用户界面 UI 进阶

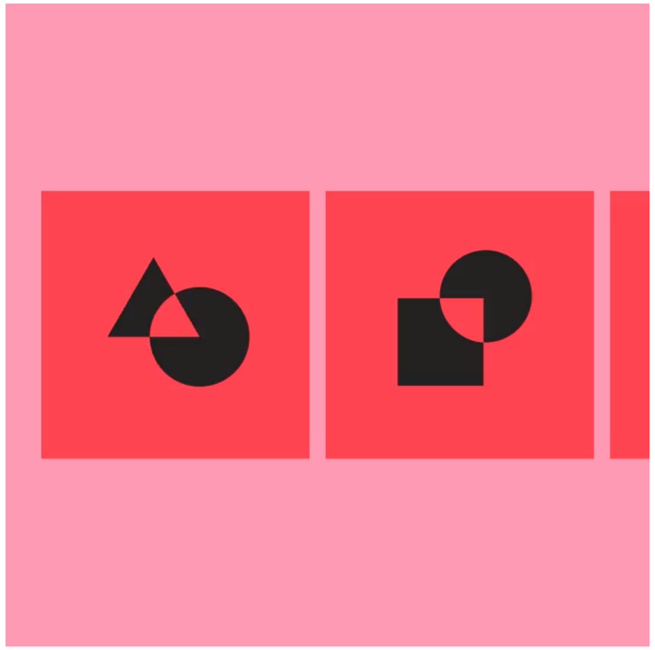
主要内容

- UI 进阶
  - 动画 Animation
  - Fragment
  - 自定义View
- 多线程编程

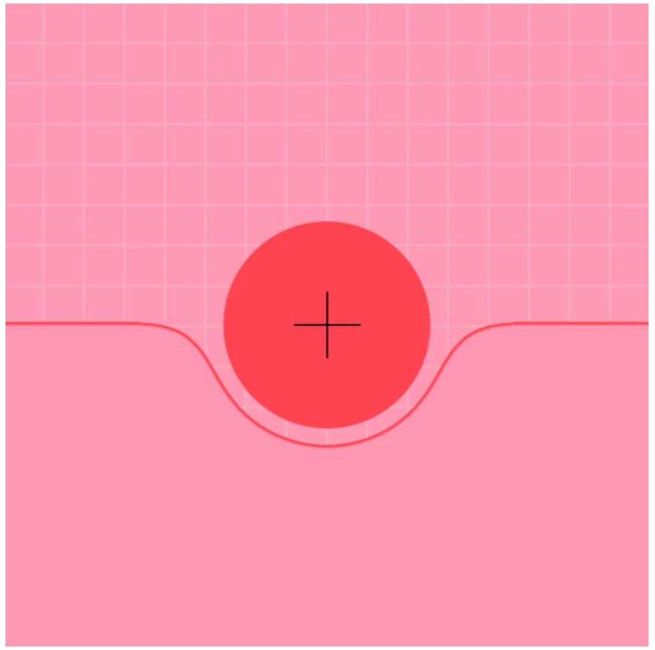
动画 Animation

- **意义**
- **属性动画**
- Activity **切换动画**
- Drawable **动画**

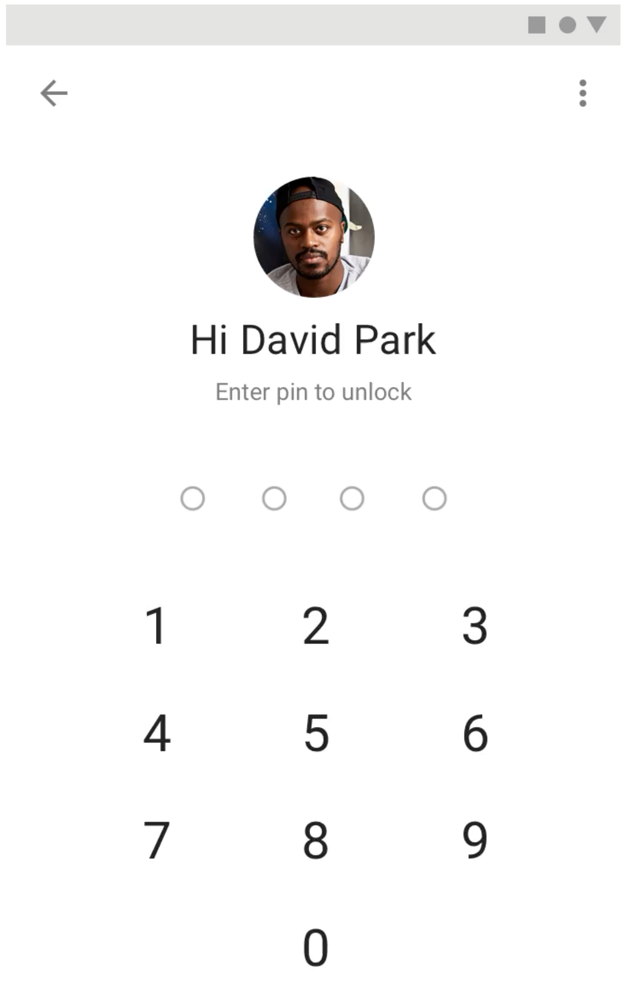
意义


Informative

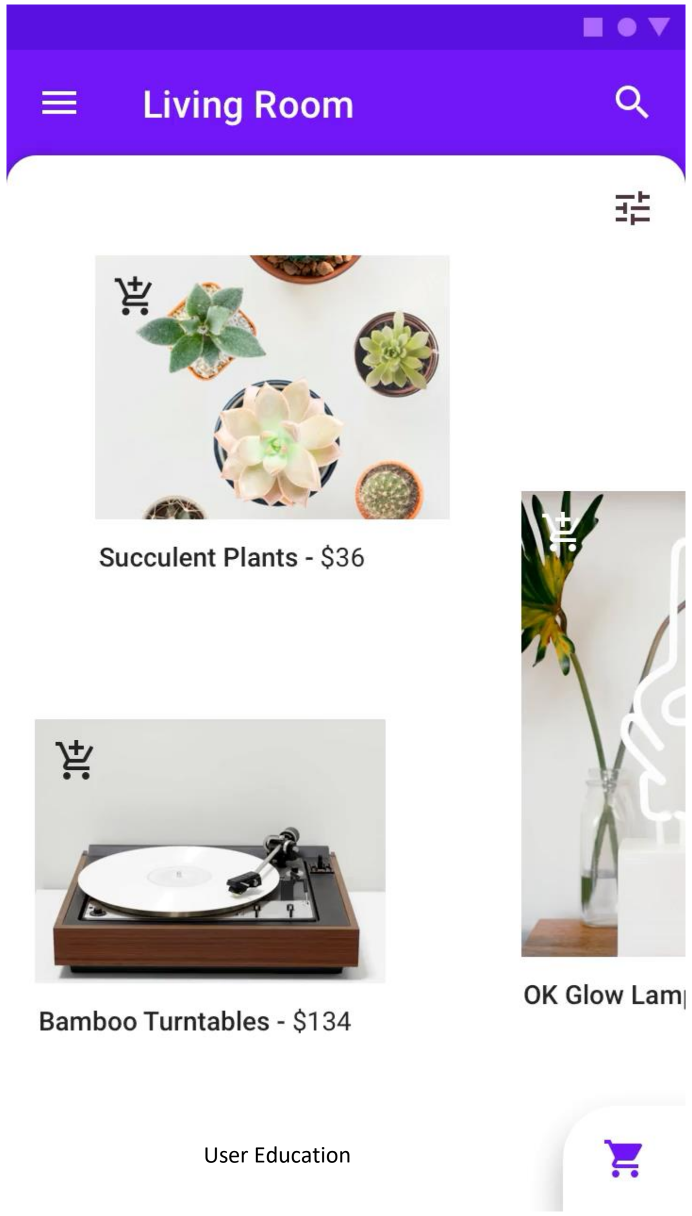
Focused


Expressive

意义



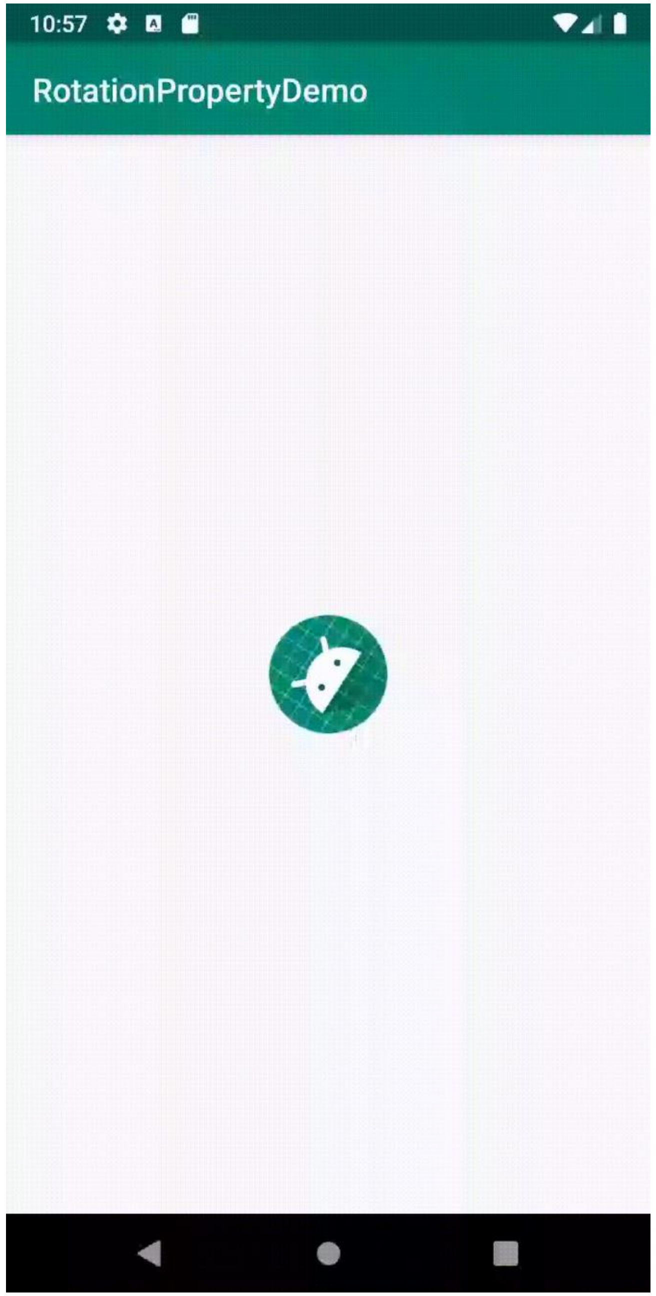
Feedback



User Education

属性动画 - 示例，旋转封面，Code

```
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();
```



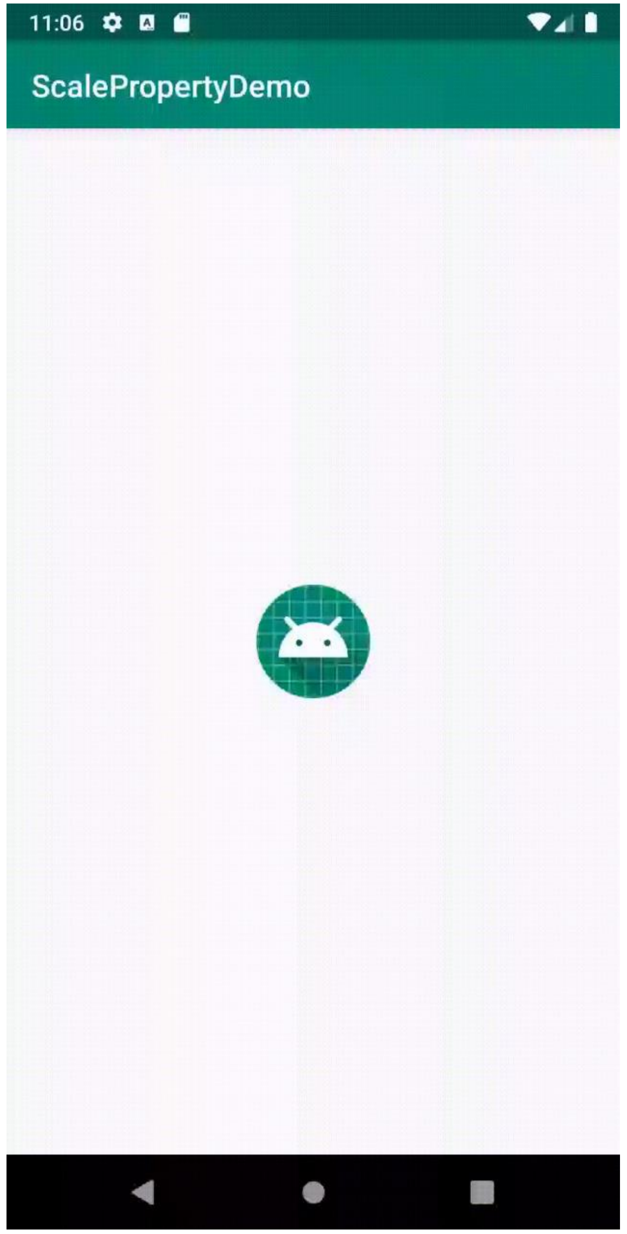
属性动画 - 示例，旋转封面，XML

```
<!— animator/rotate.xml —>
<?xml version="1.0" encoding="utf-8"?>
<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" />
```

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

属性动画 - 示例，呼吸，Code

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

ObjectAnimator scaleYAnimator = ObjectAnimator.ofFloat(imageView,
        "scaleY", 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，呼吸，XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<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>
```

属性动画 - 特点

- Property: alpha, scaleX, scaleY, rotation, rotationX, rotationY, translationX, translationY, …
- ObjectAnimator
  - Duration
  - Interpolator: Linear/AccelerateDecelerate/…
  - Repeat Count and Behavior: Infinite, Restart/Reverse
- AnimatorSet: play together or sequentially

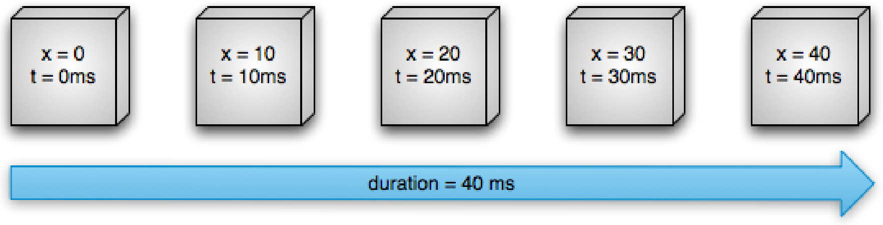
属性动画 - 特点, XML 语法

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

    <objectAnimator
        android:propertyName="string"
        android:duration="int"
        android:interpolator="@[package:]anim/interpolator_resource"
        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>
```

