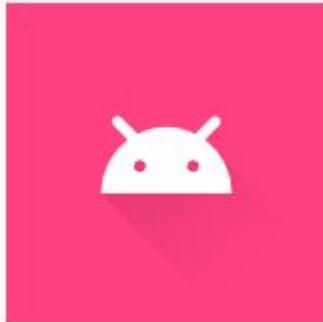




View & ViewGroup

План

- Views & ViewGroups
- Создание View
- Примеры Views
- Примеры ViewGroups
- Custom View
- Обработка тачей
- Drawable Resources
- Профилировка



checkbox

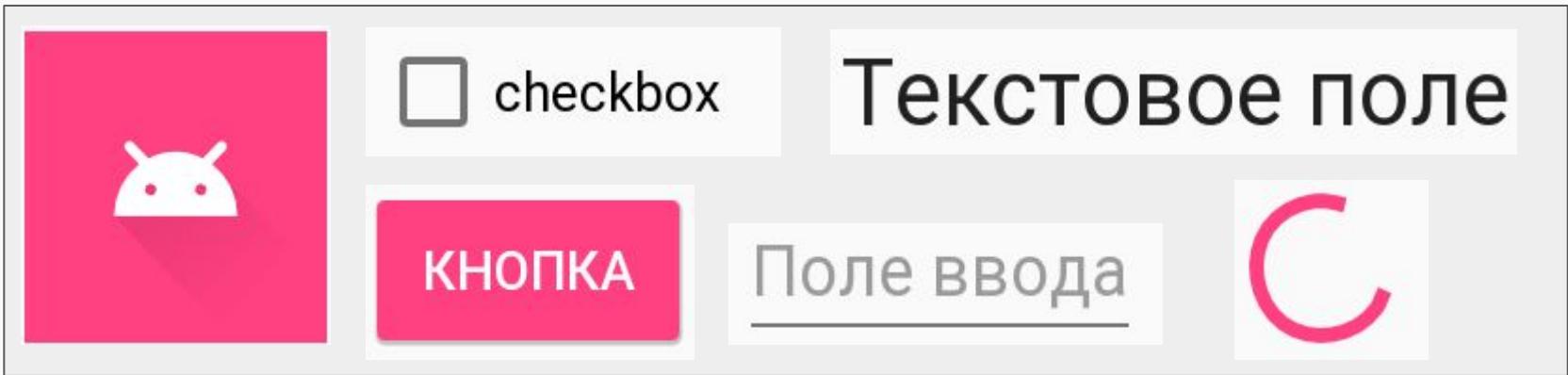
Текстовое поле

КНОПКА

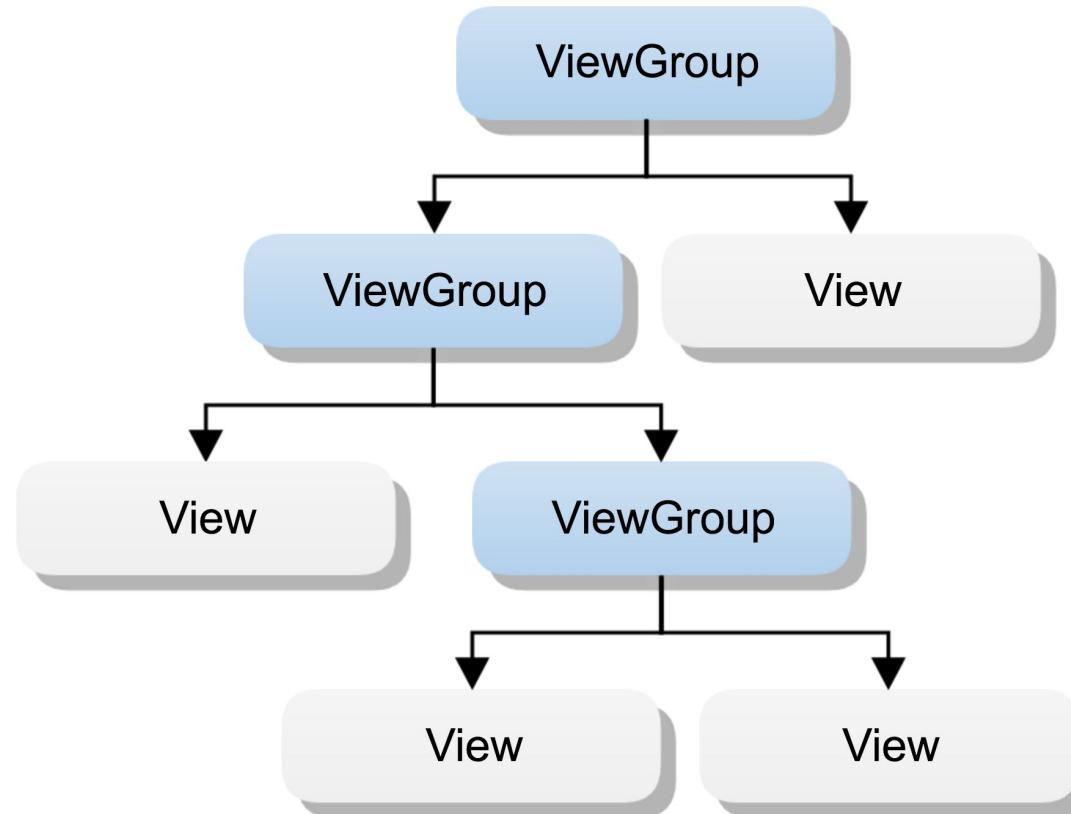


Поле ввода

ViewGroup



Дерево View



Создание view

- Через xml:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <TextView  
        android:id="@+id/textView"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_gravity="center"  
        android:text="Привет, Финтех!"  
        android:textAppearance="@style/TextAppearance.AppCompat.Headline" />  
  
</FrameLayout>
```

My Application

Привет, Финтех!

- Через код:

```
ViewGroup root = new FrameLayout( context: this );  
TextView textView = new TextView( context: this );  
textView.setText("Привет, Финтех!");  
textView.setTextAppearance( context: this, R.style.TextAppearance_AppCompat_Headline );  
root.addView(textView);
```

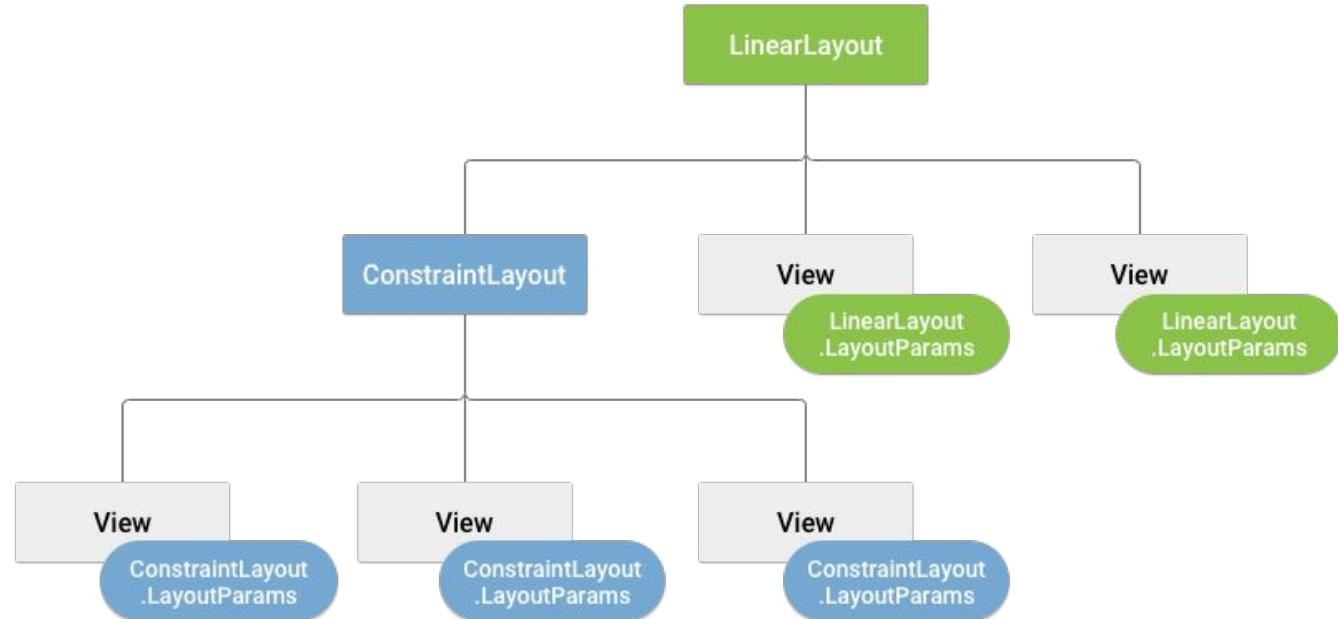


Создание view

LayoutInflater

Позволяет из xml файла получить объект View

`inflate(int resource, ViewGroup root, boolean attachToRoot)`

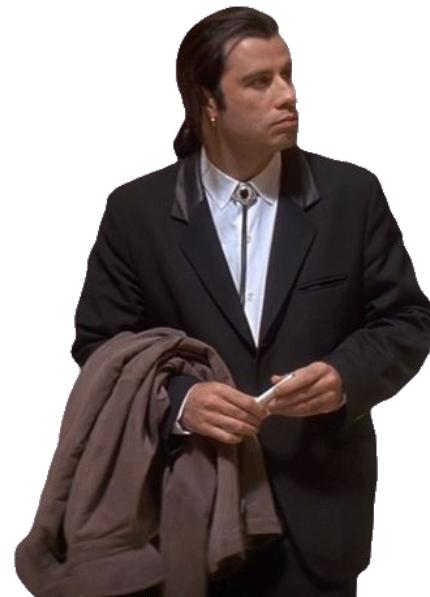


Создание view

Когда стоит создавать из **xml**, а когда из
кода?

xml

code



Создание view

Преимущества создания из xml:

- Быстро конструировать лэйаут
- Сразу видно в редакторе разметки

Преимущества создания из кода:

- Производительнее
- Можно динамически добавлять элементы

Создание view

```
<TextView  
    android:id="@+id/textView"  
    android:layout_width="200dp"  
    android:layout_height="50dp"  
    android:layout_gravity="center"  
    android:autoSizeMaxTextSize="500sp"  
    android:autoSizeMinTextSize="12sp"/>
```

Attribute autoSizeMaxTextSize is only used in API level 26 and higher (current min is 17) [more...](#) (⌘F1)

```
        android:autoSizeTextType="uniform"  
        android:maxLines="1"  
        android:text="Привет, Финтех!" />
```

```
<TextView  
    android:id="@+id/textView"  
    android:layout_width="200dp"  
    android:layout_height="50dp"  
    android:layout_gravity="center"  
    android:maxLines="1"  
    android:text="Привет, Финтех!"  
    app:autoSizeMaxTextSize="500sp"  
    app:autoSizeMinTextSize="12sp"  
    app:autoSizeStepGranularity="2sp"  
    app:autoSizeTextType="uniform" />
```

xmlns:app="http://schemas.android.com/apk/res-auto"

AppCompat*

AppCompatAutoCompleteTextView

AppCompatButton

AppCompatCheckBox

AppCompatCheckedTextView

AppCompatEditText

AppCompatImageButton

AppCompatImageView

AppCompatMultiAutoCompleteTextView

AppCompatRadioButton

AppCompatRatingBar

AppCompatSeekBar

AppCompatSpinner

AppCompatTextView

Живут в **androidx.appcompat.widget**

