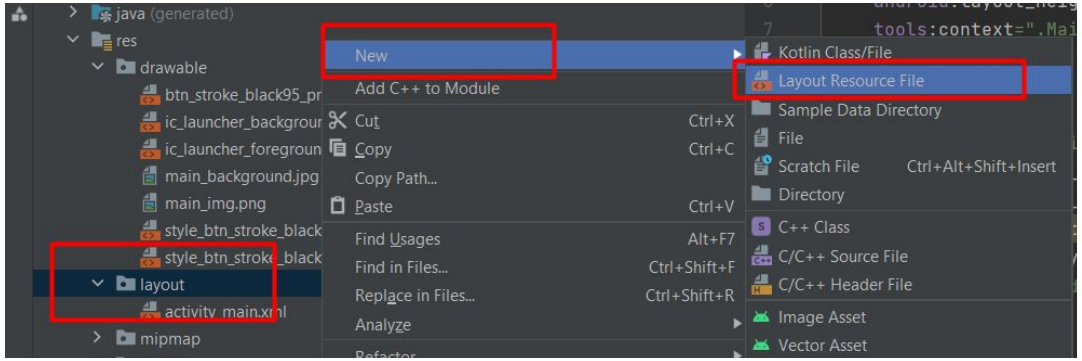
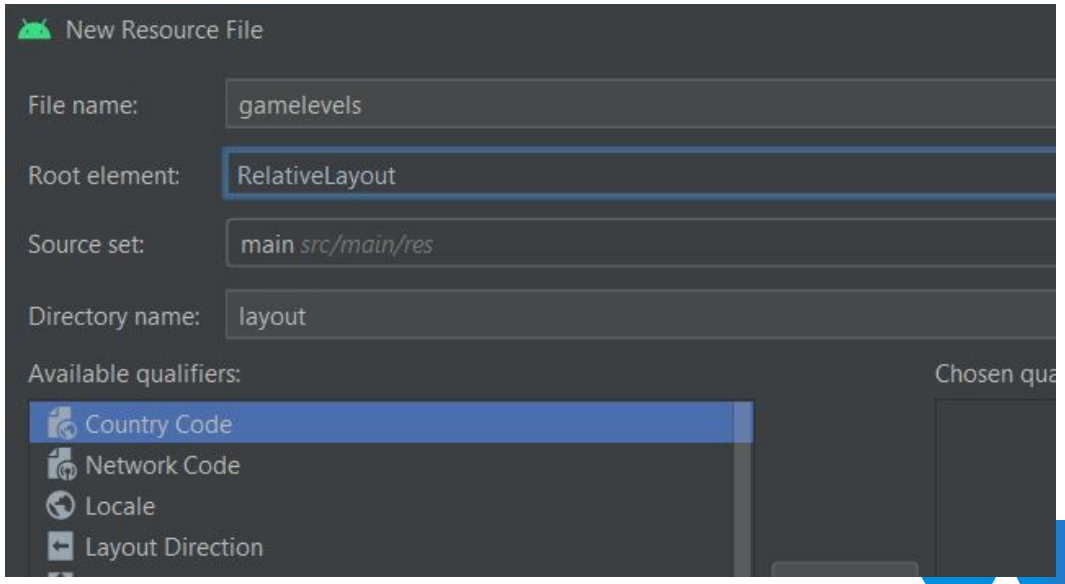
# Выбор уровня игры викторины

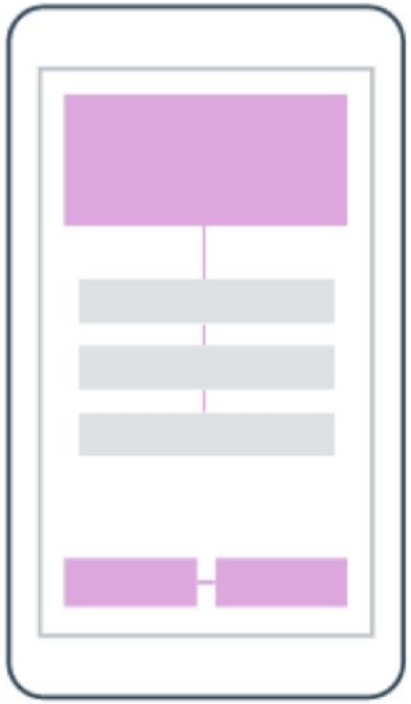
Андройд UI-design мобильного интерфейса

# Создание нового окна



# RelativeLayout

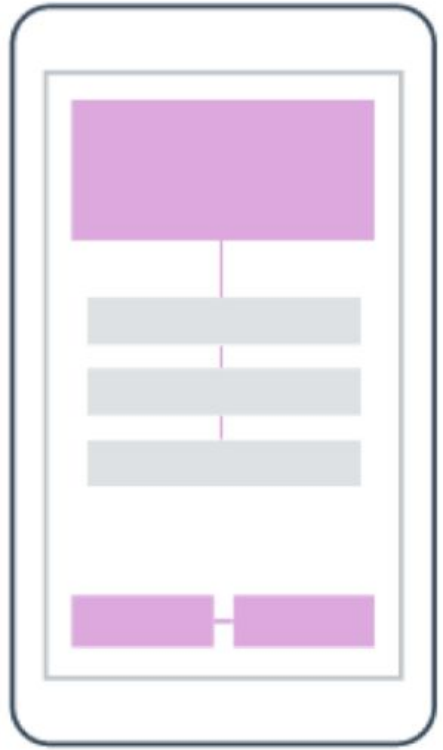


RelativeLayout

**ReiativeLayout** (относительная разметка) находится в разделе **Layouts** и позволяет дочерним компонентам определять свою позицию относительно родительского компонента или относительно соседних дочерних элементов (по идентификатору элемента).

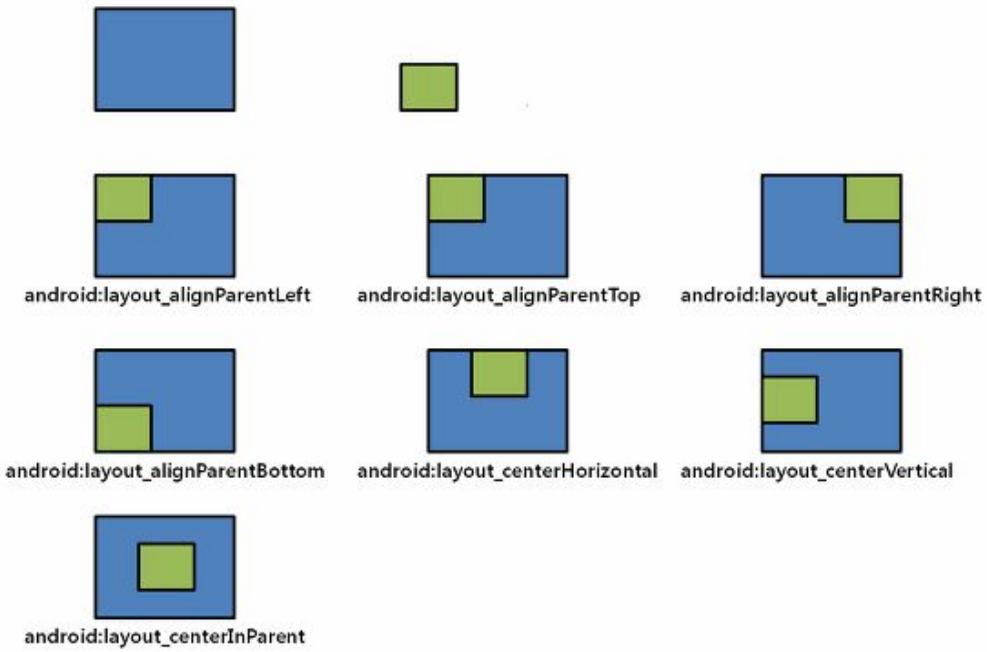
# RelativeLayout

В **RelativeLayout** дочерние элементы расположены так, что если первый элемент расположен по центру экрана, другие элементы, выровненные относительно первого элемента, будут выровнены относительно центра экрана. При таком расположении, при объявлении разметки в XML-файле, элемент, на который будут ссылаться для позиционирования другие объекты представления, должен быть объявлен раньше, чем другие элементы, которые обращаются к нему по его идентификатору.



RelativeLayout

# Атрибуты




**android:layout_alignParentBottom** - выравнивание относительно нижнего края родителя

**android:layout_alignParentLeft** - выравнивание относительно левого края родителя

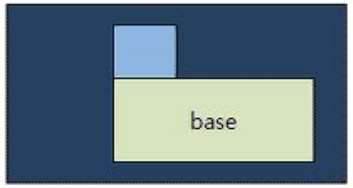
**android:layout_alignParentRight** - выравнивание относительно правого края родителя

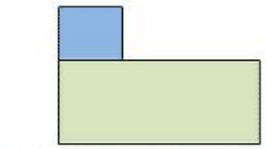
**android:layout_alignParentTop** - выравнивание относительно верхнего края родителя

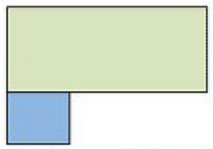
**android:layout_centerInParent** - выравнивание по центру родителя по вертикали и горизонтали

**android:layout_centerHorizontal** - выравнивание по центру родителя по горизонтали

**android:layout_centerVertical** - выравнивание по центру родителя по вертикали

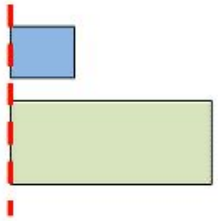


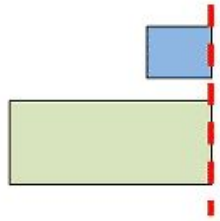


android:layout_above="base"


android:layout_below="base"

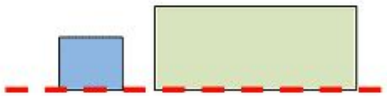

android:layout_toLeftOf="base"


android:layout_toRightOf="base"


android:layout_alignLeft="base"


android:layout_alignRight="base"


android:layout_alignTop="base"


android:layout_alignBottom="base"

# Атрибуты

Компонент можно размещать не только относительно родителя, но и относительно других компонентов. Для этого все компоненты должны иметь свой идентификатор, по которому их можно будет отличать друг от друга. В этом случае вы можете задействовать другие атрибуты.

