Использование ViewPager для создания эффекта перелистывания

BY [ADMIN](http://easyandroid.ru/author/admin/) · 26.01.2015

Перелистывание экранов, это переход между объектами, растянутыми на весь экран. Такой способ является общим для интерфейсов, вроде мастеров настройки или слайдшоу. В этом уроке показано, как создать эффект перелистывания используя компонент [ViewPager](http://developer.android.com/reference/android/support/v4/view/ViewPager.html), предоставляемый библиотекой поддержки. ViewPager позволяет создавать такой эффект автоматически. Вот как выглядит этот эффект.

Слайд-анимация. **Нажмите на экран для повторного просмотра.**

Если вы хотите забежать вперед и посмотреть готовый пример, [скачайте демо-проект](http://developer.android.com/shareables/training/Animations.zip), запустите и выберите пример Screen Slide. Смотрите следующие файлы:

* src/ScreenSlidePageFragment.java
* src/ScreenSlideActivity.java
* layout/activity\_screen\_slide.xml
* layout/fragment\_screen\_slide\_page.xml

Создаем объекты

Создаем разметку, которую позже будем использовать во фрагментах. Следующий пример включает в себя текстовое поле:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <!-- 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> |

Не забудьте также про строковый ресурс для данной разметки.

Создаем фрагмент

Создаем расширенный класс [Fragment](http://developer.android.com/reference/android/support/v4/app/Fragment.html), который будет возвращать разметку. Разметку мы создадим в методе [onCreateView()](http://developer.android.com/reference/android/app/Fragment.html" \l "onCreateView(android.view.LayoutInflater,%20android.view.ViewGroup,%20android.os.Bundle)). Вы можете создавать экземпляры данного фрагмента в родительском явлении, всякий раз, когда вам нужно открыть новую страницу и показать ее пользователю:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | 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;      }  } |

Добавляем компонент ViewPager

Компонент [ViewPager](http://developer.android.com/reference/android/support/v4/view/ViewPager.html) содержит встроенные жесты для переключения страниц, которые показывают анимацию перелистывания по умолчанию. Поэтому вам не надо создавать для этого что-либо еще. ViewPager использует адаптер [PageAdapter](http://developer.android.com/reference/android/support/v4/view/PagerAdapter.html) в качестве поставщика новых страниц. В свою очередь PageAdapter использует класс фрагмента, который мы создали ранее.

Сначала создадим разметку, содержащую компонент ViewPager:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <!-- 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" /> |

Создадим явление, которое делает следующее:

* Устанавливает компонент, который будет размещаться в ViewPager.
* Создает наследник абстрактного класса [FragmentStatePagerAdapter](http://developer.android.com/reference/android/support/v13/app/FragmentStatePagerAdapter.html) и реализует метод [getItem()](http://developer.android.com/reference/android/support/v4/app/FragmentStatePagerAdapter.html" \l "getItem(int)) для поставки экземпляров классаScreenSlidePageFragment как новых страниц. Адаптер страниц также требует реализации метода [getCount()](http://developer.android.com/reference/android/support/v4/view/PagerAdapter.html" \l "getCount()), который возвращает количество страниц адаптера, который будет создан (В нашем примере 5).
* Подключает PagerAdapter в ViewPager.
* Обрабатывает аппаратную кнопку “назад”, перемещаясь обратно по виртуальному стеку фрагментов.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61 | import android.support.v4.app.Fragment;  import android.support.v4.app.FragmentManager;  ...  public class ScreenSlidePagerActivity extends FragmentActivity {      /\*\*       \* Количество страниц       \*/      private static final int NUM\_PAGES = 5;        /\*\*       \* Виджет, который делает анимацию и позволяет листать страницы       \*/      private ViewPager mPager;        /\*\*       \* Адаптер, который предоставляет страницы для виджета mPager.       \*/      private PagerAdapter mPagerAdapter;    *@Override*      protected void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_screen\_slide);            // Создаем экземпляры ViewPager и PagerAdapter.          mPager = (ViewPager) findViewById(R.id.pager);          mPagerAdapter = new ScreenSlidePagerAdapter(getSupportFragmentManager());          mPager.setAdapter(mPagerAdapter);      }    *@Override*      public void onBackPressed() {          if (mPager.getCurrentItem() == 0) {              // Если пользователь на первом шаге, разрешаем системе обработку кнопки назад.              // Этот вызовет метод finish() в текущем явлении и вытолкнет стек.              super.onBackPressed();          } else {              // Иначе показываем предыдущий шаг              mPager.setCurrentItem(mPager.getCurrentItem() - 1);          }      }        /\*\*       \* Простой адаптер, предоставляющий 5 объектов ScreenSlidePageFragment       \*/      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;          }      }  } |

Настройка анимации с использованием PageTransformer

Для использования другой анимации, реализуйте интерфейс [ViewPager.PageTransformer](http://developer.android.com/reference/android/support/v4/view/ViewPager.PageTransformer.html) и поставьте его для ViewPager. Интерфейс предоставляет единственный метод [transformPage()](http://developer.android.com/reference/android/support/v4/view/ViewPager.PageTransformer.html" \l "transformPage(android.view.View,%20float)). В каждой точке перехода, этот метод вызывается однократно для каждой видимой страницы (как правило есть только одна видимая страница) и для соседних страниц. Например, если страница 3 видима и пользователь потащил экран к четвертой странице, метод transformPage() вызывается для двух страниц, для третей и четвертой, на каждом шаге жеста.