```
implementation 'com.android.support:appcompat-v7:26.1.0'
```

AppCompat*

Обычные view подменяются на AppCompat-аналоги в LayoutInflator

Упрощенно это происходит так:

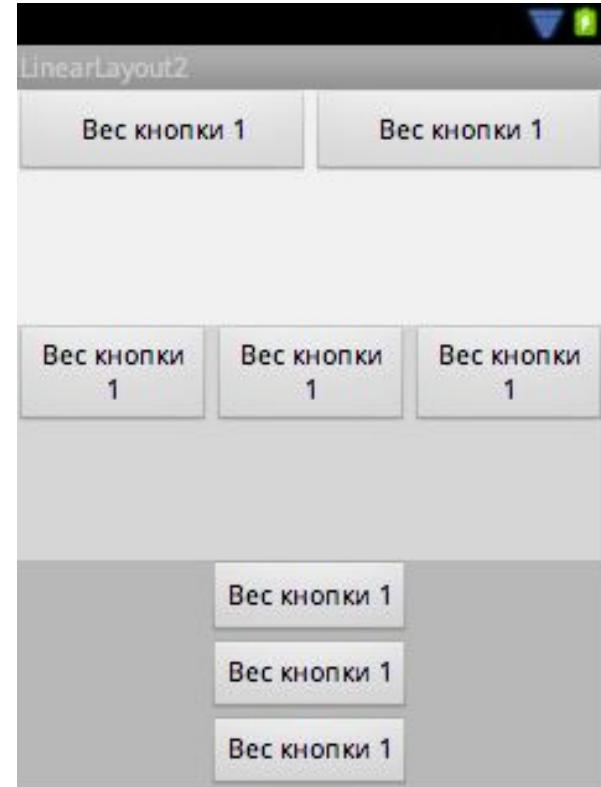
```
public final View onCreateView(View parent, final String name,
    @NonNull Context context, @NonNull AttributeSet attrs) {
    View view = null;
    switch (name) {
        case "TextView":
            view = new AppCompatTextView(context, attrs);
            break;
        case "ImageView":
            view = new AppCompatImageView(context, attrs);
            break;
        //... All AppCompatViews
    }
    return view;
}
```

AppCompat*

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        TextView textView = new TextView( context: this );  
        textView.setText("Hello World!");  
  
    }  
  
}
```

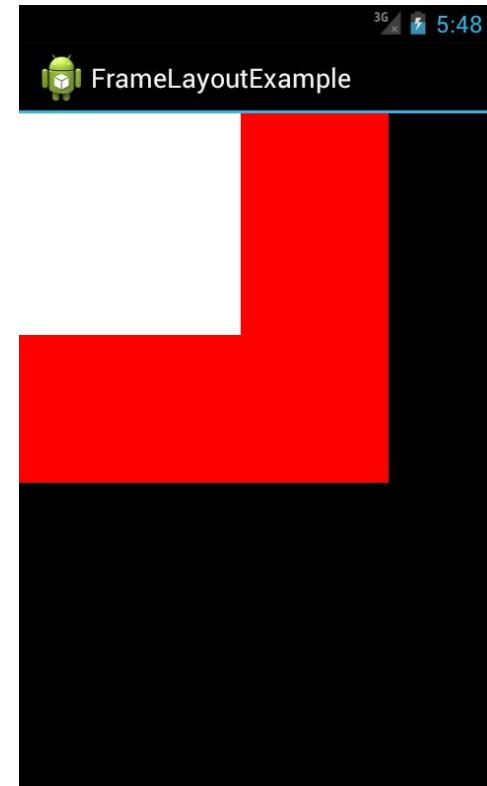
LinearLayout

- Ориентация
 - Горизонтальная
 - Вертикальная
- Элементы могут иметь “вес”



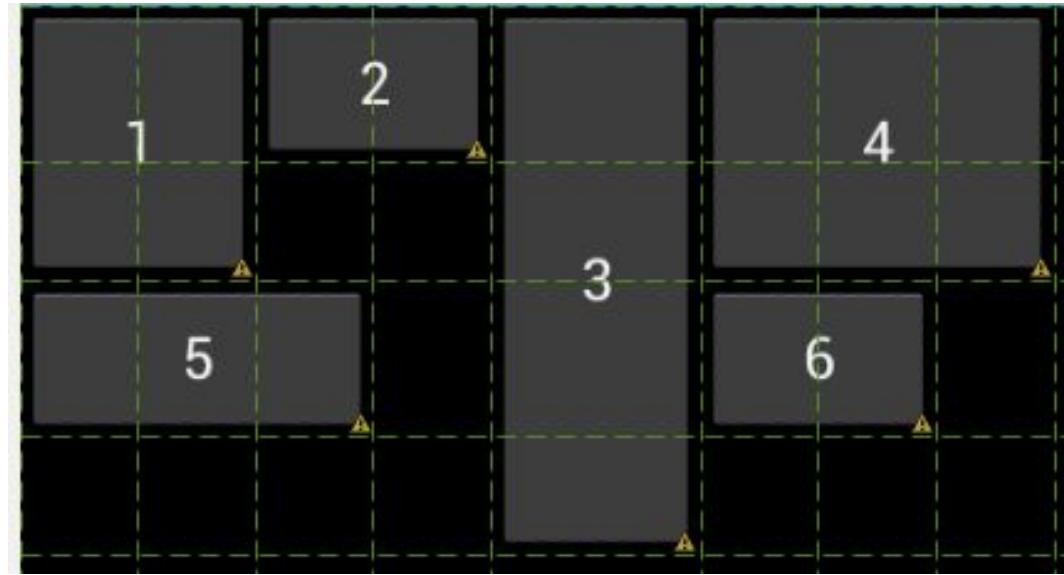
FrameLayout

- Быстрый
- Если внутри 1 View как правило можно обойтись без него



<https://developer.android.com/reference/android/widget/FrameLayout.html>

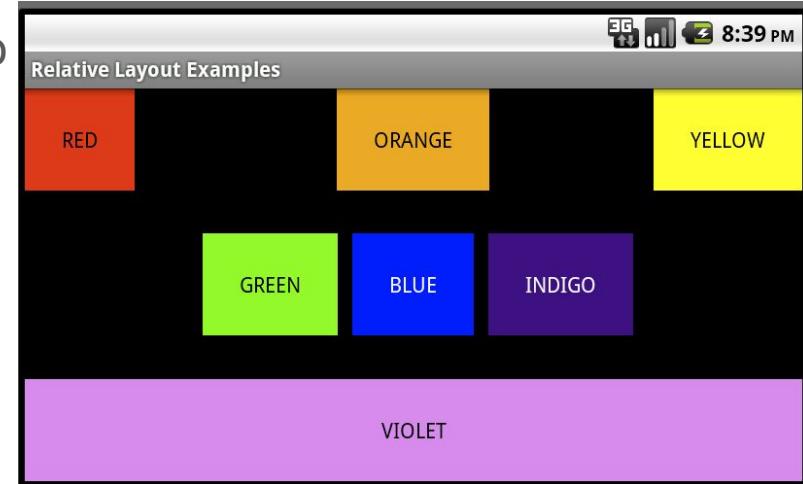
GridLayout



<https://developer.android.com/reference/android/support/v7/widget/GridLayout.html>

RelativeLayout

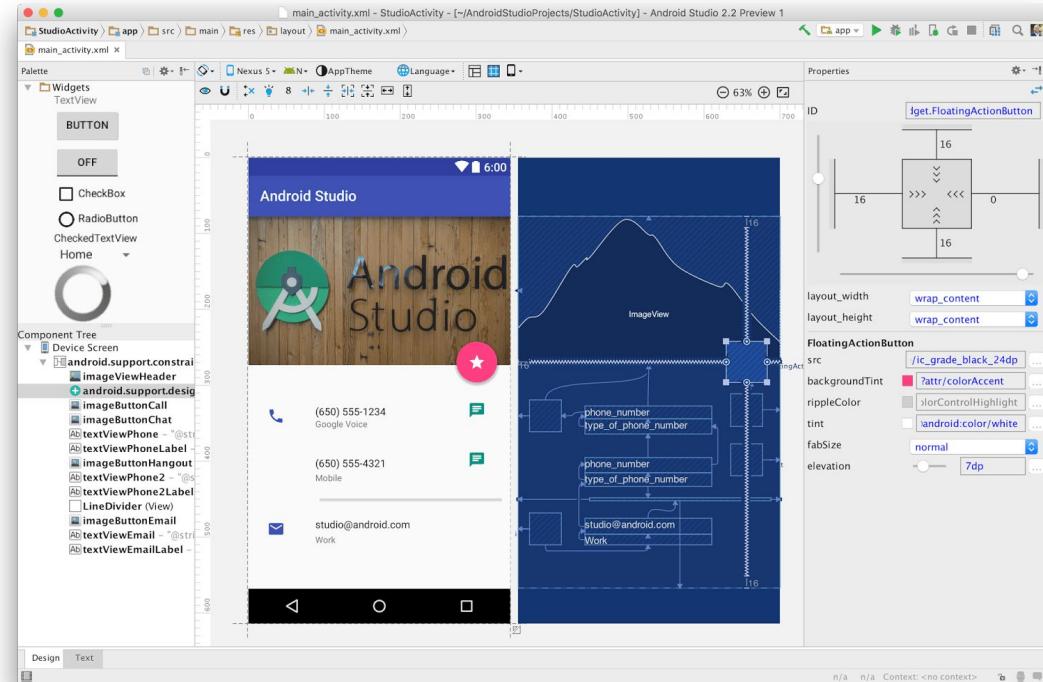
- View можно располагать относительно
 - Других View
 - Родительского контейнера



<https://developer.android.com/guide/topics/ui/layout/relative.html>

ConstraintLayout

- Живёт в **com.android.support.constraint:constraint-layout**
- Убирает лишнюю вложенность и улучшает производительность



CoordinatorLayout

- Живет в **com.android.support:design**
- Для координации view внутри него



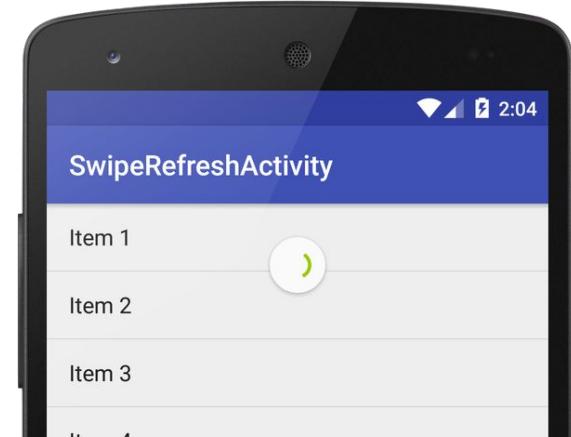
<https://habrahabr.ru/post/270121/>

CoordinatorLayout



SwipeRefreshLayout

- setRefreshing
- OnRefreshListener#onRefresh
- Только 1 дочерний элемент



<https://developer.android.com/training/swipe/add-swipe-interface.html>

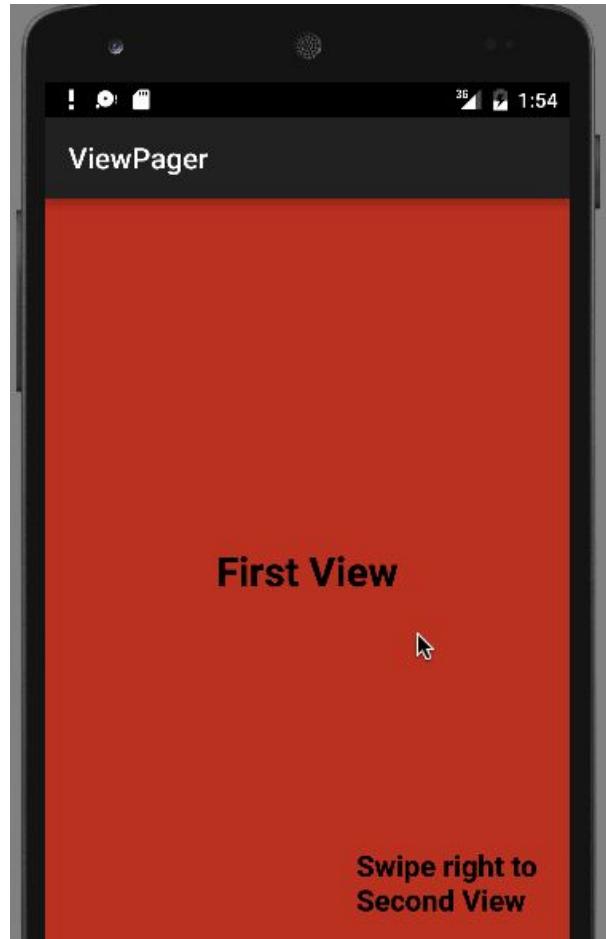
<https://developer.android.com/reference/android/support/v4/widget/SwipeRefreshLayout.html> 21

