Android UI 进阶

赵昀 | 字节跳动 Android工程师
zhaoyun.1224@bytedance.com

ByteDance 字节跳动

Overview

- Fragment
- Animation
- Exercise

Fragment

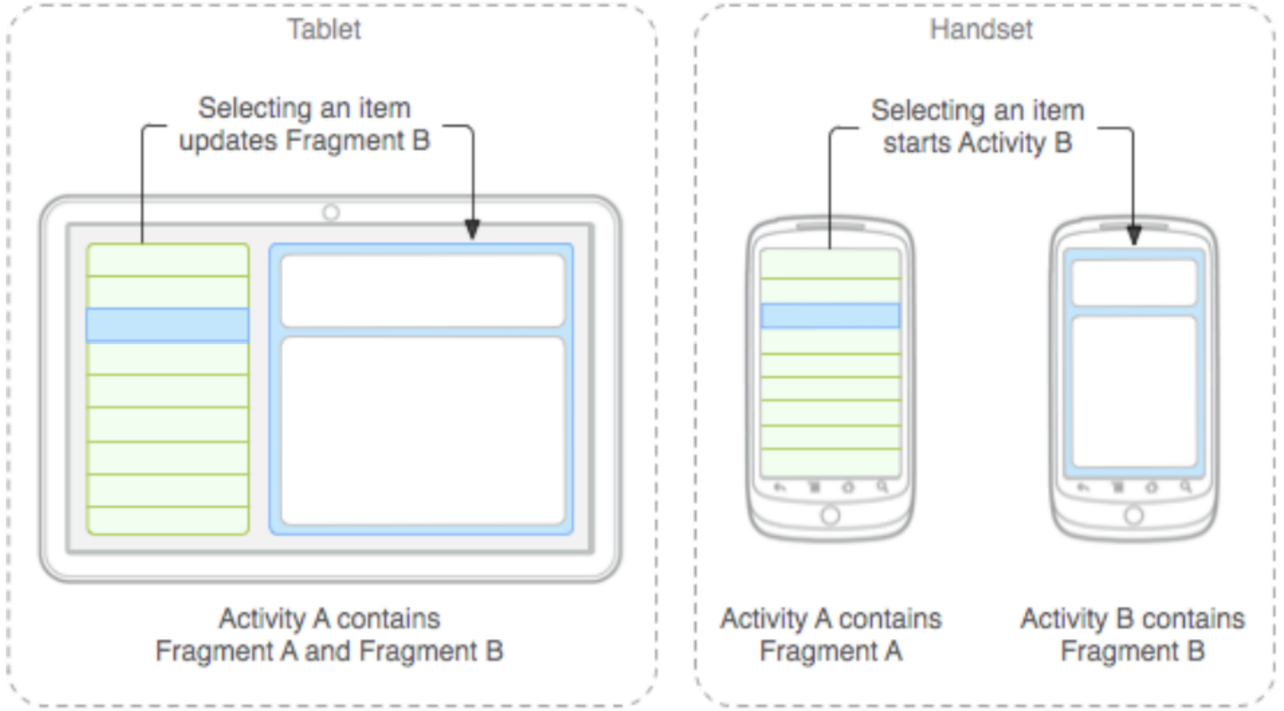
ByteDance 字节跳动

Fragment

- 概念和作用
- 生命周期和基本用法
- 结合 ViewPager 创建多 Tab 界面
- 如何和 Activity 通信

Fragment – Why

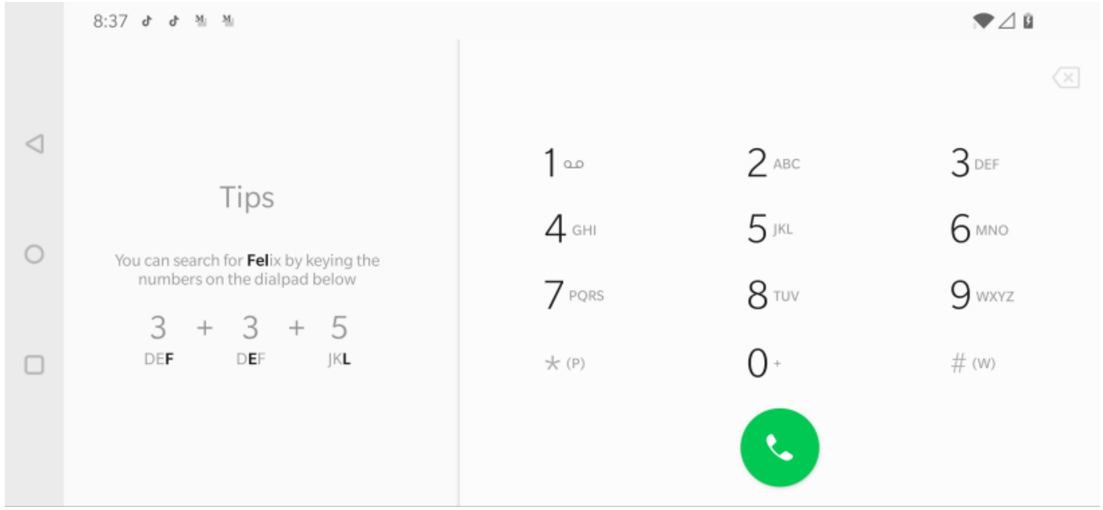
- 由来
  - 最早是为了跟进IOS平板

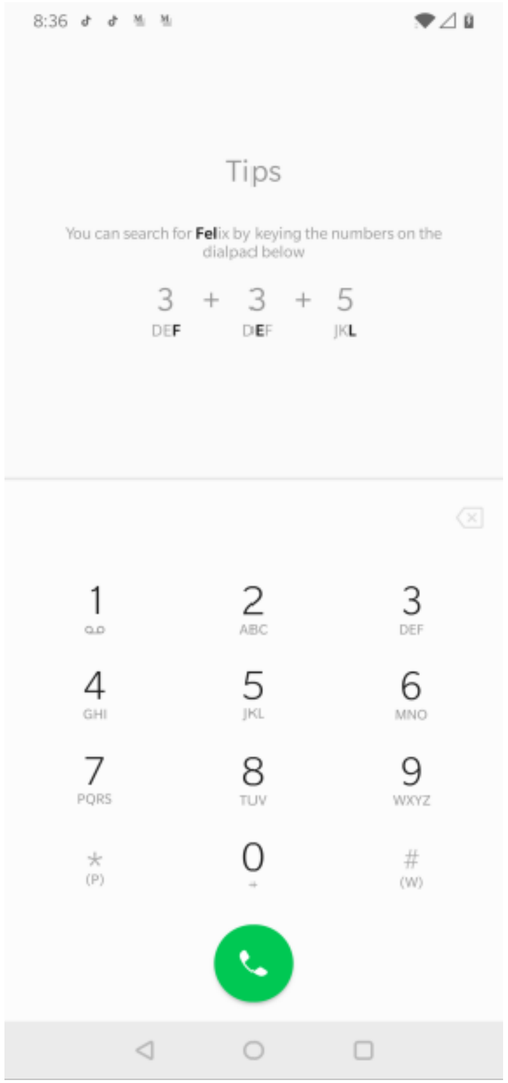


Fragment – Why

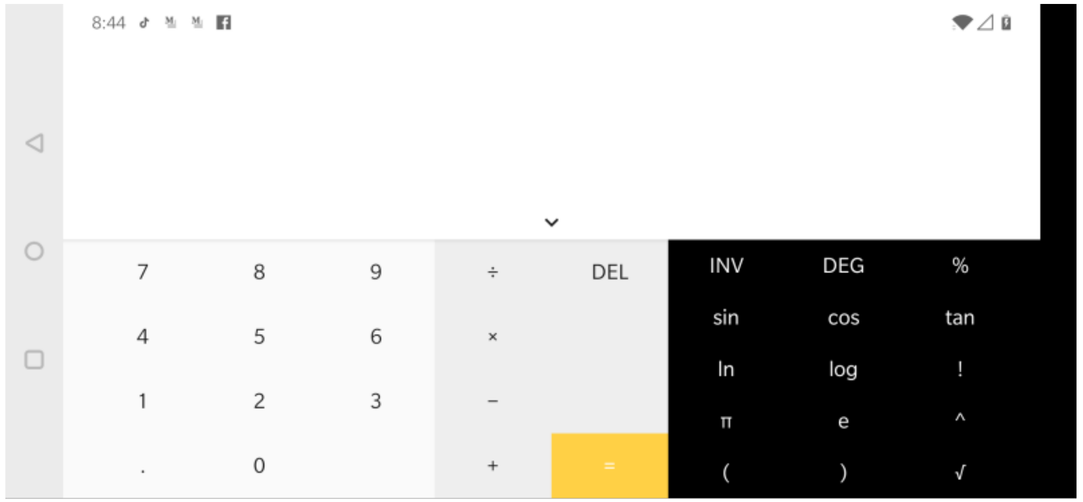
- Activity 模块化
- 可重用，灵活
- 相比View，带有生命周期概念

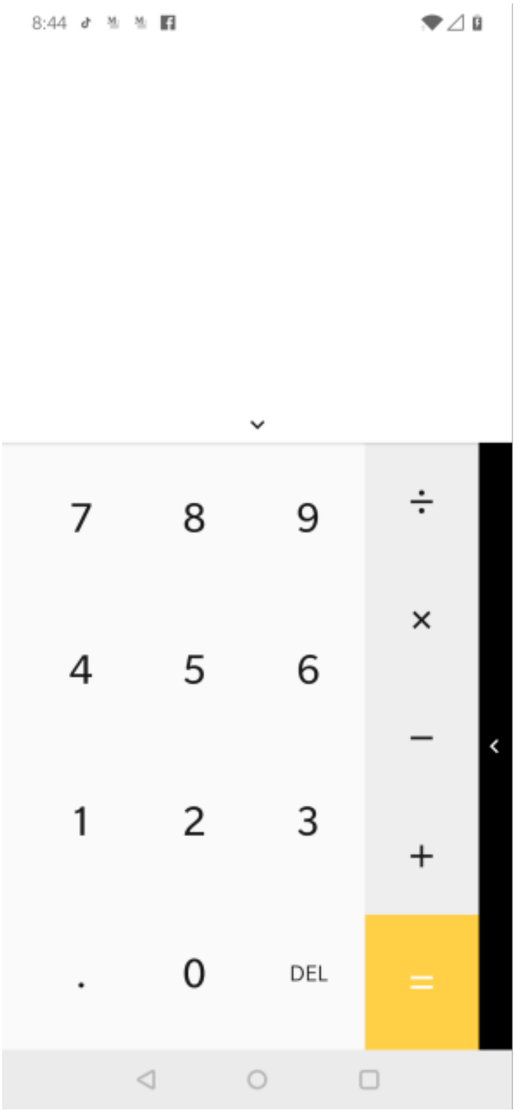
Fragment – UI重用





Fragment – UI重用



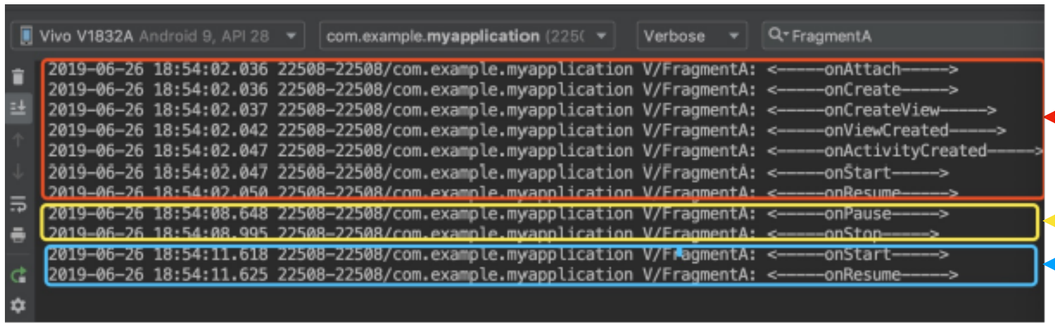


Fragment 生命周期

- onAttach/onDetach
- onCreate/onDestroy
- onCreateView/onDestroyView
- onActivityCreated
- onStart/onStop
- onResume/onPause

