TP Android: Création d'Animations

Sur Android, il existe plusieurs types d'animations, je vais commencer par vous présenter les animations entre deux activités puis les animations sur des éléments au sein d'une activité.

Animation entre deux activités

Android permet de réaliser différentes animations au moment où l'on passe d'une activité à une autre, voici les différents types d'animation possible:

- Alpha : permet de faire apparaître ou disparaître un élément
- Rotate : permet de faire effectuer une rotation à un élément
- Scale : permet de changer la taille d'un élément
- Translate : permet de faire effectuer une translation à un élément

Dans cette première partie nous utiliserons deux animations de type translate afin de donner à l'utilisateur l'impression que la première activité se trouve haut dessus de la seconde.

Projet Android Studio final et fonctionnel :

Partie 1: Création d'animations lors du changement d'activité

1ère étape : Création de deux fichiers xml correspondant à deux activités distinctes

Chaque vue contient une image et un bouton qui serviront à déclencher les animations, les deux images utilisées dans ce tutoriel sont disponibles dans l'archive contenant ce dossier.

activity_1.xml

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

    <ImageView
        android:id="@+id/img_face"
        android:src="@drawable/img"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        />

    <Button
        android:id="@+id/button_next"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Next"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        />

</RelativeLayout>
```

activity_2.xml

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


    <ImageView
        android:id="@+id/img_rocket"
        android:src="@drawable/rocket"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        />


    <Button
        android:id="@+id/button_back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Back"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        />

</RelativeLayout>
```

Nous allons maintenant créer les deux activités correspondants aux vues xml.
Pour chaque activité, il faut d'abord créer la variable correspondante au bouton contenu dans sa vue xml, et ensuite lui ajouter un click listener qui déclenche une fonction permettant d'accéder à la seconde activité.

Activity_1.java

```
public class Activity1 extends AppCompatActivity {

    private Button button_next = null;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_1);

        button_next = (Button) findViewById(R.id.button_next);
        button_next.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                goToActivity2();
            }
        });
    }

    private void goToActivity2(){
        Intent intent = new Intent(this, Activity2.class);
        startActivity(intent);
        finish();
    }
}
```

Activity_2.java

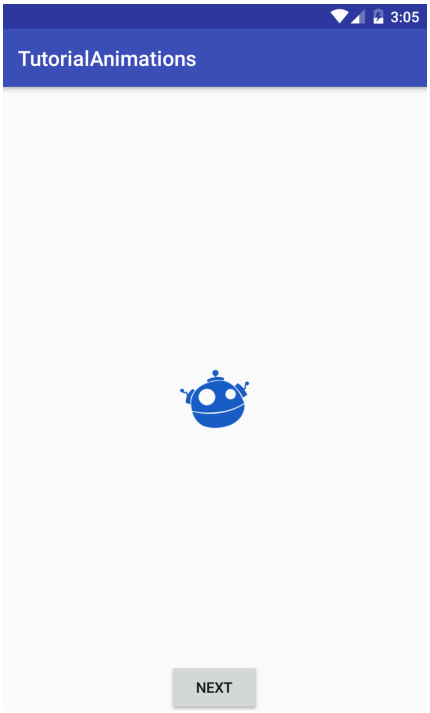
```
public class Activity2 extends AppCompatActivity {

    private Button button_back = null;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_2);

        button_back = (Button) findViewById(R.id.button_back);
        button_back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                goToActivity1();
            }
        });
    }

    private void goToActivity1(){
        Intent intent = new Intent(this, Activity1.class);
        startActivity(intent);
        finish();
    }
}
```

Ajouter les activités dans la balise <application> du manifest afin de pouvoir avoir un aperçu des deux activités :