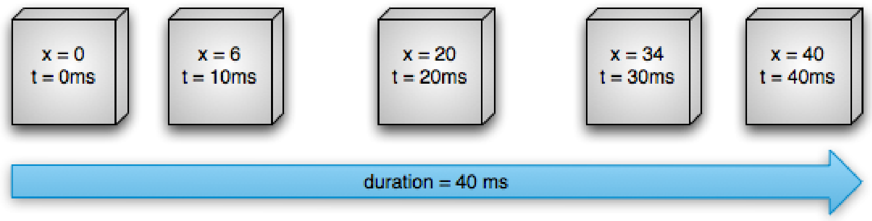
背后 - ValueAnimator，旋转封面

```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();
```

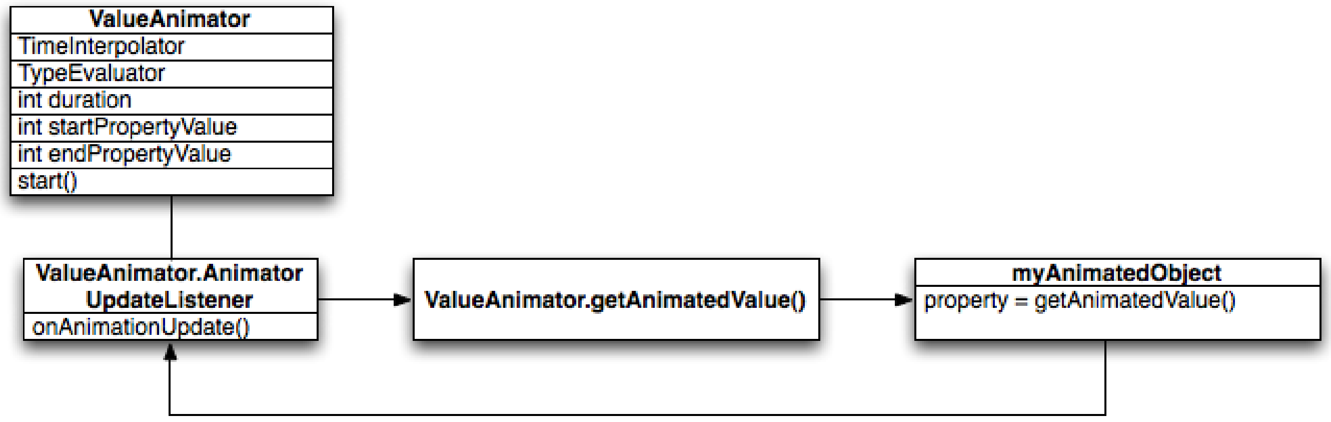
属性动画 - 原理



Linear animation



Nonlinear animation



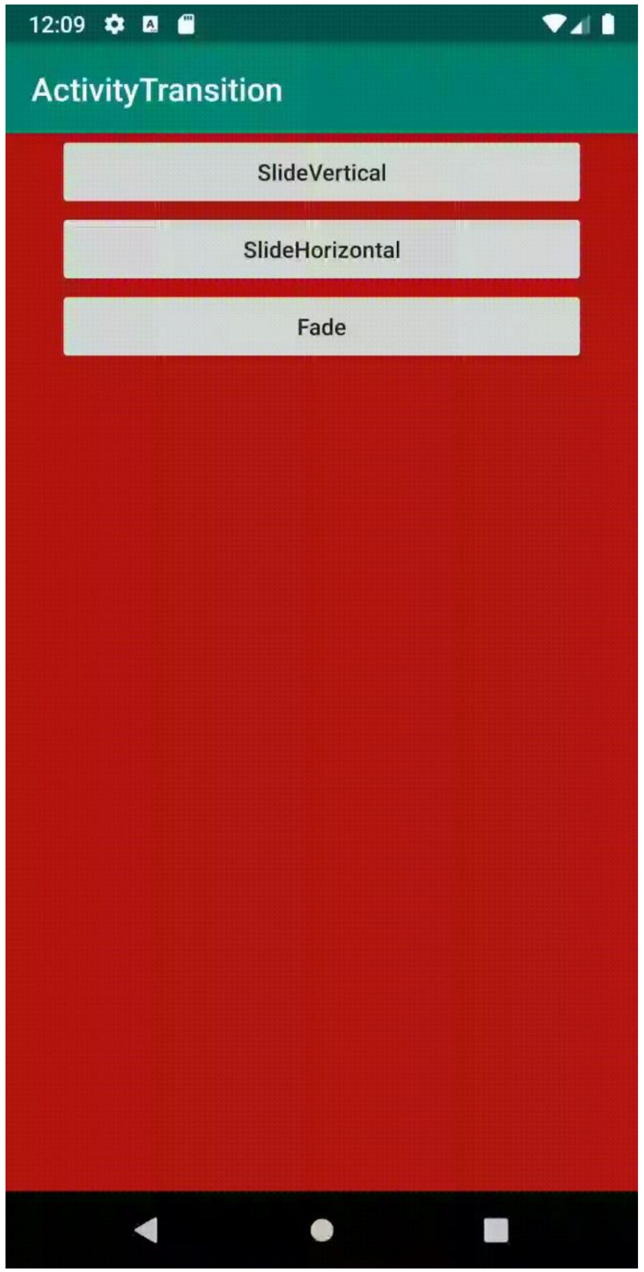
How animations are calculated

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 **的某些属性做动画**
  - **只是视觉效果**

Activity 切换动画

```java
/**
 * 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

```
// 进入动画
startActivity(new Intent(MainActivity.this, activityClass));
overridePendingTransition(android.R.anim.fade_in, android.R.anim.fade_out);

// 退出动画
@Override
public void finish() {
    super.finish();
    overridePendingTransition(android.R.anim.fade_in, android.R.anim.fade_out);
}

// anim/fade_in
alpha xmlns:android="http://schemas.android.com/apk/res/android"
        android:interpolator="@interpolator/decelerate_quad"
        android:fromAlpha="0.0" android:toAlpha="1.0"
        android:duration="@android:integer/config_longAnimTime"  />

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

Drawable 动画

- AnimationDrawable
- AnimationVectorDrawable
- Lottie

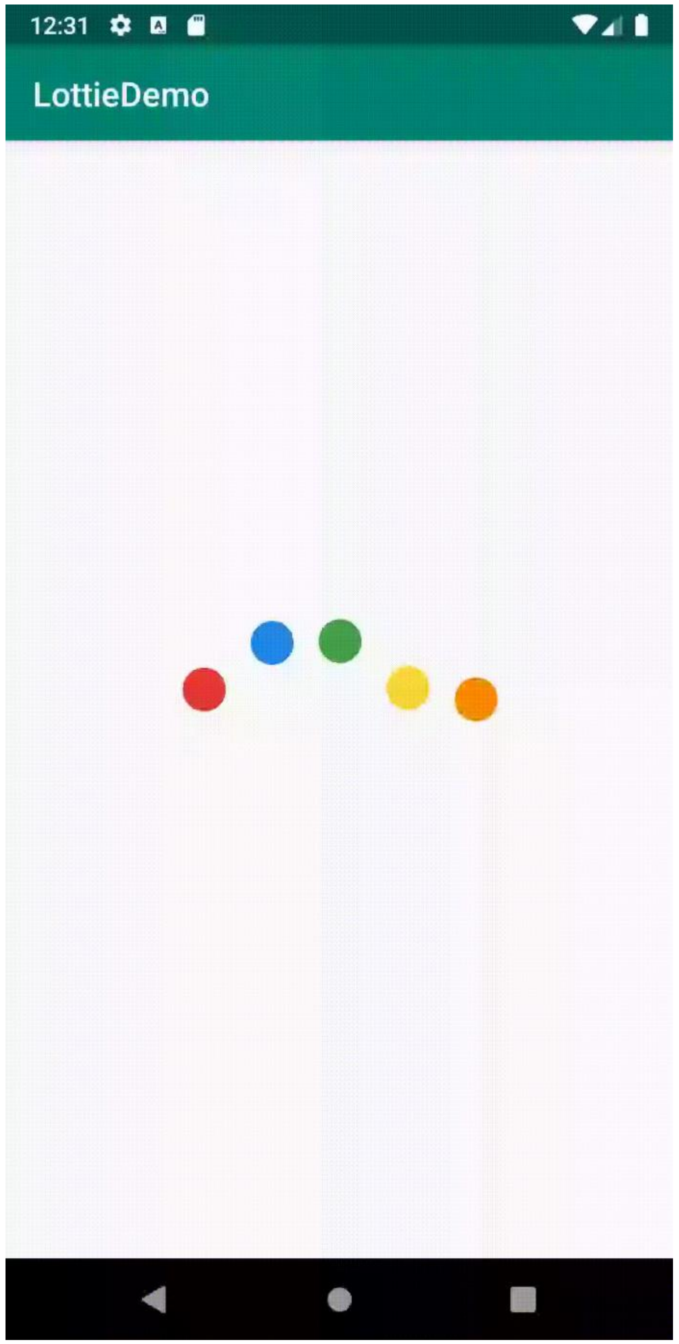
示例 - AnimationDrawable

```
// res/drawable/rocket.xml
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">
    <item android:drawable="@drawable/rocket_thrust1" android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust2" android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust3" android:duration="200" />
</animation-list>

// activity
ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
rocketImage.setBackgroundResource(R.drawable.rocket_thrust);
rocketAnimation = (AnimationDrawable) rocketImage.getBackground();
rocketAnimation.start();
```

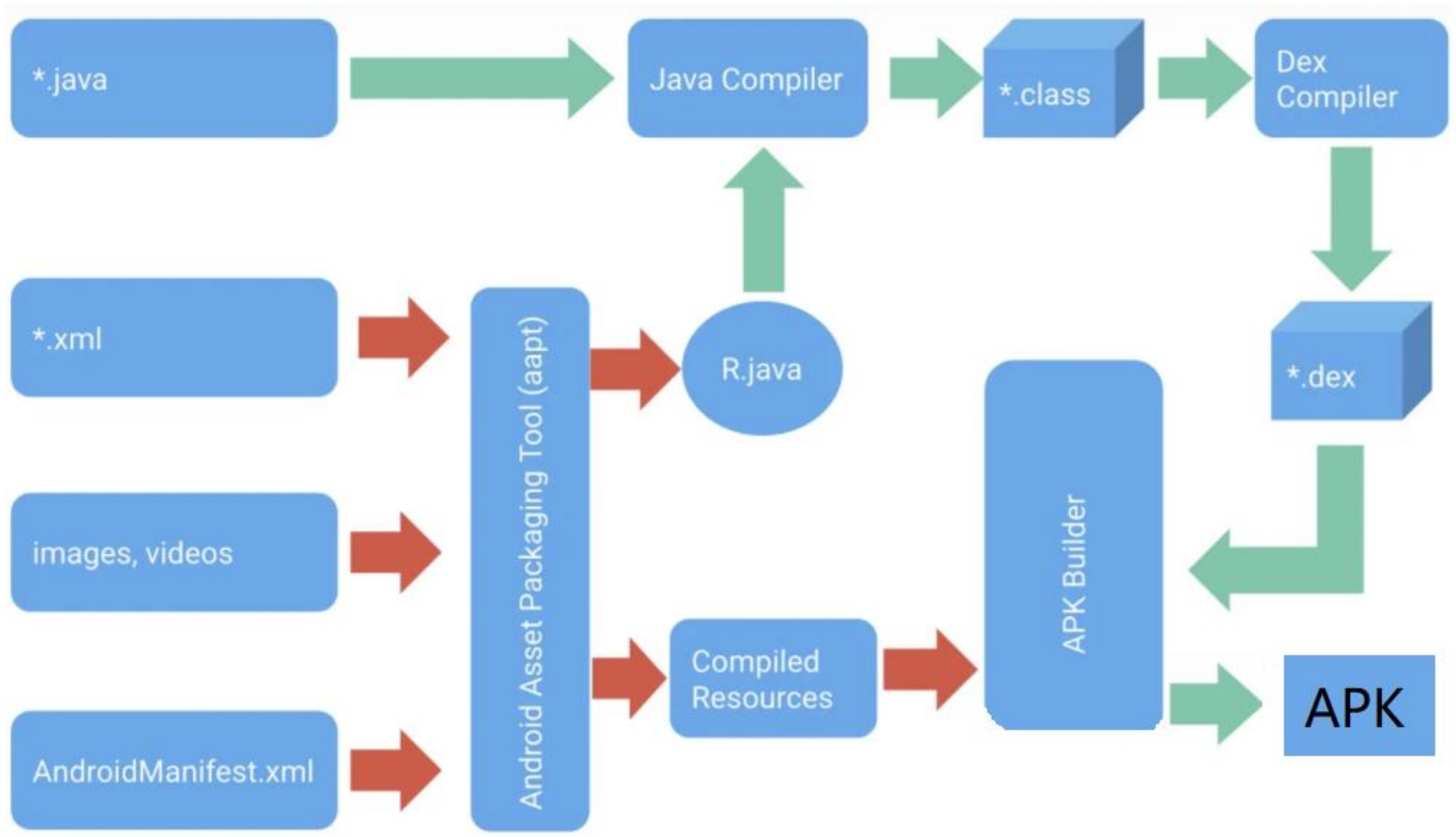
示例 - Lottie

```
// app/build.gradle, 添加依赖
dependencies {
    …
    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" />
```