Fragment 生命周期

- case1: home

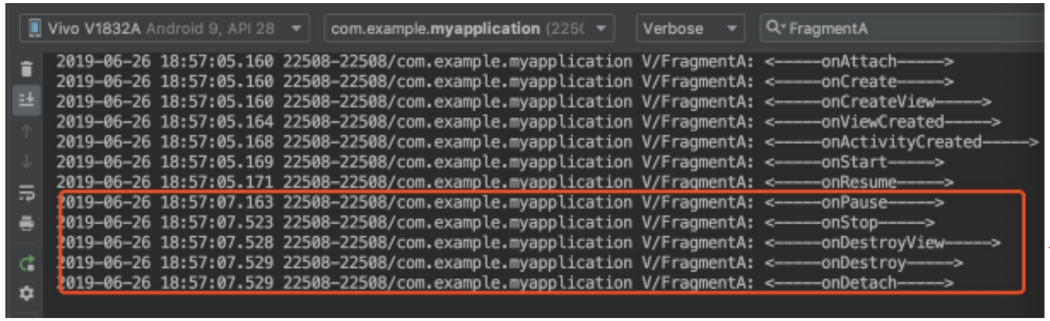


阶段1 应用启动

阶段2 按 Home 键

阶段3 重新启动应用

- case2: back



按 back 键返回主界面

静态添加 Fragment

- 定义 fragment 布局文件
- 定义 fragment 类
- 在 activity 布局文件中嵌入 fragment

示例 – 静态添加 Fragment – 1

- fragment_hello.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/hello_fragment" />
</FrameLayout>
```

示例 – 静态添加 Fragment – 2

- HelloFragment.java

```
public class HelloFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
        @Nullable ViewGroup container,
        @Nullable Bundle savedInstanceState) {
       return inflater.inflate(R.layout.fragment_hello, container, false);
    }
}
```

示例 – 静态添加 Fragment – 3

- activity_lifecycle_fragment.xml

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

    <fragment
        android:id="@+id/hello_fragment"
        android:name="com.example.chapter3.demo.fragment.HelloFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

动态添加/删除 Fragment

- 定义布局文件和 Fragment 类和静态添加相同
- Fragment 容器
  - 定义 Fragment 的位置和大小
- FragmentManager
  - 动态添加/替换/删除 Fragment
  - FragmentTransaction

示例 – 动态修改 Fragment – 1

- 在 activity 布局文件中定义 fragment 容器
- activity_dynamic_add_fragment.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

示例 – 动态修改 Fragment – 2

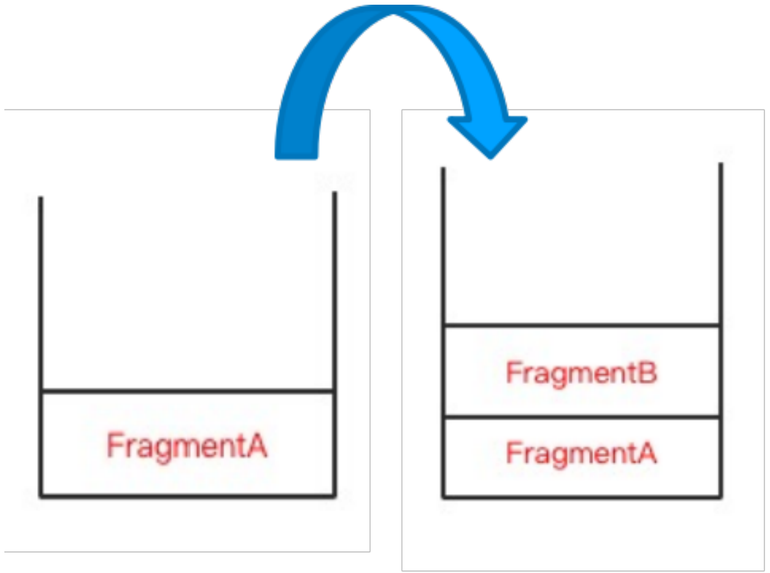
- 使用 FragmentManager 添加 Fragment
- DynamicAddFragmentActivity.java

```
public class DynamicAddFragmentActivity extends AppCompatActivity {

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

        getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.fragment_container, new HelloFragment())
                .commit();
    }
}
```

示例 – 添加到返回栈

- addToBackStack

```
getSupportFragmentManager() FragmentManager
        .beginTransaction() FragmentTransaction
        .add(R.id.fragment_container, ColorFragment.newInstance(color))
        .addToBackStack(null)
        .commit();
```



ViewPager + Fragment

- 常用于实现可滑动的多个视图
- 容器，类似于ListView/RecyclerView
- 需要通过 Adapter 配置内容
- 内容一般通过 Fragment 实现
- 可配置 TabLayout 或三方库添加 Title

示例 – ViewPager – 1

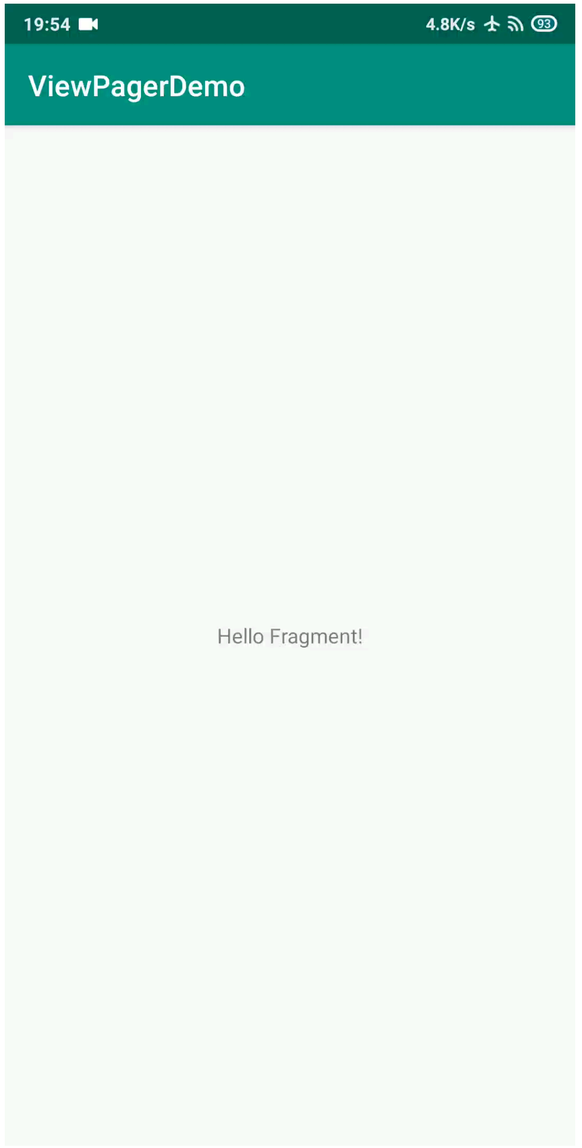
- 在布局 xml 中添加 ViewPager

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.viewpager.widget.ViewPager
        android:id="@+id/view_pager"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</FrameLayout>
```

示例 – ViewPager – 2

- 通过 Adapter 配置页面的 Fragment

```
public class ViewPagerActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_pager);
        ViewPager pager = findViewById(R.id.view_pager);
        pager.setAdapter(new FragmentPagerAdapter(getSupportFragmentManager()) {
            @Override
            public Fragment getItem(int i) {
                return new HelloFragment();
            }

            @Override
            public int getCount() {
                return 3;
            }
        });
    }
}
```



示例 – ViewPager + TabLayout – 1

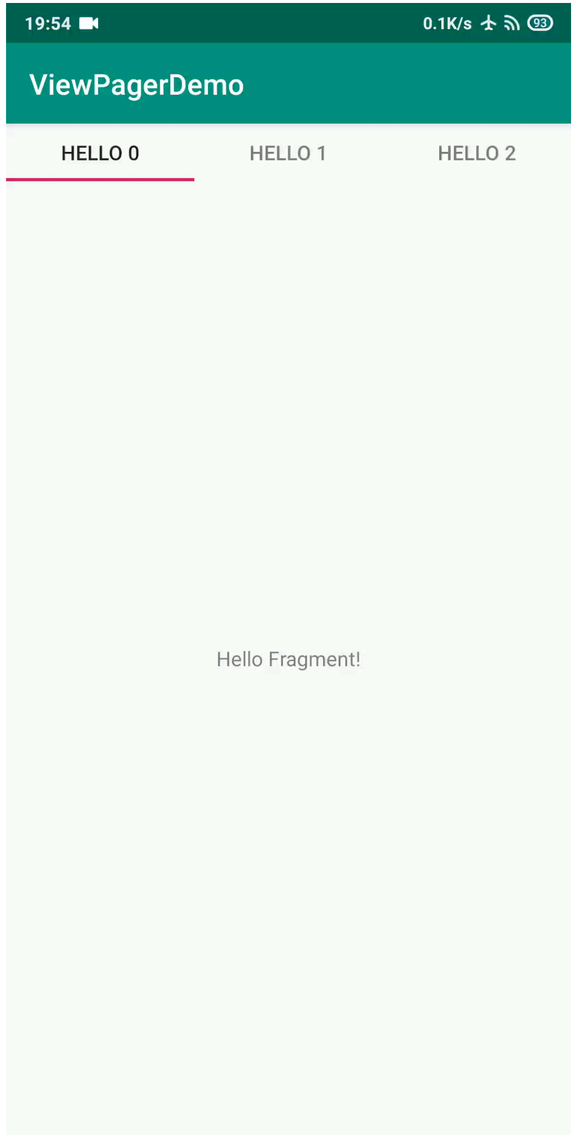
- 在布局 xml 中继续添加 TabLayout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.google.android.material.tabs.TabLayout
        android:id="@+id/tab_layout"
        android:layout_width="match_parent"
        android:layout_height="40dp" />

