Using ViewPager for Screen Slides

Screen slides are transitions between one entire screen to another and are common with UIs like setup wizards or slideshows. This lesson shows you how to do screen slides with a `ViewPager` (/reference/android/support/v4/view/ViewPager.html) provided by the support library (/tools/support-library/index.html). `ViewPager` (/reference/android/support/v4/view/ViewPager.html)s can animate screen slides automatically. Here's what a screen slide looks like that transitions from one screen of content to the next:

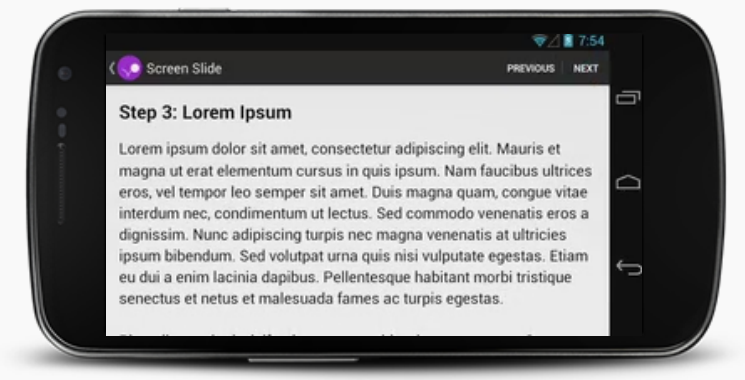
THIS LESSON TEACHES YOU TO

1. Create the Views
2. Create the Fragment
3. Add a ViewPager
4. Customize the Animation with PageTransformer

TRY IT OUT

Download the sample app

Animations.zip



Screen slide animation

Click device screen to replay movie.

If you want to jump ahead and see a full working example, download (/shareables/training/Animations.zip) and run the sample app and select the Screen Slide example. See the following files for the code implementation:

- `src/ScreenSlidePageFragment.java`
- `src/ScreenSlideActivity.java`
- `layout/activity_screen_slide.xml`
- `layout/fragment_screen_slide_page.xml`

Create the Views

Create a layout file that you'll later use for the content of a fragment. The following example contains a text view to display some text:

```
<!-- fragment_screen_slide_page.xml -->
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/content"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent" >

    <TextView style="?android:textAppearanceMedium"
        android:padding="16dp"
        android:lineSpacingMultiplier="1.2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/lorem_ipsum" />
</ScrollView>
```

Define also a string for the contents of the fragment.

Create the Fragment

Create a Fragment (/reference/android/support/v4/app/Fragment.html) class that returns the layout that you just created in the onCreateView() (/reference/android/app/Fragment.html#onCreateView(android.view.LayoutInflater, android.view.ViewGroup, android.os.Bundle)) method. You can then create instances of this fragment in the parent activity whenever you need a new page to display to the user:

```
import android.support.v4.app.Fragment;
...
public class ScreenSlidePageFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        ViewGroup rootView = (ViewGroup) inflater.inflate(
                R.layout.fragment_screen_slide_page, container, false);

        return rootView;
    }
}
```

Add a ViewPager

ViewPager (/reference/android/support/v4/view/ViewPager.html)s have built-in swipe gestures to transition through pages, and they display screen slide animations by default, so you don't need to create any. ViewPager (/reference/android/support/v4/view/ViewPager.html)s use PagerAdapter (/reference/android/support/v4/view/PagerAdapter.html)s as a supply for new pages to display, so the PagerAdapter (/reference/android/support/v4/view/PagerAdapter.html) will use the fragment class that you created earlier.

To begin, create a layout that contains a ViewPager (/reference/android/support/v4/view/ViewPager.html):

```
<!-- activity_screen_slide.xml -->
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Create an activity that does the following things:

- Sets the content view to be the layout with the ViewPager.
- Creates a class that extends the FragmentStatePagerAdapter abstract class and implements the getItem()

method to supply instances of `ScreenSlidePageFragment` as new pages. The pager adapter also requires that you implement the `getCount()` method, which returns the amount of pages the adapter will create (five in the example).

- Hooks up the `PagerAdapter` to the `ViewPager`.
- Handles the device's back button by moving backwards in the virtual stack of fragments. If the user is already on the first page, go back on the activity back stack.