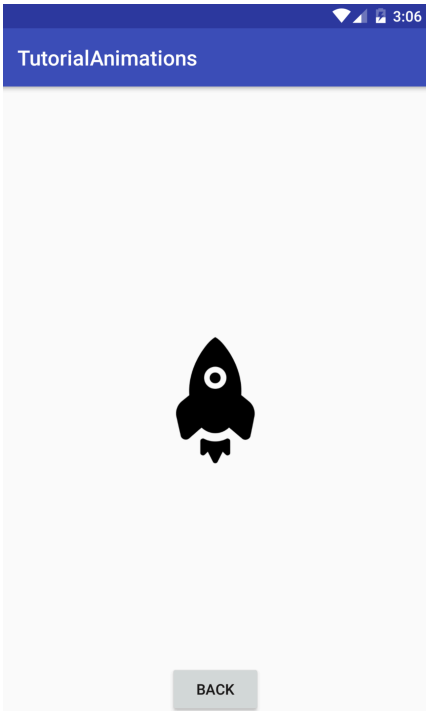
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.florianmalapel.tutorialanimations">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".Activity1">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:name=".Activity2" />

    </application>

</manifest>
```

Exécuter le projet et voici ce que vous devez obtenir :

Activity1.java



Activity2.java



Maintenant nous allons créer les 4 animations nous permettant de créer l'effet souhaité, pour cela vous devez tout d'abord créer le dossier « anim » dans le dossier « res » si celui-ci n'existe pas. Puis créer chacun des fichiers xml suivant dans le dossier « anim » précédemment créé.

bottom_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
     android:interpolator="@android:anim/linear_interpolator">
    <translate
        android:fromYDelta="100%p"
        android:toYDelta="0"
        android:duration="400"/>
</set>
```

Cette animation permet de faire entrer une vue à l'écran en la faisant glisser vers le haut.

bottom_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
     android:interpolator="@android:anim/linear_interpolator">
    <translate
        android:fromYDelta="0"
        android:toYDelta="100%p"
        android:duration="400"/>
</set>
```

Cette animation permet de faire sortir une vue de l'écran en la faisant glisser vers le bas.

top_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
     android:interpolator="@android:anim/linear_interpolator">
    <translate
        android:fromYDelta="-100%p"
        android:toYDelta="0"
        android:duration="400"/>
</set>
```

Cette animation permet de faire entrer une vue à l'écran en la faisant glisser vers le bas.

top_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
     android:interpolator="@android:anim/linear_interpolator">
    <translate
        android:fromYDelta="0"
        android:toYDelta="-100%p"
        android:duration="400"/>
</set>
```

Cette animation permet de faire sortir une vue de l'écran en la faisant glisser vers le haut.

Maintenant nous allons exploiter ces 4 animations.
Pour effectuer une animation lors du changement d'activité, android nous permet d'utiliser la méthode *overridePendingTransition(int animationActiviteEntrante, int animationActiviteSortante)* qui doit toujours suivre la méthode *startActivity(Intent intent)*:

```
startActivity(intent);
overridePendingTransition(R.anim.top_in,R.anim.bottom_out);
```

Intégration des animations dans les deux activités:

```
public class Activity1 extends AppCompatActivity {

    private Button button_next = null;
    private ImageView img_face = null;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_1);

        button_next = (Button) findViewById(R.id.button_next);
        button_next.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                goToActivity2();
            }
        });

        img_face = (ImageView) findViewById(R.id.img_face);
        img_face.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                beginAnimationImage();
            }
        });

    }

    private void goToActivity2(){
        Intent intent = new Intent(this, Activity2.class);
        startActivity(intent);
        overridePendingTransition(R.anim.bottom_in, R.anim.top_out);
    }
}
```

```
public class Activity2 extends AppCompatActivity {

    private Button button_back = null;
    private ImageView img_rocket = null;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_2);

        button_back = (Button) findViewById(R.id.button_back);
        img_rocket = (ImageView) findViewById(R.id.img_rocket);

        button_back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                goToActivity1();
            }
        });

        img_rocket.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                beginAnimationImage();
            }
        });
    }



    private void goToActivity1(){
        Intent intent = new Intent(this, Activity1.class);
        startActivity(intent);
        overridePendingTransition(R.anim.top_in,R.anim.bottom_out);
    }
}
```

Exécuter le programme, et l'animation devrait être présente lors du changement d'activité en cliquant sur les boutons respectifs à chaque activité.

Partie 2: Création d'animations au sein d'une activité

1 - Création d'une animation en utilisant un fichier xml

Créer le fichier xml suivant, permettant d'effectuer une rotation de 180° sur un élément:

rotate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate android:fromDegrees="0"
            android:toDegrees="180"
            android:pivotX="50%"
            android:pivotY="50%"
            android:duration="200"
            android:repeatMode="restart"
            android:repeatCount="infinite"
            />

</set>
```