    <androidx.viewpager.widget.ViewPager
        android:id="@+id/view_pager"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

示例 – ViewPager + TabLayout – 2

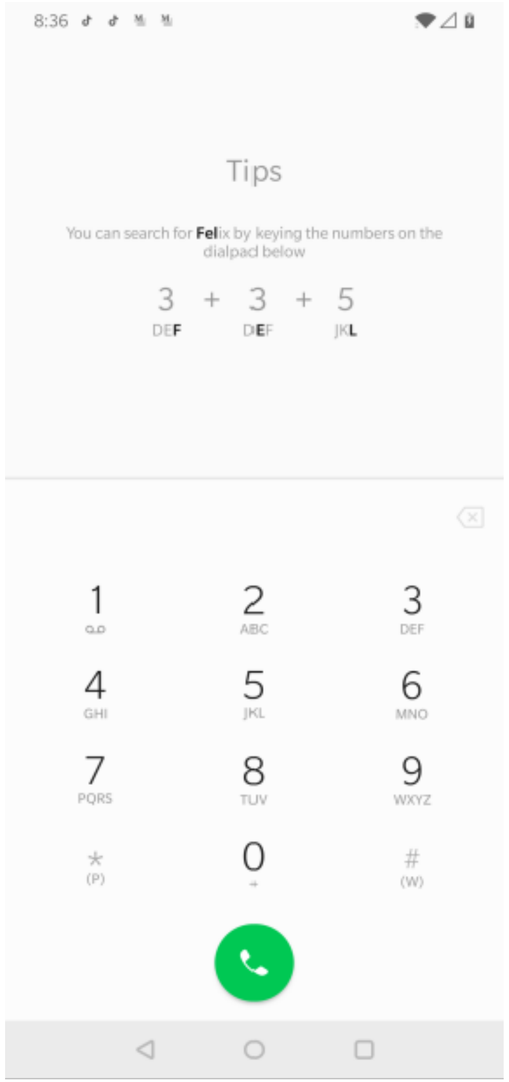
- 在代码中对 ViewPager 和 TabLayout 建立关联
- ViewPagerActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_view_pager_with_tab);
    ViewPager pager = findViewById(R.id.view_pager);
    TabLayout tabLayout = findViewById(R.id.tab_layout);
    pager.setAdapter(new FragmentPagerAdapter(getSupportFragmentManager()) {
        @Override
        public Fragment getItem(int i) {
            return new HelloFragment();
        }

        @Override
        public int getCount() {
            return PAGE_COUNT;
        }

        @Override
        public CharSequence getPageTitle(int position) {
            return "Hello " + position;
        }
    });
    tabLayout.setupWithViewPager(pager);
}
```



Fragment/Activity 之间的通信

- 构造 Fragment 时传递参数（setArguments/getArguments）
- 通过接口和回调



示例 – 通信 – 传参

```
public final class ColorFragment extends Fragment {
    private static final String KEY_EXTRA_COLOR = "extra_color";

    public static ColorFragment newInstance(int color) {
        ColorFragment cf = new ColorFragment();
        Bundle args = new Bundle();
        args.putInt(KEY_EXTRA_COLOR, color);
        cf.setArguments(args);
        return cf;
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState
        int color = Color.BLUE;
        Bundle args = getArguments();
        if (args != null) {
            color = args.getInt(KEY_EXTRA_COLOR, Color.BLUE);
        }
        View view = inflater.inflate(R.layout.fragment_color, container, false);
        view.setBackgroundColor(color);
        return view;
    }
}
```

示例 – 通信 – Listener – 1

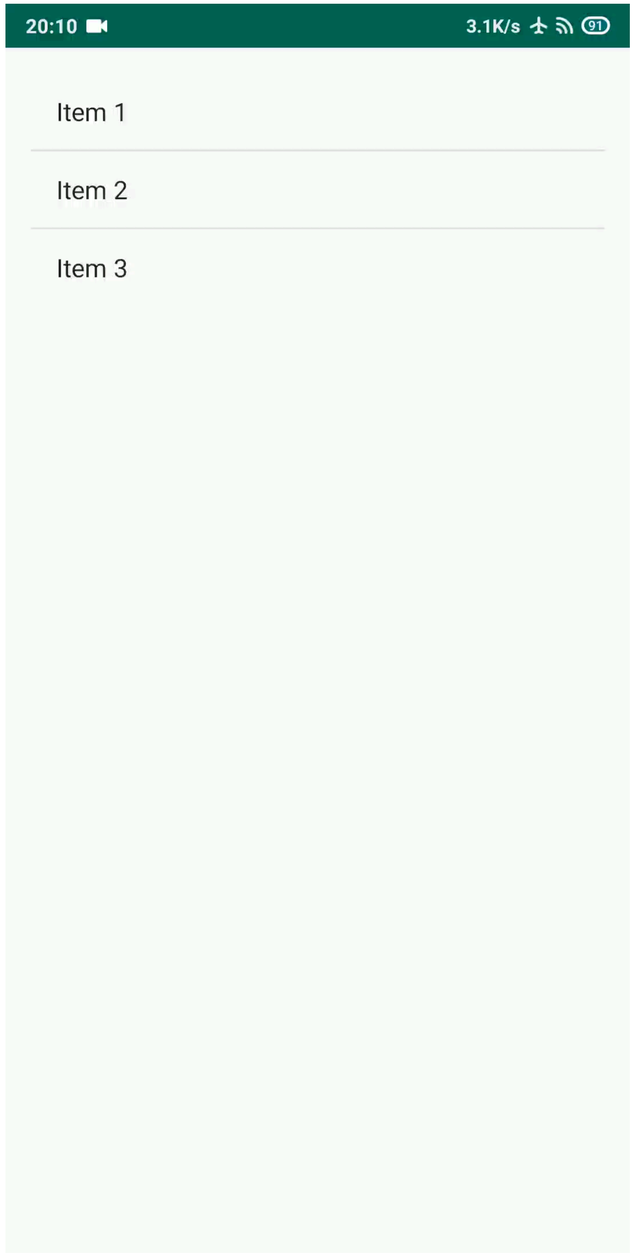
```
public final class ColorPlusFragment extends Fragment {

    public interface Listener {
        void onCollectColor(int color);
    }

    private Listener mListener;

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof Listener) {
            mListener = (Listener) context;
        }
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
        @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        Bundle args = getArguments();
        final int color = args != null ? args.getInt(KEY_EXTRA_COLOR, Color.BLUE) : Color.BLUE;
        View view = inflater.inflate(R.layout.fragment_color_plus, container, false);
        if (mListener != null) {
           mListener.onCollectColor(color);
       }
        return view;
    }
}
```

示例 – 通信 – Listener – 2

```
public class ViewPagerCommunicationActivity extends AppCompatActivity
        implements ColorPlusFragment.Listener {
    private ViewPager mViewPager;
    private RecyclerView mCollectView;
    private CollectColorAdapter mCollectAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {...}

    @Override
    public void onCollectColor(int color) { mCollectAdapter.addColor(color); }

    private static class ColorViewHolder extends RecyclerView.ViewHolder {
        public ColorViewHolder(@NonNull View itemView) { super(itemView); }
    }

    private static class CollectColorAdapter extends RecyclerView.Adapter<ColorViewHolder> {...}
}
```

示例 – Master Detail

- Portrait
  - Master Activity: Item List
  - Detail Activity: Item Detail
- Landscape
  - One Activity: List & Detail



示例 – Master Detail

- ItemsListActivity 横竖屏布局文件不同

```
public class ItemsListActivity extends FragmentActivity implements ItemsListFragment.OnItemSelectedListener {
    private boolean isTwoPane = false;

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

    private void determinePaneLayout() {
        FrameLayout fragmentItemDetail = findViewById(R.id.flDetailContainer);
        if (fragmentItemDetail != null) {
            isTwoPane = true;
            ItemsListFragment fragmentItemsList =
                    (ItemsListFragment) getSupportFragmentManager().findFragmentById(R.id.fragmentItemsList);
            fragmentItemsList.setActivateOnItemClick(true);
        }
    }
```

```
Choose Declaration
"@+id/flDetailContainer" src/main/res/layout/activity_item_detail.xml  Chapter3Demo.app
"@+id/flDetailContainer" src/main/res/layout-land/activity_items.xml   Chapter3Demo.app
```

示例 – Master Detail

- 横屏

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:showDividers="middle"
    android:baselineAligned="false"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <fragment
        android:id="@+id/fragmentItemsList"
        android:name="xxx.ItemsListFragment"
        android:layout_height="wrap_content"
        android:layout_width="0dp"
        android:layout_weight="1"
        tools:layout="@layout/fragment_items_list" />

    <View android:background="#000000"
        android:layout_width="1dp"
        android:layout_height="wrap_content"/>

    <FrameLayout
        android:id="@+id/flDetailContainer"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="3" />

</LinearLayout>
```

- 竖屏

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp">

    <fragment
        android:id="@+id/fragmentItemsList"
        android:name="xxx.ItemsListFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentTop="true"
        tools:layout="@layout/fragment_items_list" />

