VŨ THANH TÚ

ĐH CÔNG NGHỆ, ĐHQG HÀ NỘI

Android Animations

HÀ NỘI, 05/ 2012

MỤC LỤC Trang

Tổng quan

Animation (có thể dịch là hoạt ảnh) cho phép một đối tượng trên màn hình có thể thay đổi màu sắc, vị trí, kích thước hay hướng của nó theo thời gian. Animation trong Android rất thiết thực, vui nhộn và đơn giản vì vậy chúng được sử dụng rất thường xuyên.

Android 2.3 và các phiên bản trước đó hỗ trợ ba kiểu animation: **frame-by-frame animation**, **layout animation**, **view animation** (hai kiểu sau thuộc loại tweening animation).

Android 3.0 và các bản sau đó đã tăng cường khả năng Animation trong Android bằng việc giới thiệu khả năng tạo hiệu ứng động cho các thuộc tính của giao diện người dùng (UI).

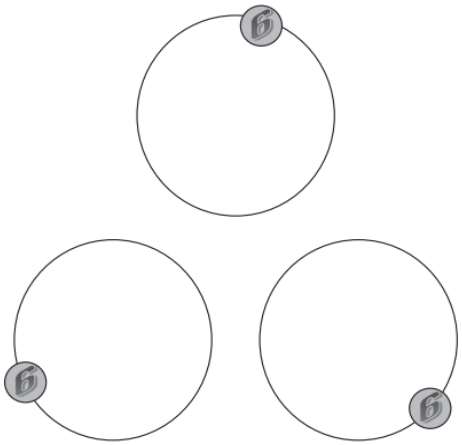
Trong tài liệu này, chúng ta sẽ lần lượt tìm hiểu về ba kiểu animation **frame-by-frame animation**, **layout animation**, **view animation** bằng việc phân tích chúng sâu hơn thông qua các ví dụ**.**

Phần 1. Frame-by-Frame Animation

Frame-by-frame animation là quá trình đơn giản, hiển thị một chuỗi các hình ảnh liên tiếp trong các khoảng thời gian ngắn để tạo ra hiệu ứng cuối cùng là đối tượng di chuyển hoặc thay đổi. Nó cũng giống như hoạt động của các máy chiếu phim vậy.

1.1. Ví dụ về frame-by-frame animation

Hình bên mô tả các vòng tròn có cùng kích cỡ với một quả bóng có màu trên mỗi vòng tròn. Chúng ta có thể tạo ra một chuỗi các vòng tròn như vậy với quả bóng đặt tại các vị trí khác nhau theo chu vi của vòng tròn. Khi bạn sử dụng nhiều khung hình như trên, bạn có thể tạo ra hiệu ứng giống như quả bóng di chuyển xung quanh vòng tròn vậy. Trong phần dưới đây, chúng ta sử dụng tám hình ảnh như vậy và chúng có tên dạng colored-ballN, được đặt trong thư mục /res/drawable.



1.2. Tạo Activity

XML Layout cho ví dụ

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView android:id="@+id/textViewId1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Debug Scratch Pad"
    />


<Button
   android:id="@+id/startFAButtonId"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Start Animation"
/>

<ImageView
        android:id="@+id/imageView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

Activity để load ImageView

```
public class FrameAnimationActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.frame_animations_layout);
    }

}
```

Bạn có thể gọi activity này qua một intent như đoạn mã dưới đây,

```
Intent intent = new Intent(inActivity, FrameAnimationActivity.class);
inActivity.startActivity(intent);
```

1.3. Thêm Animation cho Activity

Trong Android, việc thực hiện frame-by-frame animation được thông qua một class trong gói đồ họa có tên AnimationDrawable. Class Drawable cho phép animation bằng cách yêu cầu container hoặc view của nó gọi một class Runnable cơ bản vẽ lại Drawable bằng cách sử dụng

một tập các tham số khác nhau. Chú ý rằng, bạn không cần biết chi tiết những thể hiện bên trong nếu sử dụng class AnimationDrawable.