ViewPager

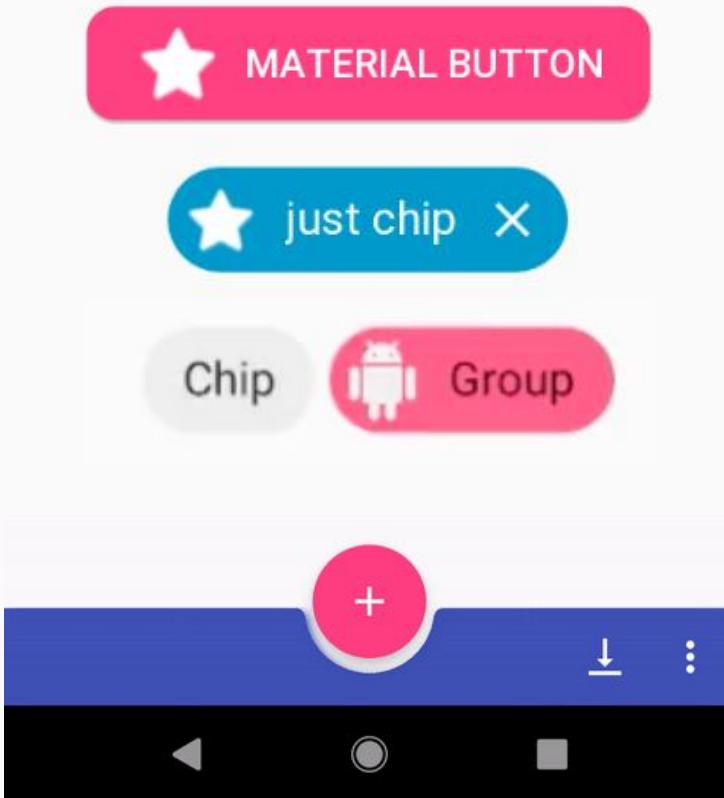
- Живёт в **androidx.viewpager.widgetViewPager**
- Работает как с View, так и с Fragment



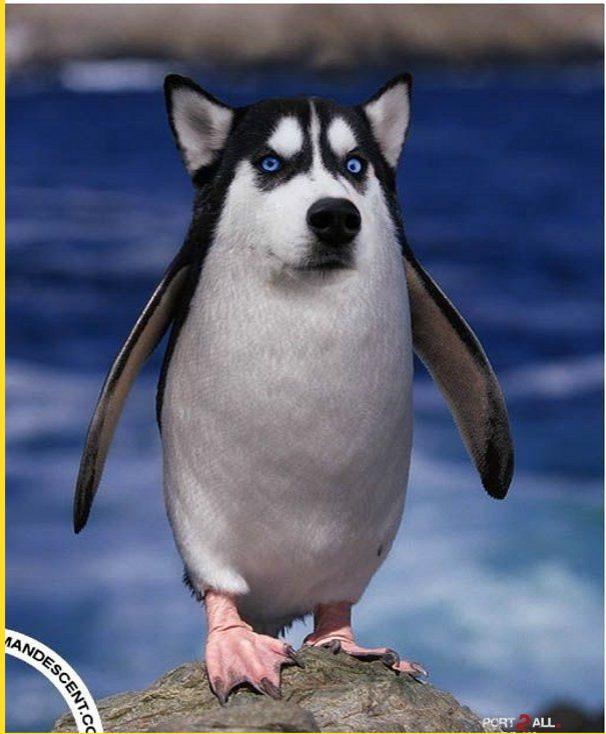
<https://developer.android.com/reference/androidx/viewpager/widgetViewPager>



New material design 2.0 views



it's `MaterialCardView`
with `strokeWidth`



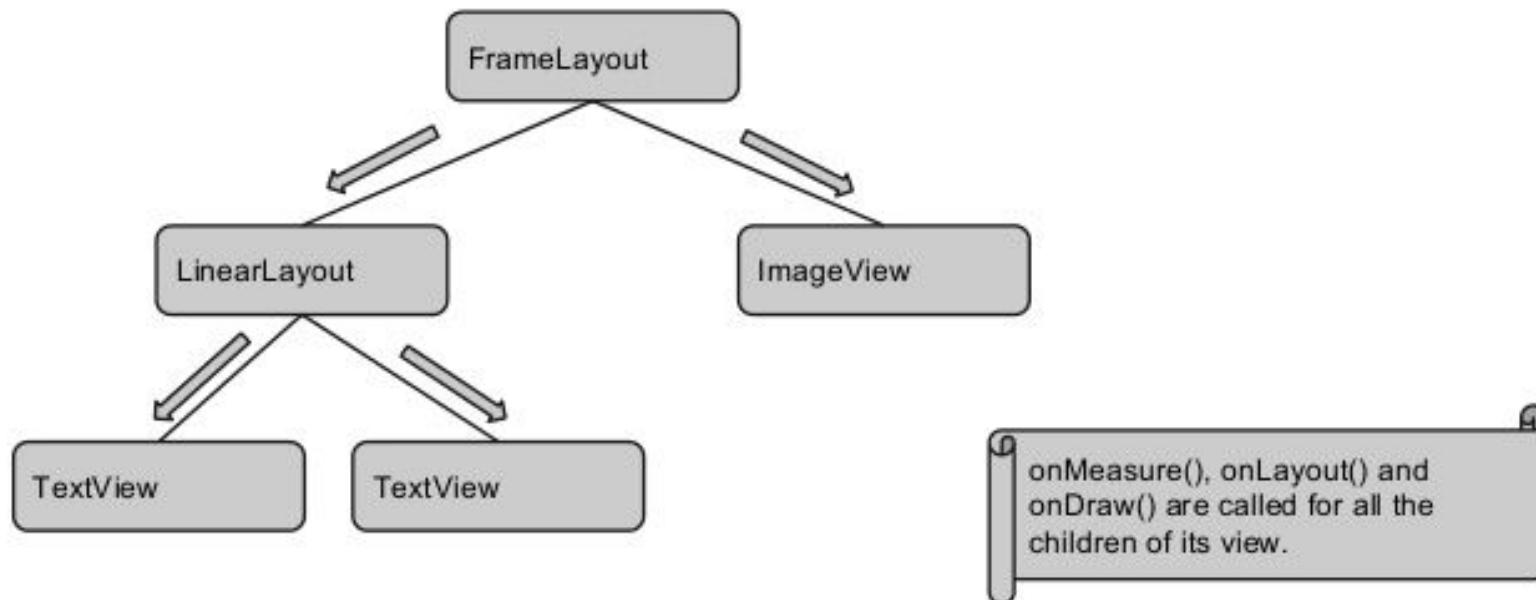
CustomView



View Rendering

- **measure** – сколько места мне нужно (в т.ч. детям) ?
- **layout** – где расположиться на экране ?
- **draw** – отрисовка !

Traversals are expensive



View Rendering: onMeasure

```
@Override  
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {  
    super.onMeasure(widthMeasureSpec, heightMeasureSpec);  
}
```

View.MeasureSpec :

- EXACTLY — ребёнок будет такого же размера
- UNSPECIFIED — ребёнок может быть настолько большим, насколько он захочет
- AT_MOST — ребёнок может быть настолько большим, как он хочет, но до определённого максимума

Нужно вызвать setMeasuredDimension() в onMeasure() после всех вычислений

View Rendering: onMeasure

View.MeasureState:

- *MEASURED_STATE_TOO_SMALL* — ребенок не доволен своим размером
- 0 — ребёнок доволен своим размером

LayoutParams

Хранит исходные данные о размерах и положении View

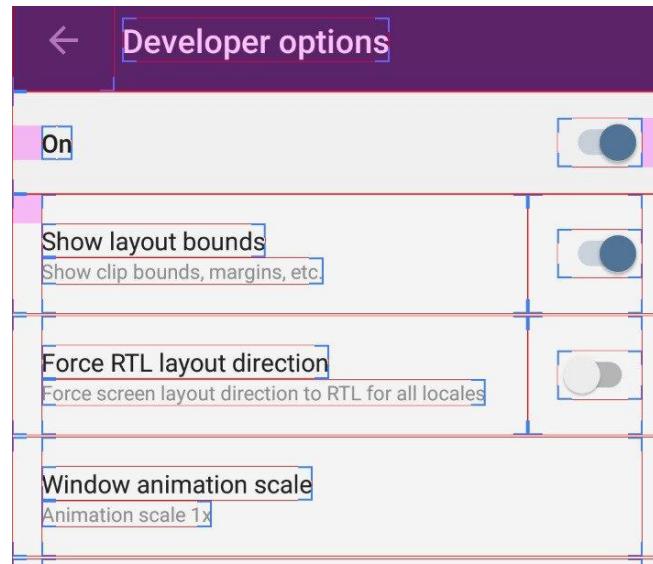
LayoutParams **extends** ViewGroup.MarginLayoutParams

```
@Override  
protected LayoutParams generateDefaultLayoutParams() {...}  
  
@Override  
protected LayoutParams generateLayoutParams(LayoutParams p) {...}  
  
@Override  
public LayoutParams generateLayoutParams(AttributeSet attrs) {...}  
  
@Override  
protected boolean checkLayoutParams(LayoutParams p) {...}
```

View Rendering: onLayout

```
@Override  
protected void onLayout(boolean changed, int left, int top, int right, int bottom) {  
    super.onLayout(changed, left, top, right, bottom);  
}
```

Позволяет присваивать размер и позицию дочерним View-компонентам



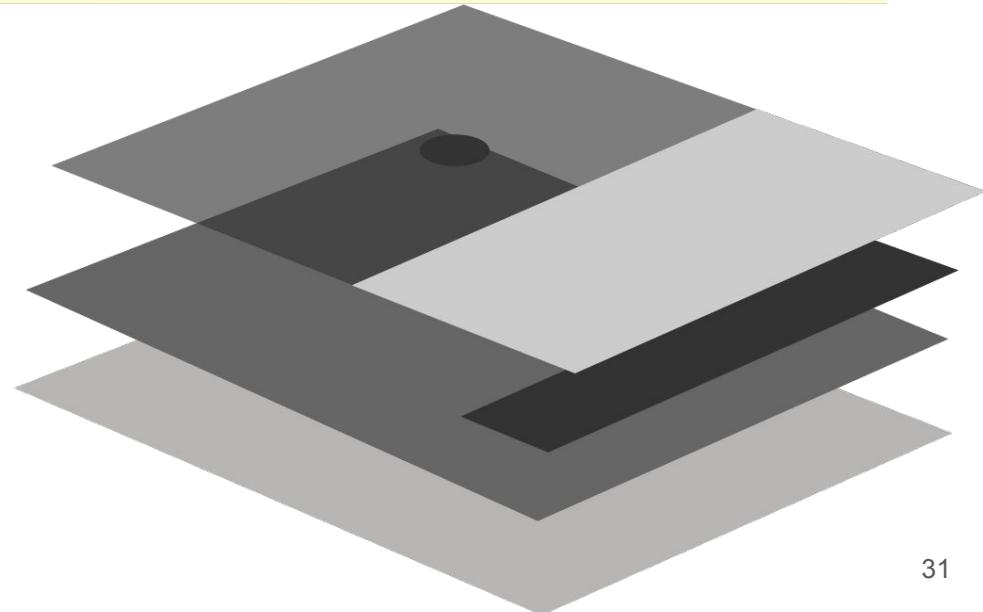
View Rendering: onDraw

```
@Override  
protected void onDraw(Canvas canvas) {  
    super.onDraw(canvas);  
    paint = new Paint();  
    RectF rectF = new RectF(x, y, x2, y2);
```

Avoid object allocations during draw/layout operations (preallocate and reuse instead)

Главные правила:

- Не создавать лишних объектов в onDraw, onMeasure, onLayout
- Избегать бессмысленных overdraw

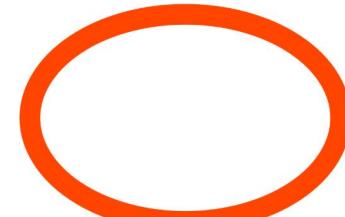


Canvas

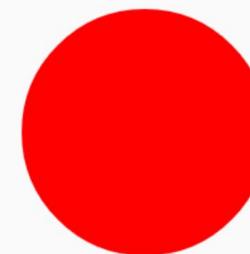
```
canvas.drawLine( startX: 10, startY: 10, stopX: 100, stopY: 10, paint);
```



```
canvas.drawoval(new RectF(x, y, x2, y2), paint);
```



```
canvas.drawCircle( cx: 10, cy: 10, radius: 100, paint);
```