</RelativeLayout>
```

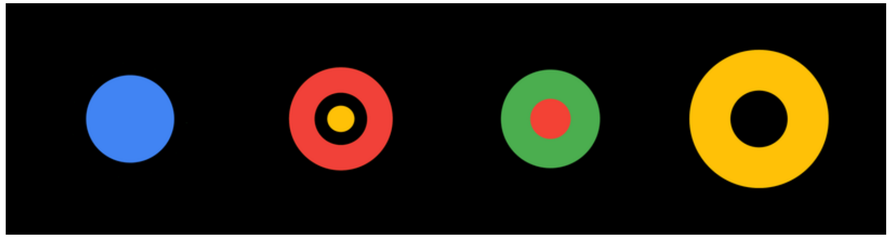
Summary

- Fragment: 灵活，可重用，迷你 Activity
- 生命周期、静态/动态添加用法
- ViewPager & Fragment
- 和 Activity 通信：Argument、Listener

Animation

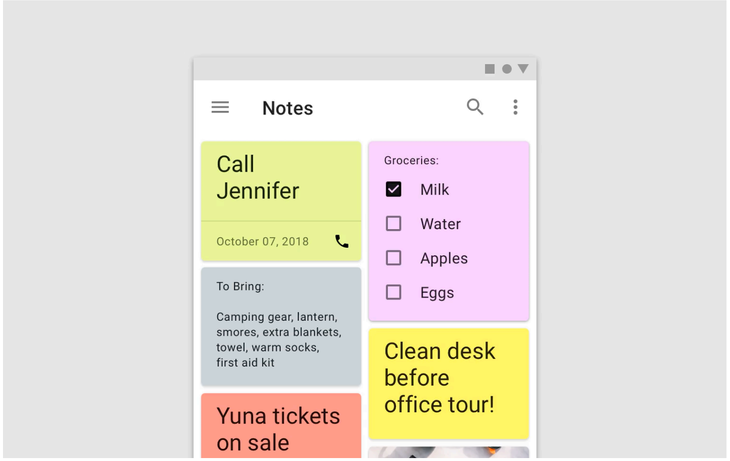
ByteDance 字节跳动

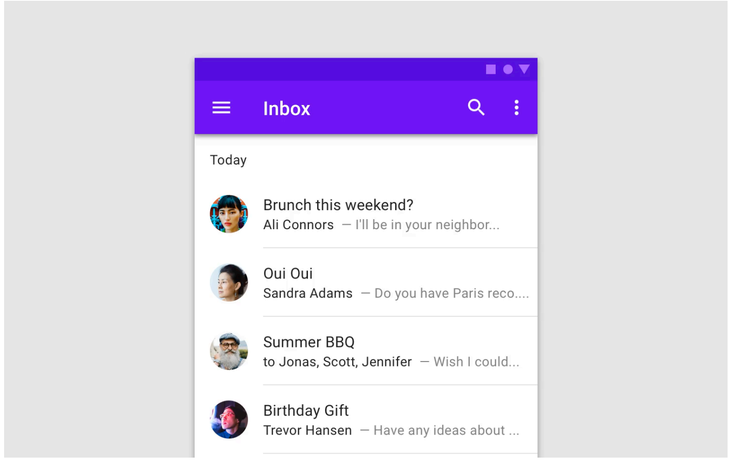
Animation



Hang on

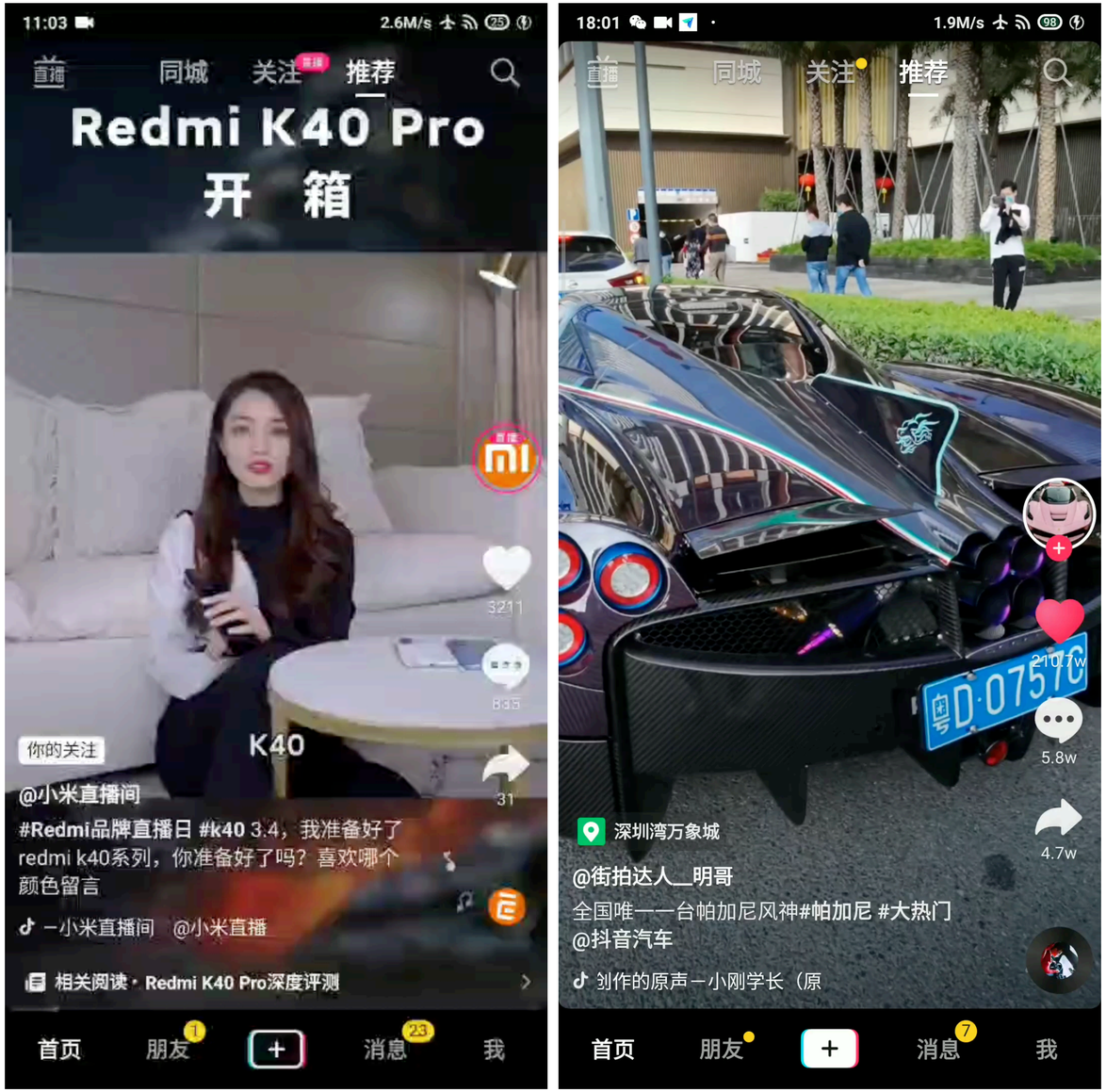
Setting up your account





Animation 意义

- 用户体验
- Informative & Focused & Expressive
- 行为引导

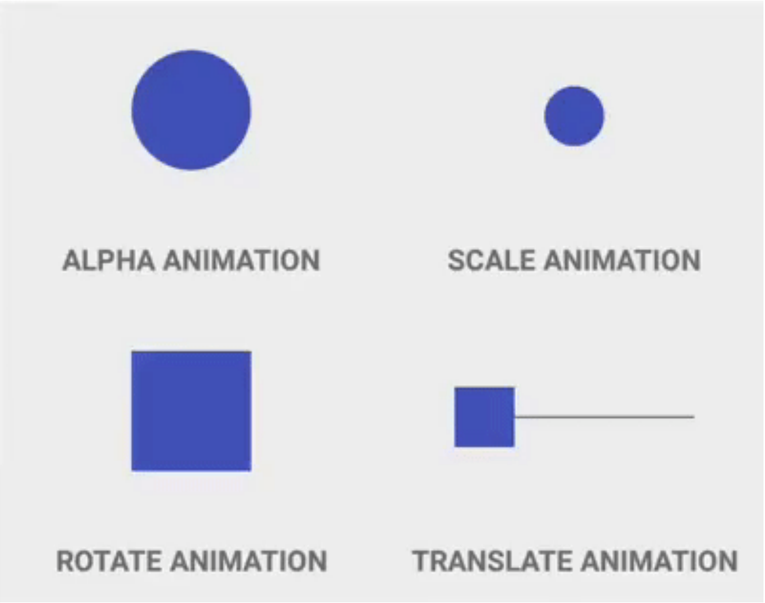


Animation 分类

- 属性动画
- 视图动画/补间动画
- 逐帧动画/drawable动画

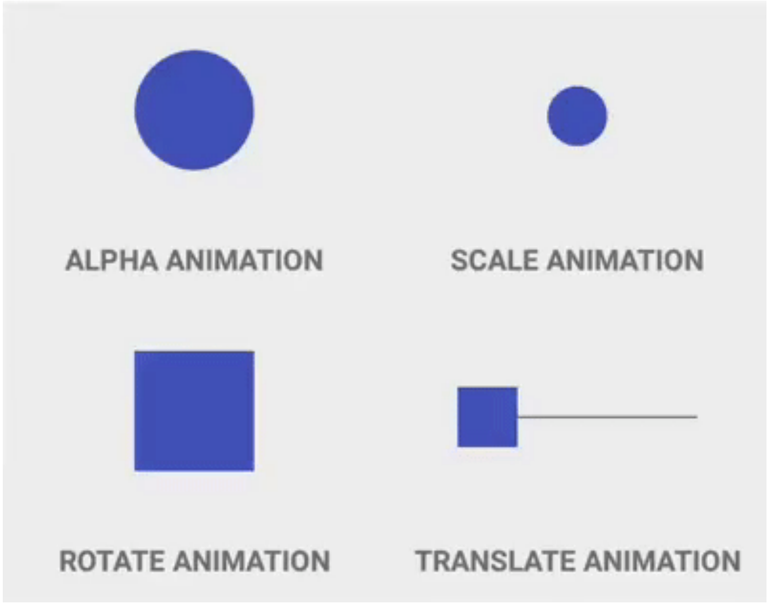
属性动画

- 基于属性的动画
- 一切可以连续变化的属性都是动画的元素
- 实现一种复杂动画，就是将动画拆解成不同属性组合的过程



属性动画 – 角色构成

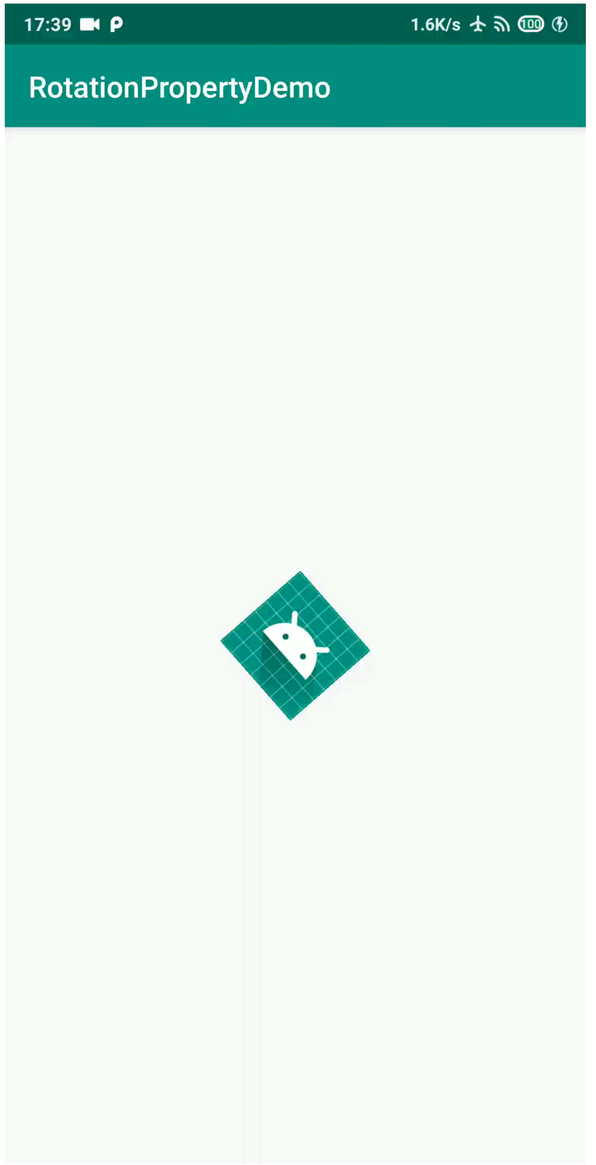
- Property：alpha, scaleX, scaleY, rotation, translationX, translationY
- StartValue, EndValue, Duration
- RepeatCount: number, infinite
- RepeatMode: restart, reverse
- TypeEvaluator: IntEvaluator, ArgbEvaluator
- Interpolator: linear/accelerate/decelerate/…



属性动画 – 示例1

- RotationPropertyActivity.java