Ajouter un clic listener sur l'image de l'activité 1 qui déclenche l'animation:

```
public class Activity1 extends AppCompatActivity {

    private Button button_next = null;
    private ImageView img_face = null;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_1);

        button_next = (Button) findViewById(R.id.button_next);
        button_next.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                goToActivity2();
            }
        });

        img_face = (ImageView) findViewById(R.id.img_face);
        img_face.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                beginAnimationImage();
            }
        });
    }

    private void goToActivity2(){
        Intent intent = new Intent(this, Activity2.class);
        startActivity(intent);
        overridePendingTransition(R.anim.bottom_in, R.anim.top_out);
    }

    private void beginAnimationImage(){
        Animation rotateAnim = AnimationUtils.loadAnimation(this,
R.anim.rotate);
        rotateAnim(2000);
        img_face.startAnimation(rotateAnim);
    }
}
```

Exécutez et testez.

2 - Création d'une animation en utilisant l'objet Animation

Ajouter à l'image de l'activité 2 un clic listener qui déclenche l'animation:

```
public class Activity2 extends AppCompatActivity {

    private Button button_back = null;
    private ImageView img_rocket = null;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_2);

        button_back = (Button) findViewById(R.id.button_back);
        img_rocket = (ImageView) findViewById(R.id.img_rocket);

        button_back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                goToActivity1();
            }
        });

        img_rocket.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                beginAnimationImage();
            }
        });
    }



    private void goToActivity1(){
        Intent intent = new Intent(this, Activity1.class);
        startActivity(intent);
        overridePendingTransition(R.anim.top_in,R.anim.bottom_out);
    }


    private void beginAnimationImage(){
        RotateAnimation rotateAnimation = new RotateAnimation(0, 540,
Animation.RELATIVE_TO_SELF,
                0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
        rotateAnimation.setDuration(2000);
        rotateAnimation.setFillAfter(true);
        img_rocket.startAnimation(rotateAnimation);
    }
}
```

Exécutez et testez.

Compléments

zoom_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
     android:fillAfter="true" >

    <scale
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:duration="1000"
        android:fromXScale="1"
        android:fromYScale="1"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toXScale="3"
        android:toYScale="3" >
    </scale>
</set>
```

zoom_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
     android:fillAfter="true" >

    <scale
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:duration="1000"
        android:fromXScale="1.0"
        android:fromYScale="1.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toXScale="0.5"
        android:toYScale="0.5" >
    </scale>
</set>
```

blink.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha android:fromAlpha="0.0"
           android:toAlpha="1.0"
           android:interpolator="@android:anim/accelerate_interpolator"
           android:duration="600"
           android:repeatMode="reverse"
           android:repeatCount="infinite"/>
</set>
```

fade_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
     android:fillAfter="true" >

    <alpha
        android:duration="1000"
        android:fromAlpha="0.0"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:toAlpha="1.0" />
</set>
```

fade_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
     android:fillAfter="true" >

    <alpha
        android:duration="1000"
        android:fromAlpha="1.0"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:toAlpha="0.0" />

</set>
```

Ajouter un listerner à un objet de type Animation:

Il est possible d'ajouter à une animation un listener permettant d'effectuer des actions spécifiques au moment où l'animation démarre, se termine, ou se répète.

```
Animation rotateAnim = AnimationUtils.loadAnimation(this, R.anim.rotate);
rotateAnim.setDuration(2000);
rotateAnim.setAnimationListener(new Animation.AnimationListener() {
     @Override
     public void onAnimationStart(Animation animation) {

     }

     @Override
     public void onAnimationEnd(Animation animation) {

     }

     @Override
     public void onAnimationRepeat(Animation animation) {

     }
 });
```