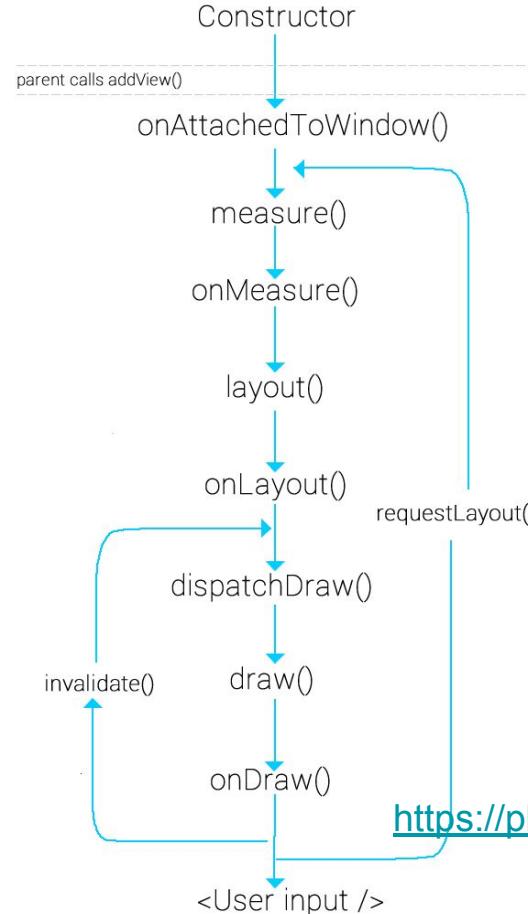


```
canvas.drawText( text: "android", x: 50, y: 100, paint);
```

ANDROID

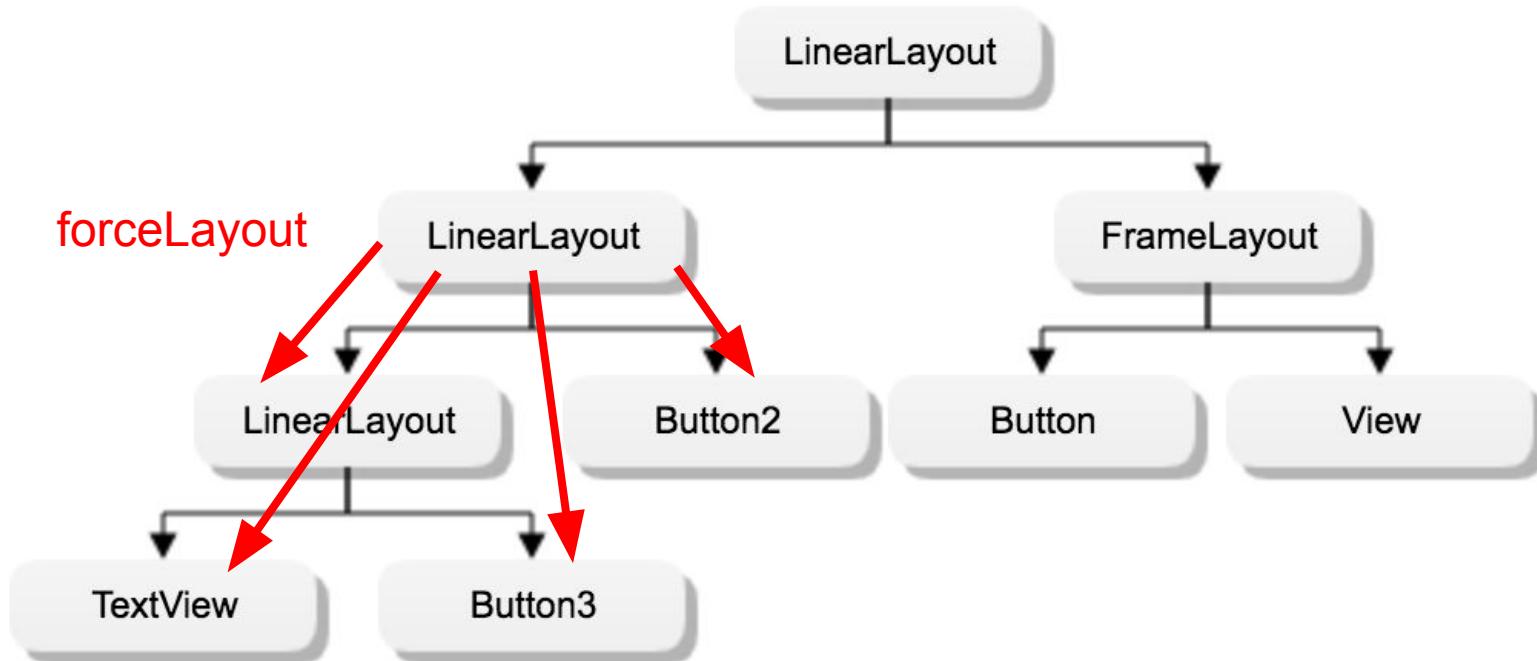
A large, stylized text "ANDROID" in green and blue, representing the output of the canvas.drawText() method.

invalidate vs requestLayout vs forceLayout



<https://plus.google.com/+ArpitMathur/posts/cT1EuBbxEqN>

invalidate vs requestLayout vs forceLayout



Рекомендации

- Использовать Frame или Linear **вместо** Relative или Linear с весами
- Избегать **лишней** вложенности
- Избегать бессмысленных **overdraw**
- **Не объявлять** лишних объектов в onDraw, onMeasure, onLayout
- Подумать перед вызовом **requestLayout()**
- Не стоит везде и всюду использовать **ConstraintLayout**

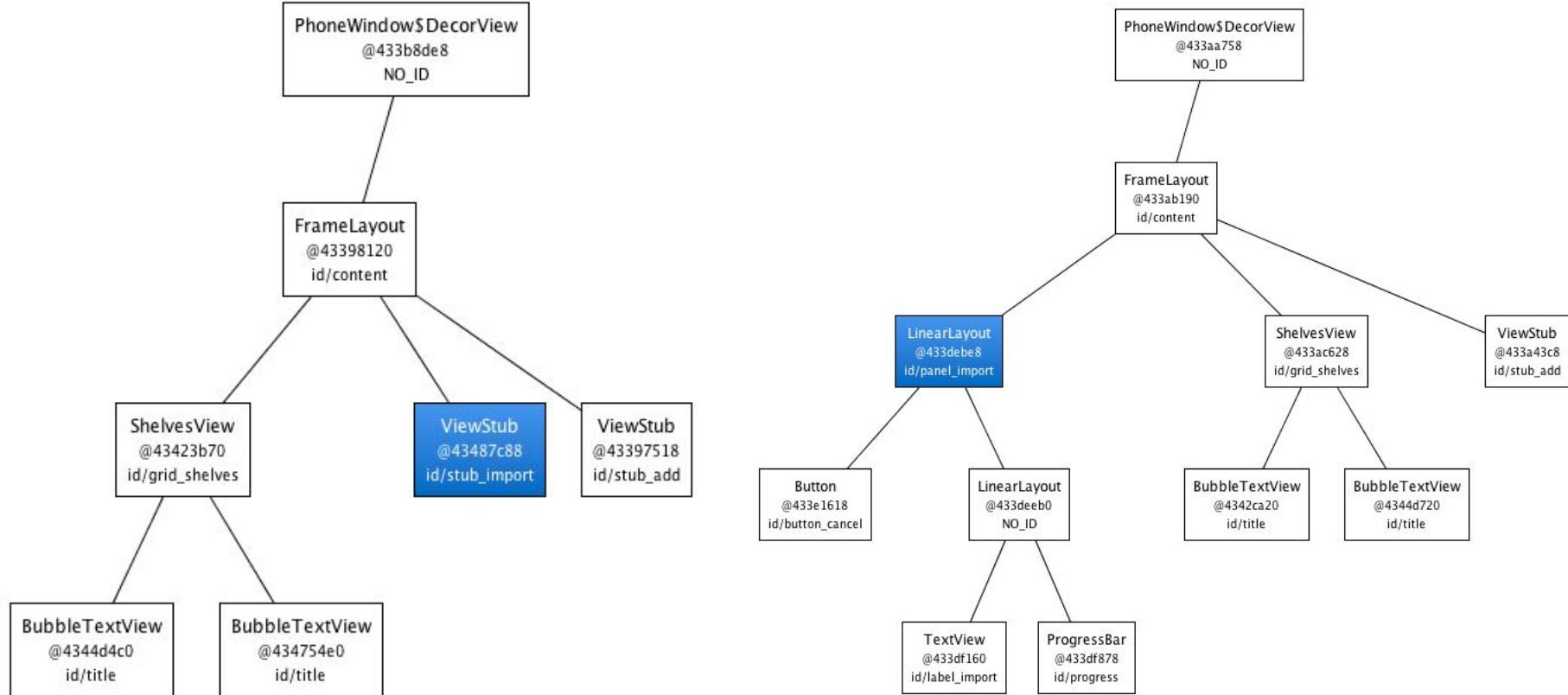
ViewStub

- Lazy загрузка View
- Атрибуты
 - layout
 - inflatedId
- Метод inflate()

<https://developer.android.com/reference/android/view/ViewStub.html>

<https://developer.android.com/training/improving-layouts/loading-ondemand.html>

ViewStub



Дополнительные XML-тэги

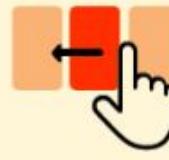
- `</merge>` – избежать лишней вложенности

`tools:showIn`

`tools:parentTag & tools:orientation`

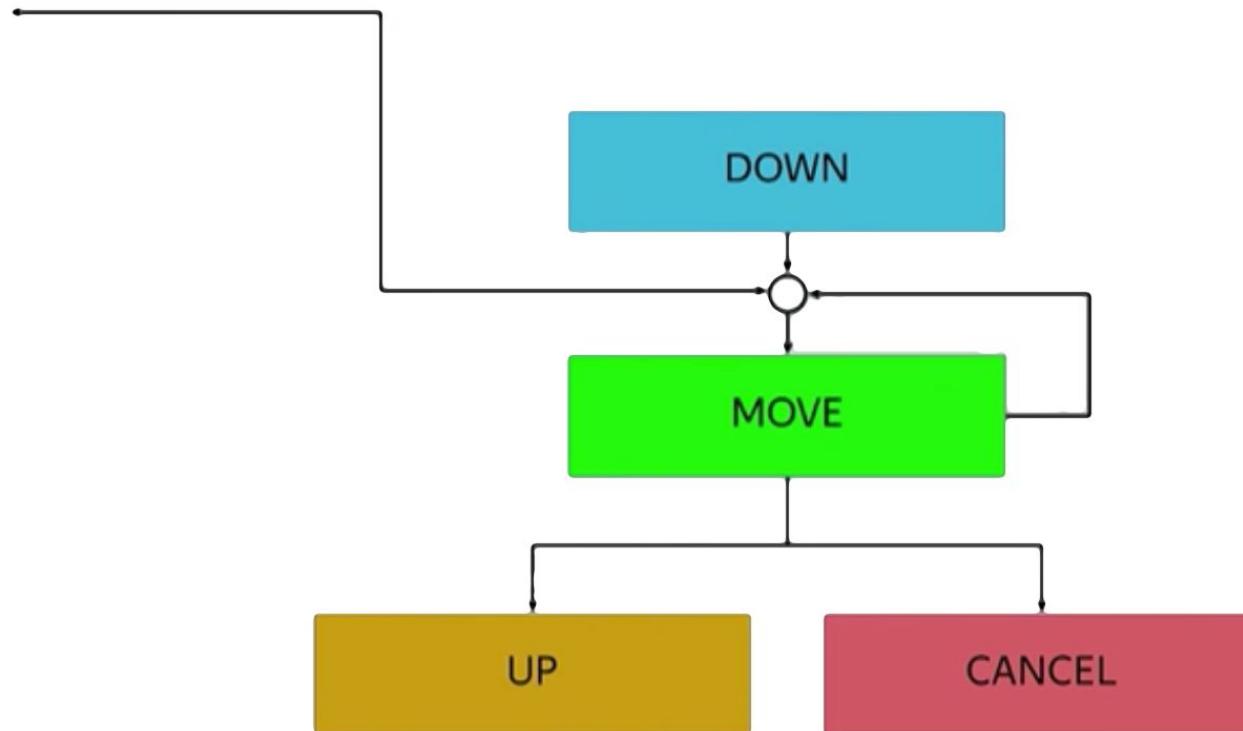
- `<include/>` – встроить другой лэйаут в текущий