```
ObjectAnimator animator = ObjectAnimator.ofFloat(findViewById(R.id.image_view),
        "rotation", 0, 360);
animator.setRepeatCount(ValueAnimator.INFINITE);
animator.setInterpolator(new LinearInterpolator());
animator.setDuration(8000);
animator.setRepeatMode(ValueAnimator.RESTART);
animator.start();
```

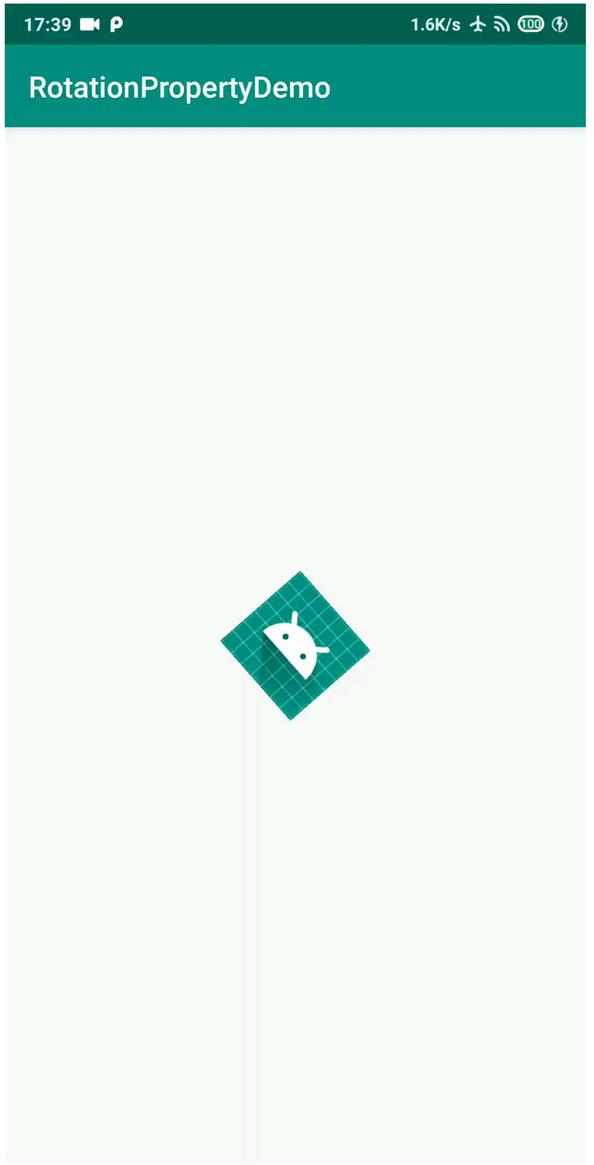


属性动画 – 示例1

- rotate.xml

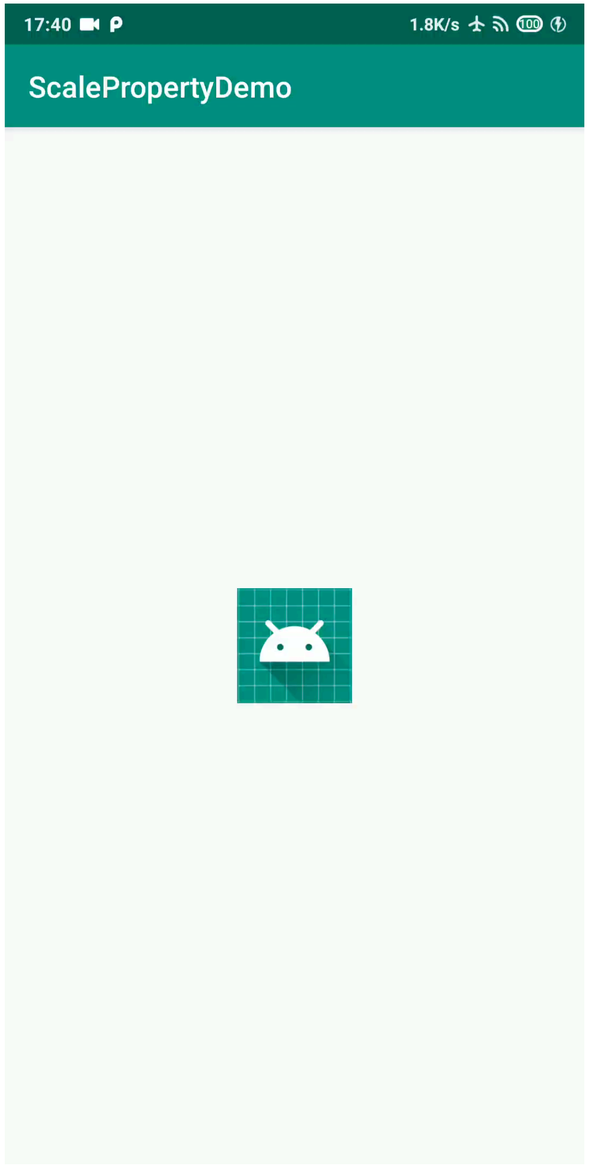
```
<!-- animator/rotate.xml -->
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="8000"
    android:propertyName="rotation"
    android:interpolator="@android:anim/linear_interpolator"
    android:repeatCount="infinite"
    android:repeatMode="restart"
    android:valueFrom="0"
    android:valueTo="360" />
```

```
// RotationPropertyActivity.java
Animator animator = AnimatorInflater.loadAnimator(this, R.animator.rotate);
animator.setTarget(findViewById(R.id.image_view));
animator.start();
```



属性动画 – 示例2

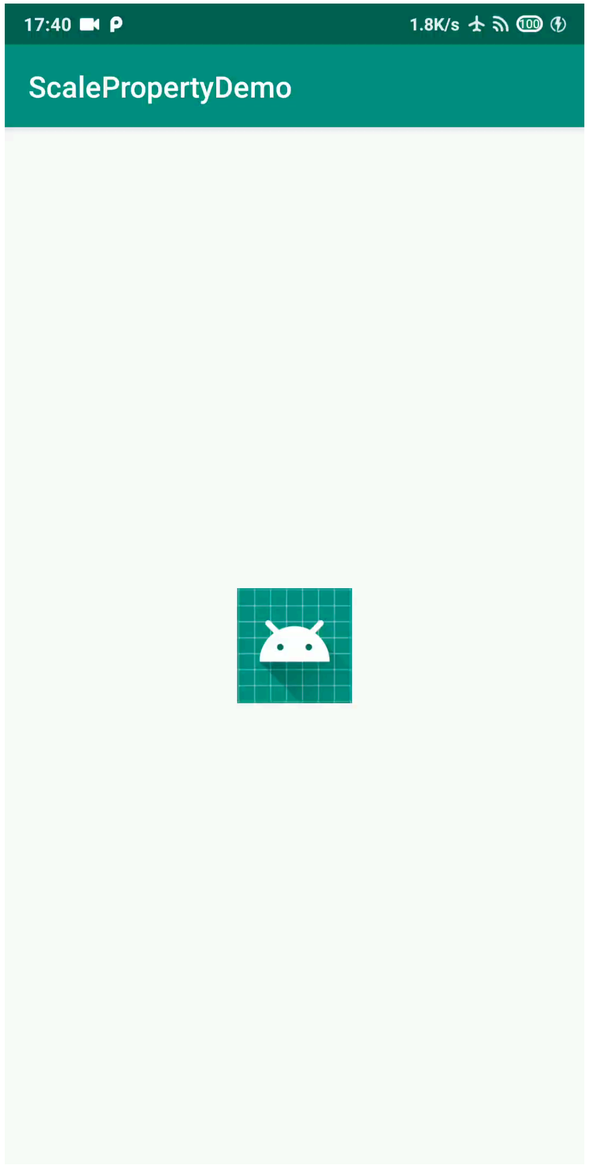
```
View imageView = findViewById(R.id.image_view);
ObjectAnimator scaleXAnimator = ObjectAnimator.ofFloat(imageView,
        propertyName: "scaleX",  ...values: 1.1f, 0.9f);
scaleXAnimator.setRepeatCount(ValueAnimator.INFINITE);
scaleXAnimator.setInterpolator(new LinearInterpolator());
scaleXAnimator.setDuration(1000);
scaleXAnimator.setRepeatMode(ValueAnimator.REVERSE);

ObjectAnimator scaleYAnimator = ObjectAnimator.ofFloat(imageView,
        propertyName: "scaleY",  ...values: 1.1f, 0.9f);
scaleYAnimator.setRepeatCount(ValueAnimator.INFINITE);
scaleYAnimator.setInterpolator(new LinearInterpolator());
scaleYAnimator.setDuration(1000);
scaleYAnimator.setRepeatMode(ValueAnimator.REVERSE);

AnimatorSet animatorSet = new AnimatorSet();
animatorSet.playTogether(scaleXAnimator, scaleYAnimator);
animatorSet.start();
```



属性动画 – 示例2

```xml
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:duration="1000"
        android:valueFrom="1.1"
        android:valueTo="0.9"
        android:propertyName="scaleX"
        android:interpolator="@android:anim/linear_interpolator"
        android:repeatMode="reverse"
        android:repeatCount="infinite" />

    <objectAnimator
        android:duration="1000"
        android:valueFrom="1.1"
        android:valueTo="0.9"
        android:propertyName="scaleY"
        android:interpolator="@android:anim/linear_interpolator"
        android:repeatMode="reverse"
        android:repeatCount="infinite" />
</set>
```

```
Animator animator = AnimatorInflater.loadAnimator( context: this, R.animator.breath);
animator.setTarget(findViewById(R.id.image_view));
animator.start();
```



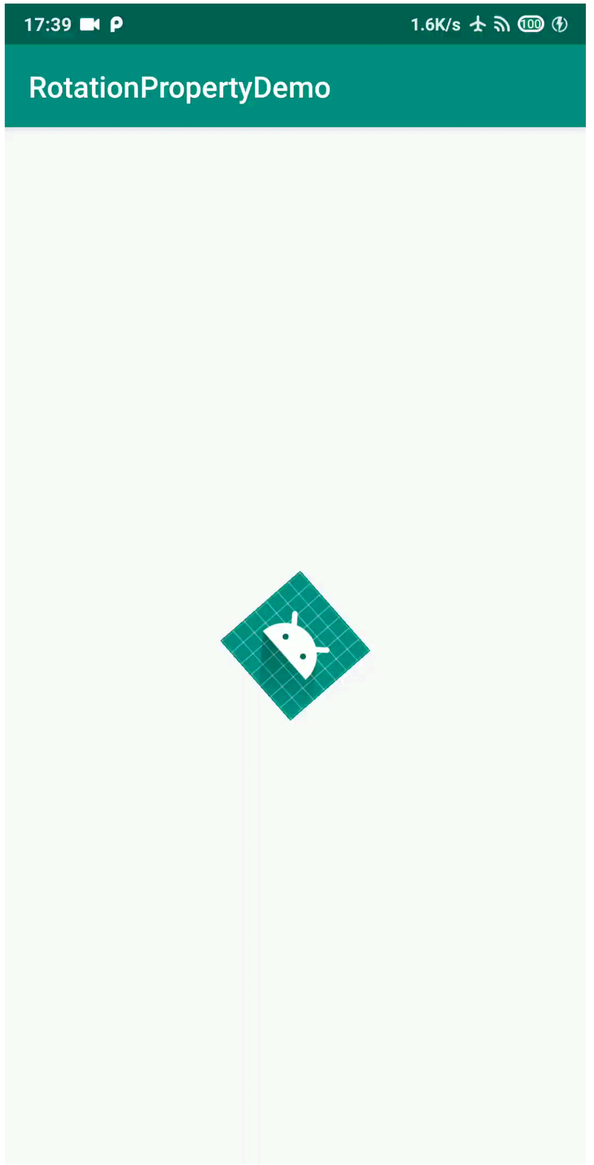
属性动画 – 特点，xml语法