В вашей реализации метода transformPage(), вы можете создать собственную анимацию, определив, какие страницы должны быть преобразованы, основываясь на положении страниц на экране. Позиция передается в метод transformPage() в виде параметра position.

Параметр position показывает, где данная страница находится по отношению к центру экрана. Это динамическое свойство, которое меняется во время перелистывания страницы. Если страница заполняет экран, ее позиция равна 0. Если страница находится вблизи правой стороны экрана, ее позиция равна 1. Если пользователь на половине пути между двух страниц, первая имеет положение -0.5, а вторая 0.5. Взяв за основу позицию страницы на экране, вы можете создать собственную анимацию, установив свойства страниц с помощью методов [setAlpha()](http://developer.android.com/reference/android/view/View.html" \l "setAlpha(float)), [setTranslationX()](http://developer.android.com/reference/android/view/View.html" \l "setTranslationX(float)) или[setScaleY()](http://developer.android.com/reference/android/view/View.html#setScaleY(float)).

После реализации интерфейса [PageTransformer](http://developer.android.com/reference/android/support/v4/view/ViewPager.PageTransformer.html), вызовите метод [setPageTransformer()](http://developer.android.com/reference/android/support/v4/view/ViewPager.html" \l "setPageTransformer(boolean,%20android.support.v4.view.ViewPager.PageTransformer)), передав в нее объект вашего класса, чтобы применить собственную анимацию. Например, если вы назвали ваш класс ZoomOutPageTransformer, установить анимацию можно таким образом:

|  |  |
| --- | --- |
| 1  2  3 | ViewPager mPager = (ViewPager) findViewById(R.id.pager);  ...  mPager.setPageTransformer(true, new ZoomOutPageTransformer()); |

Трансформер Zoom-out

Этот трансформер сначала сжимает, а затем прячет страницу при прокрутке. Когда страница становится ближе к центру, она растет обратно до нормальных размеров.

Пример использования ZoomOutPageTransformer **Щелкните по экрану, чтобы проиграть видео заново.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | 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)              // Эта страница за экраном слева              view.setAlpha(0);            } else if (position <= 1) { // [-1,1]              // Меняем стандартный переход между страницами              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);              }                // Уменьшаем масштаб (между MIN\_SCALE и 1)              view.setScaleX(scaleFactor);              view.setScaleY(scaleFactor);                // Скрываем страницу, относительно ее размера              view.setAlpha(MIN\_ALPHA +                      (scaleFactor - MIN\_SCALE) /                      (1 - MIN\_SCALE) \* (1 - MIN\_ALPHA));            } else { // (1,+Infinity]              // Эта страница за экраном справа              view.setAlpha(0);          }      }  } |

Трансформер с прозрачностью

Этот трансформер использует стандартную анимацию при прокрутке влево и анимацию с прозрачностью при прокрутке вправо.

Пример DepthPageTransformer **Щелкните по экрану, чтобы проиграть видео заново.**

***Примечание:****во время выполнения анимации с прозрачностью, стандартная анимация перелистывания еще имеет место быть, поэтому вы должны отключить анимацию перелистывания для отрицательного направления X. Например:*

|  |  |
| --- | --- |
| 1 | view.setTranslationX(-1 \* view.getWidth() \* position); |

Следующий пример показывает как противодействовать стандартной анимации перелистывания в трансформере:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | 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)              // Эта страница слева за экраном              view.setAlpha(0);            } else if (position <= 0) { // [-1,0]              // Если двигаемся на левую страницу, используем стандартную анимацию              view.setAlpha(1);              view.setTranslationX(0);              view.setScaleX(1);              view.setScaleY(1);            } else if (position <= 1) { // (0,1]              view.setAlpha(1 - position);                // Противодействуем стандартной анимации              view.setTranslationX(pageWidth \* -position);                // Масштабируем страницу (между MIN\_SCALE и 1)              float scaleFactor = MIN\_SCALE                      + (1 - MIN\_SCALE) \* (1 - Math.abs(position));              view.setScaleX(scaleFactor);              view.setScaleY(scaleFactor);            } else { // (1,+Infinity]              // Эта страница за экраном справа              view.setAlpha(0);          }      }  } |