```
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
...
public class ScreenSlidePagerActivity extends FragmentActivity {
    /**
     * The number of pages (wizard steps) to show in this demo.
     */
    private static final int NUM_PAGES = 5;

    /**
     * The pager widget, which handles animation and allows swiping horizontally to acces
     * and next wizard steps.
     */
    private ViewPager mPager;

    /**
     * The pager adapter, which provides the pages to the view pager widget.
     */
    private PagerAdapter mPagerAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_screen_slide);

        // Instantiate a ViewPager and a PagerAdapter.
        mPager = (ViewPager) findViewById(R.id.pager);
        mPagerAdapter = new ScreenSlidePagerAdapter(getSupportFragmentManager());
        mPager.setAdapter(mPagerAdapter);
    }

    @Override
    public void onBackPressed() {
        if (mPager.getCurrentItem() == 0) {
            // If the user is currently looking at the first step, allow the system to ha
            // Back button. This calls finish() on this activity and pops the back stack.
            super.onBackPressed();
        } else {
            // Otherwise, select the previous step.
            mPager.setCurrentItem(mPager.getCurrentItem() - 1);
        }
    }

    /**
     * A simple pager adapter that represents 5 ScreenSlidePageFragment objects, in
     * sequence.
     */
    private class ScreenSlidePagerAdapter extends FragmentStatePagerAdapter {
        public ScreenSlidePagerAdapter(FragmentManager fm) {
            super(fm);
        }

        @Override
        public Fragment getItem(int position) {
```

```
            return new ScreenSlidePageFragment();
        }

        @Override
        public int getCount() {
            return NUM_PAGES;
        }
    }
}
```

Customize the Animation with PageTransformer

To display a different animation from the default screen slide animation, implement the `ViewPager.PageTransformer` (/reference/android/support/v4/view/ViewPager.PageTransformer.html) interface and supply it to the view pager. The interface exposes a single method, `transformPage()` (/reference/android/support/v4/view/ViewPager.PageTransformer.html#transformPage(android.view.View, float)). At each point in the screen's transition, this method is called once for each visible page (generally there's only one visible page) and for adjacent pages just off the screen. For example, if page three is visible and the user drags towards page four, `transformPage()` (/reference/android/support/v4/view/ViewPager.PageTransformer.html#transformPage(android.view.View, float)) is called for pages two, three, and four at each step of the gesture.

In your implementation of `transformPage()` (/reference/android/support/v4/view/ViewPager.PageTransformer.html#transformPage(android.view.View, float)), you can then create custom slide animations by determining which pages need to be transformed based on the position of the page on the screen, which is obtained from the `position` parameter of the `transformPage()` (/reference/android/support/v4/view/ViewPager.PageTransformer.html#transformPage(android.view.View, float)) method.

The `position` parameter indicates where a given page is located relative to the center of the screen. It is a dynamic property that changes as the user scrolls through the pages. When a page fills the screen, its position value is `0`. When a page is drawn just off the right side of the screen, its position value is `1`. If the user scrolls halfway between pages one and two, page one has a position of -0.5 and page two has a position of 0.5. Based on the position of the pages on the screen, you can create custom slide animations by setting page properties with methods such as `setAlpha()` (/reference/android/view/View.html#setAlpha(float)), `setTranslationX()` (/reference/android/view/View.html#setTranslationX(float)), or `setScaleY()` (/reference/android/view/View.html#setScaleY(float)).

When you have an implementation of a `PageTransformer` (/reference/android/support/v4/view/ViewPager.PageTransformer.html), call `setPageTransformer()` (/reference/android/support/v4/view/ViewPager.html#setPageTransformer(boolean, android.support.v4.view.ViewPager.PageTransformer)) with your implementation to apply your custom animations. For example, if you have a `PageTransformer` (/reference/android/support/v4/view/ViewPager.PageTransformer.html) named `ZoomOutPageTransformer`, you can set your custom animations like this:

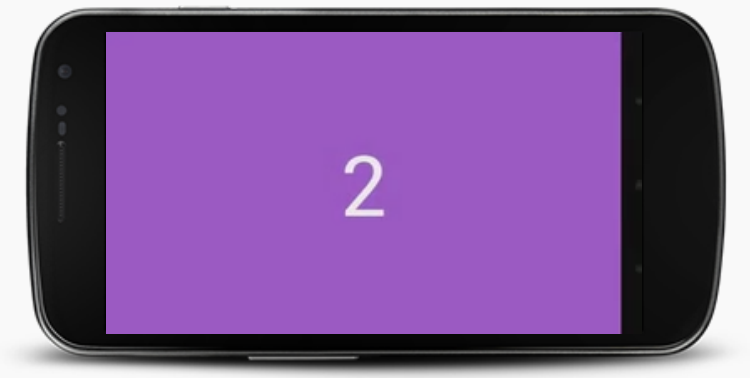
```
ViewPager mPager = (ViewPager) findViewById(R.id.pager);
...
mPager.setPageTransformer(true, new ZoomOutPageTransformer());
```

See the Zoom-out page transformer (#zoom-out) and Depth page transformer (#depth-page) sections for examples and videos of a `PageTransformer` (/reference/android/support/v4/view/ViewPager.PageTransformer.html).

Zoom-out page transformer

This page transformer shrinks and fades pages when scrolling between adjacent pages. As a page gets closer

to the center, it grows back to its normal size and fades in.



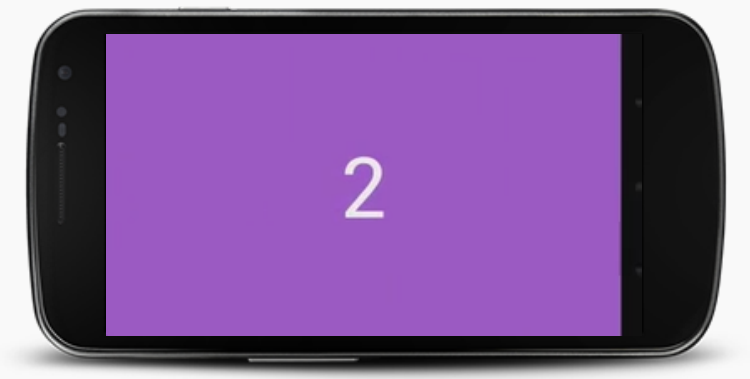
ZoomOutPageTransformer example

Click device screen to replay movie.