**android:layout_below** - размещается под указанным компонентом

**android:layout_alignLeft** - выравнивается по левому краю указанного компонента

android:layout_alignRight - выравнивается по правому краю указанного компонента

**android:layout_alignTop** - выравнивается по верхнему краю указанного компонента

**android:layout_alignBottom** - выравнивается по нижнему краю указанного компонента

**android:layout_toLeftOf** - правый край компонента размещается слева от указанного компонента

**android:layout_toRightOf** - левый край компонент размещается справа от указанного компонента

# Атрибуты

Чтобы компоненты "не прилипали" друг к другу, используются атрибуты, добавляющие пространство между ними.

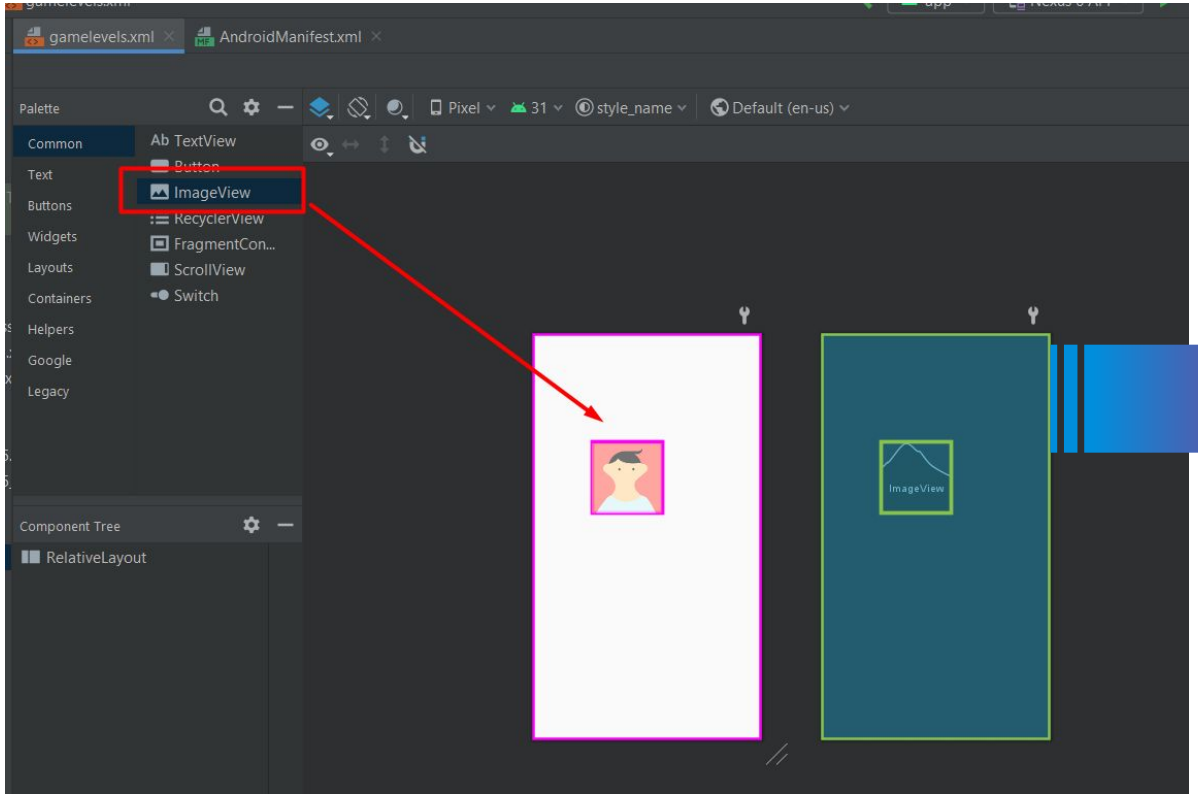
- android:layout_marginTop
- android:layout_marginBottom
- android:layout_marginLeft
- android:layout_marginRight

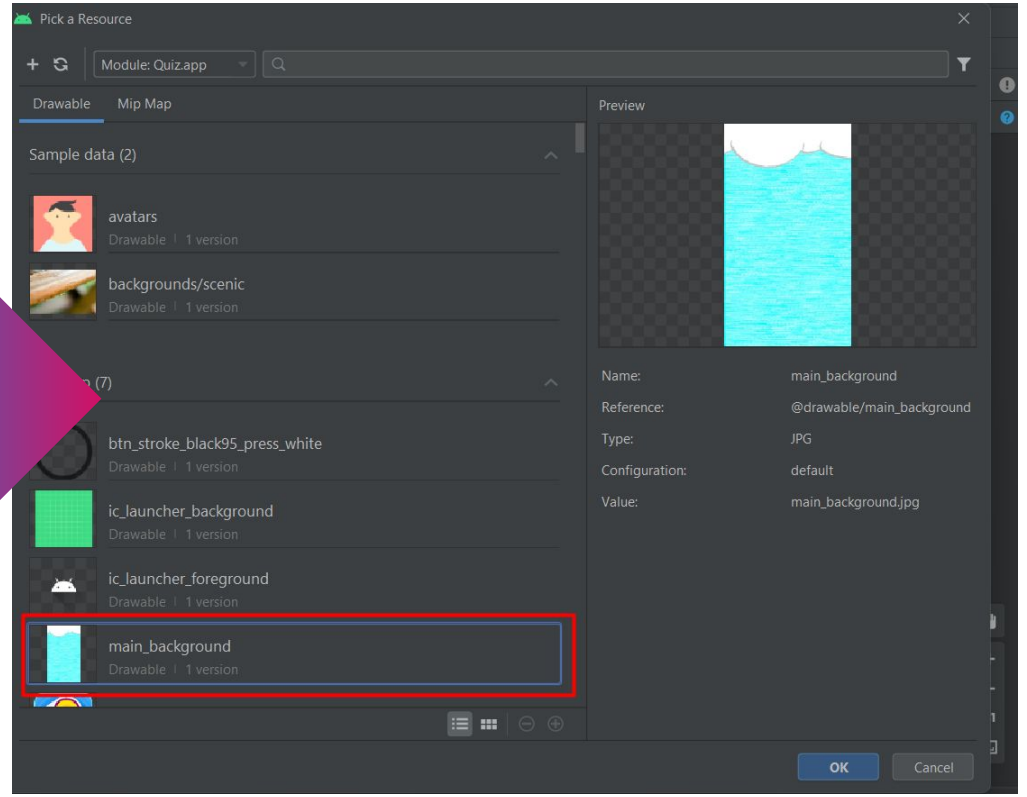
# Вертикальная ориентация

gamelevels.xml | AndroidManifest.xml

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

# Добавление фона



# Добавление фона

**Сделаем так, чтобы фон пропорционально растягивался на весь экран**

```
<ImageView
    android:id="@+id/imageView2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scaleType="centerCrop"
    app:srcCompat="@drawable/main_background" />
```

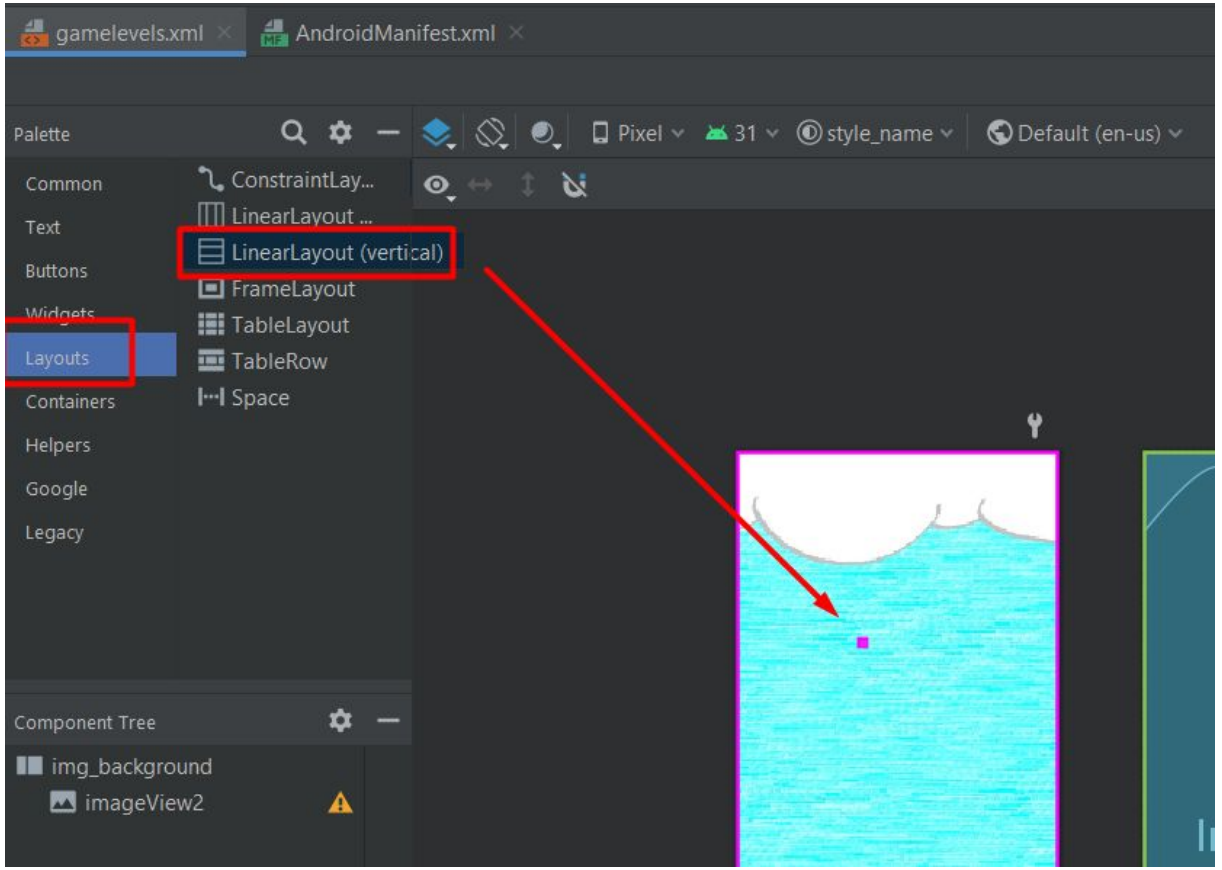
**match_parent** – означает, что элемент займет всю доступную ему в родительском элементе ширину/высоту.