Android 编译过程



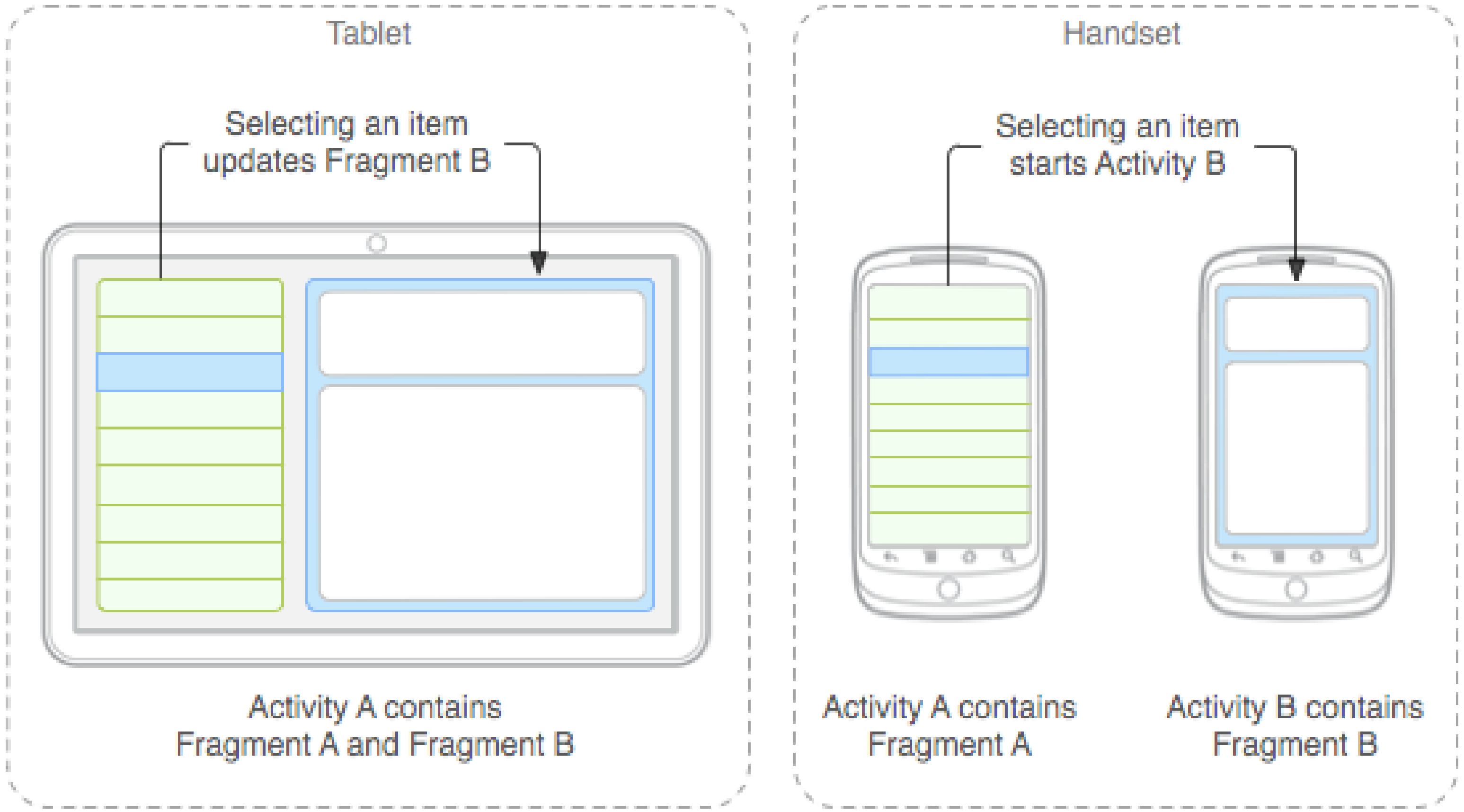
主要内容

- UI 进阶
  - ➢ 动画 Animation
  - ➢ Fragment
  - ➢ 自定义View
- 多线程编程

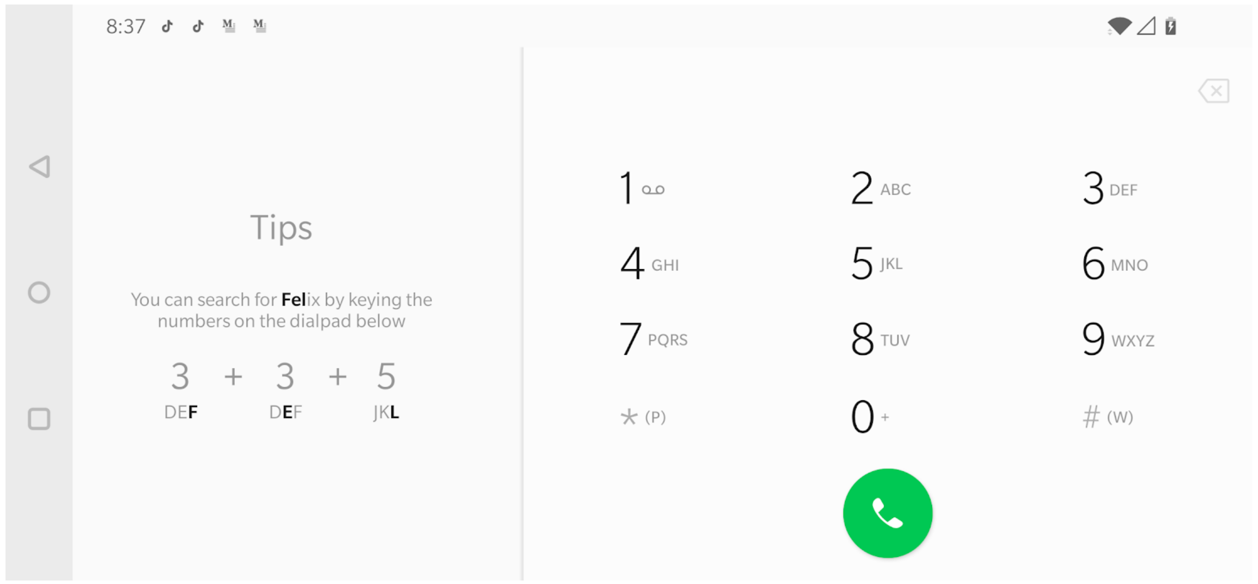
Fragment

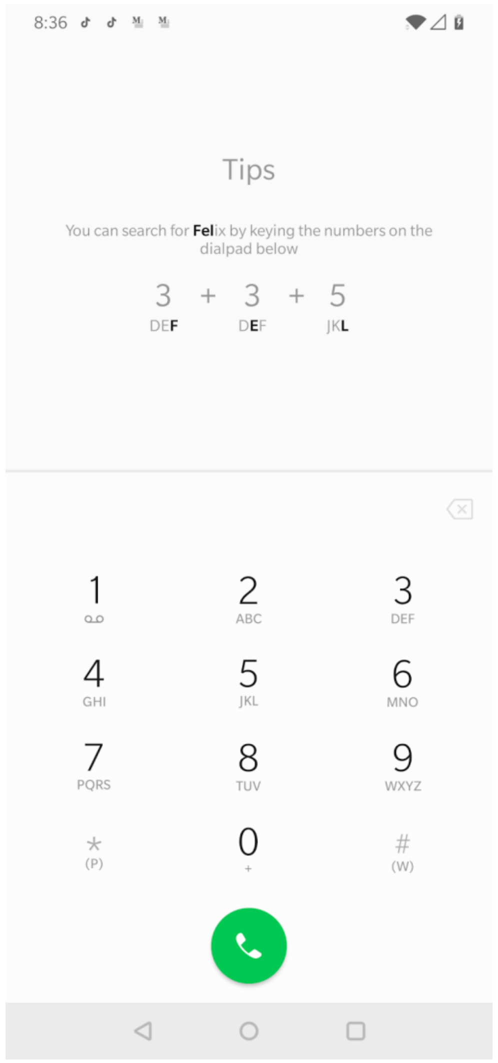
- 概念和作用
- 生命周期和基本用法
- 结合 ViewPager 创建多 Tab 界面
- 如何和 Activity 通信

Fragment - UI 重用

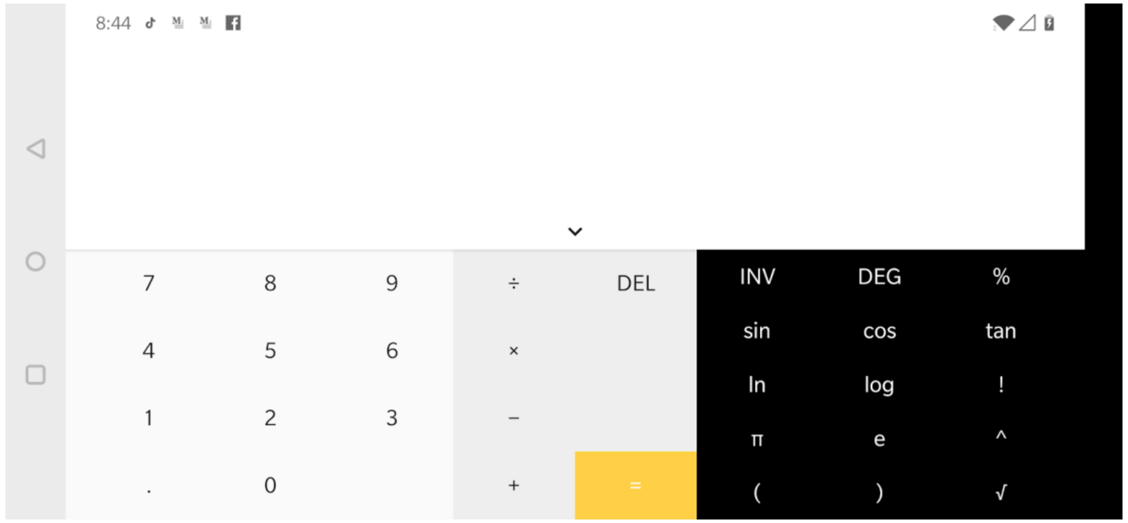

Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

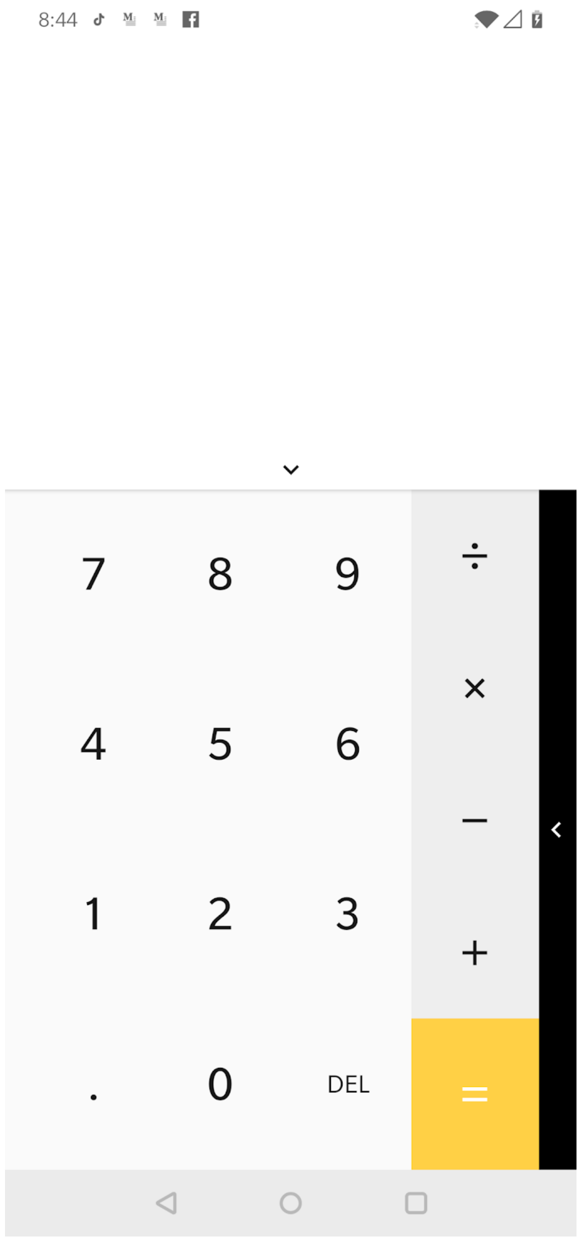
Fragment - Responsive Design





Fragment - Responsive Design





Fragment - Why

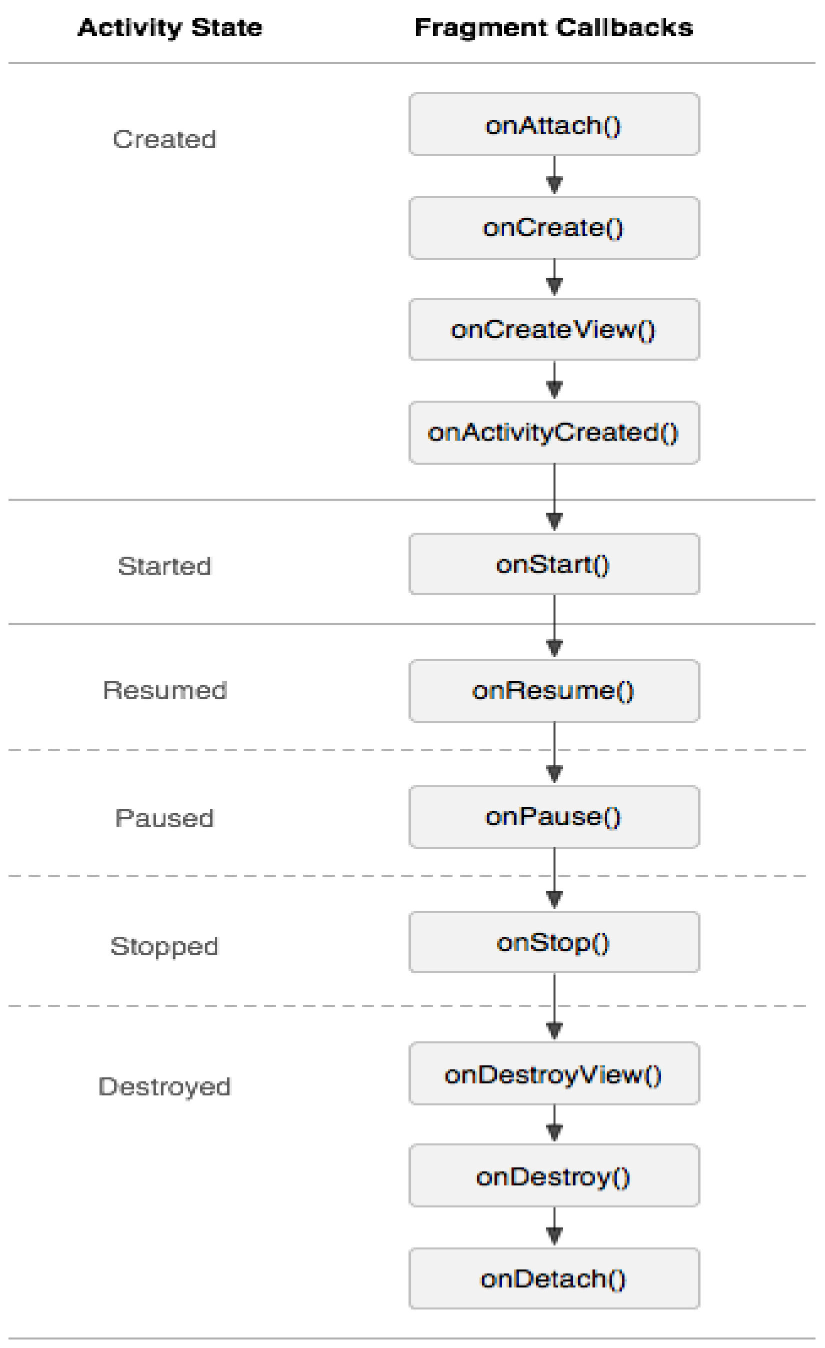
- Activity 模块化
- 相比 View，带有生命周期管理
- 可重用，灵活

使用哪个?