```
<set
  android:ordering=["together" | "sequentially"]>

    <objectAnimator
        android:propertyName="string"
        android:duration="int"
        android:valueFrom="float | int | color"
        android:valueTo="float | int | color"
        android:startOffset="int"
        android:repeatCount="int"
        android:repeatMode=["repeat" | "reverse"]
        android:valueType=["intType" | "floatType"]/>

    <animator
        android:duration="int"
        android:valueFrom="float | int | color"
        android:valueTo="float | int | color"
        android:startOffset="int"
        android:repeatCount="int"
        android:repeatMode=["repeat" | "reverse"]
        android:valueType=["intType" | "floatType"]/>
    <set>
        ...
    </set>
</set>
```

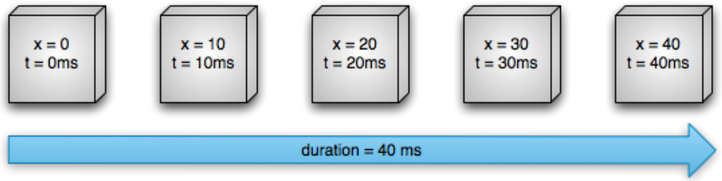
背后 – ValueAnimator，旋转封面

- RotationPropertyActivity.java

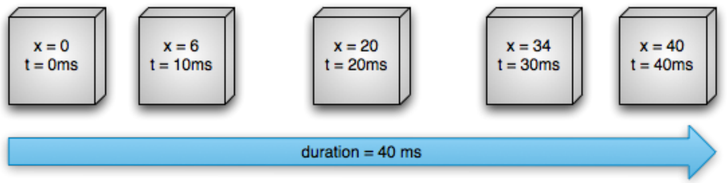
```
final View v = findViewById(R.id.image_view);
ValueAnimator valueAnimator = ValueAnimator.ofFloat(0, 360);
valueAnimator.setRepeatCount(ValueAnimator.INFINITE);
valueAnimator.setInterpolator(new LinearInterpolator());
valueAnimator.setRepeatMode(ValueAnimator.RESTART);
valueAnimator.setDuration(8000);
valueAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
        v.setRotation((float) animation.getAnimatedValue());
    }
});
valueAnimator.start();
```



属性动画 – 原理

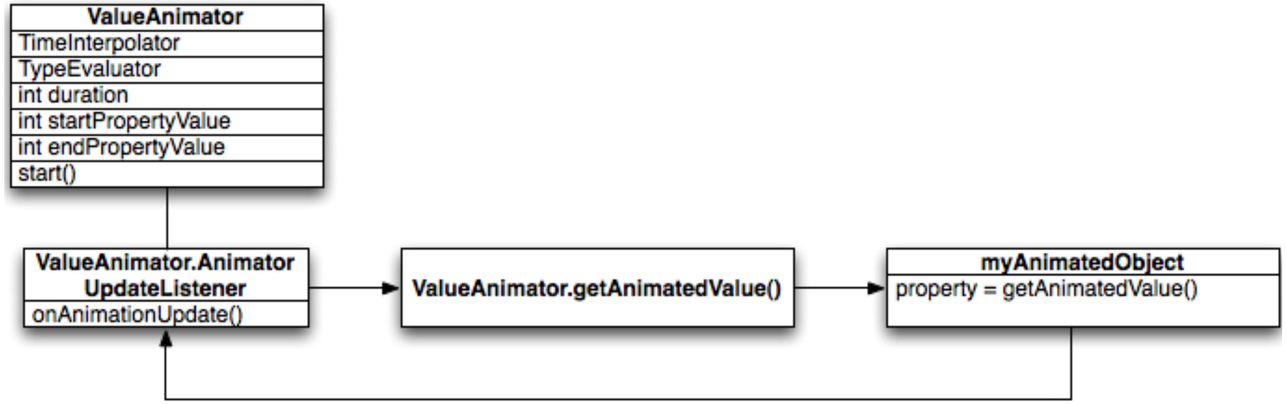


线性动画



非线性动画

- TimeInterpolator: 插值器，根据时间完成度计算动画完成度
- TypeEvaluator: 估值器，根据动画完成度计算具体属性值



动画如何计算动画

属性动画 – 自定义属性

- 操作的对象可以不是View

```
class WrapperView {
    private View mTarget;

    public WrapperView(View mTarget) {
        this.mTarget = mTarget;
    }
    public int getHeight() {
        return mTarget.getLayoutParams().height;
    }
    public void setHeight(int height) {
        mTarget.getLayoutParams().height = height;
        mTarget.requestLayout();
    }
}
```

属性动画

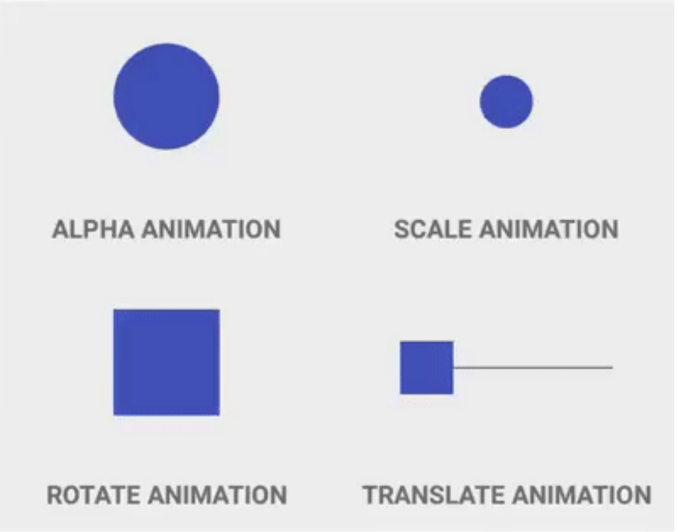
- ObjectAnimator 操作的属性必须具有get/set方法，不然无法生效

```
ObjectAnimator objectAnimator = ObjectAnimator.ofFloat(mTestButton1, propertyName: "Height", ...values: 100);
objectAnimator.start();

  ValueAnimator animator = ValueAnimator.ofFloat(0f,300f);
```

The setter for this property does not match the expected signature (public void setHeight(float arg) more... (⌘F1)

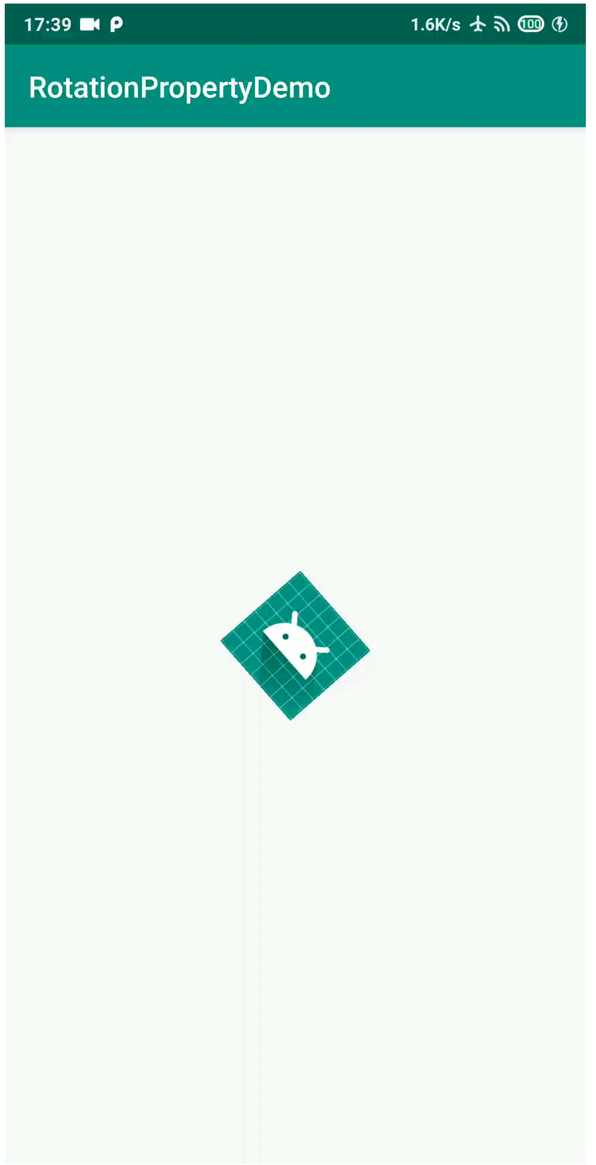
属性动画 vs 视图动画

- 属性动画：android.animation
- 视图动画：android.view.animation
  - 只能对 View 做动画
  - 只能对 View 的某些绘制属性做动画
  - 只是视觉效果



视图动画 – 示例1

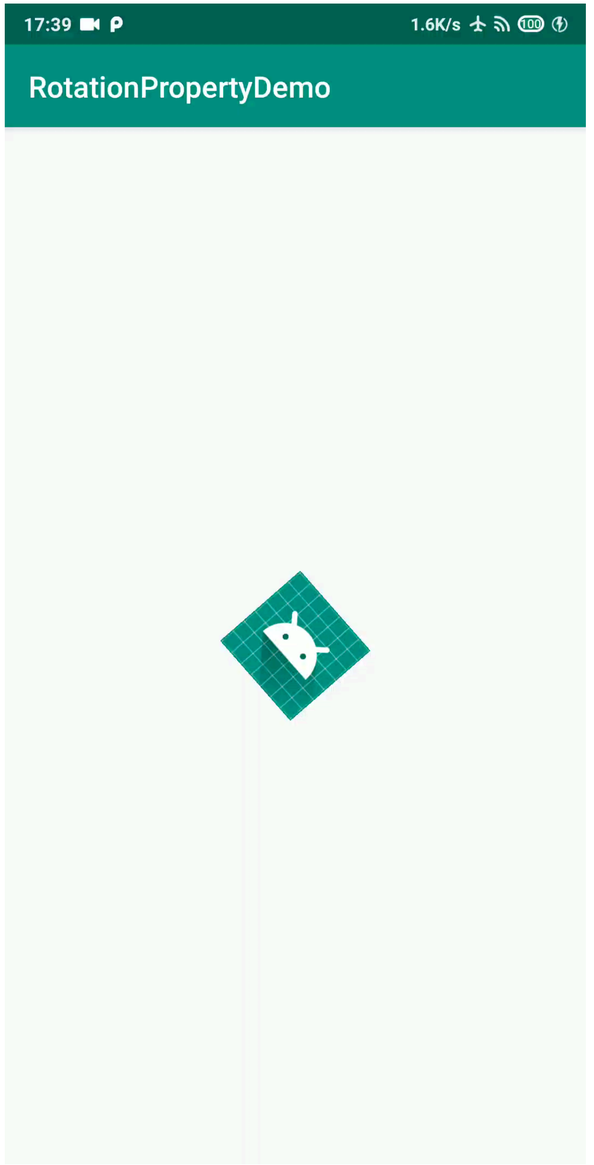
```
RotateAnimation rotateAnimation = new RotateAnimation(
        0, 360,
        Animation.RELATIVE_TO_SELF, 0.5f,
        Animation.RELATIVE_TO_SELF, 0.5f);
rotateAnimation.setDuration(1000);
mImage.startAnimation(rotateAnimation);
```



视图动画 – 示例2

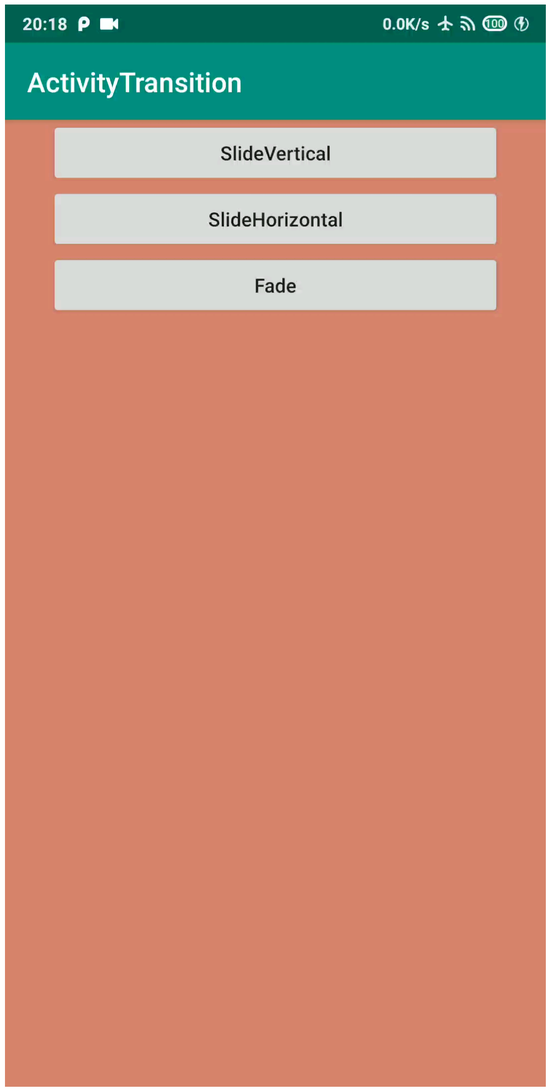
```xml
<!-- anim/rotate.xml -->
<set xmlns:android="http://schemas.android.com/apk/res/android" >
    <rotate
        android:duration="1000"
        android:fromDegrees="0"
        android:interpolator="@android:anim/accelerate_decelerate_interpolator"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toDegrees="+360" />