**android:scaleType="centerCrop"** – также размещает картинку в центре, но учитывает ширину или высоту контейнера. Режим попытается сделать так, чтобы ширина (или высота) картинки совпала с шириной (или высотой) контейнера, а остальное обрезается.

# LinearLayout

## Добавим на экран

```
<LinearLayout
    android:id="@+id/container1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

</LinearLayout>
```



# Вертикальная ориентация

```
android:orientation="horizontal"
android:orientation="vertical"
```

В студии
макет **LinearLayout** представлен
двумя вариантами
- **Horizontal** и **Vertical**.
Макет **LinearLayout** выравнивает все дочерние объекты в одном направлении — вертикально или горизонтально. Направление задается при помощи атрибута ориентации **android:orientation**:

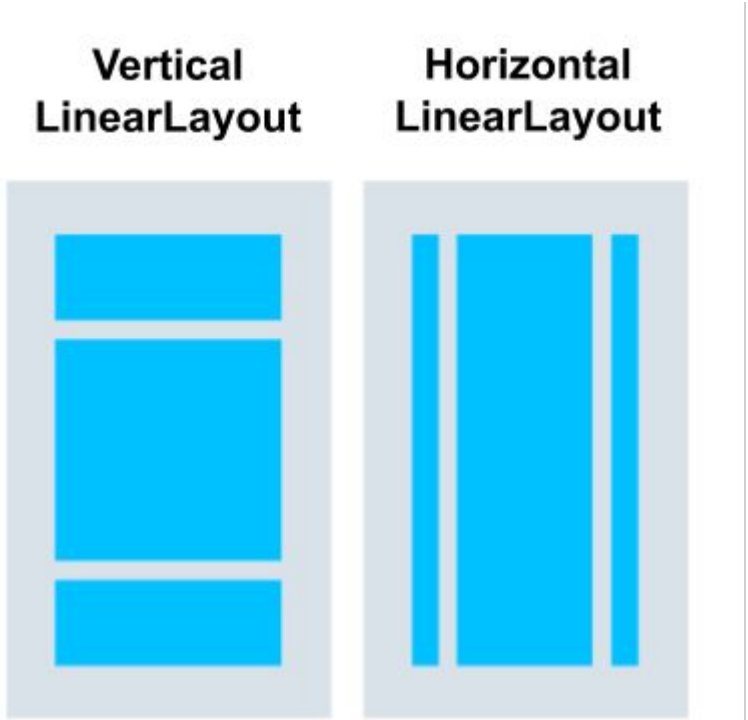
Все дочерние элементы помещаются в стек один за другим, так что вертикальный список компонентов будет иметь только один дочерний элемент в ряду независимо от того, насколько широким он является. Горизонтальное расположение списка будет размещать элементы в одну строку с высотой, равной высоте самого высокого дочернего элемента списка.

# LinearLayout

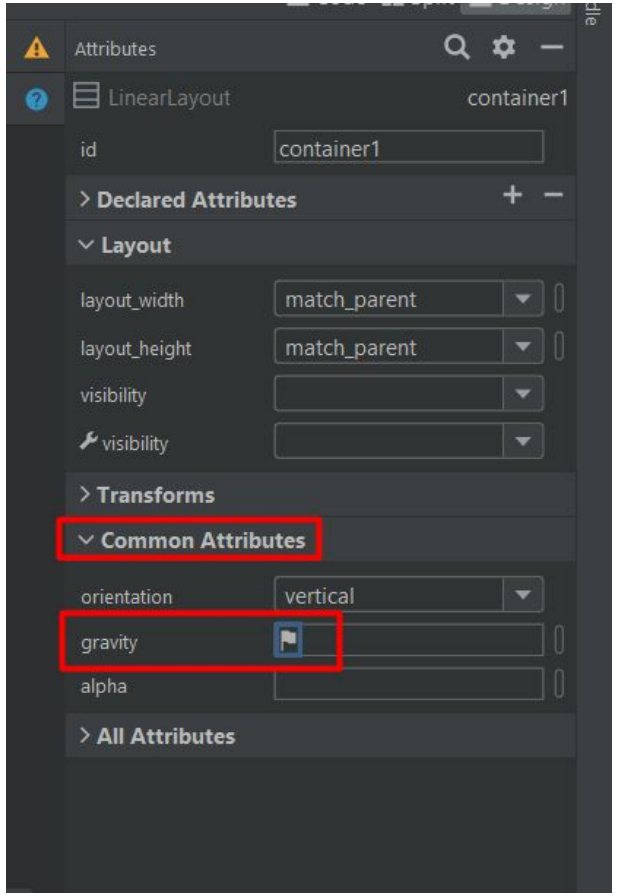
У разметки **LinearLayout** есть интересный атрибут **android:layout_weight**, который назначает индивидуальный вес для дочернего элемента. Этот атрибут определяет "важность" представления и позволяет этому элементу расширяться, чтобы заполнить любое оставшееся пространство в родительском представлении. Заданный по умолчанию вес является нулевым.

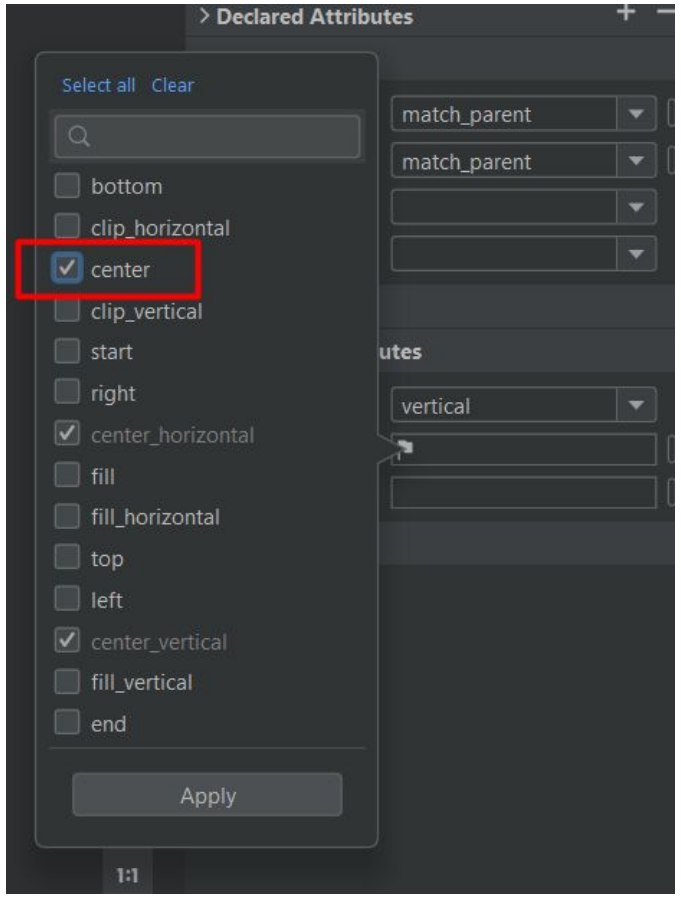
Например, если есть три текстовых поля, и двум из них объявлен вес со значением 1, в то время как другому не даётся никакого веса (0), третье текстовое поле без веса не будет расширяться и займёт область, определяемую размером текста, отображаемого этим полем. Другие два расширятся одинаково, чтобы заполнить остаток пространства, не занятого третьим полем. Если третьему полю присвоить вес 2 (вместо 0), это поле будет объявлено как "более важное", чем два других, так что третье поле получит 50% общего пространства, в то время как первые два получат по 25% общего пространства.

Также можно указать атрибут **android:weightSum**. Если атрибуту присвоить значение 100, то можно указывать вес дочерних элементов в удобном виде, как в процентах. Такой способ широко используется веб-мастерами при вёрстке.



# LinearLayout (container1)

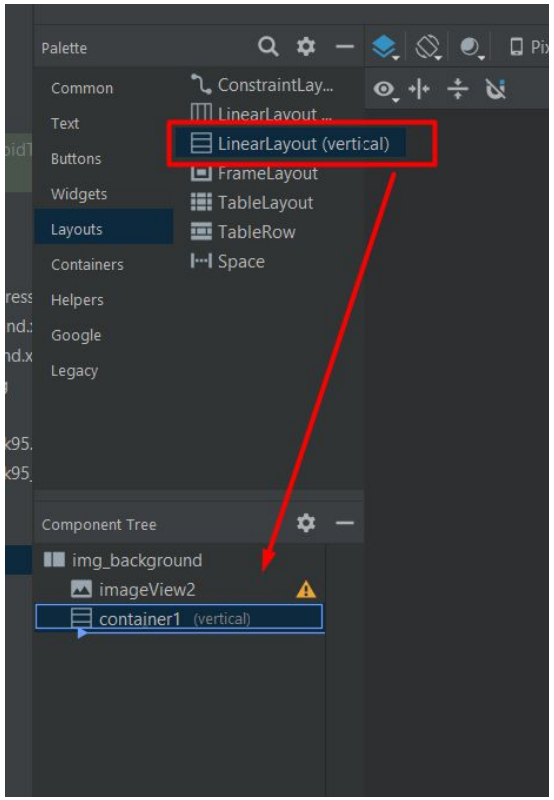



Атрибут **gravity** задает позиционирование содержимого внутри визуального элемента.

**center** – элементы размещаются по центру

# LinearLayout (вложенный container2)




```
<LinearLayout
    android:id="@+id/container1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <LinearLayout
        android:id="@+id/container2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

    </LinearLayout>
```

**wrap_content** – установка размера представления в w**rap_content** заставит его расширяться только настолько, чтобы содержать значения (или дочерние элементы управления), которые он содержит. Для элементов управления - таких как текстовые поля **(TextView)** или изображения **(ImageView)** – это обернет отображаемый текст или изображение. Для элементов макета он изменит размер макета в соответствии с элементами управления / макетами, добавленными в качестве его дочерних элементов.