Library	Package
Support Library	android.support.v4.app.Fragment
AndroidX Library	androidx.fragment.app.Fragment
Native	android.app.Fragment

生命周期

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



示例 - Lifecycle

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

示例 - Lifecycle - 1

- fragment_hello.xml

```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">

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

示例 - Lifecycle - 2

- **HelloFragment.java**

```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 - Lifecycle - 3

- activity_fragment.xml

```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

- 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

```
<?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">

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

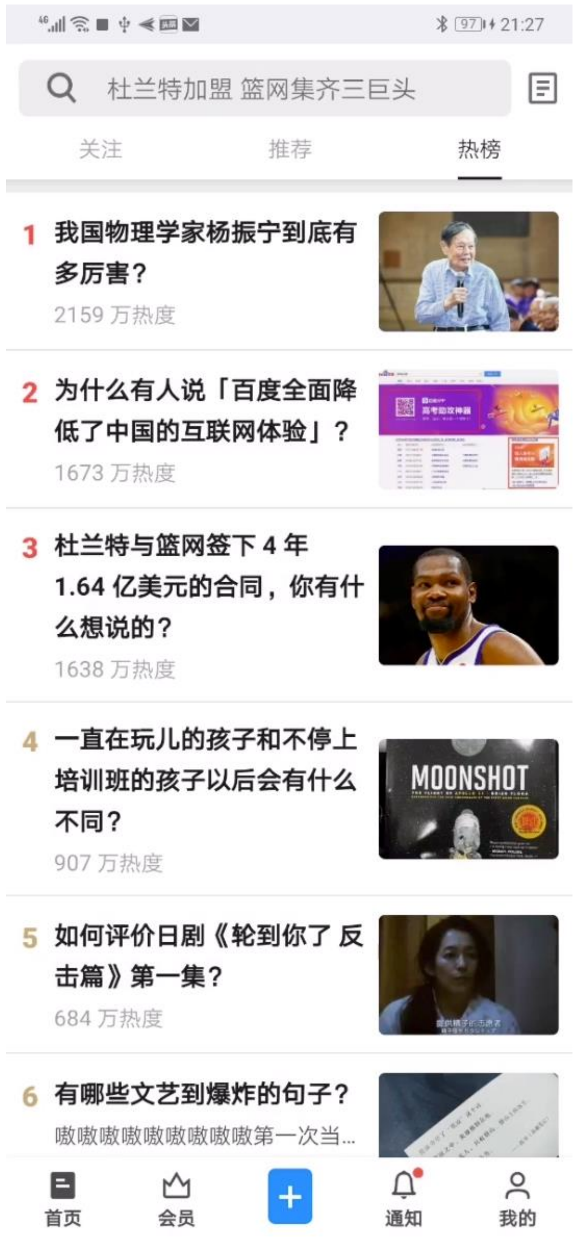
示例 - 动态修改 Fragment - 2

- 使用 FragmentManager 添加 Fragment

```
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();
    }
}
```

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 来实现
- 可配合 TabLayout 或三方库添加 Title



21:27

杜兰特加盟 篮网集齐三巨头

关注 推荐 热榜

1 我国物理学家杨振宁到底有多厉害？
2159 万热度

2 为什么有人说「百度全面降低了中国的互联网体验」？
1673 万热度

3 杜兰特与篮网签下 4 年 1.64 亿美元的合同，你有什么想说的？
1638 万热度

4 一直在玩儿的孩子和不停上培训班的孩子以后会有什么不同？
907 万热度

5 如何评价日剧《轮到你了 反击篇》第一集？
684 万热度

6 有哪些文艺到爆炸的句子？
嗷嗷嗷嗷嗷嗷嗷嗷嗷第一次当...

首页 会员 通知 我的

示例 - ViewPager - 1

- 在布局 xml 中添加 ViewPager

```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.support.v4.view.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

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

   <android.support.design.widget.TabLayout
      android:id="@+id/tab_layout"
      android:layout_width="match_parent"
      android:layout_height="40dp" />

   <android.support.v4.view.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

```
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

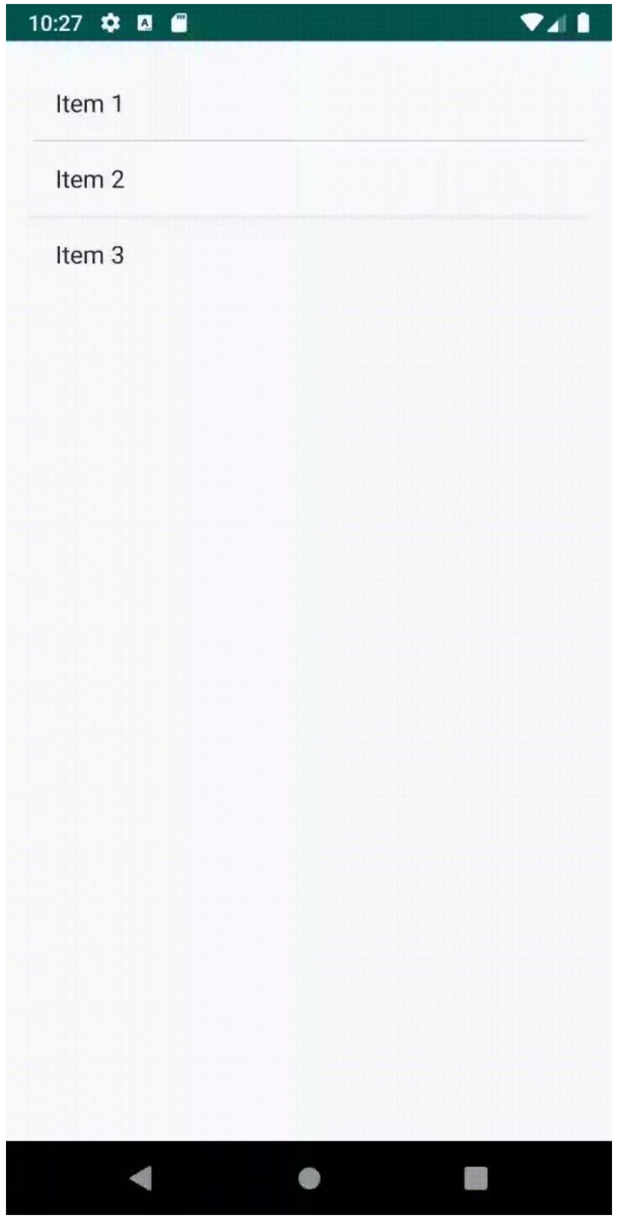
```java
public final class ColorPlusFragment extends Fragment {

    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;
        }
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
            @Nullable ViewGroup container,
            @Nullable Bundle savedInstanceState) {
        ...
        // fire event when needed
        if (mListener != null) {
            mListener.onCollectColor(color);
        }
        return view;
    }
}
```

示例 - 通信 - Listener - 2

```
public class ViewPagerCommunicationActivity extends AppCompatActivity
        implements ColorPlusFragment.Listener {

    …

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

    …
}
```

示例 - Master Detail

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



Android Developer

- 开发者官网：https://developer.android.com
- 开发者官网中文站：https://developer.android.google.cn/

Android Studio

- AS 快捷键：https://developer.android.google.cn/studio/intro/keyboard-shortcuts
- AS 用户指南：https://developer.android.google.cn/studio/intro

Animation

- Material Design - Motion: https://material.io/design/motion/
- Property Animation: https://developer.android.google.cn/guide/topics/graphics/prop-animation

Animation

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

Fragment

- Fragment Overview:
  https://developer.android.google.cn/guide/components/fragments

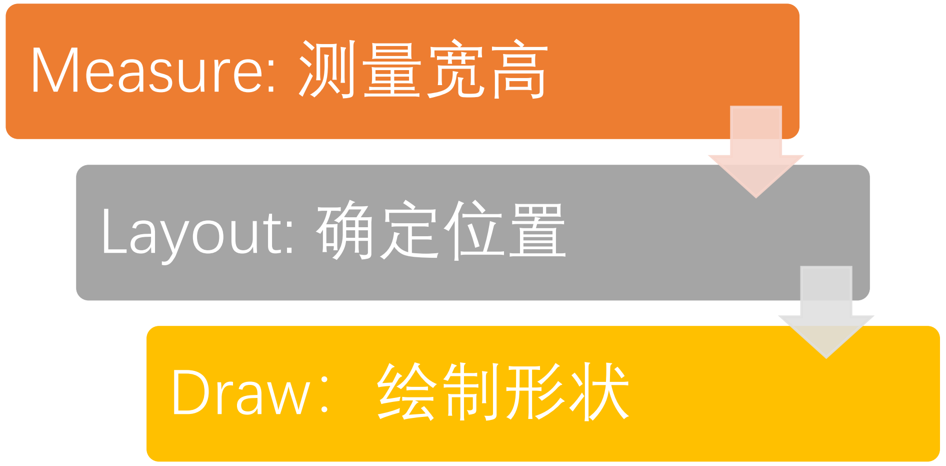
???