<https://developer.android.com/training/improving-layouts/reusing-layouts.html>



Обработка тачей

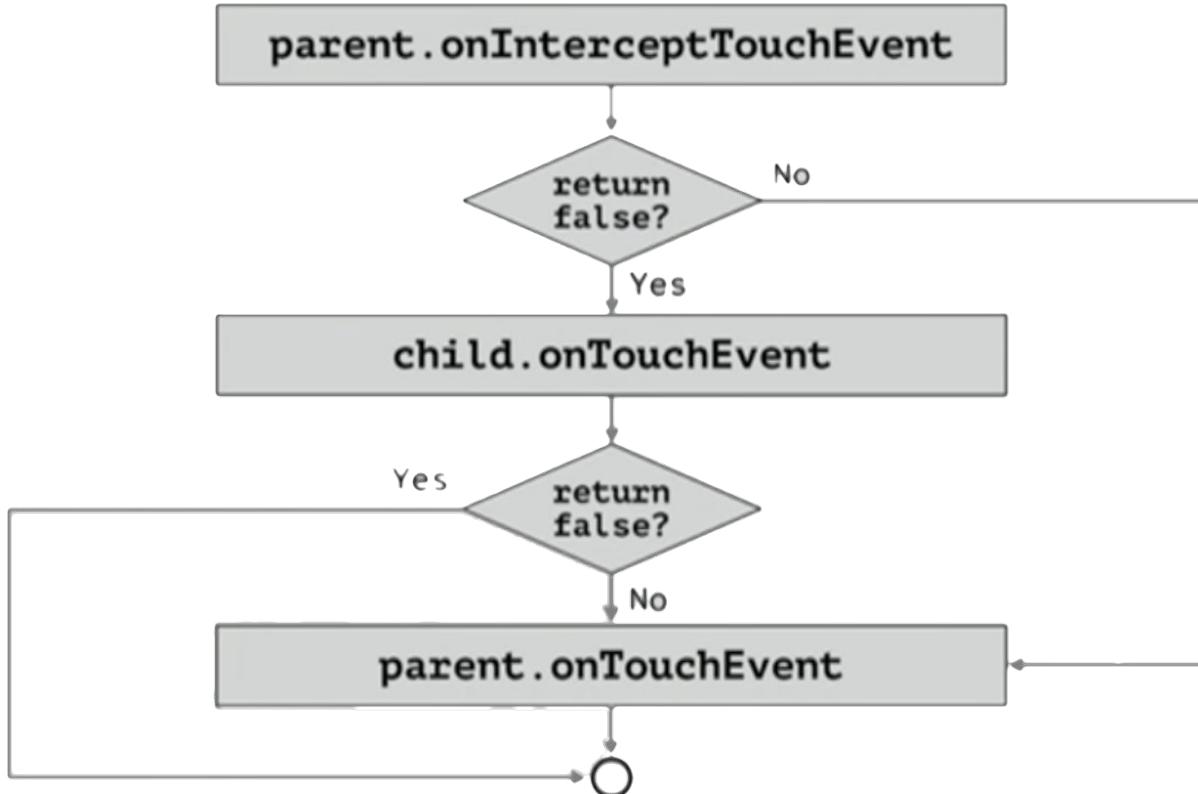
MotionEvent Actions



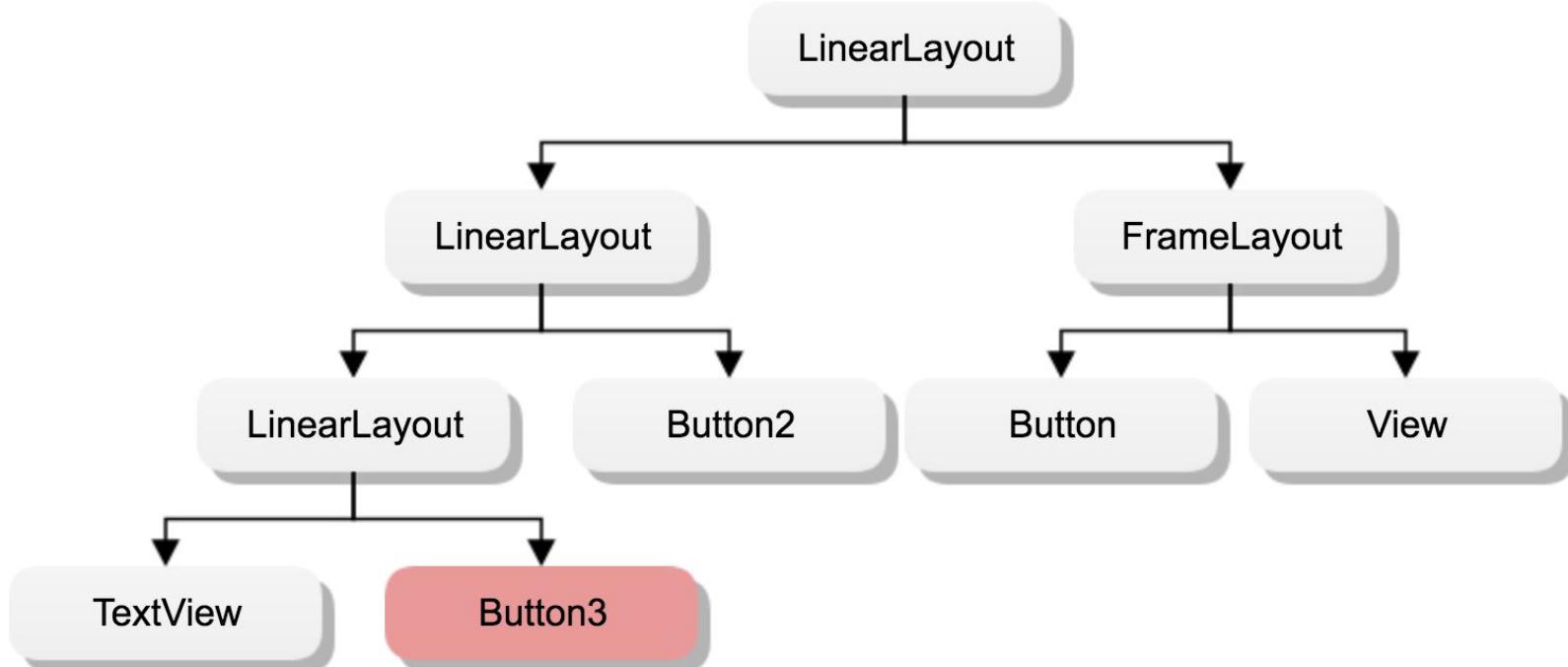
Обработка тачей

```
@Override
public boolean onTouchEvent(MotionEvent event){
    int action = MotionEventCompat.getActionMasked(event);
    switch(action) {
        case (MotionEvent.ACTION_DOWN) :
            Log.d(DEBUG_TAG, "Action was DOWN");
            return true;
        case (MotionEvent.ACTION_MOVE) :
            Log.d(DEBUG_TAG, "Action was MOVE");
            return true;
        case (MotionEvent.ACTION_UP) :
            Log.d(DEBUG_TAG, "Action was UP");
            return true;
        case (MotionEvent.ACTION_CANCEL) :
            Log.d(DEBUG_TAG, "Action was CANCEL");
            return true;
        default :
            return super.onTouchEvent(event);
    }
}
```

Обработка тачей



Обработка тачей



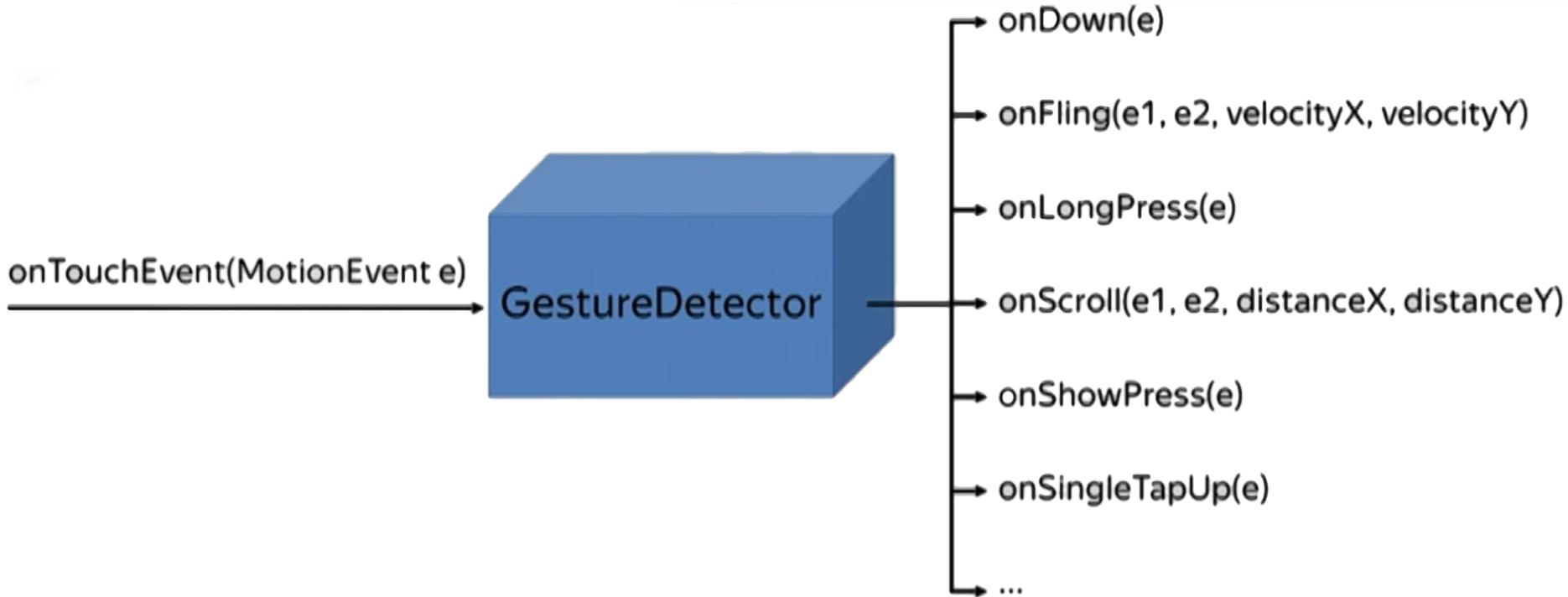
Определение скорости

```
velocityTracker = VelocityTracker.obtain();
velocityTracker.addMovement(motionEvent);
velocityTracker.computeCurrentVelocity(1000);

float velocityX = velocityTracker.getXVelocity();
float velocityY = velocityTracker.getYVelocity();

velocityTracker.recycle();
```

Gesture Detector



Перерыв на какаушко



Custom Views

- Компоновка нескольких View
- Наследование от существующего View(TextView)
- Наследование от класса View



<https://developer.android.com/training/custom-views/create-view.html>

<https://developer.android.com/guide/topics/ui/custom-components.html>

Custom Views: constructors

```
public CustomView(Context context) {  
    super(context);  
}  
  
public CustomView(Context context, @Nullable AttributeSet attrs) {  
    super(context, attrs);  
}  
  
public CustomView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {  
    super(context, attrs, defStyleAttr);  
}
```

<http://blog.danlew.net/2016/07/19/a-deep-dive-into-android-view-constructors/>

Custom Views: constructors

```
public AppCompatTextView(Context context) {
    this(context, attrs: null);
}

public AppCompatTextView(Context context, AttributeSet attrs) {
    this(context, attrs, android.R.attr.textViewStyle);
}

public AppCompatTextView(Context context, AttributeSet attrs, int defStyleAttr) {
    super(TintContextWrapper.wrap(context), attrs, defStyleAttr);

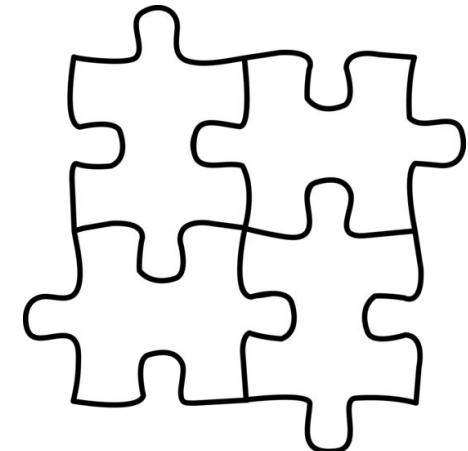
    mBackgroundTintHelper = new AppCompatBackgroundHelper( view: this );
    mBackgroundTintHelper.loadFromAttributes(attrs, defStyleAttr);

    mTextHelper = AppCompatTextHelper.create(this);
    mTextHelper.loadFromAttributes(attrs, defStyleAttr);
    mTextHelper.applyCompoundDrawablesTints();
}
```