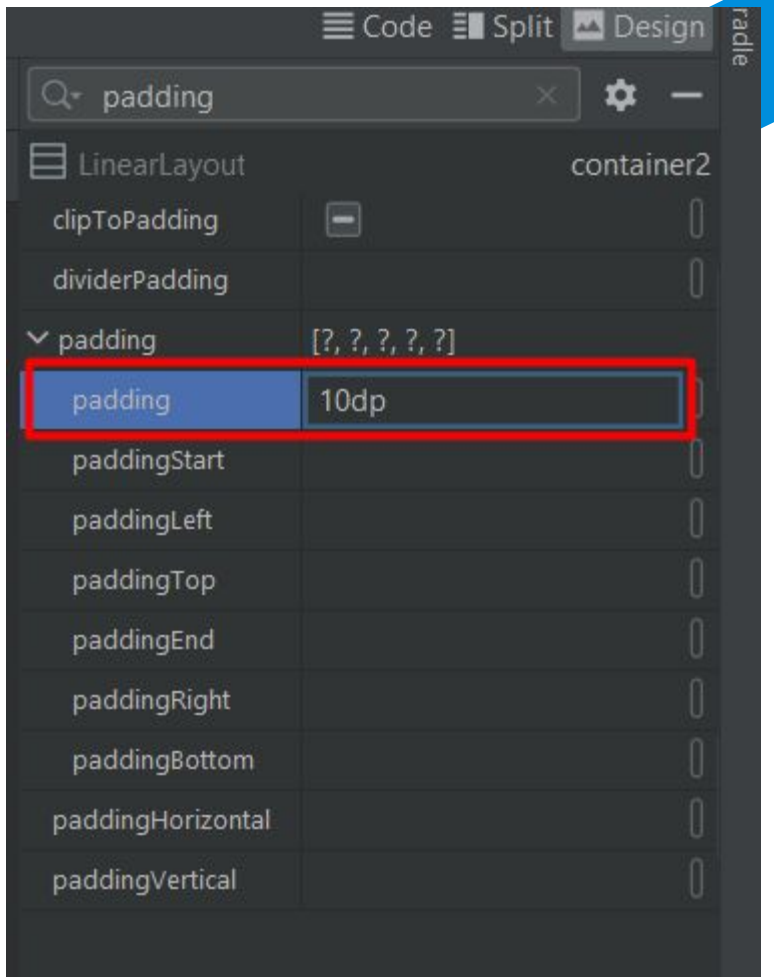
# padding

Для установки внутренних отступов применяется атрибут **android:padding**.

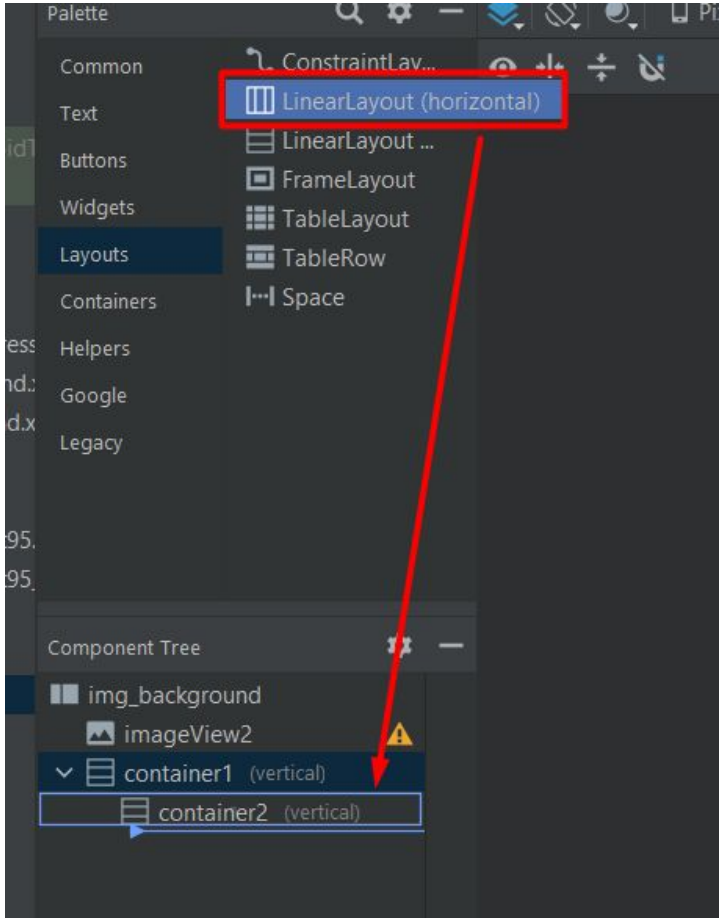
Он устанавливает отступы контента от всех четырех сторон контейнера.

Можно устанавливать отступы только от одной стороны контейнера, применяя следующие атрибуты:

- **android:paddingLeft**
- **android:paddingRight**
- **android:paddingTop**
- **android:paddingBottom**.



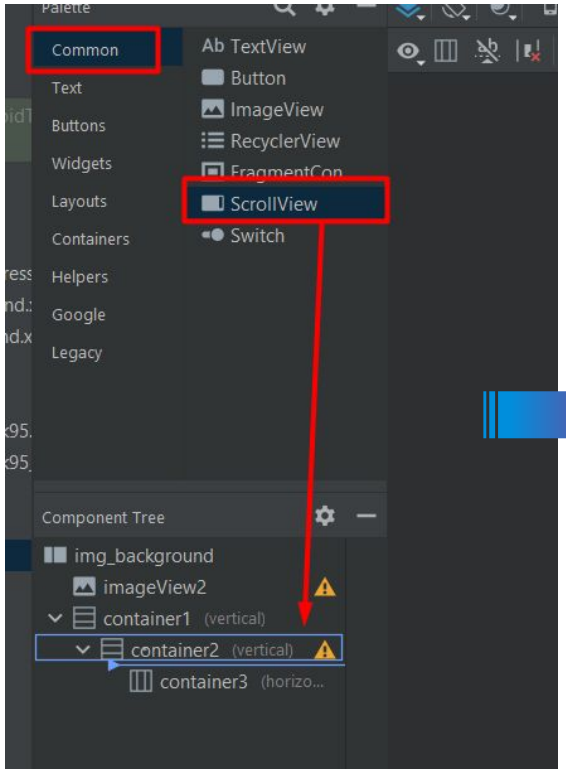
# LinearLayout (вложенный container3)

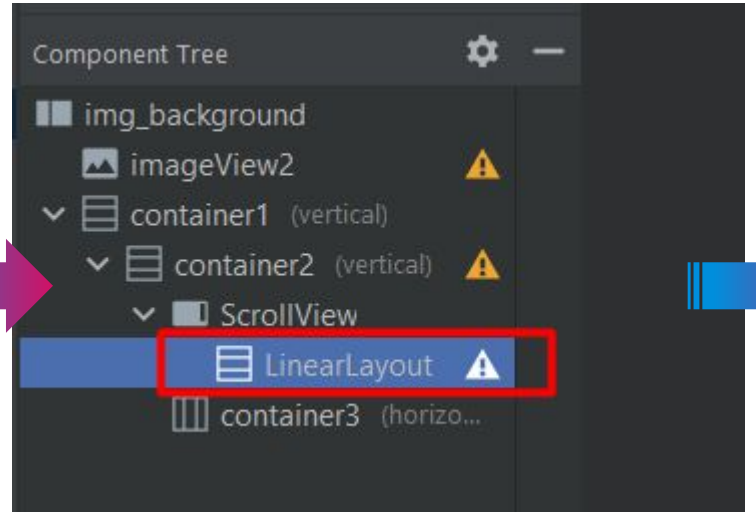



```
<LinearLayout
    android:id="@+id/container1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <LinearLayout
        android:id="@+id/container2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="10dp">

        <LinearLayout
            android:id="@+id/container3"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="horizontal">

        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

# ScrollView

- Добавим возможность прокрутки ScrollView



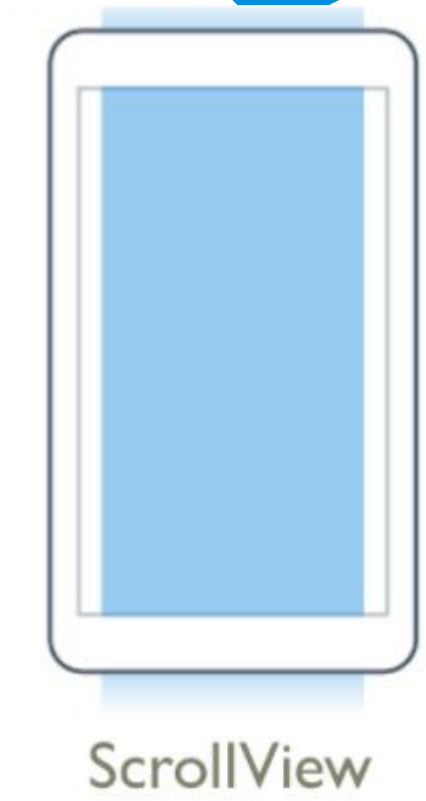

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/container4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" />
</ScrollView>
```