- getCount
- interface 回调

主要内容

- UI 进阶
  - 动画 Animation
  - Fragment
  - 自定义View
- 多线程编程

View绘制的三个重要步骤



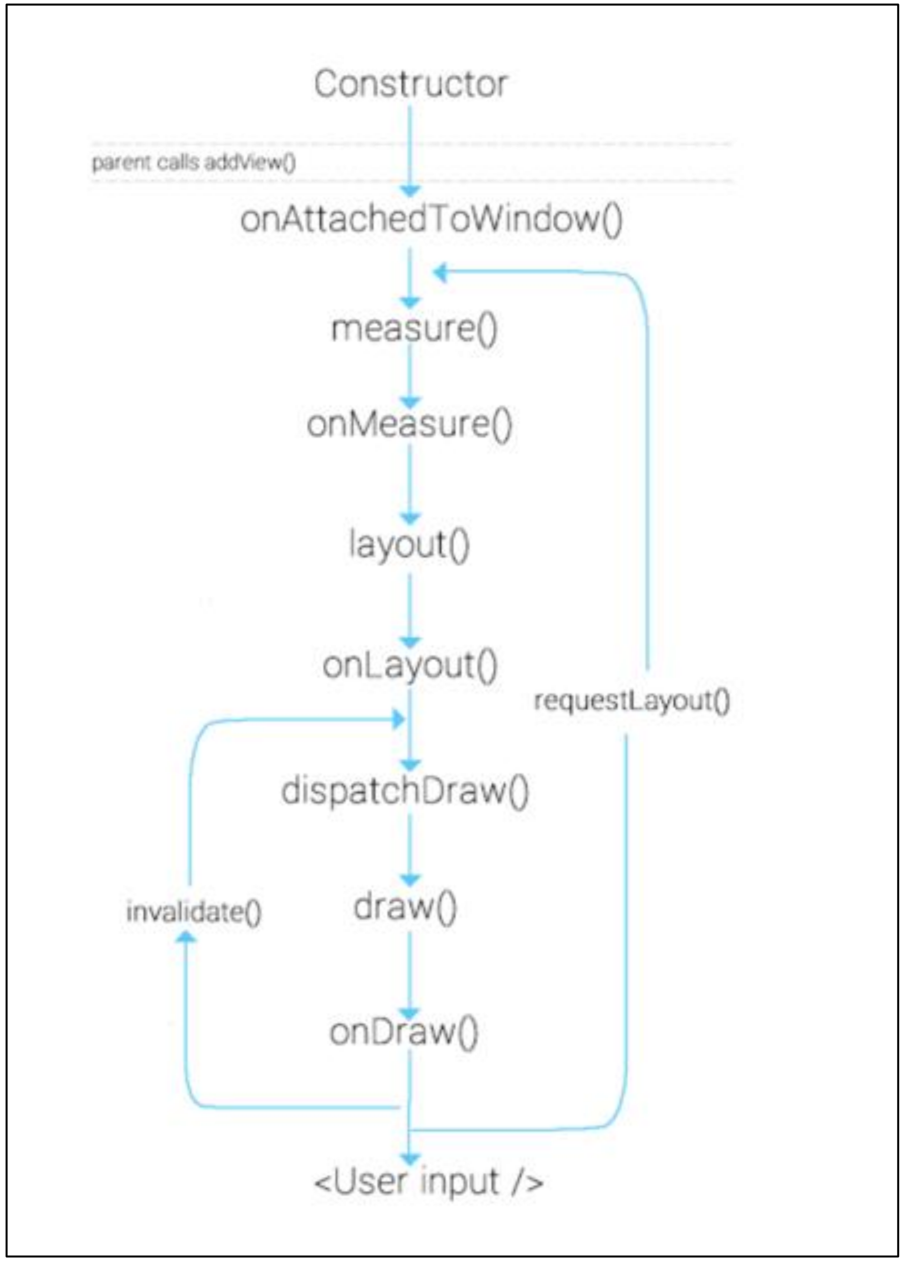
举例说明：

1. **首先画一个100 x 100的照片框，需要尺子测量出宽高的长度（measure过程）**

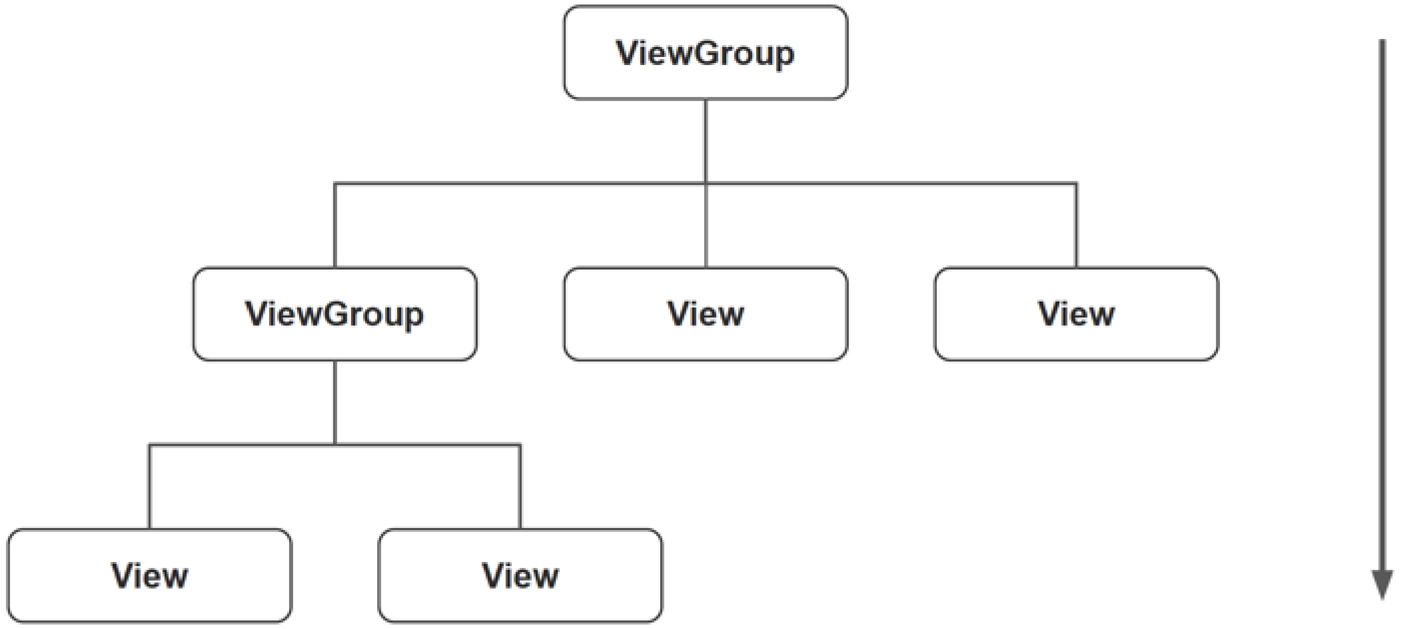
2. **然后确定照片框在屏幕中的位置（layout过程）**

3. **最后借助尺子用手画出我们的照片框（draw过程）**

View绘制的三个重要步骤

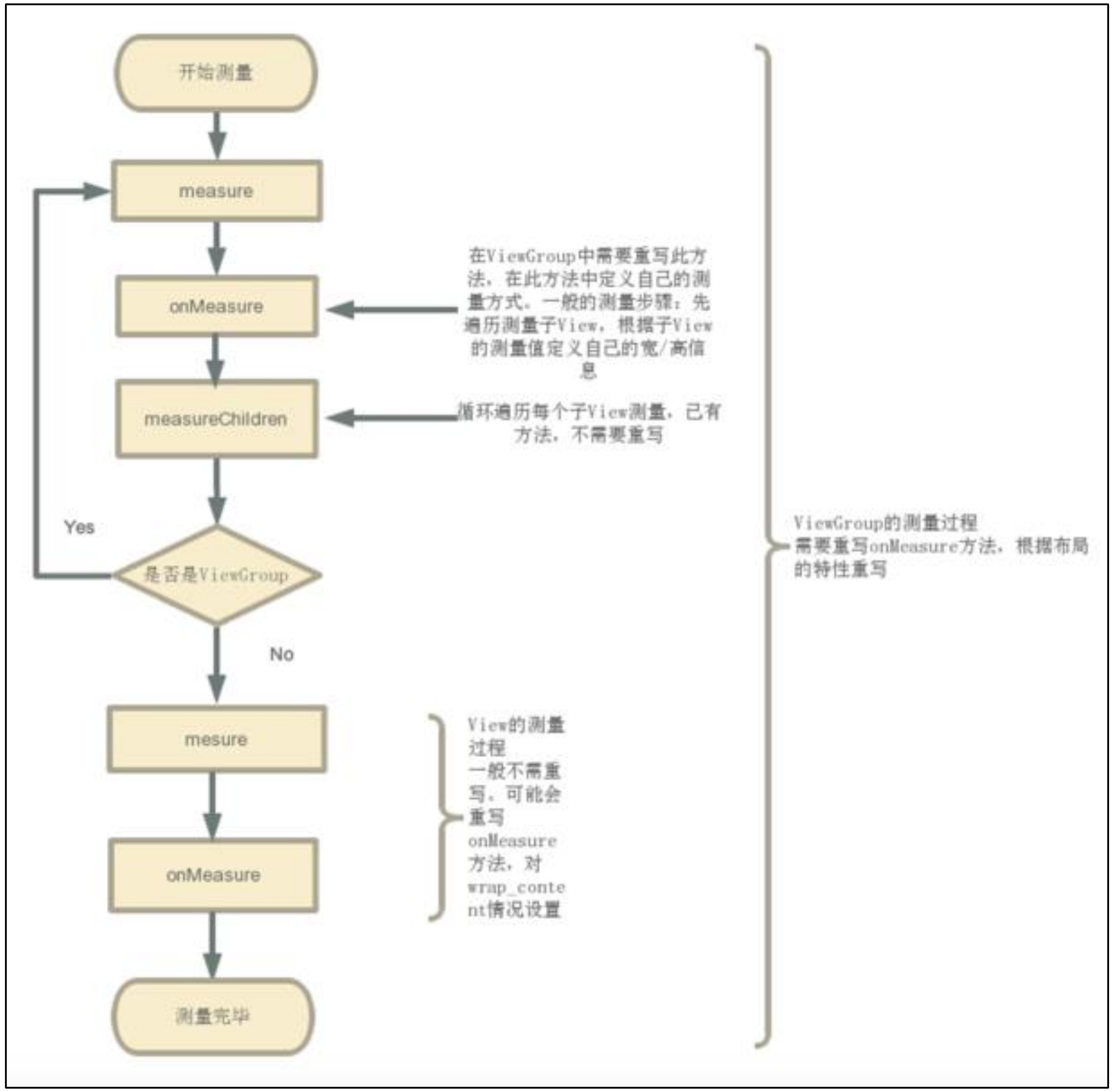


扩展：详解 ViewTree 及 View / ViewGroup 绘制流程

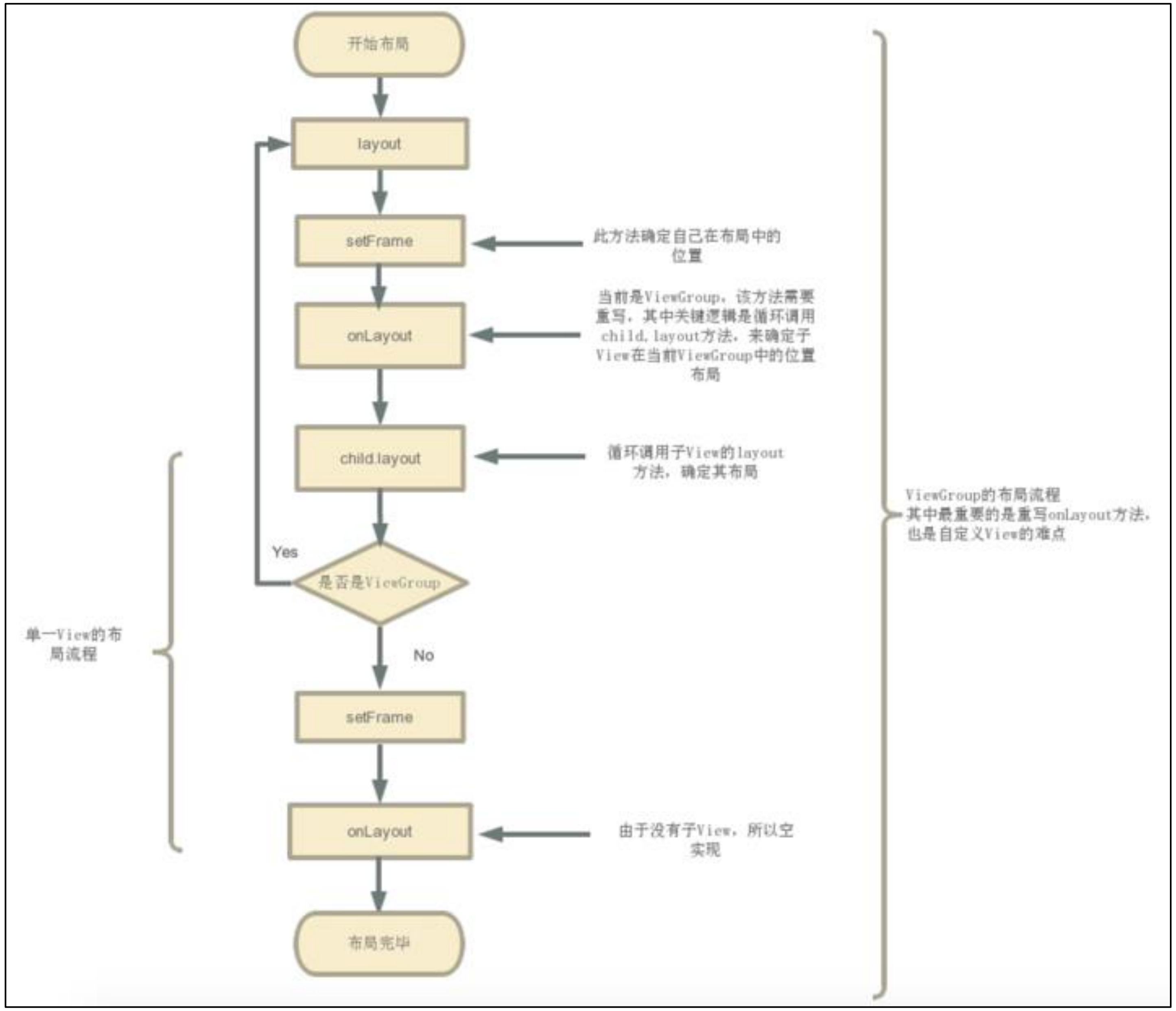


自上而下递归进行:
Measure
Layout
Draw

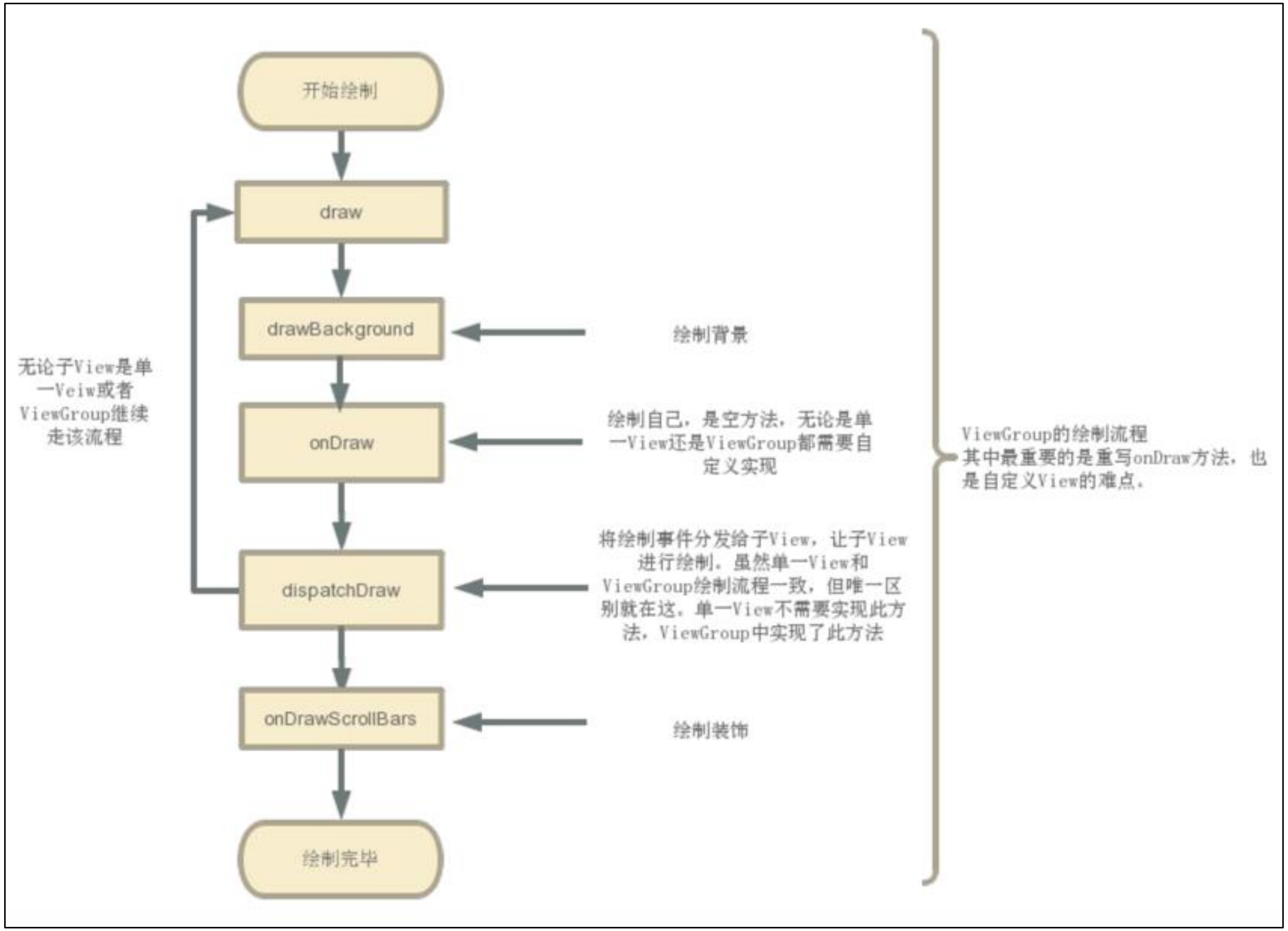
扩展：ViewGroup 绘制


开始测量
measure
onMeasure
在ViewGroup中需要重写此方法，在此方法中定义自己的测量方式。一般的测量步骤：先遍历测量子View，根据子View的测量值定义自己的宽/高信息
measureChildren
循环遍历每个子View测量，已有方法，不需要重写
Yes
是否是ViewGroup
No
mesure
onMeasure
View的测量过程
一般不需重写，可能会重写onMeasure方法，对wrap_content情况设置
测量完毕
ViewGroup的测量过程
需要重写onMeasure方法，根据布局的特性重写