Để sử dụng class AnimationDrawable, đầu tiên chúng ta tạo file XML định nghĩa danh sách các frame đặt trong /res/drawable

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
        android:oneshot="false">
    <item android:drawable="@drawable/colored_ball1" android:duration="50" />
    <item android:drawable="@drawable/colored_ball2" android:duration="50" />
    <item android:drawable="@drawable/colored_ball3" android:duration="50" />
    <item android:drawable="@drawable/colored_ball4" android:duration="50" />
    <item android:drawable="@drawable/colored_ball5" android:duration="50" />
    <item android:drawable="@drawable/colored_ball6" android:duration="50" />
    <item android:drawable="@drawable/colored_ball7" android:duration="50" />
    <item android:drawable="@drawable/colored_ball8" android:duration="50" />
    <item android:drawable="@drawable/colored_ball9" android:duration="50" />
 </animation-list>
```

Các tag trong animation-list về cơ bản sẽ được chuyển thành các đối tượng AnimationDrawable đại diện cho tập các hình ảnh. Sau đó, cần thiết lập Drawable này như là background resource cho ImageView,

```
view.setBackgroundResource(Resource.drawable.schemasframe_animation);
```

Sau khi đã thiết lập, bạn có thể truy cập đối tượng AnimationDrawable như sau

```
Object  backgroundObject = view.getBackground();
AnimationDrawable ad = (AnimationDrawable)backgroundObject;
```

Khi đã có đối tượng AnimationDrawable, ta có thể sử dụng hai phương thức start() và stop() để bắt đầu hay kết thúc animation. Hai phương thức quan trọng khác là setOneShot() (chạy animation một lần và sau đó dừng lại) và addFrame() (thêm một frame mới sử dụng một đối tượng Drawable và thiết lập thời gian hiển thị của nó). Đặt chúng cạnh nhau trong một đoạn mã hoàn chỉnh như sau

```
public class FrameAnimationActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.frame_animations_layout);
        this.setupButton();
    }

    private void setupButton()
    {
       Button b = (Button)this.findViewById(R.id.startFAButtonId);
       b.setOnClickListener(
             new Button.OnClickListener(){
                public void onClick(View v)
                {
                   parentButtonClicked(v);
                }
             });
    }
```

```
    private void parentButtonClicked(View v)
    {
        animate();
    }
    private void animate()
    {
         ImageView imgView = (ImageView)findViewById(R.id.imageView);
         imgView.setVisibility(ImageView.VISIBLE);
         imgView.setBackgroundResource(R.drawable.frame_animation);

         AnimationDrawable frameAnimation = (AnimationDrawable) imgView.getBackground();

         if (frameAnimation.isRunning())
         {
         frameAnimation.stop();
         }
         else
         {
         frameAnimation.stop();
         frameAnimation.start();
         }
    }
}//eof-class
```

Ghi chú: Tham khảo chương trình hoàn chỉnh trong thư mục **SampleFrameAnimation** trong file đính kèm.

Phần 2. Layout Animation

Giống như **frame-by-frame, layout animation** cũng khá đơn giản. Giống như tên gọi của nó, layout animation dành riêng cho một số loại view được bố trí một cách cụ thể. Chẳng hạn, bạn có thể sử dụng layout animation với ListView và GridView (hai loại view thường dùng layout để bố cục). Cụ thể, bạn sẽ sử dụng layout animation để thêm các hiệu ứng trực quan cho mỗi item trong một ListView hay GridView được hiển thị.

Trên thực tế, bạn có thể dùng loại animation này cho tất cả những loại được dẫn xuất từ một ViewGroup. Khác với frame-by-frame, layout animation không thông qua các khung hình lặp lại mà thay vào đó nó thay đổi các thuộc tính khác nhau của một view theo thời gian.

Mỗi một view trong Android có một ma trận biến đổi để ánh xạ view tới màn hình. Bằng cách thay đổi ma trận đó theo một số cách bạn có thể thực hiện phép co, phép quay, tịnh tiến,... Ví dụ,

việc thay đổi độ trong suốt của view trong khoảng 0 đến 1, bạn có thể thực hiện phép biến đổi mà ta gọi là alpha animation.

Layout animation thuộc kiểu tweening animation.

2.1. Các kiểu Tweening Animation cơ bản

- *Scale animation* (Co): Làm cho một view nhỏ hơn hoặc lớn hơn dọc theo trục *x* hoặc dọc theo trục *y*. Bạn cũng có thể chỉ định animation diễn ra xung quanh một điểm chốt (*pivot point*).
- *Rotate animation* (Quay): Quay một view quanh một điểm chốt theo một góc quay xác định.
- *Translate animation* (Tịnh tiến): Tịnh tiến một view dọc theo trục *x* hoặc dọc theo trục *y*.
- *Alpha animation* (Alpha): Thay đổi độ trong suốt của một view.

Ghi chú: Tham khảo chương trình minh họa trong thư mục **AndroidAnimButtons** trong file đính kèm.

Bạn có thể định nghĩa những animation trên như một file XML đặt trong /res/anim. Ví dụ,

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@android:anim/accelerate_interpolator">
   <scale
         android:fromXScale="1"
         android:toXScale="1"
         android:fromYScale="0.1"
         android:toYScale="1.0"
         android:duration="500"
         android:pivotX="50%"
         android:pivotY="50%"
         android:startOffset="100" />
</set>
```