# ScrollView

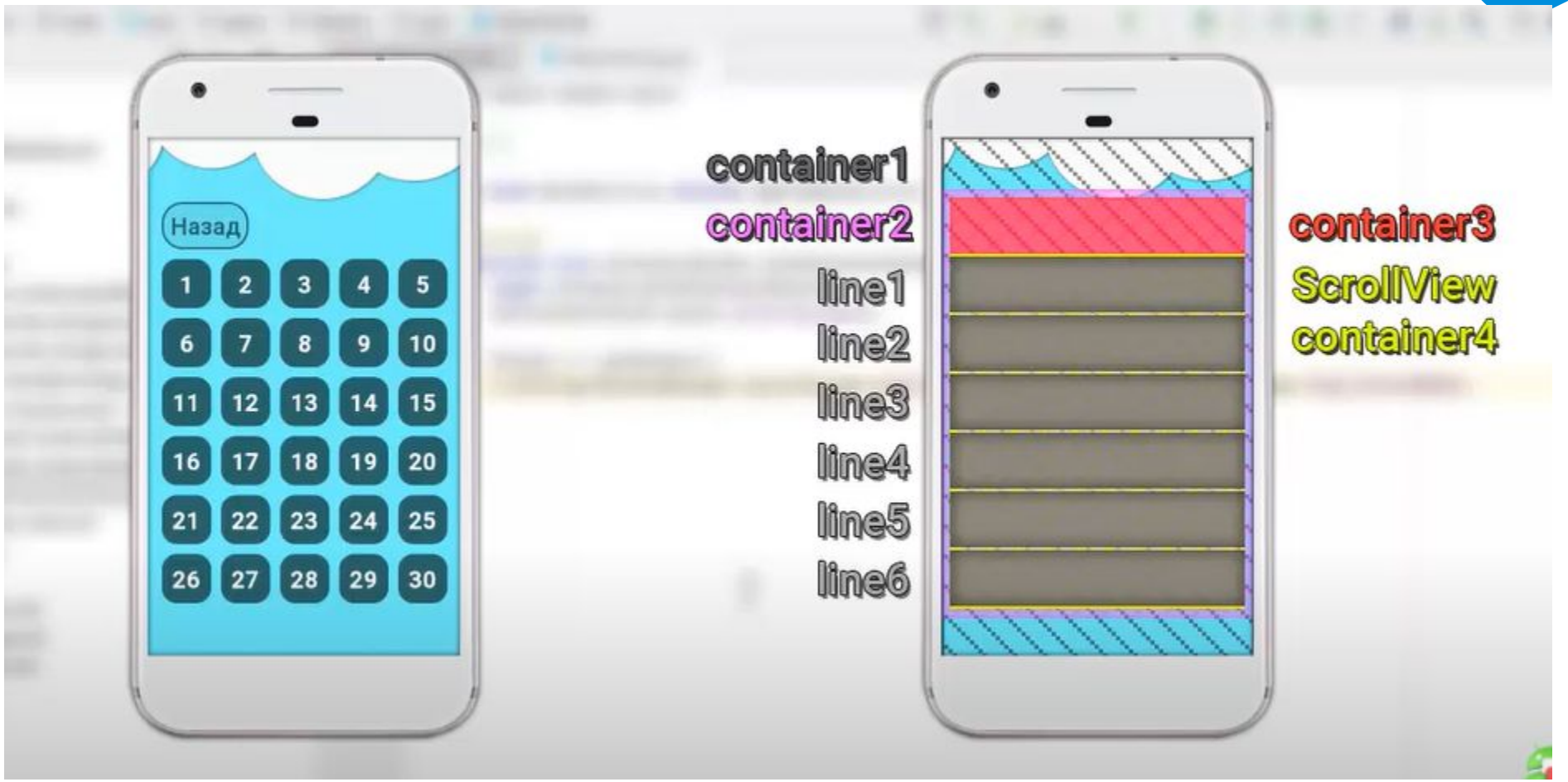
При большом количестве информации, которую нужно поместить на экране приходится использовать полосы прокрутки. В Android существуют специальные компоненты **ScrollView** и **HorizontalScrollView**, которые являются контейнерными элементами и наследуются от **ViewGroup**. Обратите внимание, что класс **TextView** использует свою собственную прокрутку и не нуждается в добавлении отдельных полос прокрутки. Но использование отдельных полос даже с **TextView** может улучшить вид вашего приложения и повышает удобство работы для пользователя.

На панели инструментов компоненты можно найти в разделе **Containers**.

В контейнеры **ScrollView** и **HorizontalScrollView** можно размещать только один дочерний элемент (обычно **LinearLayout**), который в свою очередь может быть контейнером для других элементов. Виджет **ScrollView**, несмотря на свое название, поддерживает только вертикальную прокрутку, поэтому для создания вертикальной и горизонтальной прокрутки необходимо использовать **ScrollView** в сочетании с **HorizontalScrollView**. Обычно **ScrollView** используют в качестве корневого элемента, а **HorizontalScrollView** в качестве дочернего.

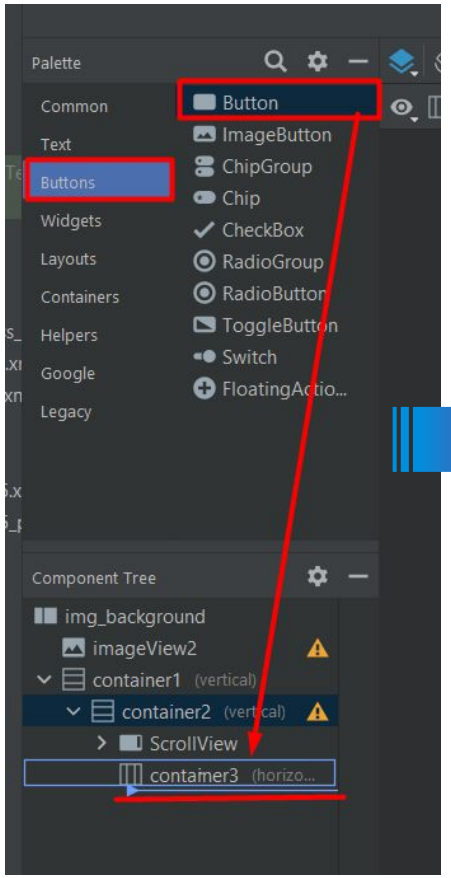


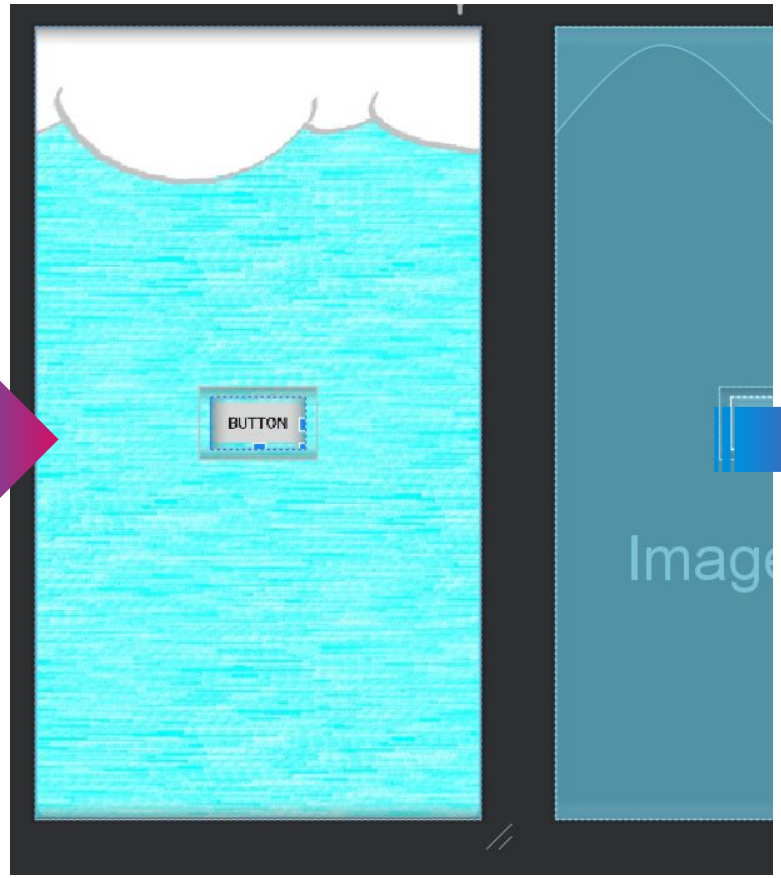
# Расположение container (1,2,3)



# Button

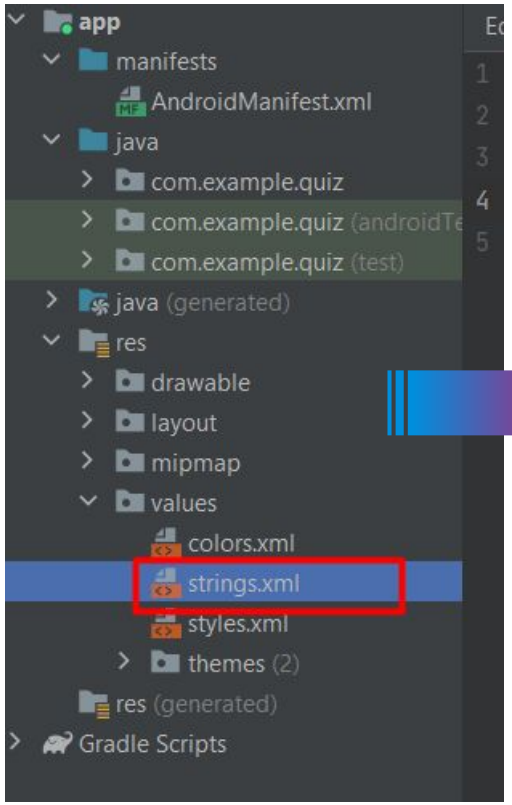
- Добавим кнопку
- Изменим ID
- Удалим вес



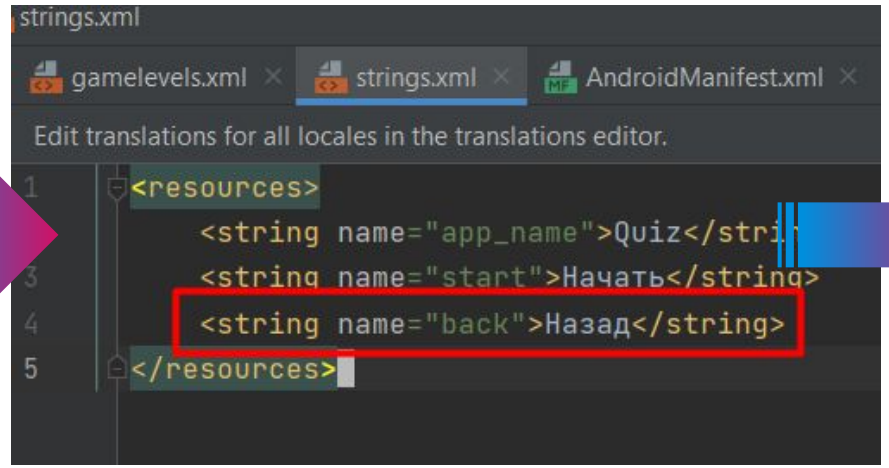




```
<LinearLayout
    android:id="@+id/container3"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <Button
        android:id="@+id/button_back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button" />
</LinearLayout>
```

# strings.xml




- Добавим текст для кнопки «Назад»