Compound

Новое View образуется группировкой уже имеющихся

- Наследуемся от лейаута
- В конструкторе
 - инфлейтим иерархию вьюх
 - читаем атрибуты
 - инициализируем создаваемое View
 - создаем нужные листенеры
- Добавляем свойства, геттеры и сеттеры
- Переопределяем методы родителя

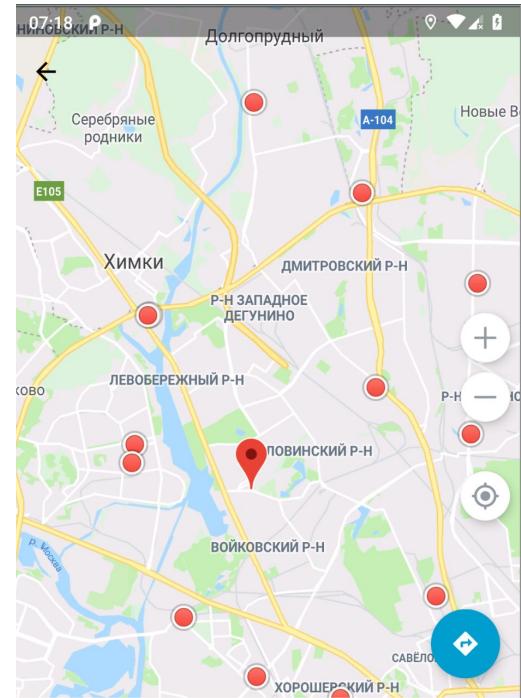


Следует обратить внимание на <merge> тэг

<https://developer.android.com/guide/topics/ui/custom-components.html#compound>

Compound

```
public class CinemaMapView extends LinearLayout {  
    private final TextView cinemaTitleView;  
    private final StationLinearLayout stationLayout;  
    private final TextView addressView;  
    public CinemaMapView(Context context) {...}  
    public CinemaMapView(Context context, AttributeSet attrs) {...}  
    public CinemaMapView(Context context, AttributeSet attrs, int  
defStyleAttr) {  
        super(context, attrs, defStyleAttr);  
        inflate(context, R.layout.widget_cinema_map_header, this);  
        setOrientation(VERTICAL);  
  
        cinemaTitleView = findViewById(R.id.cinema_title);  
        stationLayout = findViewById(R.id.cinema_station_layout);  
        addressView = findViewById(R.id.cinema_address);  
    }  
  
    public void setData(CinemaMapHeaderData cinema) {  
        cinemaTitleView.setText(cinema.getName());  
        stationLayout.setStations(cinema.getStations());  
        addressView.setText(cinema.getAddress());  
    }  
}
```



Киномакс-Водный

• Водный стадион

ул. Головинское шоссе, 5, 3 этаж

Compound

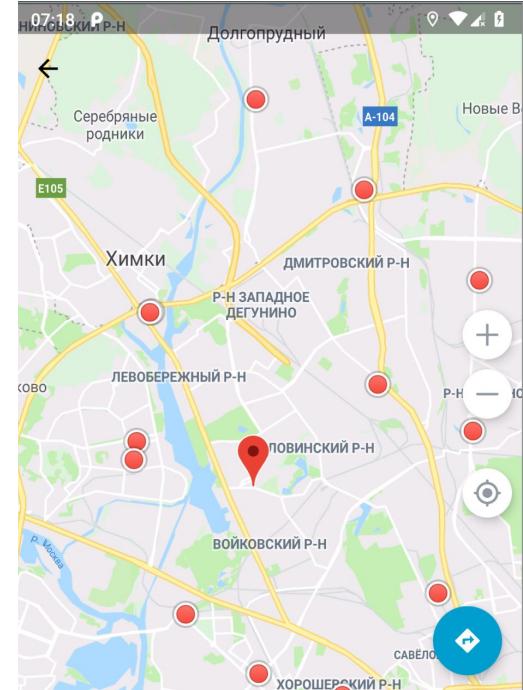
```
<merge xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:minHeight="@dimen/cinema_bottom_sheet_peek_height"
    android:orientation="vertical"
    tools:parentTag="android.widget.LinearLayout">

    <TextView
        android:id="@+id/cinema_title"
        tools:text="Киномакс-Водный"
        ... />

    <ru.tinkoff.mb.booking.ui.cinema.hub.StationLinearLayout
        android:id="@+id/cinema_station_layout"
        ... />

    <TextView
        android:id="@+id/cinema_address"
        tools:text="ул. Головинское шоссе, 5, 3 этаж"
        ... />

</merge>
```



Киномакс-Водный

● Водный стадион

ул. Головинское шоссе, 5, 3 этаж

Subclass Existed View

Наследуемся от существующей в SDK или библиотеке View и изменяем ее под себя

Примеры:

- AppCompat*View
- TextView с кастомным шрифтом
- **Button с прогрессом**



<https://developer.android.com/guide/topics/ui/custom-components.html#modifying>

Subclass View

```
public class MbSpinner extends AppCompatSpinner {  
    private ErrorStateDrawableController errorStateDrawableController;  
  
    public MbSpinner(Context context) {...}  
    public MbSpinner(Context context, AttributeSet attrs) {...}  
    public MbSpinner(Context context, AttributeSet attrs, int defStyleAttr) {  
        super(context, attrs, defStyleAttr);  
        init();  
    }  
  
    public void init() {  
        Drawable drawable =  
getBackground().getCurrent().getConstantState().newDrawable().mutate();  
        drawable.setColorFilter(ContextCompat.getColor(getContext(), R.color.dividers),  
PorterDuff.Mode.SRC_IN);  
        setBackground(drawable);  
        errorStateDrawableController = new ErrorStateDrawableController(this);  
    }  
  
    public void markError(boolean mark) {  
        errorStateDrawableController.markError(mark);  
    }  
}
```

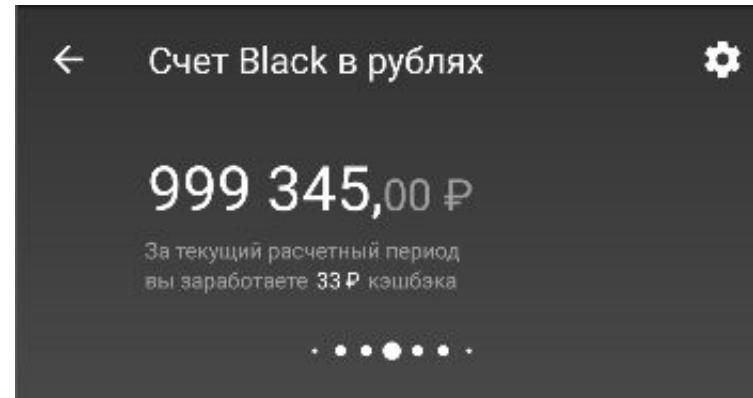
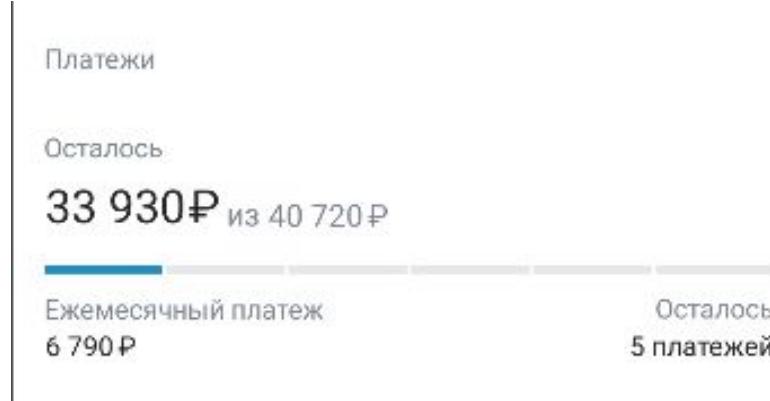
Subclass View

- Наследуемся от android.view.View
- Самостоятельно реализуем
 - onMeasure
 - onLayout
 - onDraw
 - всю остальную логику



<https://developer.android.com/guide/topics/ui/custom-components.html#custom>

Subclass View



Custom Attributes

- Объявляем атрибуты в values/attrs.xml

```
<resources>
    <declare-styleable name="Custom">
        <attr name="radius" format="dimension|reference" />
        <attr name="border_color" format="color" />
    </declare-styleable>
</resources>
```

-
- Обрабатываем в конструкторе

```
private void applyAttributes(AttributeSet attrs) {
    TypedArray array = getContext().obtainStyledAttributes(attrs,
R.styleable.Custom);
    setBorderRadius(array.getDimension(R.styleable.Custom_radius, 10));
    setBorderColor(array.getColor(R.styleable.Custom_border_color, Color.RED));
    array.recycle();
}
```

Ресурсы

Каталог	Тип ресурсов
animator	Файлы XML, которые определяют анимации свойств.
anim	Файлы XML, которые определяют анимации преобразований.
color	Файлы XML, которые определяют список состояний цветов.
drawable	Файлы растровых изображений (.png, .9.png, .jpg, .gif) или файлы XML, которые составляют следующие подтипы графических ресурсов: растровые изображения, списки состояний, формы и т. д.
mipmap	Графические файлы для значков запуска с различной плотностью.
layout	Файлы XML, которые определяют макет пользовательского интерфейса.
menu	Файлы XML, которые определяют меню приложения, такие как меню параметров, контекстные меню или вложенные меню.
raw	Произвольные файлы для сохранения в исходной форме.
values	Файлы XML, которые содержат простые значения, такие как строки, целые числа и цвета.
xml	Произвольные XML-файлы, которые можно читать в режиме выполнения. Здесь должны сохраняться различные файлы конфигурации XML)

Ресурсы

Каталог	Тип ресурсов
animator	Файлы XML, которые определяют анимации свойств .
anim	Файлы XML, которые определяют анимации преобразований .
color	Файлы XML, которые определяют список состояний цветов.
drawable	Файлы растровых изображений (.png, .9.png, .jpg, .gif) или файлы XML, которые составляют следующие подтипы графических ресурсов: растровые изображения, списки состояний, формы и т. д.
mipmap	Графические файлы для значков запуска с различной плотностью.
layout	Файлы XML, которые определяют макет пользовательского интерфейса.
menu	Файлы XML, которые определяют меню приложения, такие как меню параметров, контекстные меню или вложенные меню.
raw	Произвольные файлы для сохранения в исходной форме.
values	Файлы XML, которые содержат простые значения, такие как строки, целые числа и цвета.
xml	Произвольные XML-файлы, которые можно читать в режиме выполнения. Здесь должны сохраняться различные файлы конфигурации XML)

Drawable-ресурсы

- Bitmap images
- Shapes
- Layer Lists
- Nine patch images
- Vector images
- State Lists
- etc.