Ta có thể thấy, các tham số trong file XML trên đều có "from" và "to" vì chúng ta phải xác định giá trị bắt đầu và kết thúc cho animation.

2.2. Ví dụ về layout animation

Khi có một ListView, bạn có thể thêm animation cho nó để tạo ra các hiệu ứng trực quan cho mỗi item trong ListView. Ví dụ, bạn có thể dùng scale animation để làm một view lớn dần lên từ kích thước 0 (zero) tớ kích thước thật của nó theo trục *y*. Hình ảnh quan sát được giống một dòng text bắt đầu như một đường kẻ ngang và sau đó "to béo hơn" (fatter) để đạt đến kích thước thực tế của nó.

2.3. Tạo Activity và ListView

XML Layout định nghĩa ListView

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

    <ListView
        android:id="@+id/list_view_id"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        />
</LinearLayout>
```

Đoạn mã cho Layout-Animation Activity

```
public class LayoutAnimationActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_layout);
        setupListView();
    }
    private void setupListView()
    {
          String[] listItems = new String[] {
                "Item 1", "Item 2", "Item 3",
                "Item 4", "Item 5", "Item 6",
          };

          ArrayAdapter<String> listItemAdapter =
                   new ArrayAdapter<String>(this
                           ,android.R.layout.simple_list_item_1
                           ,listItems);
          ListView lv = (ListView)this.findViewById(R.id.list_view_id);
          lv.setAdapter(listItemAdapter);
    }
}
```

Bạn có thể gọi activity này qua một intent như đoạn mã dưới đây,

```
Intent intent = new Intent(inActivity, LayoutAnimationActivity.class);
inActivity.startActivity(intent);
```

Như các activity khác, ta cần đăng kí LayoutAnimationActivity trong file AndroidManifest.xml

```
<activity android:name=".LayoutAnimationActivity"
                  android:label="View Animation Test Activity">
        </activity>
```

2.4. Tạo animation cho ListView

Định nghĩa Scale Animation trong một XML File

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@android:anim/accelerate_interpolator">
   <scale
         android:fromXScale="1"
         android:toXScale="1"
         android:fromYScale="0.1"
         android:toYScale="1.0"
         android:duration="500"
         android:pivotX="50%"
         android:pivotY="50%"
         android:startOffset="100" />
</set>
```

Theo trục *x*, độ khuếch đại (magnification) bắt đầu từ 1 và vẫn giữ nguyên là 1, nghĩa là list item không thay đổi theo trục *x*. Theo trục *y*, độ khuếch đại bắt đầu từ 0.1 và tới 1, nghĩa là đối tượng sẽ bắt đầu với kích thước nhỏ gấp 10 lần kích thước gốc của nó và sau đó to dần lên đến kích thước gốc này.

Phép co ở đây thực hiện trong 500 mili giây. Điểm chốt sẽ nằm ở giữa (50%) theo cả hướng *x* và *y*. Giá trị startOffset là số mili giây để đợi trước khi bắt đầu animation.