# Стиль для кнопки «Назад»

- Применим НАШ стиль для кнопки «Назад»

```
<Button
    android:id="@+id/button_back"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/btn_stroke_black95_press_white"
    android:text="@string/back" />
inearLayout>
```

- Укажем размер текста

```
<Button
    android:id="@+id/button_back"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/btn_str
    android:text="@string/back"
    android:textSize="24sp" />
```

# Кнопка «Назад»

- Укажем отступы

```
<Button
    android:id="@+id/button_back"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/btn_stroke
    android:text="@string/back"
    android:textSize="24sp"
    android:paddingLeft="5dp"
    android:paddingRight="5dp"/>
/LinearLayout>
```

- Укажем цвет текста

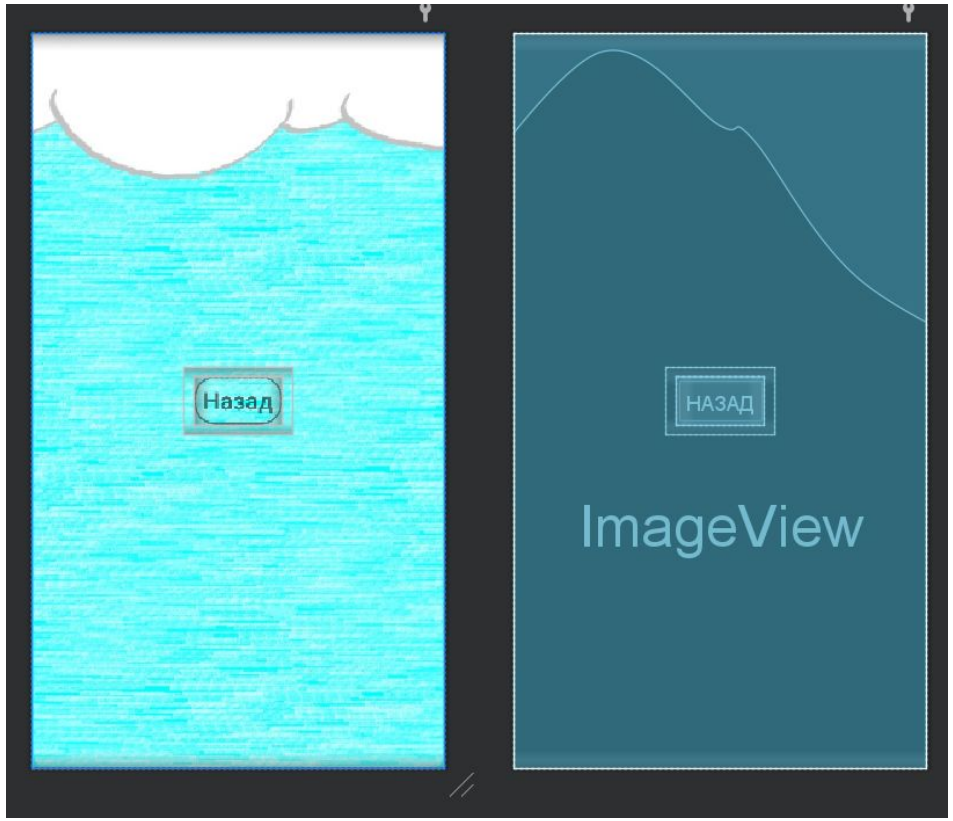
```
<Button
    android:id="@+id/button_back"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/btn_stroke_bl
    android:text="@string/back"
    android:textSize="24sp"
    android:paddingLeft="5dp"
    android:paddingRight="5dp"
    android:textColor="@color/black"/>
/LinearLayout>
```

# Кнопка «Назад»

- Уберем ВЕРХНИЙ регистр

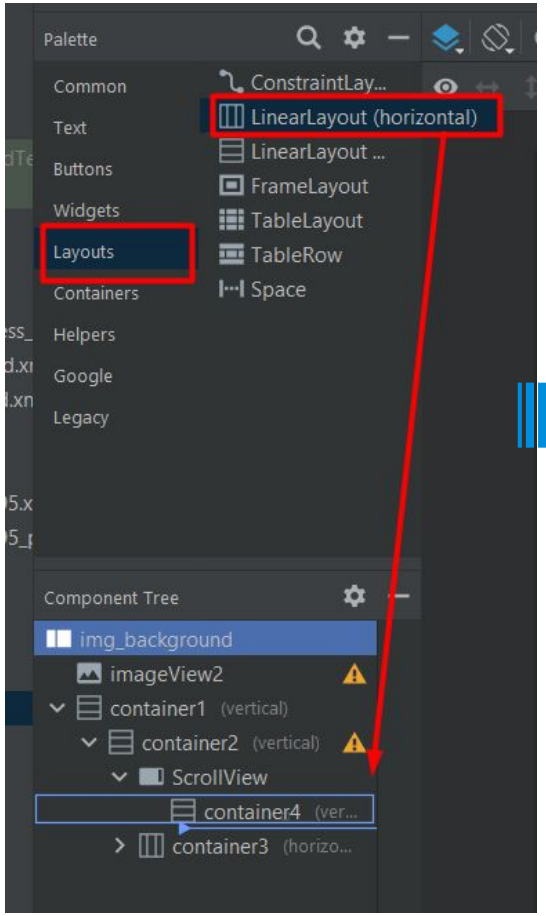
```
<Button
    android:id="@+id/button_back"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/btn_strok
    android:text="@string/back"
    android:textSize="24sp"
    android:paddingLeft="5dp"
    android:paddingRight="5dp"
    android:textColor="@color/black"
    android:textAllCaps="false" />
/LinearLayout>
```

- Кнопка ГОТОВА!!!



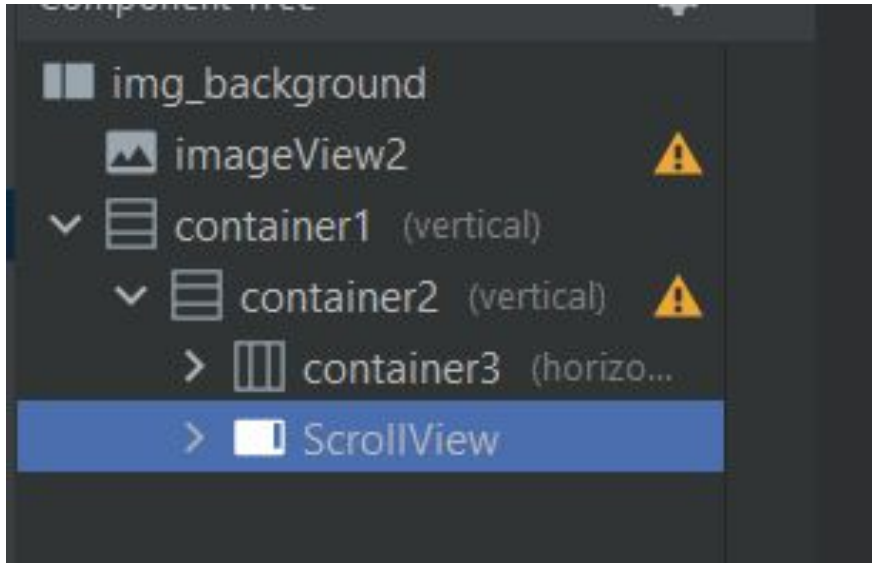
# Уровни

- Разделим по 5 уровней в одной линии, для этого нам потребуется новый контейнер



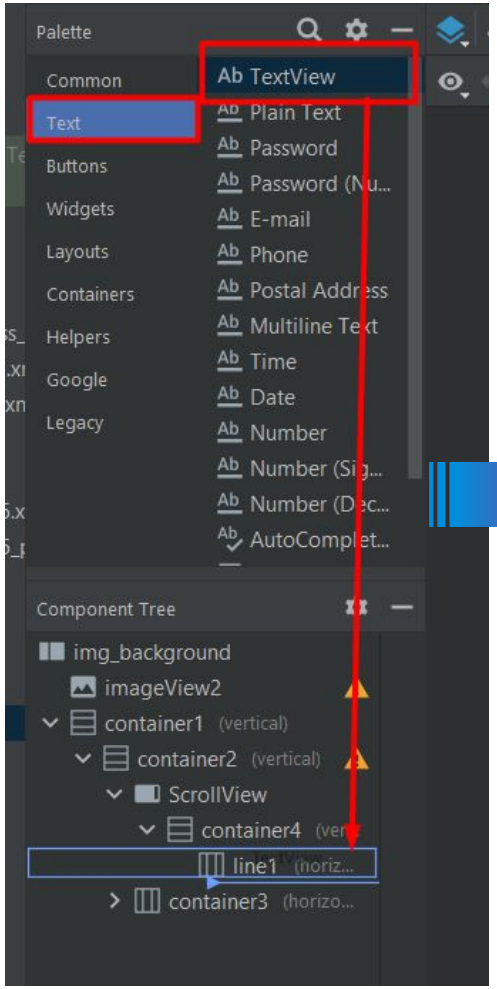

```
<LinearLayout
    android:id="@+id/container4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <LinearLayout
        android:id="@+id/line1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
    </LinearLayout>
```