<https://developer.android.com/guide/topics/resources/providing-resources.html?hl=ru>

<https://developer.android.com/guide/topics/resources/drawable-resource.html?hl=ru>

Shapes

- Можно рисовать line, oval и rectangle
- Закрашивать градиентом
- Скруглять углы
- Делать границы

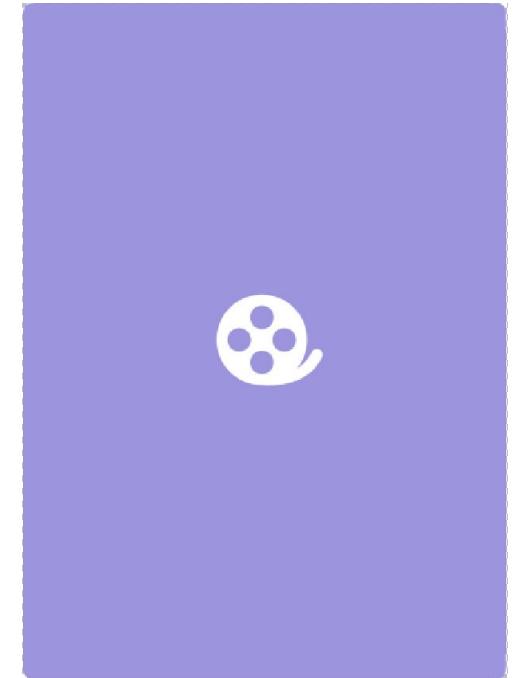
Нужен для рисования простых фигур без помощи дизайнера

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <gradient
        android:angle="90"
        android:endColor="@color/n1_50"
        android:startColor="@android:color/transparent" />
</shape>
```

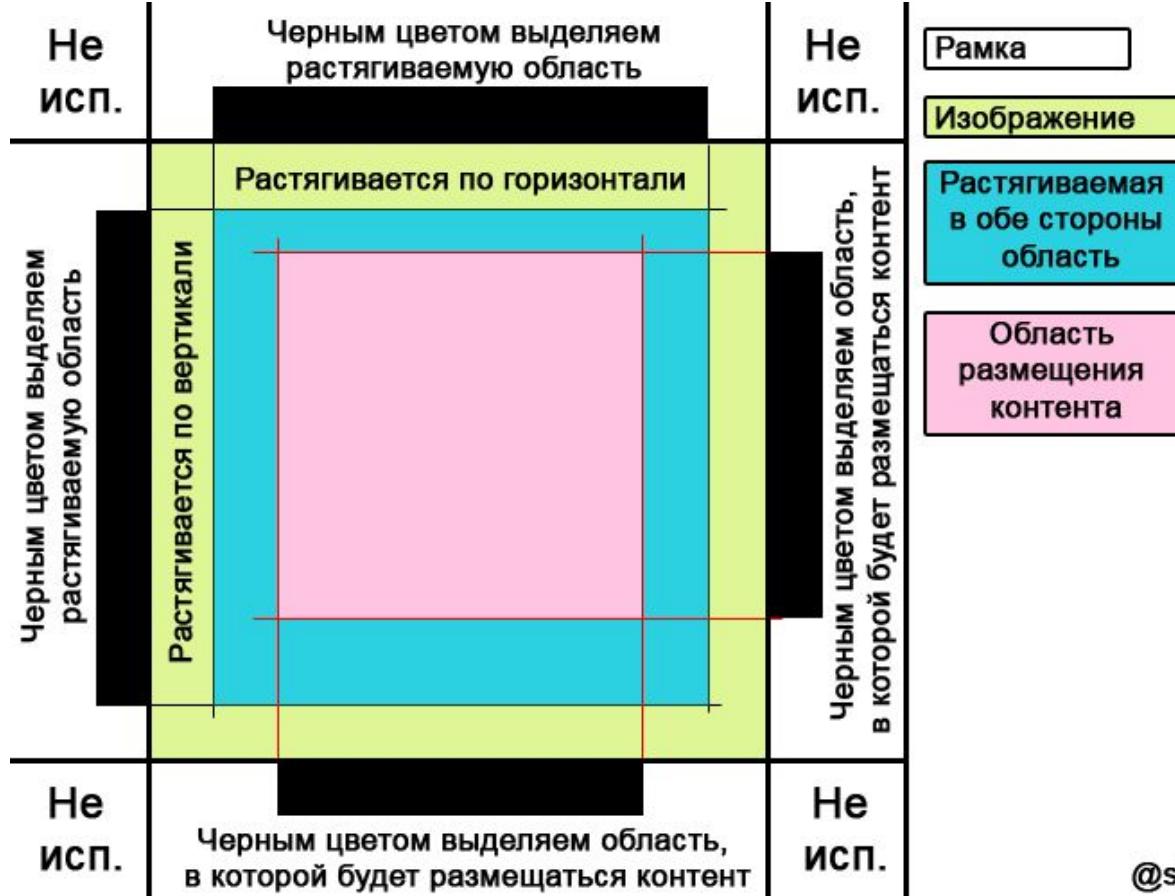


Layer List

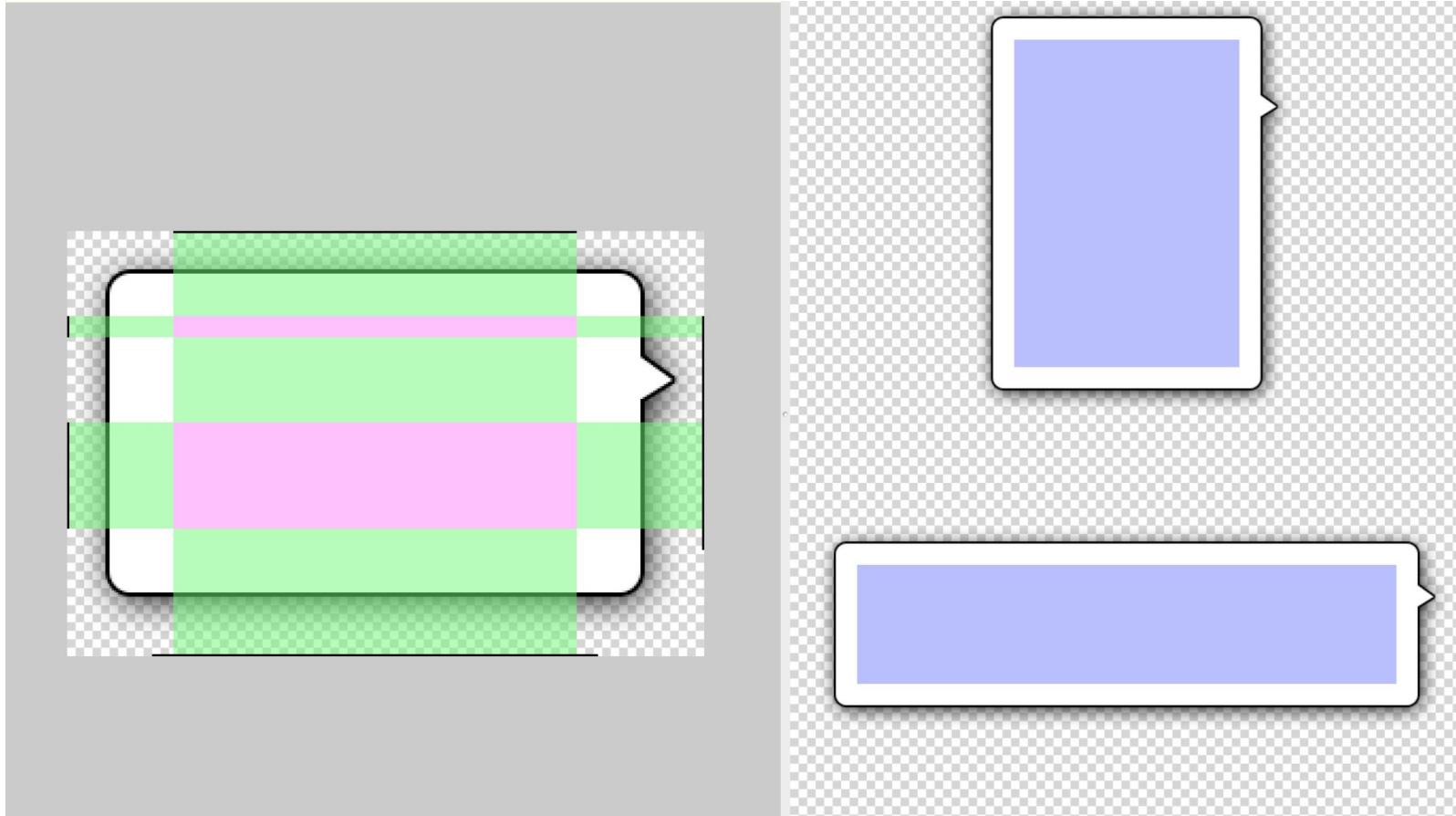
```
<layer-list xmlns:android="http://schemas.android.com/
apk/res/android">
    <item>
        <shape android:shape="rectangle">
            <corners android:radius="2dp" />
            <solid
                android:color="@color/movie_placeholder" />
            <size
                android:width="100dp"
                android:height="140dp" />
        </shape>
    </item>
    <item android:drawable="@drawable/ic_film"
        android:gravity="center" />
</layer-list>
```



9-patch

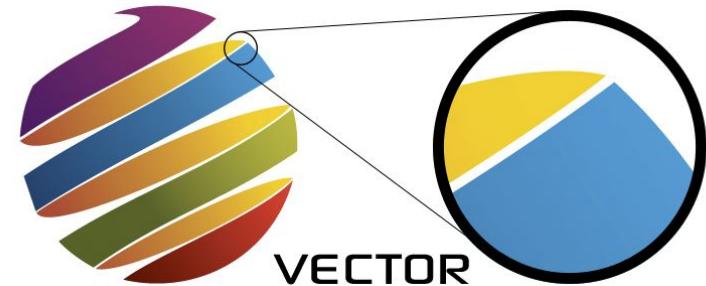


9-patch



Vector Drawables

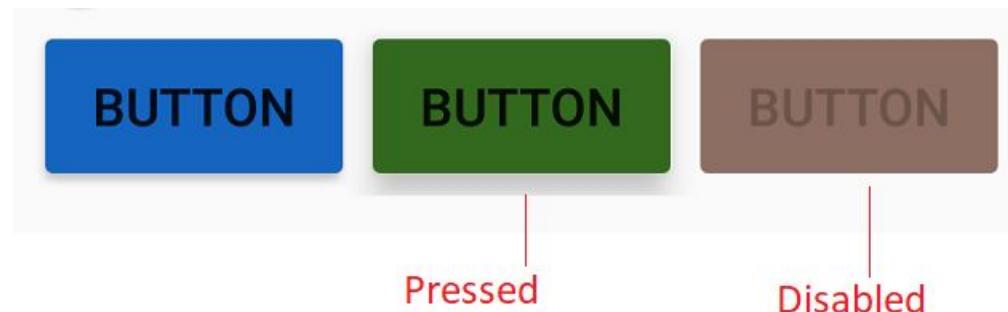
- Не нужно нарезать графику под разные плотности
- SVG -> XML
- Обратная совместимость (api < 21)
 - `vectorDrawables.useSupportLibrary = true` в defaultConfig
 - `android:src -> app:srcCompat`
- Без обратной совместимости – нарезка в api < 21



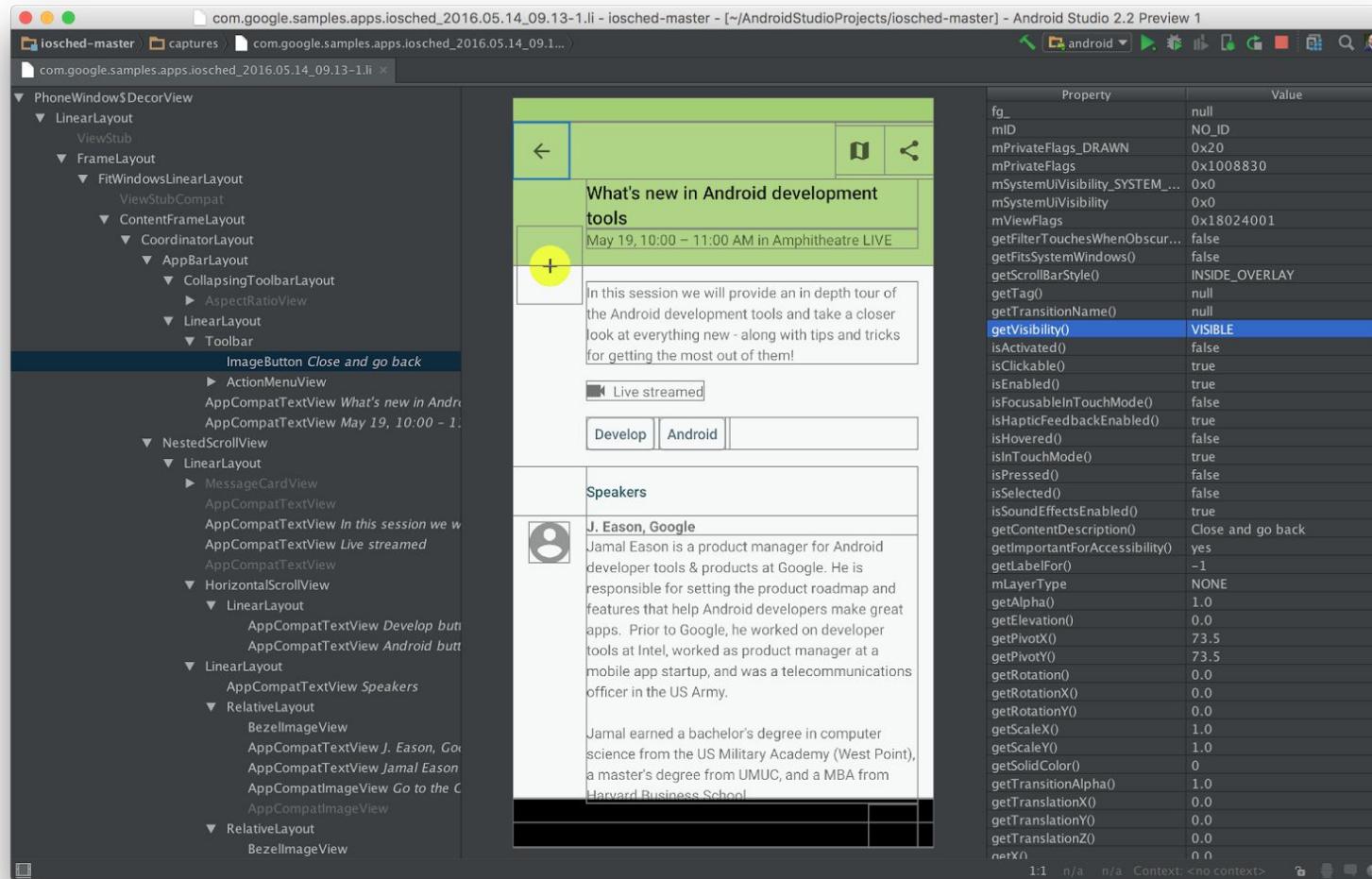
<https://developer.android.com/guide/topics/graphics/vector-drawable-resources.html>

State Lists