扩展：ViewGroup 绘制


开始布局
layout
setFrame
此方法确定自己在布局中的位置
onLayout
当前是ViewGroup，该方法需要重写，其中关键逻辑是循环调用child.layout方法，来确定子View在当前ViewGroup中的位置布局
child.layout
循环调用子View的layout方法，确定其布局
Yes
是否是ViewGroup
No
单一View的布局流程
setFrame
onLayout
由于没有子View，所以空实现
布局完毕
ViewGroup的布局流程 其中最重要的是重写onLayout方法，也是自定义View的难点

ViewGroup绘制



无论子View是单一Veiw或者ViewGroup继续走该流程

绘制背景

绘制自己，是空方法，无论是单一View还是ViewGroup都需要自定义实现

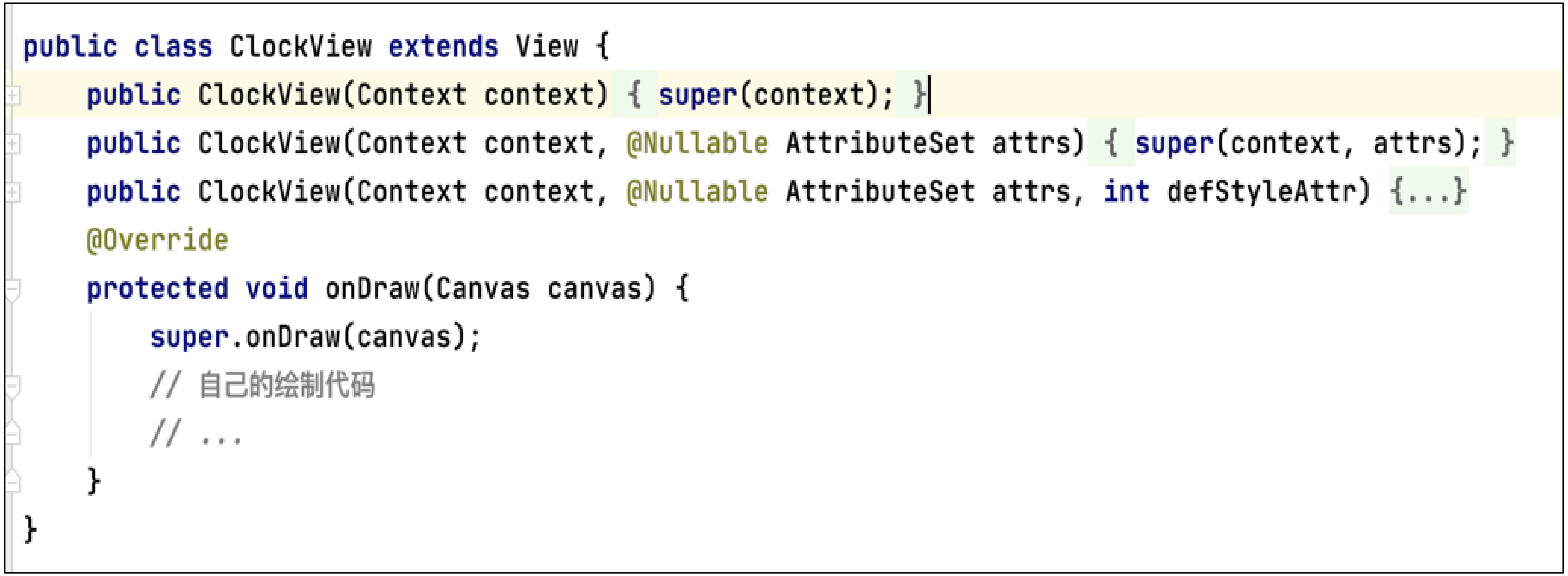
将绘制事件分发给子View，让子View进行绘制。虽然单一View和ViewGroup绘制流程一致，但唯一区别就在这。单一View不需要实现此方法，ViewGroup中实现了此方法

绘制装饰

ViewGroup的绘制流程
其中最重要的是重写onDraw方法，也是自定义View的难点。

自定义View-重写onDraw

自定义View最常见操作 - 重写onDraw

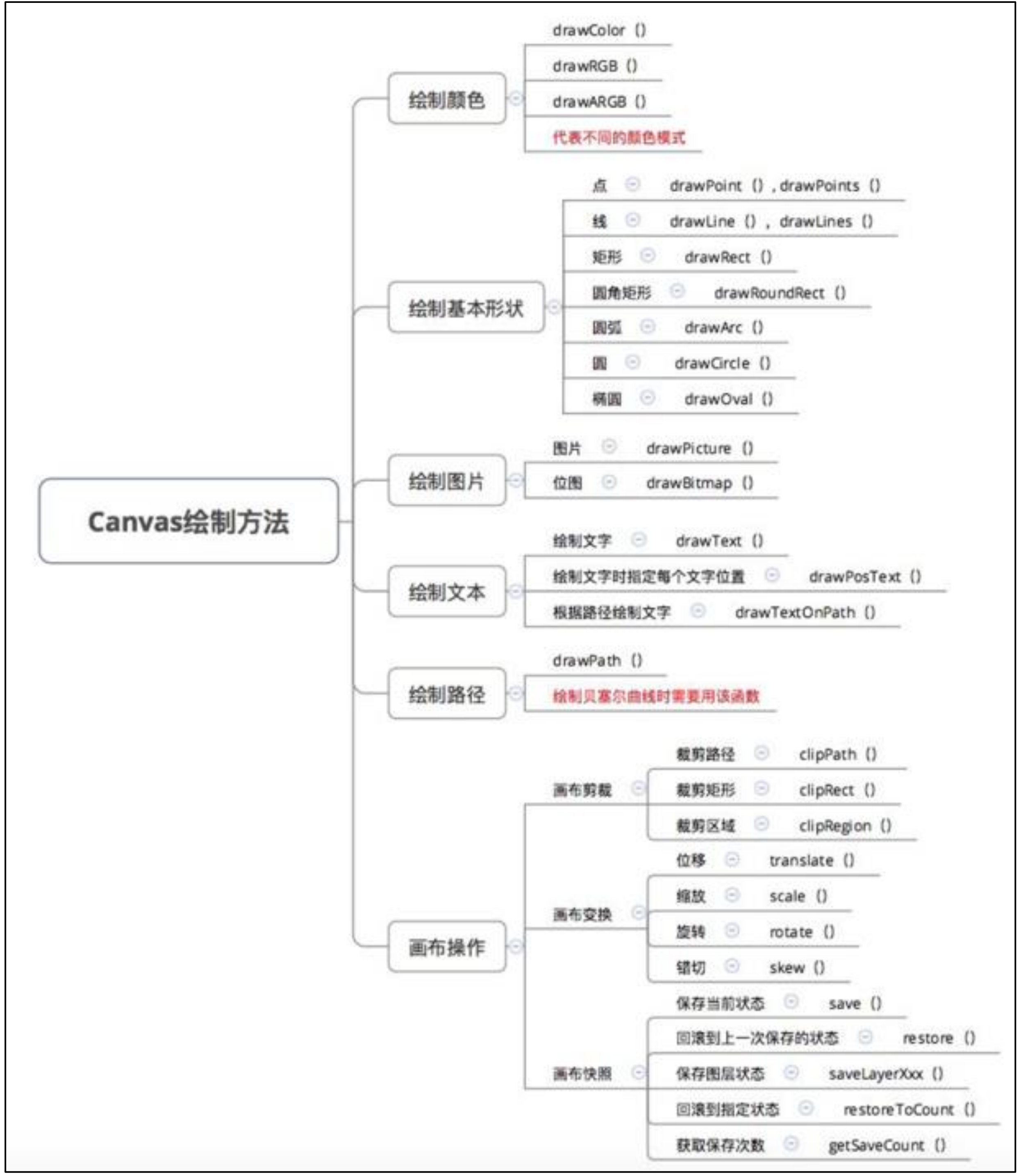


```java
public class ClockView extends View {
    public ClockView(Context context) { super(context); }
    public ClockView(Context context, @Nullable AttributeSet attrs) { super(context, attrs); }
    public ClockView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {...}
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        // 自己的绘制代码
        // ...
    }
}
```

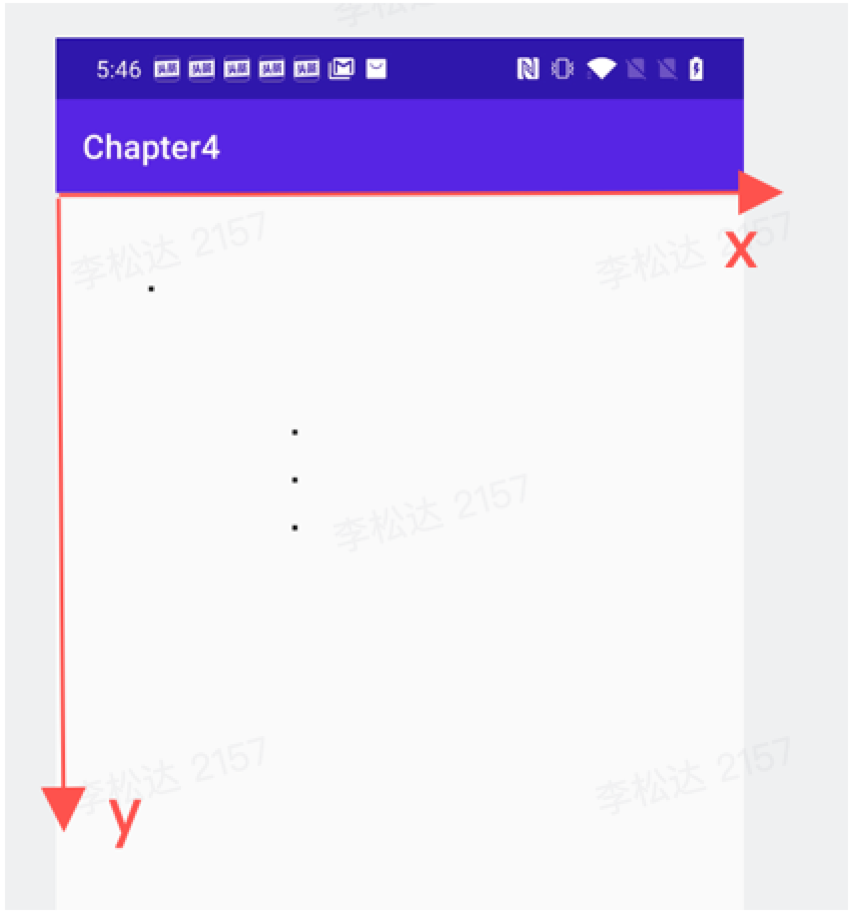
自定义View-重写onDraw

自定义View最常见操作 - 重写onDraw

概念解析：
1. Canvas：画布
2. Paint：画笔

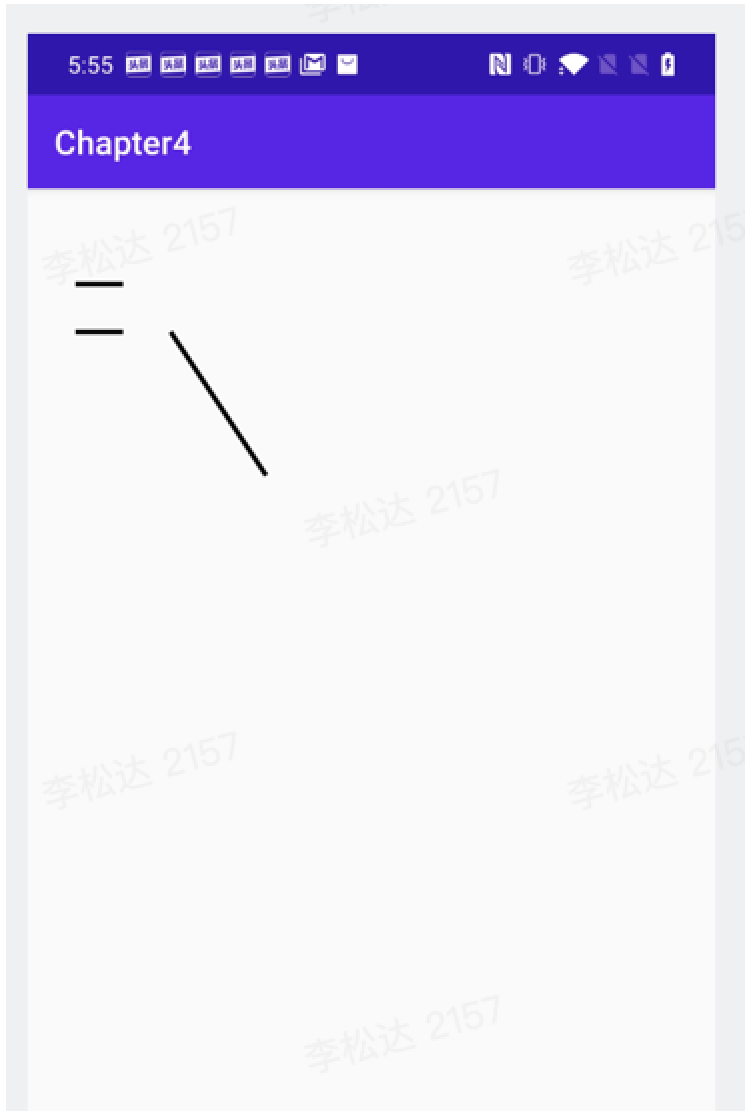


View绘制-点



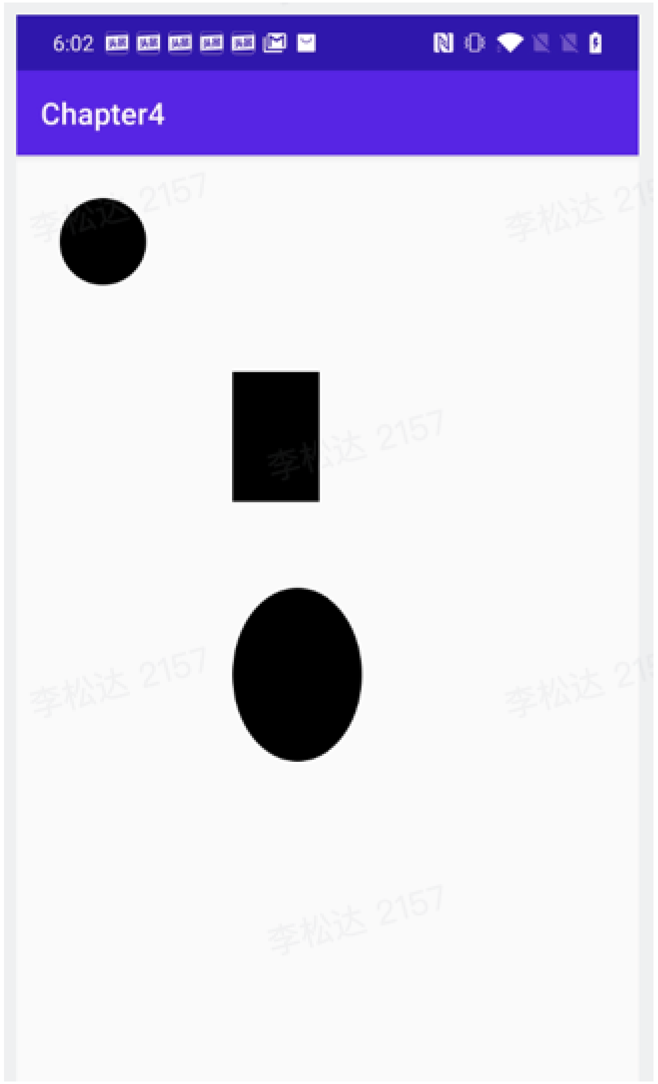
```java
public class CustomView extends View {
    private Paint mPaint;
    public CustomView(Context context) {
        super(context);
        init();
    }
    public CustomView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        init();
    }
    public CustomView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }
    private void init() {
        mPaint = new Paint();
        mPaint.setColor(Color.BLACK);
        mPaint.setStyle(Paint.Style.FILL);
        mPaint.setAntiAlias(true);
        mPaint.setStrokeWidth(10f);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawPoint( x: 200,  y: 200, mPaint);
        canvas.drawPoints(new float[]{
                500, 500,
                500, 600,
                500, 700
        }, mPaint);
    }
}
```