# Расположение контейнеров



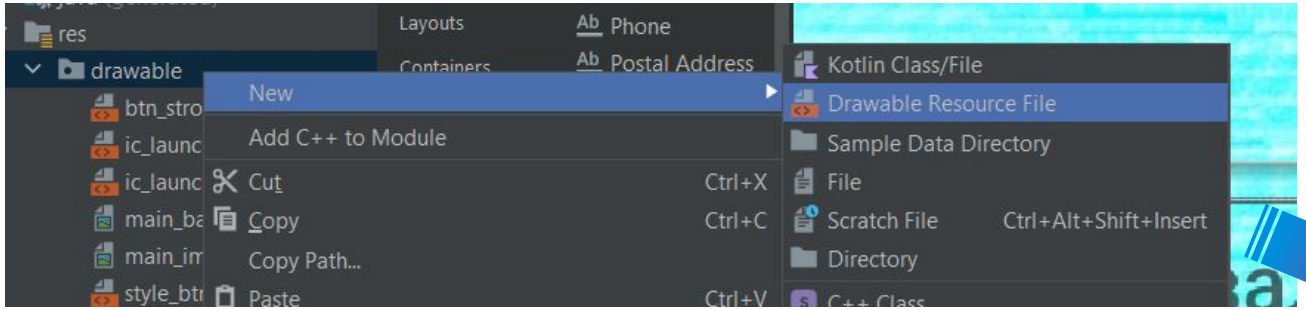
# Уровни

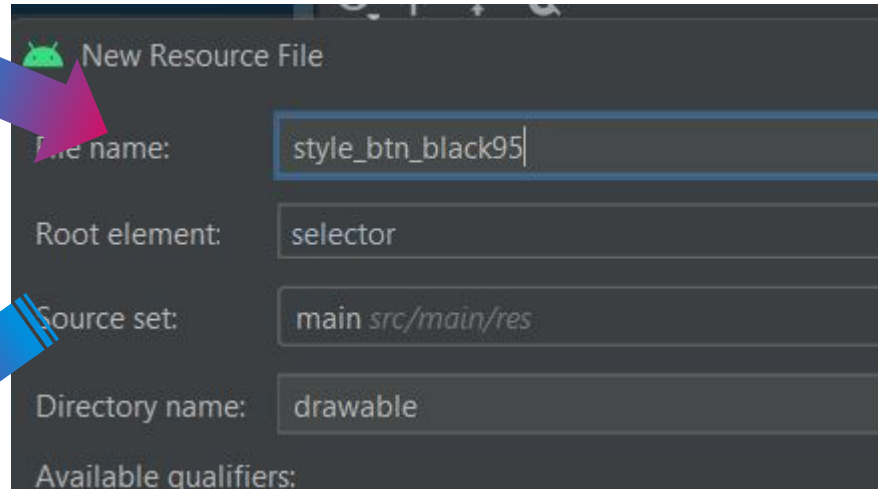
- Кнопкой может быть и текст и картинка




```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/container4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <LinearLayout
            android:id="@+id/line1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="TextView" />
        </LinearLayout>
    </LinearLayout>
</ScrollView>
```

# Стиль для Уровней (не нажата)

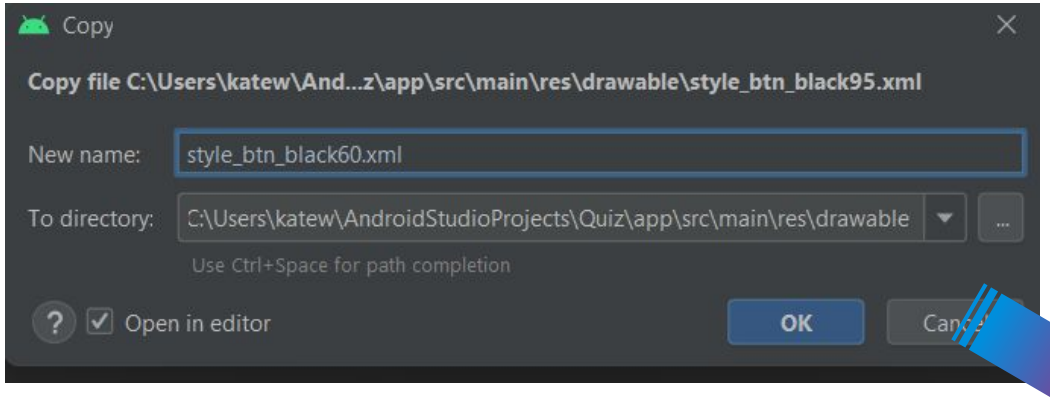



```xml
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="15dp">
    </corners>
    <solid android:color="@color/black">
    </solid>
</shape>
```

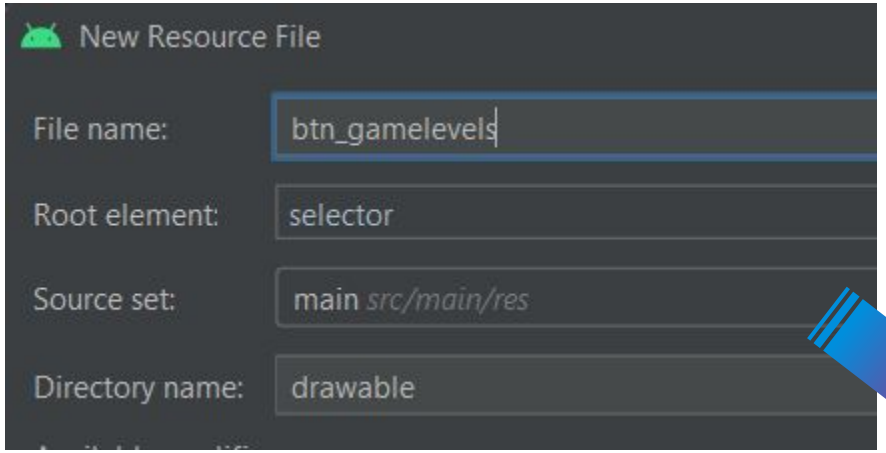
# Стиль для Уровней (нажата)

- Скопируем стиль для НЕНАЖАТОГО уровня и вставим в папку Drawable




```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="15dp">
    </corners>
    <solid android:color="@color/black60">
    </solid>
</shape>
```

# Единый стиль для уровней




```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="false"
        android:drawable="@drawable/style_btn_black95">
    </item>
    <item android:state_pressed="true"
        android:drawable="@drawable/style_btn_black60">
    </item>
</selector>
```

# Уровни

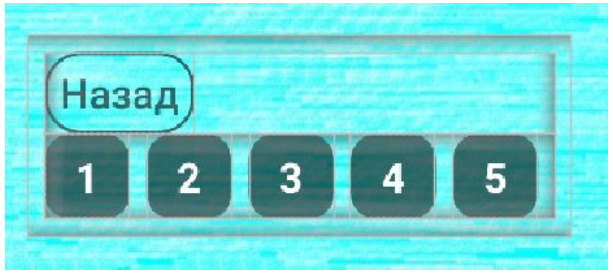
```
<LinearLayout
    android:id="@+id/line1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_marginRight="10dp"
        android:background="@drawable/btn_gamelevels"
        android:gravity="center"
        android:text="1"
        android:textColor="@color/white"
        android:textSize="24sp"
        android:textStyle="bold" />
</LinearLayout>
```

# Уровни (line1)

TextView нужно скопировать и вставить ниже. Изменить id и текст

Повторить это действие еще три раза.

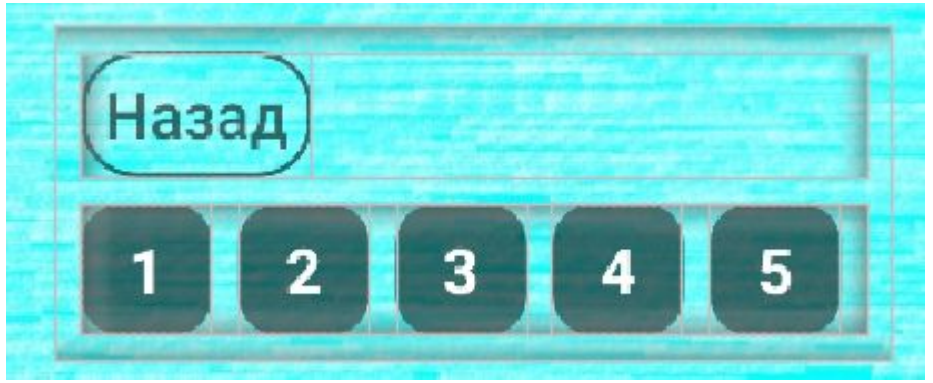
Должо быть 5 TextView




```
<TextView
    android:id="@+id/textView1"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginRight="10dp"
    android:background="@drawable/btn_gamelevels"
    android:gravity="center"
    android:text="1"
    android:textColor="@color/white"
    android:textSize="24sp"
    android:textStyle="bold" />
<TextView
    android:id="@+id/textView2"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginRight="10dp"
    android:background="@drawable/btn_gamelevels"
    android:gravity="center"
    android:text="2"
    android:textColor="@color/white"
    android:textSize="24sp"
    android:textStyle="bold" />
</LinearLayout>
```

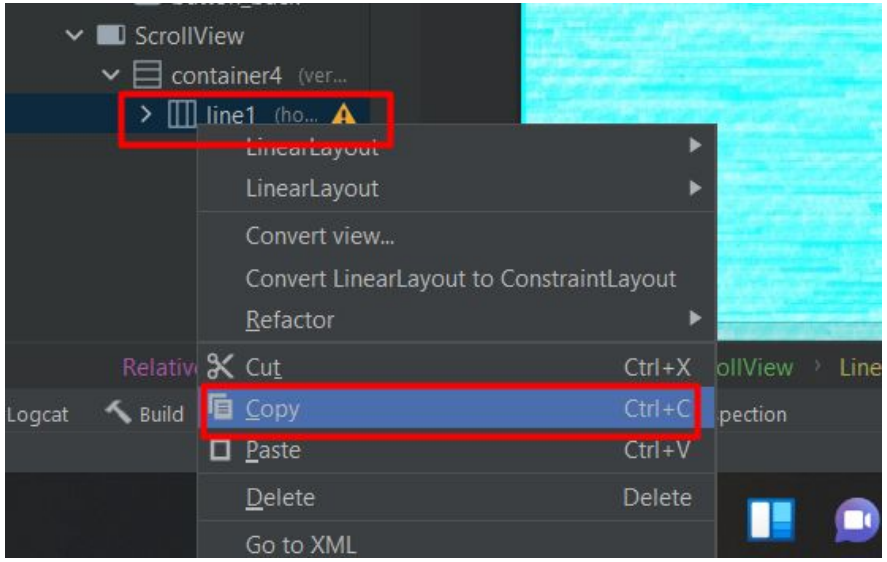
# Отступ от кнопки «Назад»