Định nghĩa Layout-Controller XML File

```
<layoutAnimation xmlns:android="http://schemas.android.com/apk/res/android"
        android:delay="30%"
        android:animationOrder="reverse"
        android:animation="@anim/scale"/>
```

Giả sử tên tập tin này là list_layout_controller.xml. Đây là tập tin trung gian cần thiết. Nó xác định rằng các animation trong list cần phải tiến hành ngược lại (reverse) và animation cho mỗi item sẽ bắt đầu với sự chậm trễ 30% đối với tổng thời gian animation.

Ta cần sửa file XML list_layout để ListView trỏ tới file list_layout_controller.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

    <ListView
        android:id="@+id/list_view_id"
        android:persistentDrawingCache="animation|scrolling"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layoutAnimation="@anim/list_layout_controller" />

</LinearLayout>
```

Các dòng thay đổi được in đậm.

File alpha.xml để test Alpha Animation

```
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
       android:interpolator="@android:anim/accelerate_interpolator"
       android:fromAlpha="0.0" android:toAlpha="1.0" android:duration="1000" />
```

Alpha animation có trách nhiệm kiểm soát sự phai nhạt dần của màu sắc. Trong ví dụ này, ta đang yêu cầu alpha animation để đi từ "không thể nhìn thấy" (invisible color) đến đầy đủ màu sắc (full color) trong 1000 mili giây hay 1 giây. Thời gian cần thiết tối thiểu là 1 giây, nếu không rất khó để nhận ra sự thay đổi này. Mỗi khi muốn thay đổi animation của một item riêng lẻ, ta cần thay đổi file XML trung gian để trỏ tới file animation mới này.

```
<layoutAnimation xmlns:android="http://schemas.android.com/apk/res/android"
        android:delay="30%"
        android:animationOrder="reverse"
        android:animation="@anim/alpha"/>
```

Dòng thay đổi được in đậm.

Tiếp theo ta thử một animation là kết hợp của việc thay đổi vị trí và thay đổi màu gradient.

Kết hợp Translate và Alpha Animation

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@android:anim/accelerate_interpolator">
        <translate android:fromYDelta="-100%" android:toYDelta="0"
android:duration="500" />
        <alpha android:fromAlpha="0.0" android:toAlpha="1.0"
android:duration="500" />
</set>
```

Translate animation sẽ dịch chuyển text từ trên xuống dưới trong không gian hiển thị. Alpha animation sẽ thay đổi màu gradient từ không thể nhìn thấy đến có thể thấy. Thời gian thiết lập là 500 đủ để cho phép người dùng nhận thức được sự thay đổi. Tất nhiên, chúng ta cần thay đổi file layoutAnimation trung gian lần nữa để tham chiếu đến file này. Giả sử file XML kết hợp ở trên có tên là translate_alpha.xml, ta sẽ thay đổi file layoutAnimation trung gian như sau

```
 <layoutAnimation xmlns:android="http://schemas.android.com/apk/res/android"
        android:delay="30%"
        android:animationOrder="reverse"
        android:animation="@anim/translate_alpha"/>
```

Dòng thay đổi được in đậm.

Đoạn mã dưới đây cho chúng ta thấy cách sử dụng rotate animation

```
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
      android:interpolator="@android:anim/accelerate_interpolator"
      android:fromDegrees="0.0"
      android:toDegrees="360"
      android:pivotX="50%"
      android:pivotY="50%"
      android:duration="500" />
```

Đoạn mã trên thể hiện việc quay mỗi text item trong list một vòng xung quanh điểm giữa của mỗi text item. Tiếp theo, bạn cần thay đổi file layout controller XML và file ListView XML layout và sau đó chạy ứng dụng.

Ghi chú: Tham khảo chương trình hoàn chỉnh trong thư mục **SampleLayoutAnimation** trong file đính kèm.

Ngoài ra, còn có một công cụ khác liên quan đến layout animation đó là **interpolator**.

Phần 3. View Animation

View animation là loại phức tạp nhất trong ba loại animation. View animation cho phép ta tạo hiệu ứng động với view tùy ý bằng việc thao tác với ma trận biến đổi.