</set>
```

```
loadAnimation = AnimationUtils.loadAnimation(this, R.anim.rotate);
mImage.startAnimation(loadAnimation);
```



Activity 切换动画

```
/**
 * Call immediately after one of the flavors of {@link #startActivity(Intent)}
 * or {@link #finish} to specify an explicit transition animation to
 * perform next.
 *
 * <p>As of {@link android.os.Build.VERSION_CODES#JELLY_BEAN} an alternative
 * to using this with starting activities is to supply the desired animation
 * information through a {@link ActivityOptions} bundle to
 * {@link #startActivity(Intent, Bundle)} or a related function.  This allows
 * you to specify a custom animation even when starting an activity from
 * outside the context of the current top activity.
 *
 * @param enterAnim A resource ID of the animation resource to use for
 * the incoming activity.  Use 0 for no animation.
 * @param exitAnim A resource ID of the animation resource to use for
 * the outgoing activity.  Use 0 for no animation.
 */
public void overridePendingTransition(int enterAnim, int exitAnim)
```



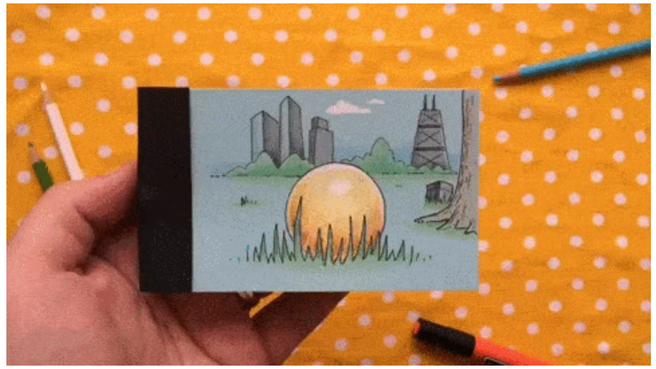
Activity 切换动画 – 示例 FadeInOut

```
// 进入动画
startActivity(new Intent(TransitionActivity.this, TransitionActivity.class));
overridePendingTransition(R.anim.fade_in, R.anim.fade_out);


// 退出动画
super.finish();
overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
```

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


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

逐帧动画/Drawable 动画

- 逐帧动画可以被当作一种特殊的drawable对象
- 逐帧动画会按次序播放一系列图片
- 逐帧动画会一次性将所有图片加载到内存中，会有OOM风险



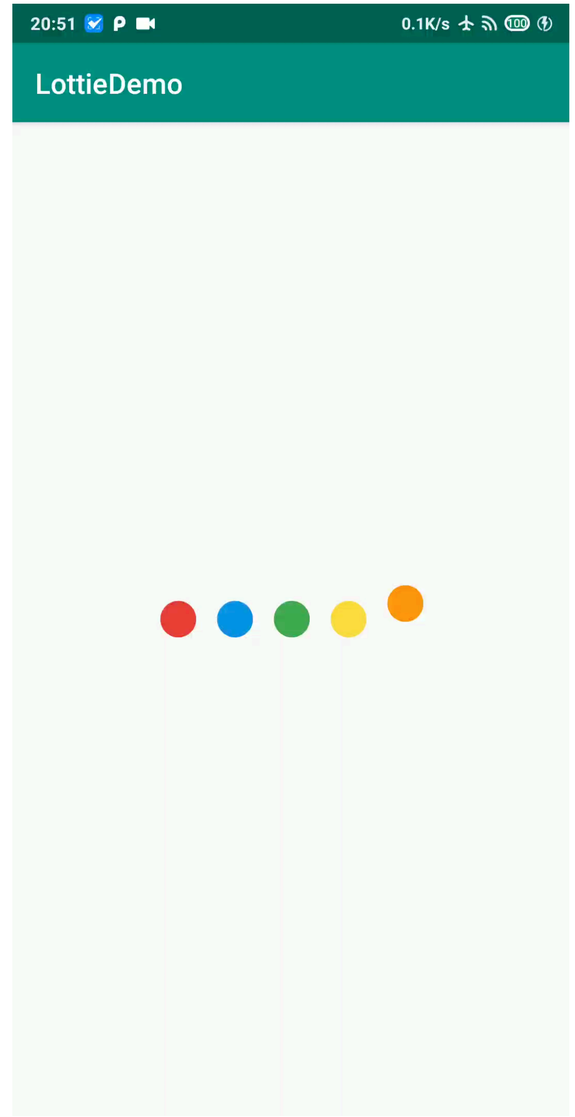
示例 – AnimationDrawable

```
// res/drawable/anim_list.xml
<animation-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/one" android:duration="500"/>
    <item android:drawable="@drawable/two" android:duration="500"/>
    <item android:drawable="@drawable/three" android:duration="500"/>
    <item android:drawable="@drawable/four" android:duration="500"/>
</animation-list>
```

```
// activity
mImage = findViewById(R.id.image_frame);
mImage.setBackgroundResource(R.drawable.anim_list);
AnimationDrawable drawable = (AnimationDrawable) mImage.getBackground();
drawable.start();
```

Lottie

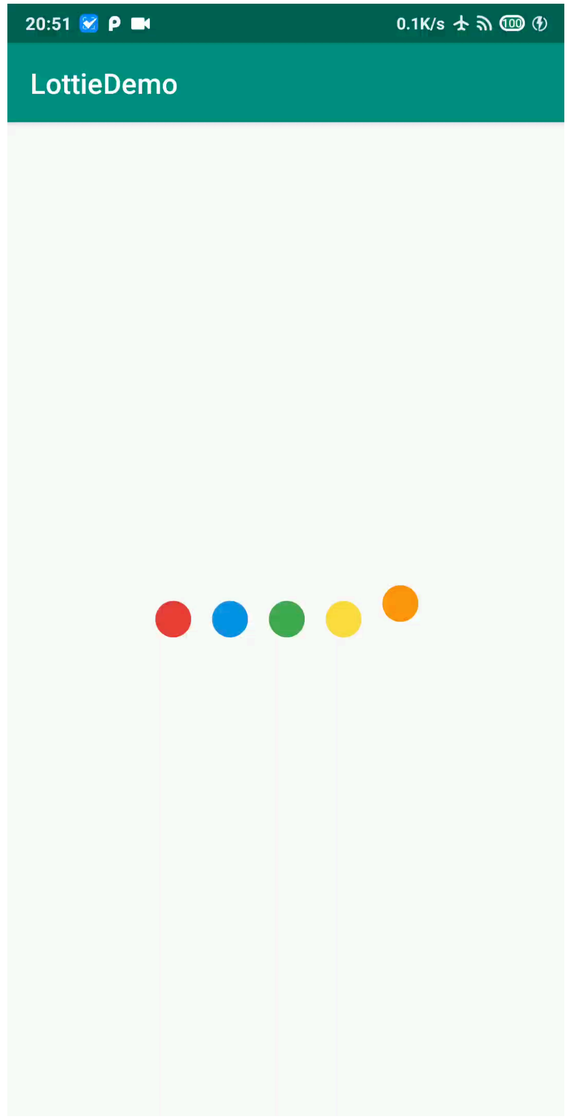
- airbnb公司的开源库
- 可以直接导入AE制作的动画素材
- 本质是将所有动画元素抽象成绘制属性

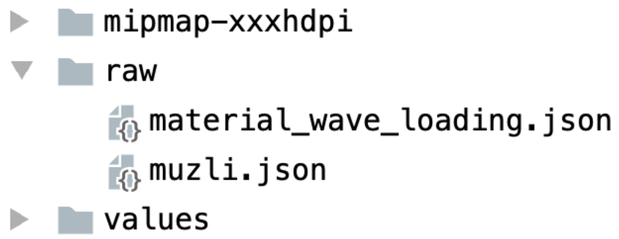


Lottie – 示例