```
<selector xmlns:android="http://schemas.android.com/apk/res/android"  
    android:constantSize=["true" | "false"]  
    android:dither=["true" | "false"]  
    android:variablePadding=["true" | "false"]>  
    <item  
        android:drawable="@+[package:]drawable/drawable_resource"  
        android:state_pressed=["true" | "false"]  
        android:state_selected=["true" | "false"]  
        android:state_checkable=["true" | "false"]  
        android:state_checked=["true" | "false"]  
        android:state_enabled=["true" | "false"]  
        ...  
    </item>
```

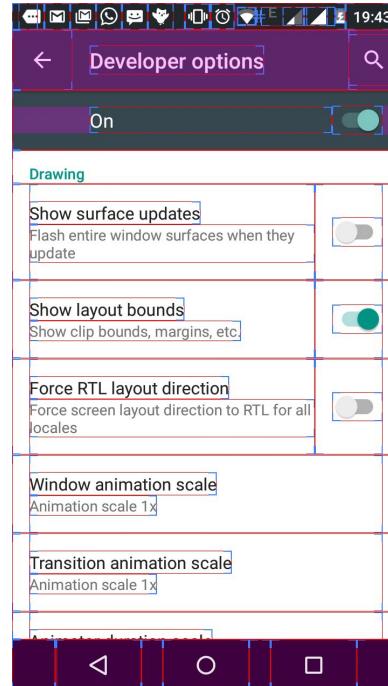


Layout Inspector



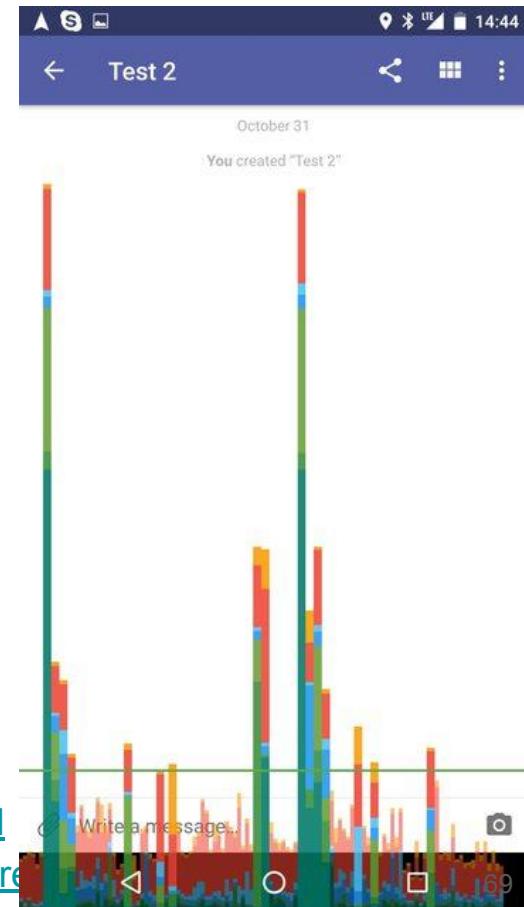
View Debug Tools

Settings -> Developer Options -> Drawing-> Show layout bounds



GPU Profiler

Settings -> Developer Options -> Monitoring -> Profile GPU Rendering

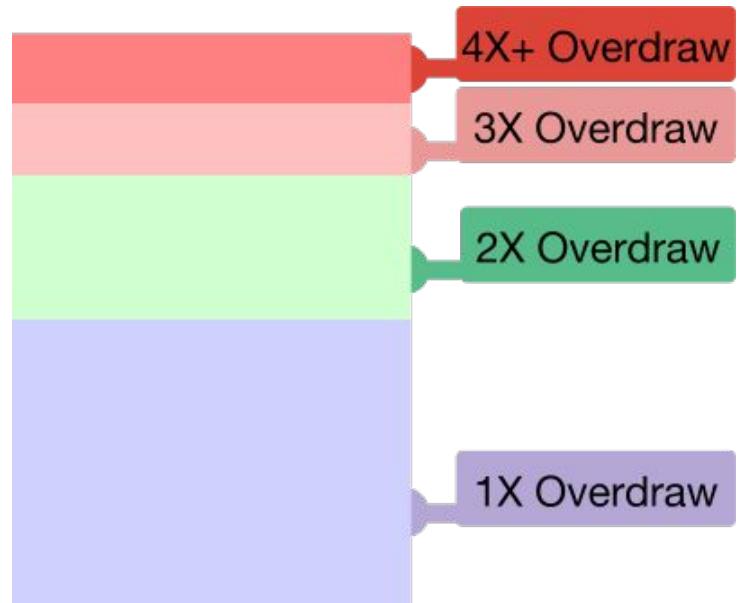
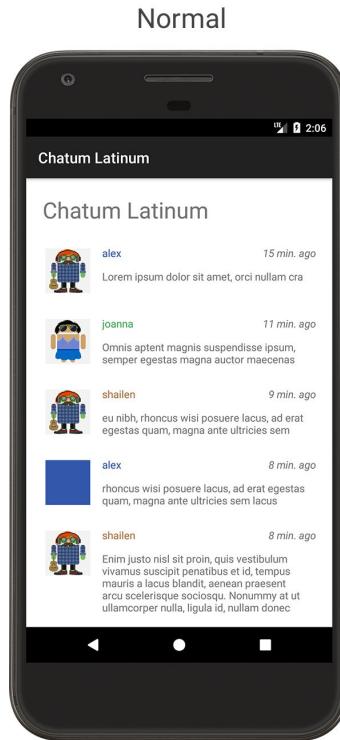


<https://developer.android.com/topic/performance/rendering/profile-gpu.html>

https://developer.android.com/studio/profile/inspect-gpu-rendering#profile_re

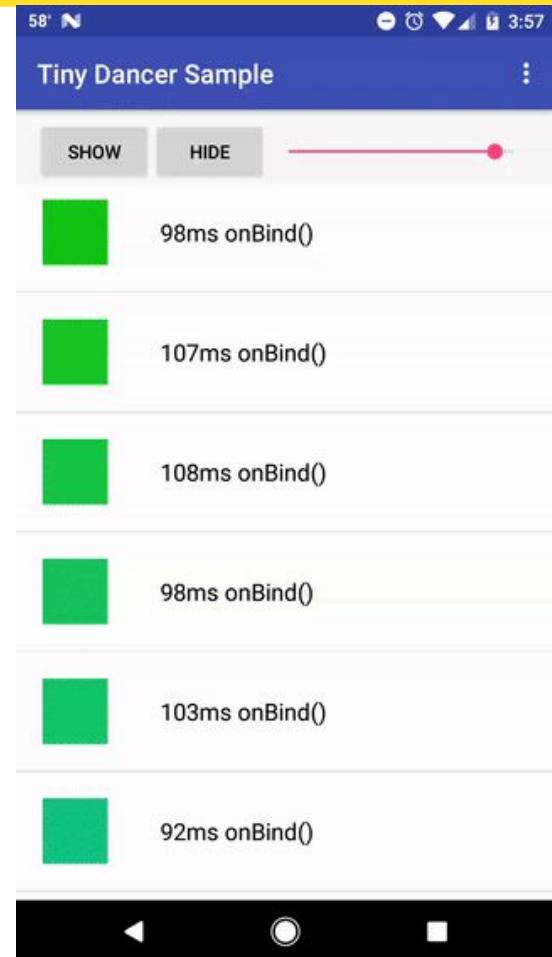
View Debug Tools

Settings -> Developer Options -> Hardware accelerated rendering -> Debug GPU overdraw



TinyDancer

<https://github.com/friendlyrobotnyc/TinyDancer>



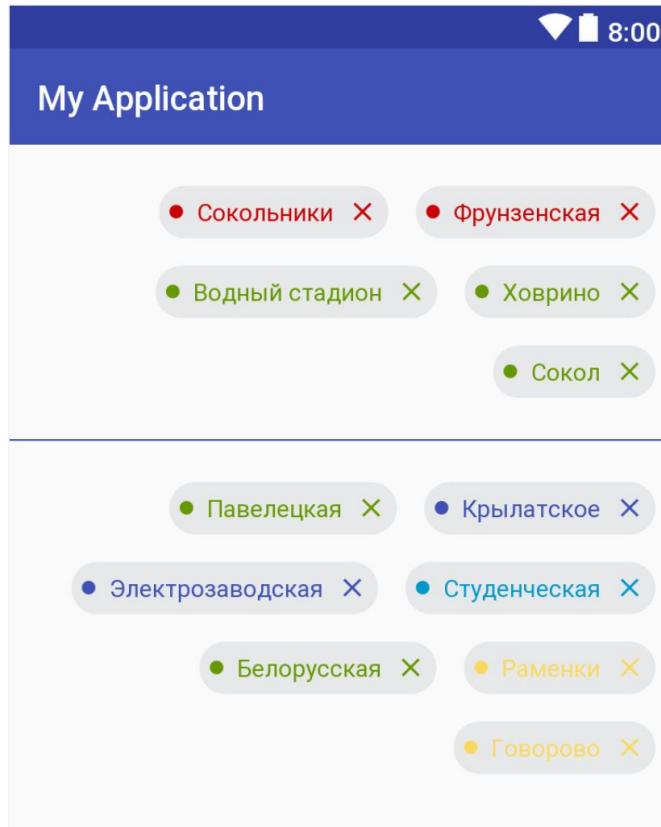
Резюме

- Узнали, что такое Views и ViewGroups
- Научились создавать View из кода и из xml-разметки
- Узнали как происходит отрисовка View и обработка тачей
- Сделали первые шаги в написании кастомных View
- Рассмотрели разные виды Drawable-ресурсов
- Узнали способы профилировки

Ссылки

- <https://developer.android.com/guide/topics/ui/controls.html>
- <http://android-developers.blogspot.ru/2009/03/android-layout-tricks-3-optimize-with.html>
- <https://developer.android.com/training/improving-layouts/optimizing-layout.html>
- <https://developer.android.com/training/animation/index.html>
- <https://developer.android.com/training/transitions/index.html>
- <https://android-developers.blogspot.ru/2016/02/android-support-library-232.html>
- <https://developer.android.com/training/improving-layouts/index.html>
- <https://developer.android.com/training/custom-views/custom-drawing#layoutevent>
- <https://www.udacity.com/course/android-performance--ud825>

Домашнее задание



Написать кастомную View Group, в которую будут добавляться выюхи таким образом: View добавляются в строку. Если длины строки не хватает на размещение view, то view переносится на следующую строчку. View должны быть одинаковые по высоте, но разные по ширине, все view прилипают к правому краю, по нажатию на view она удаляется из первого контейнера и появляется на последнем месте во втором и наоборот.



Спасибо за внимание

Перейдем к практике



Оплата картой ×

Снятие наличных ×

Штрафы ×

Исходящие платежи ×

Платежи в бюджет ×

Налоговые платежи ×

Другое ×

Возврат средств ×

Взнос наличных ×

Начисление процентов ×

Другое ×