```
public class ZoomOutPageTransformer implements ViewPager.PageTransformer {
    private static final float MIN_SCALE = 0.85f;
    private static final float MIN_ALPHA = 0.5f;

    public void transformPage(View view, float position) {
        int pageWidth = view.getWidth();
        int pageHeight = view.getHeight();

        if (position < -1) { // [-Infinity,-1)
            // This page is way off-screen to the left.
            view.setAlpha(0);

        } else if (position <= 1) { // [-1,1]
            // Modify the default slide transition to shrink the page as well
            float scaleFactor = Math.max(MIN_SCALE, 1 - Math.abs(position));
            float vertMargin = pageHeight * (1 - scaleFactor) / 2;
            float horzMargin = pageWidth * (1 - scaleFactor) / 2;
            if (position < 0) {
                view.setTranslationX(horzMargin - vertMargin / 2);
            } else {
                view.setTranslationX(-horzMargin + vertMargin / 2);
            }

            // Scale the page down (between MIN_SCALE and 1)
            view.setScaleX(scaleFactor);
            view.setScaleY(scaleFactor);

            // Fade the page relative to its size.
            view.setAlpha(MIN_ALPHA +
                    (scaleFactor - MIN_SCALE) /
                    (1 - MIN_SCALE) * (1 - MIN_ALPHA));

        } else { // (1,+Infinity]
            // This page is way off-screen to the right.
            view.setAlpha(0);
        }
    }
}
```

Depth page transformer

This page transformer uses the default slide animation for sliding pages to the left, while using a "depth" animation for sliding pages to the right. This depth animation fades the page out, and scales it down linearly.



DepthPageTransformer example

Click device screen to replay movie.

Note: During the depth animation, the default animation (a screen slide) still takes place, so you must counteract the screen slide with a negative X translation. For example:

```
view.setTranslationX(-1 * view.getWidth() * position);
```

The following example shows how to counteract the default screen slide animation in a working page transformer:

```
public class DepthPageTransformer implements ViewPager.PageTransformer {
    private static final float MIN_SCALE = 0.75f;

    public void transformPage(View view, float position) {
        int pageWidth = view.getWidth();

        if (position < -1) { // [-Infinity,-1)
            // This page is way off-screen to the left.
            view.setAlpha(0);

        } else if (position <= 0) { // [-1,0]
            // Use the default slide transition when moving to the left page
            view.setAlpha(1);
            view.setTranslationX(0);
            view.setScaleX(1);
            view.setScaleY(1);

        } else if (position <= 1) { // (0,1]
            // Fade the page out.
            view.setAlpha(1 - position);

            // Counteract the default slide transition
            view.setTranslationX(pageWidth * -position);

            // Scale the page down (between MIN_SCALE and 1)
            float scaleFactor = MIN_SCALE
                    + (1 - MIN_SCALE) * (1 - Math.abs(position));
            view.setScaleX(scaleFactor);
            view.setScaleY(scaleFactor);

        } else { // (1,+Infinity]
```

```
            // This page is way off-screen to the right.
            view.setAlpha(0);
        }
    }
}
```