3.1. Tìm hiểu View Animation

Trong Android, việc hiển thị một view được thông qua một ma trận biến đổi. Trong các ứng dụng đồ họa, bạn sử dụng ma trận biến đổi để chuyển đổi một view theo một cách nào đó. Quá trình này đưa đầu vào là tập các tọa độ điểm ảnh và sự phối màu thành một tập các tọa độ điểm ảnh và sự phối màu mới. Sau quá trình chuyển đổi, bạn sẽ thấy một bức tranh được thay đổi về kích thước, vị trí, hướng hoặc màu sắc. Với một tập các tọa độ đầu vào, chúng ta có thể biểu diễn những phép biến đổi đó thông qua phép nhân ma trận.

Android đưa ra ma trận biến đổi của một view bằng việc cho phép bạn đăng ký một đối tượng animation với view đó.

Chúng ta sẽ xem xét một ví dụ. Chẳng hạn, bạn bắt đầu với một activity, ở đó bạn đặt một ListView với một vài items, tương tự như trong ví dụ "Layout Animation". Chúng ta tạo một nút bấm ở trên cùng màn hình để bắt khi click vào đó ListView animation sẽ bắt đầu.

File XML Layout cho View-Animation Activity

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file is at /res/layout/list_layout.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

<Button
   android:id="@+id/btn_animate"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Start Animation"
/>

    <ListView
        android:id="@+id/list_view_id"
        android:persistentDrawingCache="animation|scrolling"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        />
</LinearLayout>
```

Bây giờ, bạn có thể tạo activity để hiển thị view và thiết lập nút bấm Start Animation.

Đoạn mã cho View-Animation Activity, trước Animation

```
public class ViewAnimationActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_layout);
        setupListView();
        this.setupButton();
    }
    private void setupListView()
    {
          String[] listItems = new String[] {
                "Item 1", "Item 2", "Item 3",
                "Item 4", "Item 5", "Item 6",
          };

          ArrayAdapter<String> listItemAdapter =
                   new ArrayAdapter<String>(this
                           ,android.R.layout.simple_list_item_1
                           ,listItems);
          ListView lv = (ListView)this.findViewById(R.id.list_view_id);
          lv.setAdapter(listItemAdapter);
    }
    private void setupButton()
    {
       Button b = (Button)this.findViewById(R.id.btn_animate);
       b.setOnClickListener(
           new Button.OnClickListener(){
             public void onClick(View v)
             {
                animateListView();
```

```
            }
        });
    }

}
```

Đăng kí với file AndroidManifest.xml

```
<activity android:name=".ViewAnimationActivity"
                android:label="View Animation Test Activity">
        </activity>
```

và sau đó ta thể gọi activity này từ bất kỳ menu item nào qua intent

```
Intent intent = new Intent(this, ViewAnimationActivity.class);
startActivity(intent);
```

3.2. Thêm Animation

Mục đích của chúng ta trong ví dụ này là thêm animation tới ListView, bạn cần một class ViewAnimation nhận được từ android.view.animation.Animation. Sau đó, cần override phương thức applyTransformation để sửa ma trận biến đổi. Khi đã có class ViewAnimation, bạn có thể làm như sau trong classListView

```
ListView lv = (ListView)this.findViewById(R.id.list_view_id);
lv.startAnimation(new ViewAnimation());
```

Đoạn mã cho ViewAnimation class

```
public class ViewAnimation2 extends Animation
{
    public ViewAnimation2(){}
    @Override
    public void initialize(int width, int height,
                                int parentWidth, int parentHeight)
    {
        super.initialize(width, height, parentWidth, parentHeight);
        setDuration(2500);
        setFillAfter(true);
        setInterpolator(new LinearInterpolator());
    }
    @Override
    protected void applyTransformation(float interpolatedTime, Transformation t) {
        final Matrix matrix = t.getMatrix();
        matrix.setScale(interpolatedTime, interpolatedTime);
    }
}
```

Đoạn mã cho View-Animation Activity, bao gồm Animation

```
public class ViewAnimationActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState)
```

