       }
       ...
        } else if (v == mCheatButton) {
            boolean answerIsTrue = mQuestionBank[mCurrentIndex].isAnswerTrue();
            // Create Intent to Start New Activity
            Intent intent = CheatActivity.newIntent(QuizActivity.this,
                 answerIsTrue);
            // Start Activity with code REQUEST_CODE_CHEAT
            startActivityForResult(intent, REQUEST_CODE_CHEAT);
        }
    }
```

Case Study: Demo_GeoQuiz II

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable
         Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode != Activity.RESULT_OK) {
            return;
        }
        // If user cheats, run the code here back in MainActivity.java
        if (requestCode == REQUEST_CODE_CHEAT) {
            if (data == null) {
                return;
            }
            mIsCheater = CheatActivity.wasAnswerShown(data);
        }
    }
```

Case Study: Demo_GeoQuiz III

CheatActivity.java

```
    public static Intent newIntent(Context packageContext, boolean answerIsTrue)
          {
        Intent intent = new Intent(packageContext, CheatActivity.class);
        intent.putExtra(EXTRA_ANSWER_IS_TRUE, answerIsTrue);
        return intent;
    }

    private void setAnswerShownResult(boolean isAnswerShown) {
        Intent data = new Intent();
        data.putExtra(EXTRA_ANSWER_SHOWN, isAnswerShown);
        setResult(RESULT_OK, data);
    }

    public static boolean wasAnswerShown(Intent result) {
        return result.getBooleanExtra(EXTRA_ANSWER_SHOWN, false);
    }
```

See: Demo_GeoQuiz

Case Study: Demo_Drawables I

```
public class MainActivity extends AppCompatActivity implements View.
     OnClickListener {
    private Button load_image, take_photo, share_image;
    private ImageView imageholder;
    Intent action;
    Uri loaded_image;
    private final int GET_PICTURE = 1;
    private final int TAKE_PICTURE = 2;
    ...
    @Override
    public void onClick(View view) {
        if (view == load_image) {
            action.setType("image/*");
            action.setAction(Intent.ACTION_GET_CONTENT);
            startActivityForResult(Intent.createChooser(action, "Select Photo
                 from Gallery"), GET_PICTURE);
        } else if (view == take_photo) {
            // Take Photo
            action.setAction(Intent.ACTION_GET_CONTENT);
            action = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(action, TAKE_PICTURE);
        }


```

Case Study: Demo_Drawables II

```
        // Share Button
        else {
            action.setAction(Intent.ACTION_SEND);
            action.setType("image/*");
            action.putExtra(Intent.EXTRA_SUBJECT, "Hello World, from LE/
                 EECS4443");
            action.putExtra(Intent.EXTRA_TEXT, "Sharing an image from Shogz");
            action.putExtra(Intent.EXTRA_STREAM, loaded_image);
            startActivity(Intent.createChooser(action, "Share Image"));
        }
    }
    ...
```

Case Study: Demo_Drawables III

```java

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == RESULT_OK && data != null) {
            if (requestCode == GET_PICTURE) {
                try {
                    share_image.setEnabled(true);
                    Uri image = data.getData();
                    loaded_image = data.getData();
                    assert image != null;
                    // Read the data from the image resource (Uri image)
                    InputStream imageStream = getContentResolver().openInputStream(image);
                    // Convert the input stream into a bitmap
                    Bitmap b = BitmapFactory.decodeStream(imageStream);
                    // Convert the Bitmap into a Drawable Resource
                    Drawable drawable = new BitmapDrawable(getResources(), b);
                    // Set the image in ImageView (Auxiliary Method)
                    imageholder.setImageDrawable(drawable);
                    // Redraw the View to ensure the new image is loaded properly
                    imageholder.invalidate();
                    imageStream.close();
```

Case Study: Demo_Drawables IV

```

                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            } else if (requestCode == TAKE_PICTURE) {
                share_image.setEnabled(true);
                Bitmap photo = (Bitmap) Objects.requireNonNull(data.getExtras())
                        .get("data");
                Drawable drawable = new BitmapDrawable(getResources(), photo);
                imageholder.setImageDrawable(drawable);
                imageholder.invalidate();
            }
        }
    }
```

See: Demo_Drawables

Action Bar: Basic Implementation

1. Design & implement your menu in res $>$ menu $>$ α.xml
2. Override onCreateOptionsMenu(Menu menu)
3. Override onOptionsItemSelected(@NonNull MenuItem item)
4. Profit 8)

Case Study: Demo_Drawables I

Menu.xml

```
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/action_settings"
        android:title="@string/settings"
        app:showAsAction="never"
        />
    <item
        android:id="@+id/action_help"
        android:title="@string/help"
        app:showAsAction="never"
        />
</menu>
```

Case Study: Demo_Drawables II

MainActivity.java

```java
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        // "Listener" for the menu item(s) defined in res > menu
        int selected = item.getItemId();
        if (selected == R.id.action_settings) {
            Toast.makeText(getApplicationContext(), "Settings Activated (Stub)",
                    Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(getApplicationContext(), "Settings Activated (Stub)",
                    Toast.LENGTH_SHORT).show();
        }
        return super.onOptionsItemSelected(item);
    }
```

Case Study: Demo_Drawables III

See: Demo_Drawables

Fragments (A Brief Introduction)

- A Fragment represents a reusable portion of your app's UI.
- A fragment will:
  1. FragmentManager → Define and manage its own layout
  2. Have its own lifecycle
  3. Transactions, Commits → Handle its own input events.
- Fragments can't live on their own. They must be hosted by an activity or another fragment. The fragment's view hierarchy becomes part of, or attaches to, the host's view hierarchy.

Introduction

Definition

“Animations can add visual cues that notify users about what’s going on in your app. They are especially useful when the UI changes state, such as when new content loads or new actions become available. Animations also add a polished look to your app, which gives it a higher quality look and feel.”

https://developer.android.com/develop/ui/views/animations/overview

Introduction

You can implement animations by:

1. Implementing your own behaviour(s) with timers
2. Property Animation | Overview
   - Duration (Default $\rightarrow$ 300ms)
   - Time Interpolation
   - Repeat Count & Behaviour
   - Animator Sets (Sequences, Chains)
   - Frame Refresh Delay (Default $\rightarrow$ 10ms)
3. View Animation (XML)

Case Study: Demo_Drawables

In MainActivity.java (onActivityResult(· · ·))

```
...
share_image.setEnabled(true);
Bitmap photo = (Bitmap) Objects.requireNonNull(data.getExtras()).get("data");
Drawable drawable = new BitmapDrawable(getResources(), photo);
imageholder.setImageDrawable(drawable);
imageholder.invalidate();
ObjectAnimator animation = ObjectAnimator.ofFloat(imageholder, "rotation", 0f,
     360f);
animation.setDuration(2000).setRepeatCount(1);
animation.setRepeatMode(ValueAnimator.REVERSE);
animation.start();
...
```

See: Demo_Drawables

Wrap-Up

Some other sources for you to gain hands-on learning...

1. Idently | Animating Widgets with Kotlin
2. Get animated (Android Dev Summit '18)
3. Codeible | ValueAnimator in Minutes
4. NomTek | Animation Library Demos

Conclusion

Conclusion

Thank you for your attention!
Questions?