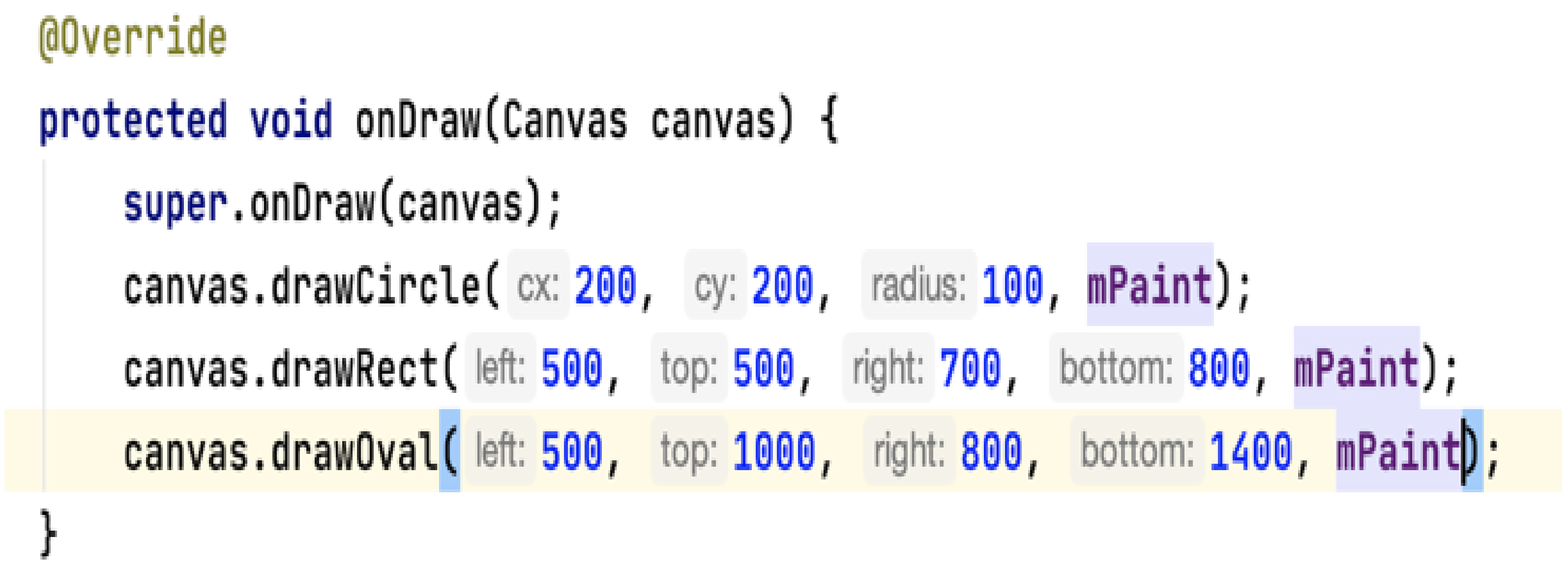
View绘制-线



```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.drawLine( startX: 300,  startY: 300,  stopX: 500,  stopY: 600, mPaint);
    canvas.drawLines(new float[]{
            100, 200, 200, 200,
            100, 300, 200, 300
    }, mPaint);
}
```

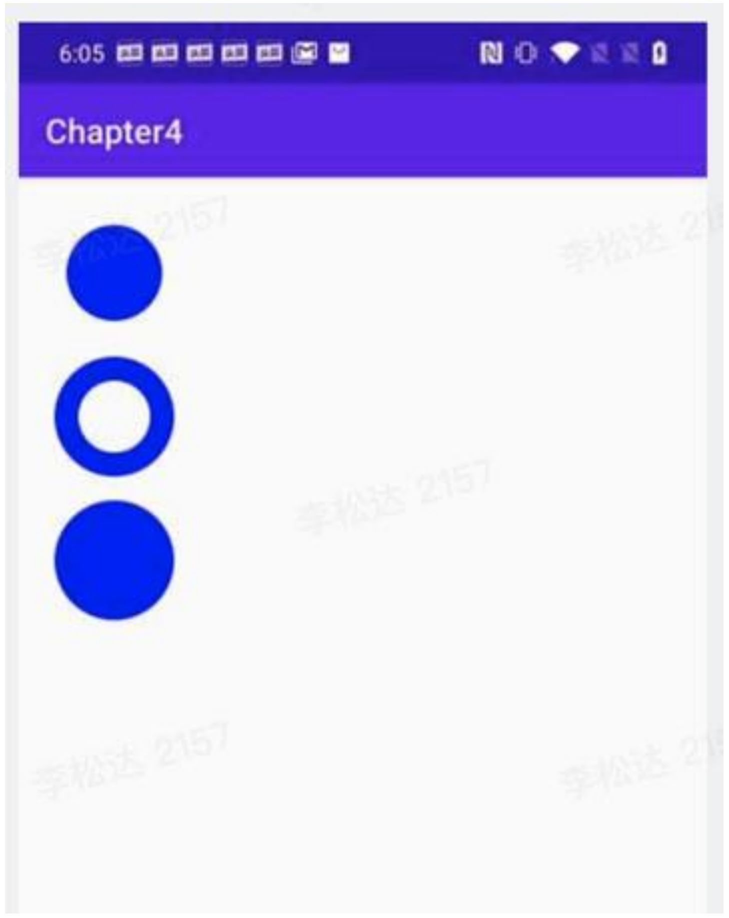
View绘制-圆

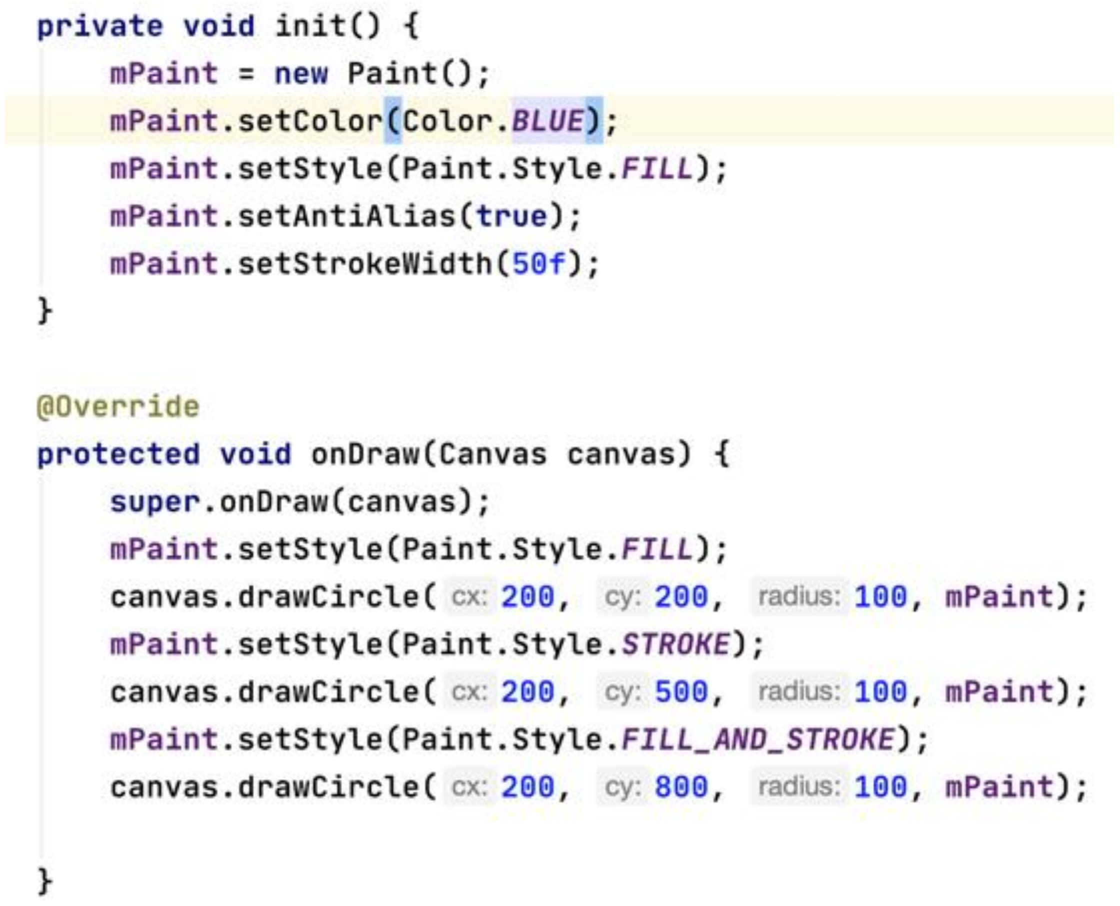




```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.drawCircle( cx: 200,  cy: 200,  radius: 100, mPaint);
    canvas.drawRect( left: 500,  top: 500,  right: 700,  bottom: 800, mPaint);
    canvas.drawOval( left: 500,  top: 1000,  right: 800,  bottom: 1400, mPaint);
}
```

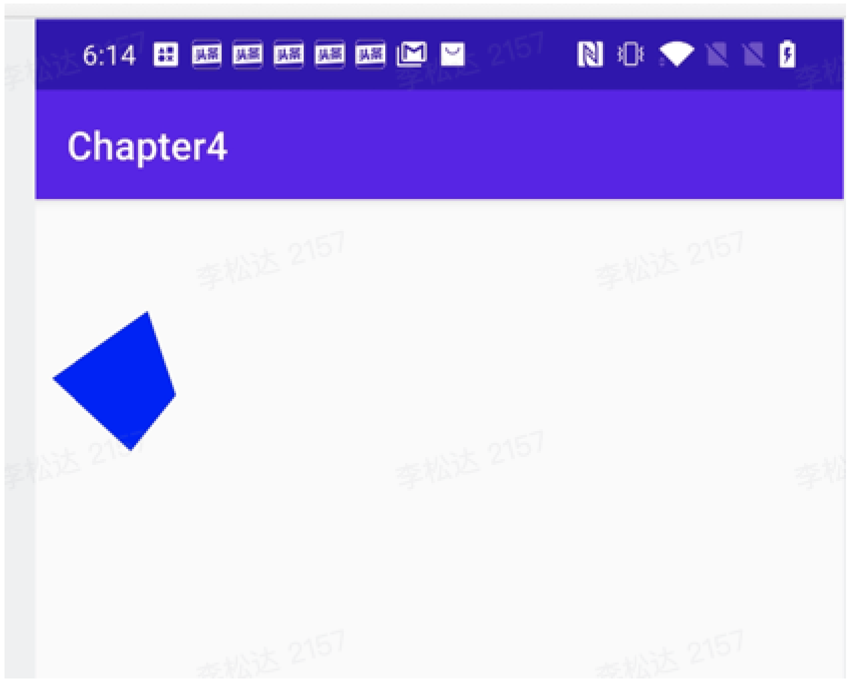
View绘制-填充

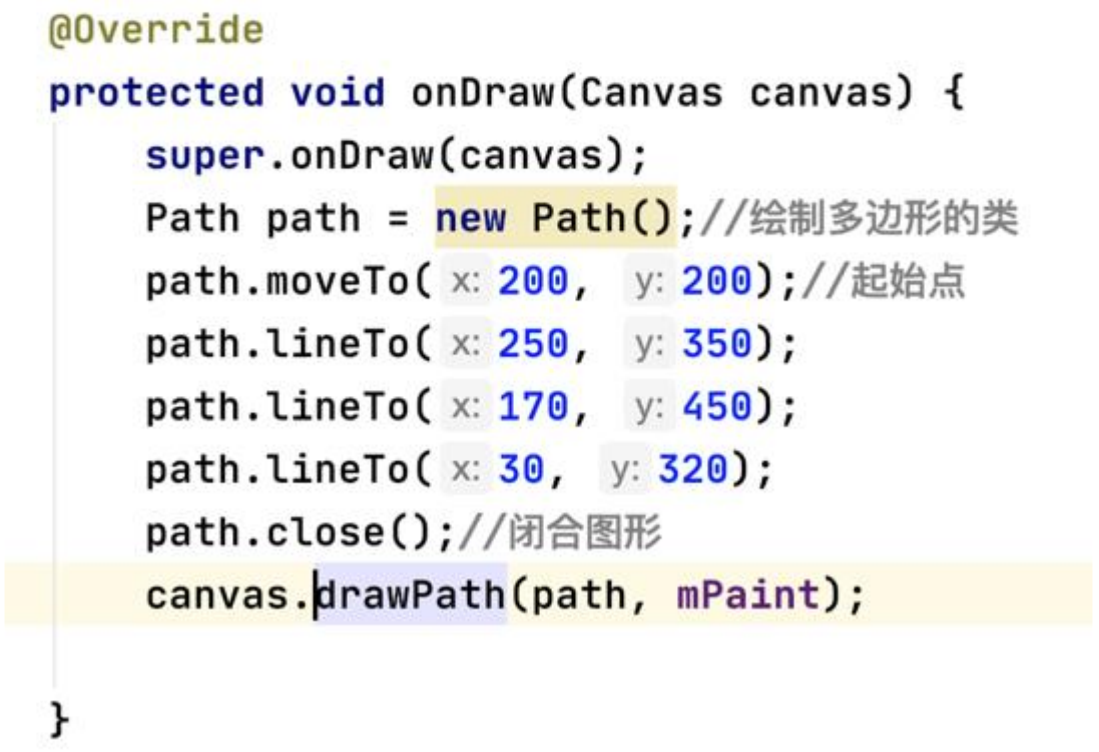




```
private void init() {
    mPaint = new Paint();
    mPaint.setColor(Color.BLUE);
    mPaint.setStyle(Paint.Style.FILL);
    mPaint.setAntiAlias(true);
    mPaint.setStrokeWidth(50f);
}

@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    mPaint.setStyle(Paint.Style.FILL);
    canvas.drawCircle( cx: 200,  cy: 200,  radius: 100, mPaint);
    mPaint.setStyle(Paint.Style.STROKE);
    canvas.drawCircle( cx: 200,  cy: 500,  radius: 100, mPaint);
    mPaint.setStyle(Paint.Style.FILL_AND_STROKE);
    canvas.drawCircle( cx: 200,  cy: 800,  radius: 100, mPaint);

}
```