```
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_layout);
        setupListView();
        this.setupButton();
    }
    private void setupListView()
    {
          String[] listItems = new String[] {"Item 1", "Item 2", "Item 3", "Item 4", "Item 5", "Item 6",
};

          ArrayAdapter<String> listItemAdapter = new ArrayAdapter<String>(this,
                           android.R.layout.simple_list_item_1, listItems);
          ListView lv = (ListView)this.findViewById(R.id.list_view_id);
          lv.setAdapter(listItemAdapter);
    }
    private void setupButton()
    {
       Button b = (Button)this.findViewById(R.id.btn_animate);
       b.setOnClickListener(
           new Button.OnClickListener(){
             public void onClick(View v)
             {
                animateListView();
             }
           });
    }
    private void animateListView()
    {
          ListView lv = (ListView)this.findViewById(R.id.list_view_id);
          lv.startAnimation(new ViewAnimation2());
    }
}
```

Khi chạy đoạn mã này, bạn sẽ cảm thấy cái gì đó kì lạ. Thay vì lớn dần từ giữa màn hình, ListView lại lớn dần từ góc trên bên trái. Để có được hiệu ứng mong muốn, bạn phải chuyển toàn bộ view sao cho trung tâm của view khớp với trung tâm của animation (trên cùng bên trái). Sau đó, áp dụng ma trận và di chuyển view trở lại trung tâm trước đó. Ta viết lại đoạn mã thực hiện điều này

View Animation sử dụng preTranslate và postTranslate

```
public class ViewAnimation3 extends Animation {
        float centerX, centerY;
    public ViewAnimation3(){}

    @Override
    public void initialize(int width, int height, int parentWidth, int parentHeight) {
        super.initialize(width, height, parentWidth, parentHeight);
        centerX = width/2.0f;
        centerY = height/2.0f;
        setDuration(2500);
        setFillAfter(true);
        setInterpolator(new LinearInterpolator());
    }
    @Override
    protected void applyTransformation(float interpolatedTime, Transformation t) {
        final Matrix matrix = t.getMatrix();
        matrix.setScale(interpolatedTime, interpolatedTime);
        matrix.preTranslate(-centerX, -centerY);
        matrix.postTranslate(centerX, centerY);
    }
}
```

Các phương thức preTranslate và postTranslate sẽ thiết lập một ma trận trước khi scale và sau khi scale. Điều này tương tương với ba ma trận biến đổi. Đoạn mã

```
matrix.setScale(interpolatedTime, interpolatedTime);
matrix.preTranslate(-centerX, -centerY);
matrix.postTranslate(centerX, centerY);
```

tương đương với việc di chuyển tới một trung tâm khác, scale nó và di chuyển nó về trung tâm ban đầu.

Ghi chú: Tham khảo chương trình hoàn chỉnh trong thư mục **SampleViewAnimation** trong file đính kèm.

Tìm hiểu sâu hơn về các thuộc tính, phương thức của đối tượng ma trận tại

http://developer.android.com/reference/android/graphics/Matrix.html

3.3. Sử dụng camera để cảm nhận được độ sâu

Gói đồ họa trong Android còn cung cấp cho ta class Camera cho phép cảm nhận được độ sâu bằng cách chiếu một hình ảnh 2D di chuyển trong không gian 3D vào một mặt 2D. Ví dụ, bạn có thể di chuyển một ListView trở lại từ màn hình bằng việc dịch 1300 pixel theo trục *z* và quay 360 độ quanh trục *y*.

```
public class ViewAnimation extends Animation {
        float centerX, centerY;
    public ViewAnimation(float cx, float cy)
    {
        centerX = cx;
        centerY = cy;
    }

    @Override
    public void initialize(int width, int height, int parentWidth, int parentHeight) {
        super.initialize(width, height, parentWidth, parentHeight);
        setDuration(2500);
        setFillAfter(true);
        setInterpolator(new LinearInterpolator());

    }

    @Override
    protected void applyTransformation(float interpolatedTime, Transformation t) {
        applyTransformationNew(interpolatedTime,t);
    }

    protected void applyTransformationNew(float interpolatedTime, Transformation t)
    {
```