```
dependencies {
// app/build.gradle 添加依赖
    implementation 'com.airbnb.android:lottie:2.7.0'
}
```

```
<com.airbnb.lottie.LottieAnimationView
    android:id="@+id/animation_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    app:lottie_autoPlay="true"
    app:lottie_loop="true"
    app:lottie_rawRes="@raw/material_wave_loading" />
```





Summary

- 动画意义
- 属性动画
  - ObjectAnimator & AnimatorSet
  - 原理
  - vs 视图动画
- Lottie

Reference

ByteDance 字节跳动

Fragment

- 官方文档：https://developer.android.com/guide/fragments
- 中文翻译：上、下
- Fragment 源码解读：https://juejin.cn/post/6844904086437904398
- ViewPager基本使用

Animation

- 属性动画：https://developer.android.com/guide/topics/graphics/prop-animation
- 贝塞尔曲线可视化：https://cubic-bezier.com/
- 非官方总结：https://juejin.cn/post/6844903465211133959
- 其他阅读材料：属性动画（上手篇、进阶篇）

Animation

- Lottie Android 指南：https://airbnb.io/lottie/#/android
- Lottie 资源：https://lottiefiles.com/

课件及作业

- https://github.com/bytedance-android-camp-sjtu-2021/chapter-3

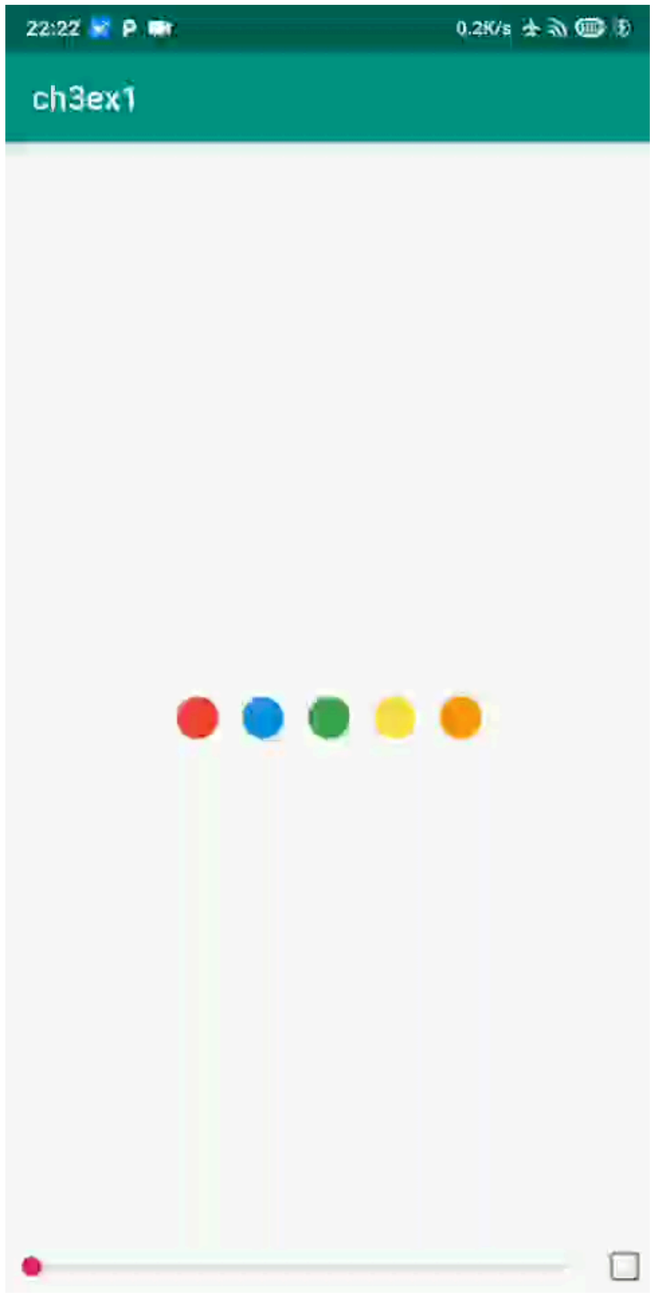
Exercise

ByteDance 字节跳动

练习1 – ch3ex1

- 引入 Lottie 库实现简单的图标动画
- 在 activity_main.xml 中添加 LottieAnimationView
- 在 SeekBar 的回调中修改 LottieAnimationView 的进度

练习1 – ch3ex1

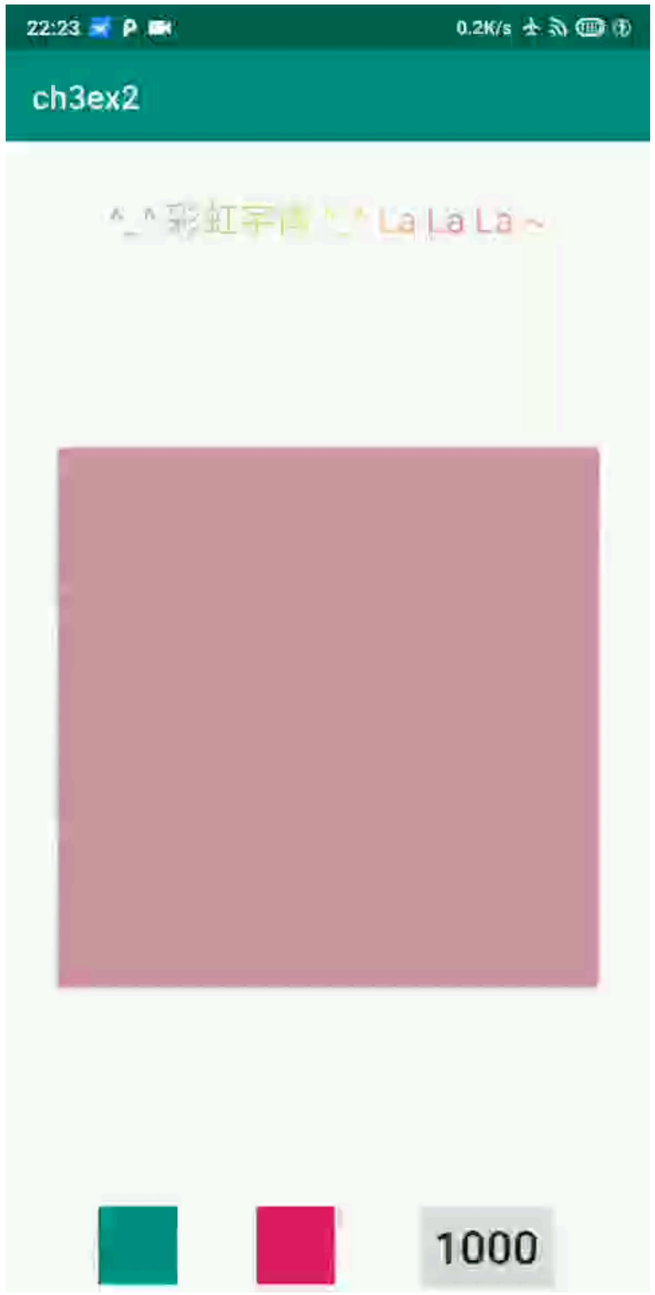


练习2 – ch3ex2

- 使用属性动画，练习AnimatorSet 和 scale/fade 等基本动画样式

    1. 添加 scale 动画
    2. 添加 alpha 动画
    3. 组合到 AnimatorSet 中

练习2 – ch3ex2

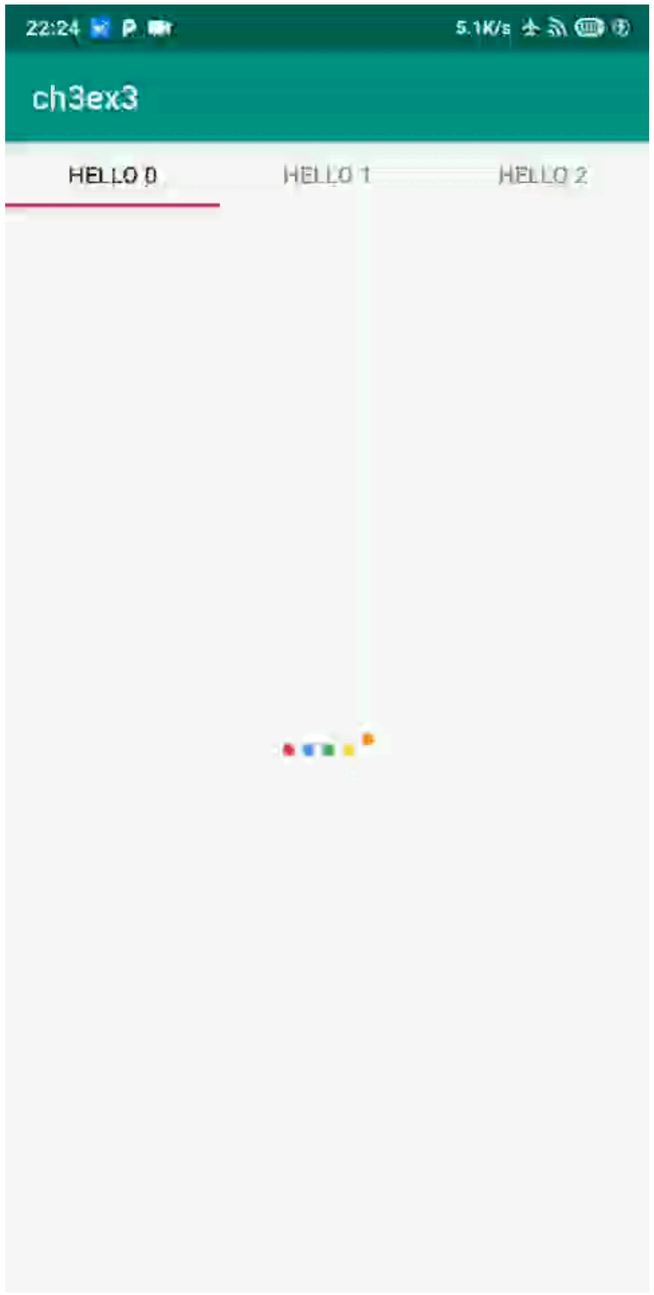


练习3 – ch3ex3

- 使用 ViewPager 和 Fragment 做一个简单的好友列表界面

    1. 使用 ViewPager 和 Fragment 做可滑动界面
    2. 使用 TabLayout 添加 Tab 支持
    3. 对于好友列表 Fragment，使用 Lottie 实现 Loading 效果，5s后展示实际列表，要求这里的动效是淡入淡出

练习3 – ch3ex3



THANKS.

ByteDance字节跳动