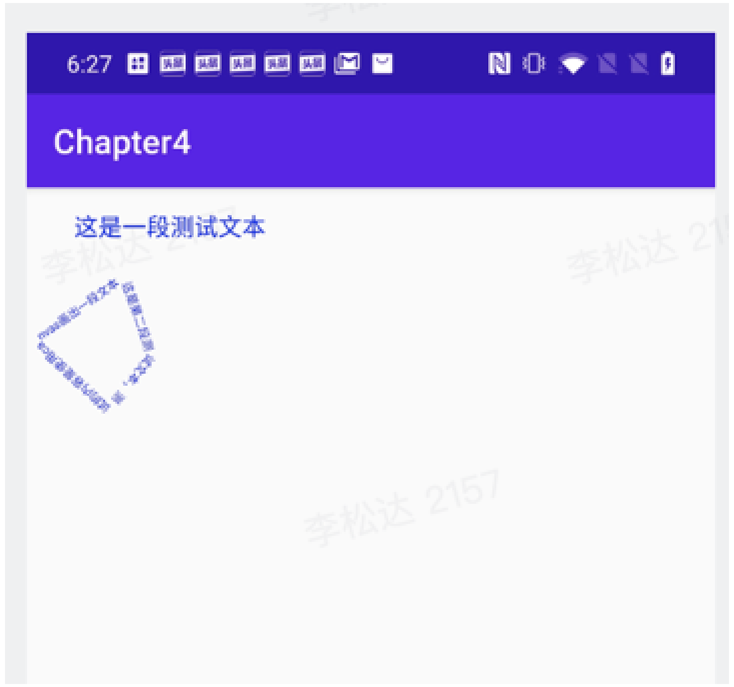
View绘制-不规则图形

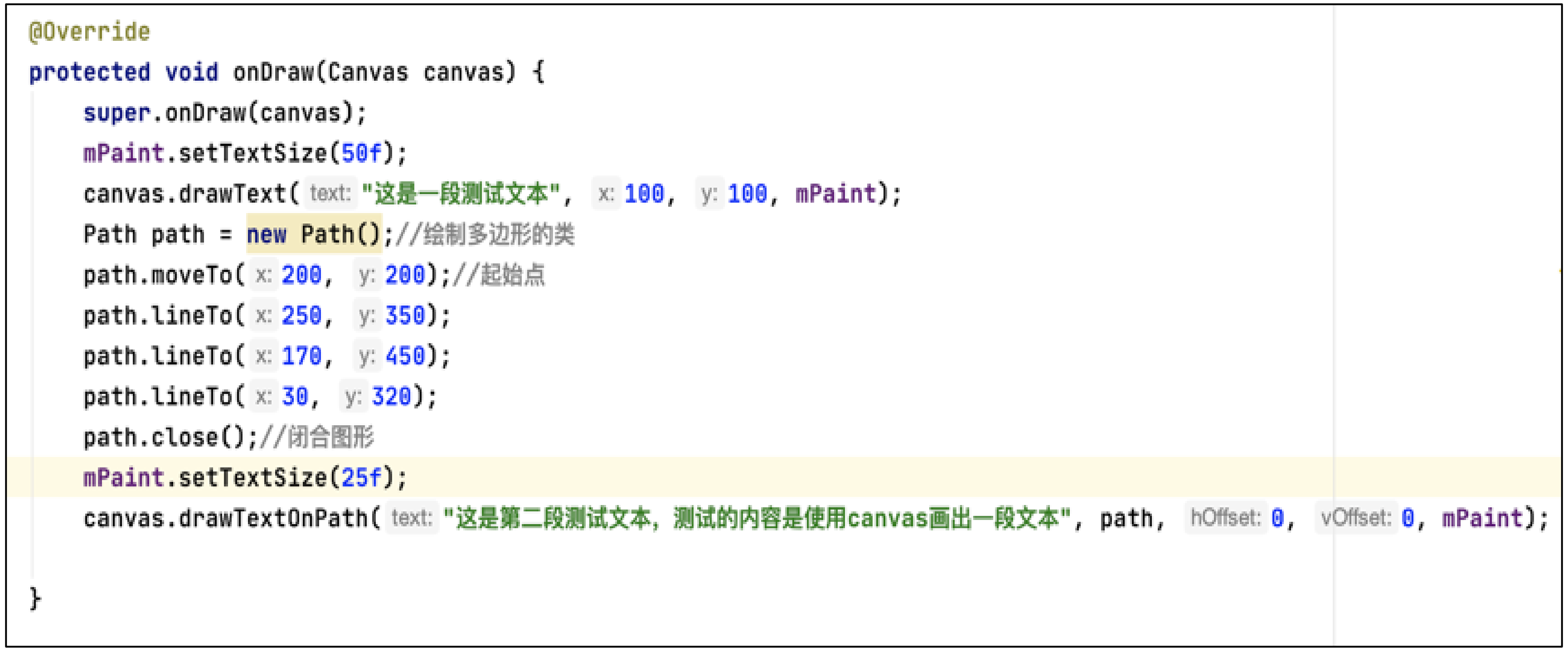




```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Path path = new Path();//绘制多边形的类
    path.moveTo( x: 200,  y: 200);//起始点
    path.lineTo( x: 250,  y: 350);
    path.lineTo( x: 170,  y: 450);
    path.lineTo( x: 30,  y: 320);
    path.close();//闭合图形
    canvas.drawPath(path, mPaint);

}
```

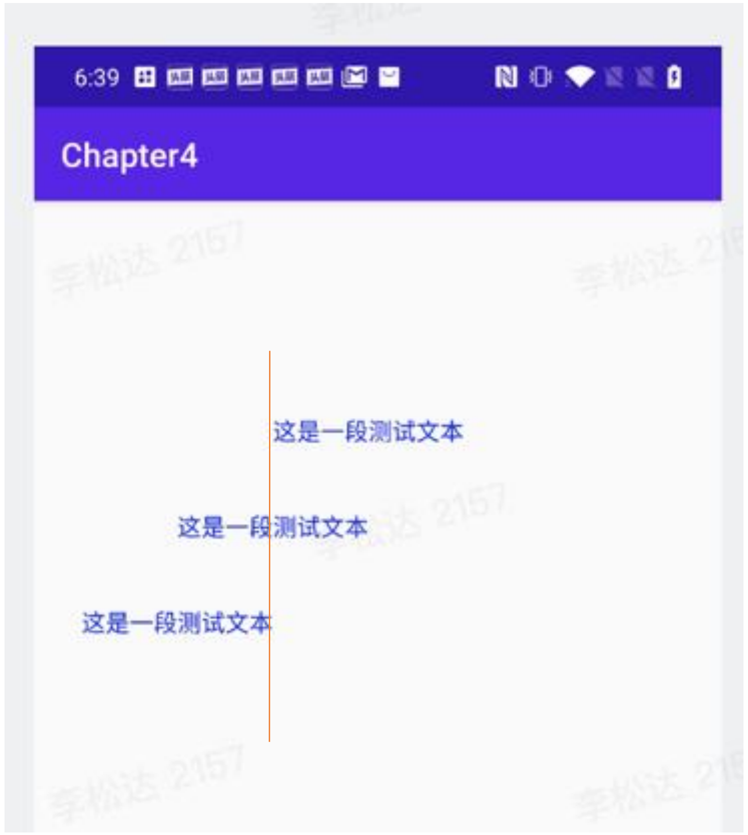
View绘制-画文本





```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    mPaint.setTextSize(50f);
    canvas.drawText( text: "这是一段测试文本",  x: 100,  y: 100, mPaint);
    Path path = new Path();//绘制多边形的类
    path.moveTo( x: 200,  y: 200);//起始点
    path.lineTo( x: 250,  y: 350);
    path.lineTo( x: 170,  y: 450);
    path.lineTo( x: 30,  y: 320);
    path.close();//闭合图形
    mPaint.setTextSize(25f);
    canvas.drawTextOnPath( text: "这是第二段测试文本，测试的内容是使用canvas画出一段文本", path,  hOffset: 0,  vOffset: 0, mPaint);

}
```

View绘制-画文本



```java
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    mPaint.setTextSize(50f);
    mPaint.setTextAlign(Paint.Align.LEFT);
    canvas.drawText( text: "这是一段测试文本",  x: 500,  y: 500, mPaint);
    mPaint.setTextAlign(Paint.Align.CENTER);
    canvas.drawText( text: "这是一段测试文本",  x: 500,  y: 700, mPaint);
    mPaint.setTextAlign(Paint.Align.RIGHT);
    canvas.drawText( text: "这是一段测试文本",  x: 500,  y: 900, mPaint);

}
```

自定义view总结

View的绘制流程：

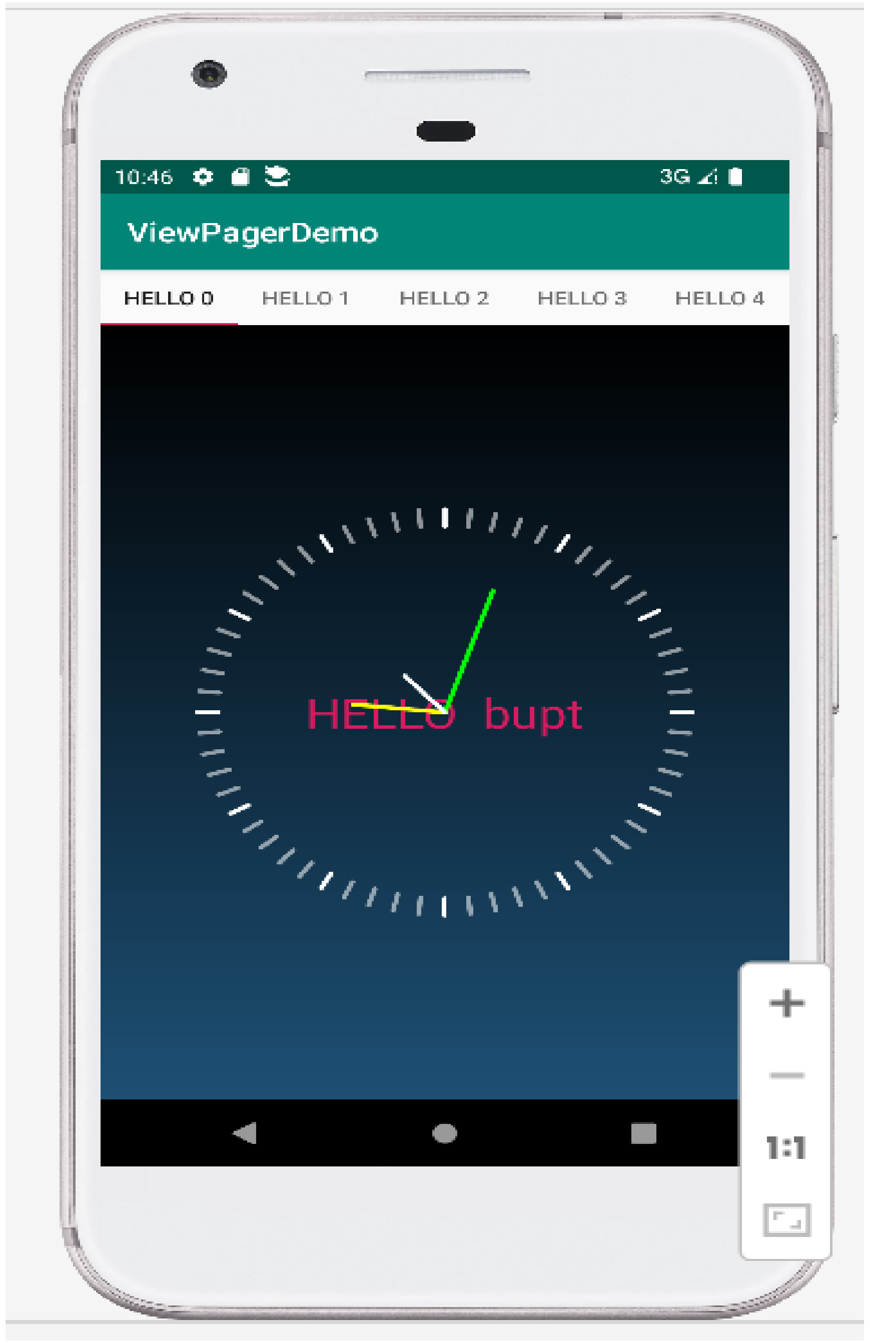
- 重要绘制流程：
  - ✓ Measure：测量
  - ✓ Layout：布局
  - ✓ Draw：绘制
- 以及几个重要函数：
  - ✓ onSizeChanged
  - ✓ invalidate
  - ✓ requestLayout
- 理解 ViewTree 及 ViewGroup 的Measure / Layout / Draw的流程
- View自定义绘制：
  - ✓ 绘制图形：点、线、圆形、椭圆、矩形、圆角矩形
  - ✓ 绘制文字：文字的测量

课堂作业

时钟App

作业基本要求：
1. 绘制时钟界面，包括表盘、时针、分针、秒针
2. 时针、分针、秒针需要跳动

升级要求：
在ViewPager 间切换图形时钟与数字时钟


10:46
3G
ViewPagerDemo
HELLO 0
HELLO 1
HELLO 2
HELLO 3
HELLO 4
HELLO bupt
+
—
1:1