```
        //Log.d("d","transform:" + interpolatedTime);
        final Matrix matrix = t.getMatrix();
        camera.save();
        camera.translate(0.0f, 0.0f, (1300 - 1300.0f * interpolatedTime));
        camera.rotateY(360 * interpolatedTime);
        camera.getMatrix(matrix);

        matrix.preTranslate(-centerX, -centerY);
        matrix.postTranslate(centerX, centerY);
        camera.restore();
    }
}
```

3.4. Lớp AnimationListener

Android sử dụng một listener interface có tên AnimationListener để giám sát các sự kiện animation. Bạn có thể lắng nghe các sự kiện animation bằng việc hiện thực AnimationListener interface và thiết lập hiện thực đối với class Animation.

```
public class ViewAnimationListener
implements Animation.AnimationListener {

    private ViewAnimationListener(){}

    public void onAnimationStart(Animation animation)
    {
        Log.d("Animation Example", "onAnimationStart");
    }
    public void onAnimationEnd(Animation animation)
    {
        Log.d("Animation Example", "onAnimationEnd");
    }
    public void onAnimationRepeat(Animation animation)
    {
        Log.d("Animation Example", "onAnimationRepeat");
    }
}
```

Class ViewAnimationListener chỉ log message. Bạn có thể update phương thức animateListView trong ví dụ về view-animation để đặt animation listener vào account

```
private void animateListView()
    {
          ListView lv = (ListView)this.findViewById(R.id.list_view_id);
          ViewAnimation animation = new ViewAnimation();
          animation.setAnimationListener(new ViewAnimationListener());
          lv.startAnimation(animation);
    }
```

3.5. Một số chú ý về ma trận biến đổi

Như chúng ta đã thấy, ma trận là chìa khóa để biến đổi view và animation. Dưới đây là một số phương thức chủ yếu trên một ma trận:

matrix.reset();

matrix.setScale();

matrix.setTranslate()

matrix.setRotate();

matrix.setSkew();

Phương thức đầu tiên sẽ reset ma trận về một ma trận đơn vị, nó sẽ không gây ra thay đổi khi áp dụng vào view. setScale thực hiện việc thay đổi kích thước ma trận. setTranslate thực hiện việc thay đổi vị trí để mô phỏng chuyển động. setRotate thực hiện việc thay đổi hướng. setSkew thực hiện việc "bóp méo" một view.

Bạn có thể liên kết hoặc nhân các ma trận để tạo nên các hiệu ứng phức tạp hơn.

Ví dụ với m1, m2, m3 là các ma trận:

m1.setScale();

m2.setTranlate()

m3.concat(m1,m2)

Việc chuyển đổi một view bởi m1 để được một view mới, sau đó chuyển đổi view mới đó với m2 thì cũng tương đương với việc chuyển đổi view ban đầu bởi m3.

Một preTranslate như m1.preTranslate(m2) tương đương với

m1.concat(m2,m1)

tương tự, m1.postTranslate(m2) tương đương với

m1.concat(m1,m2).

Mở rộng, đoạn code dưới đây

matrix.setScale(interpolatedTime, interpolatedTime);

matrix.preTranslate(-centerX, -centerY);

matrix.postTranslate(centerX, centerY);

tương đương với

Matrix matrixPreTranslate = new Matrix();

matrixPreTranslate.setTranslate(-centerX, -centerY);

Matrix matrixPostTranslate = new Matrix();

matrixPostTranslate.setTranslate(cetnerX, centerY);

matrix.concat(matrixPreTranslate,matrix);

matrix.postTranslate(matrix,matrixpostTranslate);

Tài liệu tham khảo

[1] Pro Android 3, Satya Komatineni, Dave MacLean , Sayed Y. Hashimi

[2] *http://developer.android.com/reference/android/view/animation/package-summary.html*: You can discover various animation-related APIs here, including interpolators.

[3] *http://developer.android.com/guide/topics/resources/animation-resource.html*: You will find XML tags for various animation types here.

[4] *http://www.hascode.com/2010/09/playing-around-with-the-android-animation-framework/*: Playing around with the Android Animation Framework

[5] *http://android-er.blogspot.com/2012/02/apply-animation-on-button.html*: Apply animation on Button