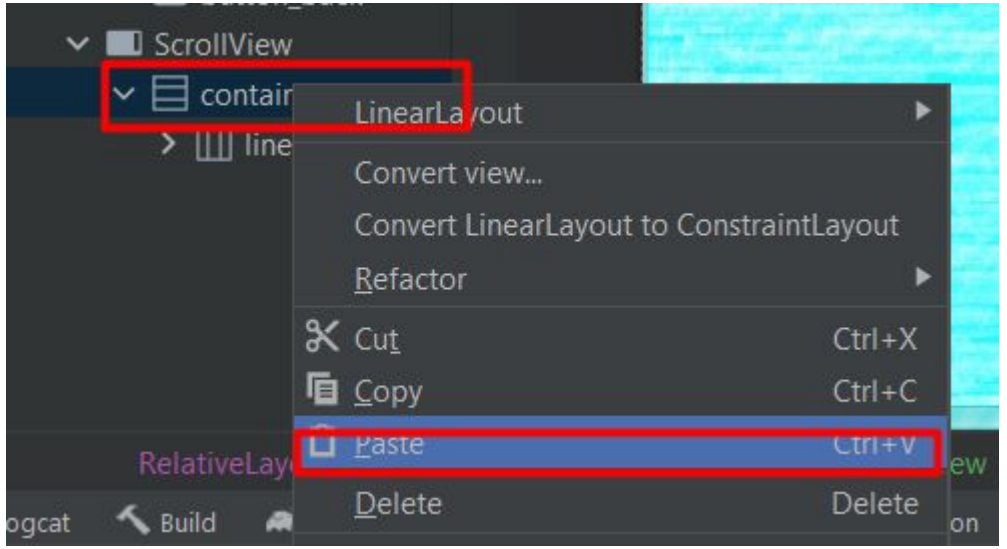
```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="10dp">

    <LinearLayout
        android:id="@+id/container4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
```



# Уровни (line2)


- Скопируем контейнер line1
- Вставим в container4
- Изменим поля id и text вручную



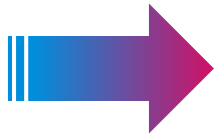
# Уровни (line2)

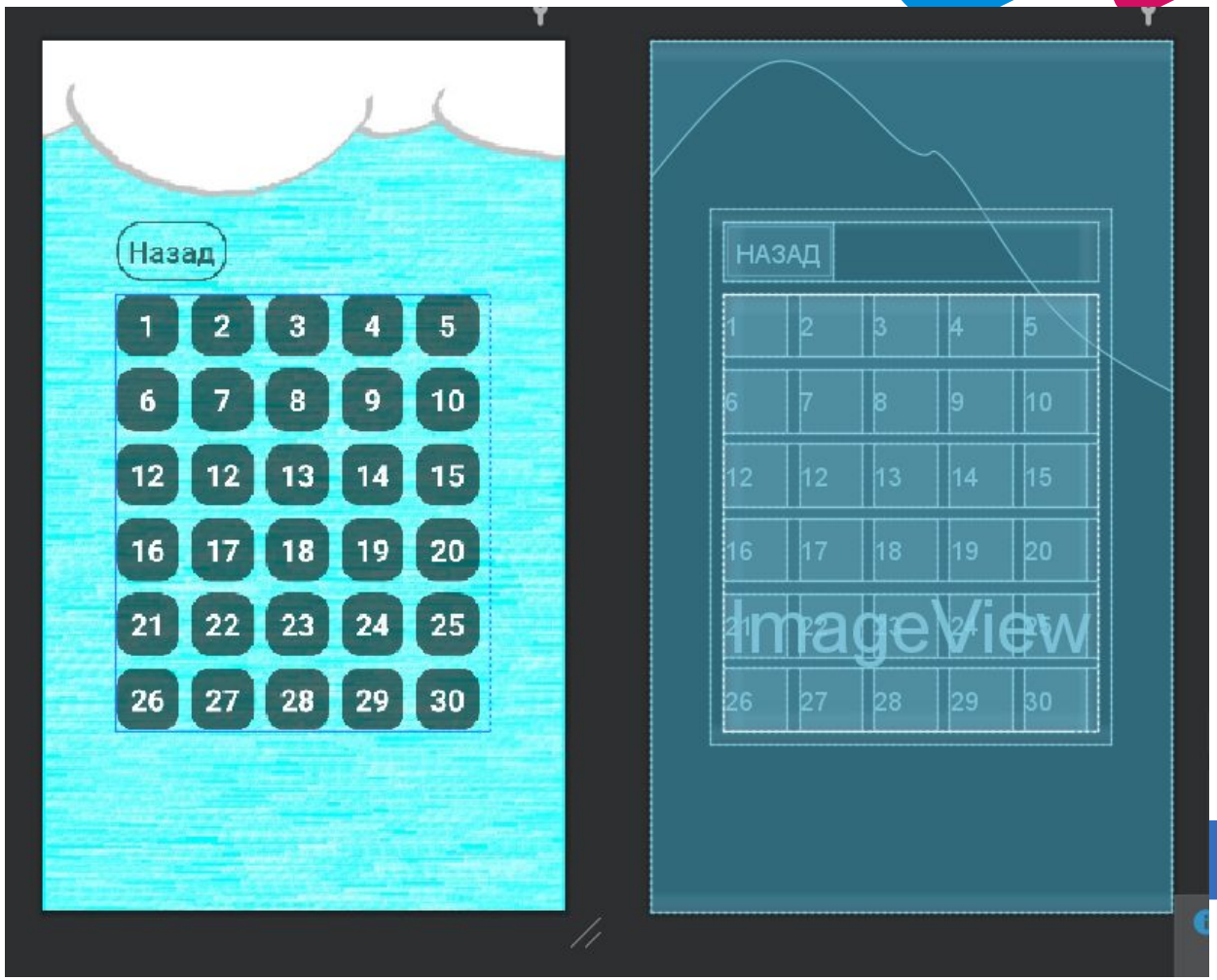
- Добавим отступ
- Повторим еще 4 раза (должно быть 6 line и 30 уровней)

```
<LinearLayout
    android:id="@+id/line2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="10dp">
```

# Уровни

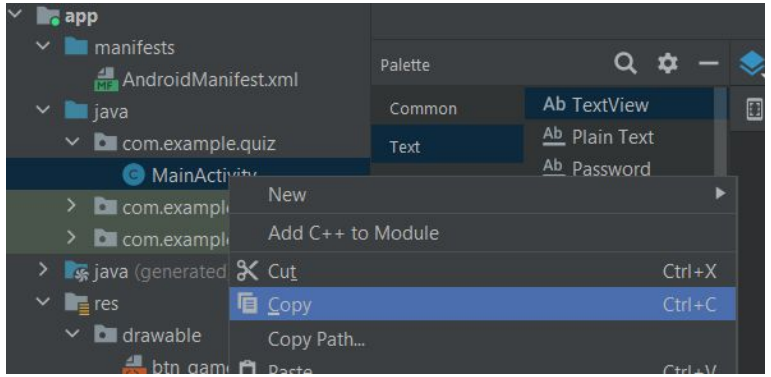
```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="10dp">

    <LinearLayout
        android:id="@+id/container4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <LinearLayout...>

        <LinearLayout...>

        <LinearLayout...>

        <LinearLayout...>

        <LinearLayout...>

        <LinearLayout...>
    </LinearLayout>
</ScrollView>
```

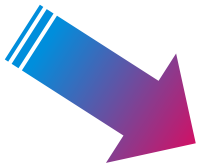


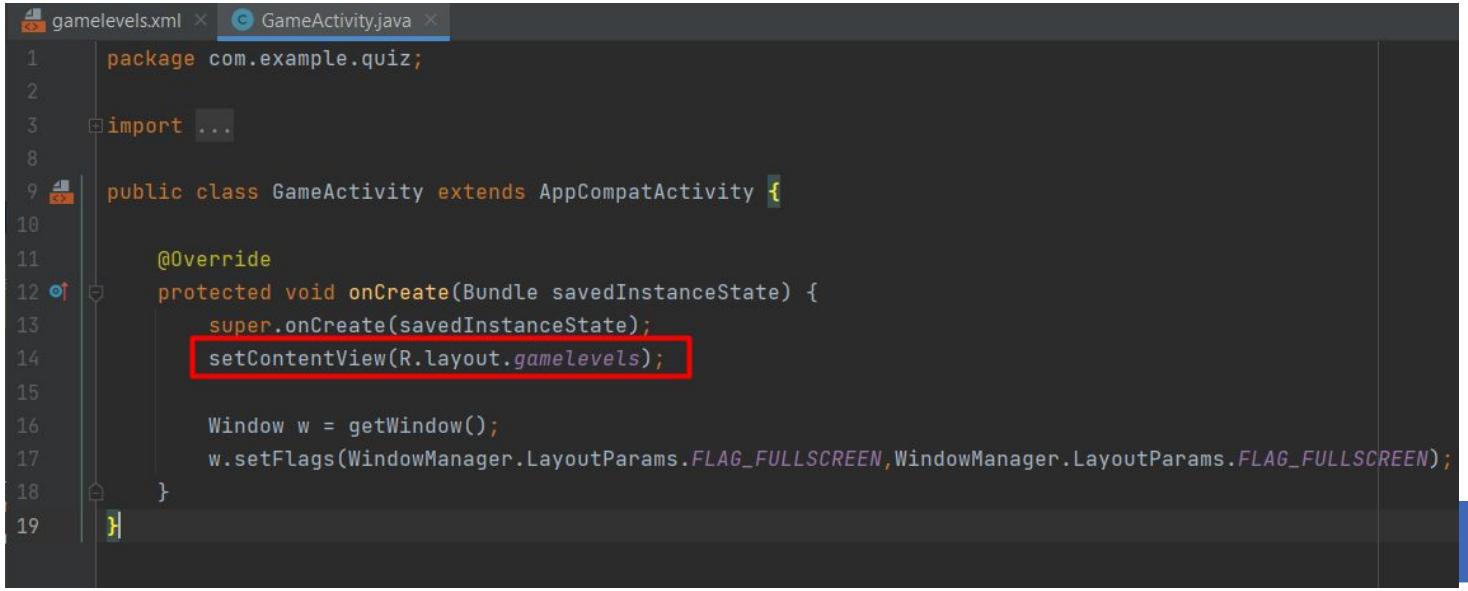


# GameLevels.java

- Скопируйте MainActivity.java
- Вставьте в ту же папку
- Назовите GameLevels





# AndroidManifest.xml

- Откройте AndroidManifest.xml
- Скопируем описание первого Activity и вставим ниже
- <action android:name="android.intent.action.MAIN" /> удаляем, так как этот атрибут только для файла, который открывается первым

AndroidManifest.xml

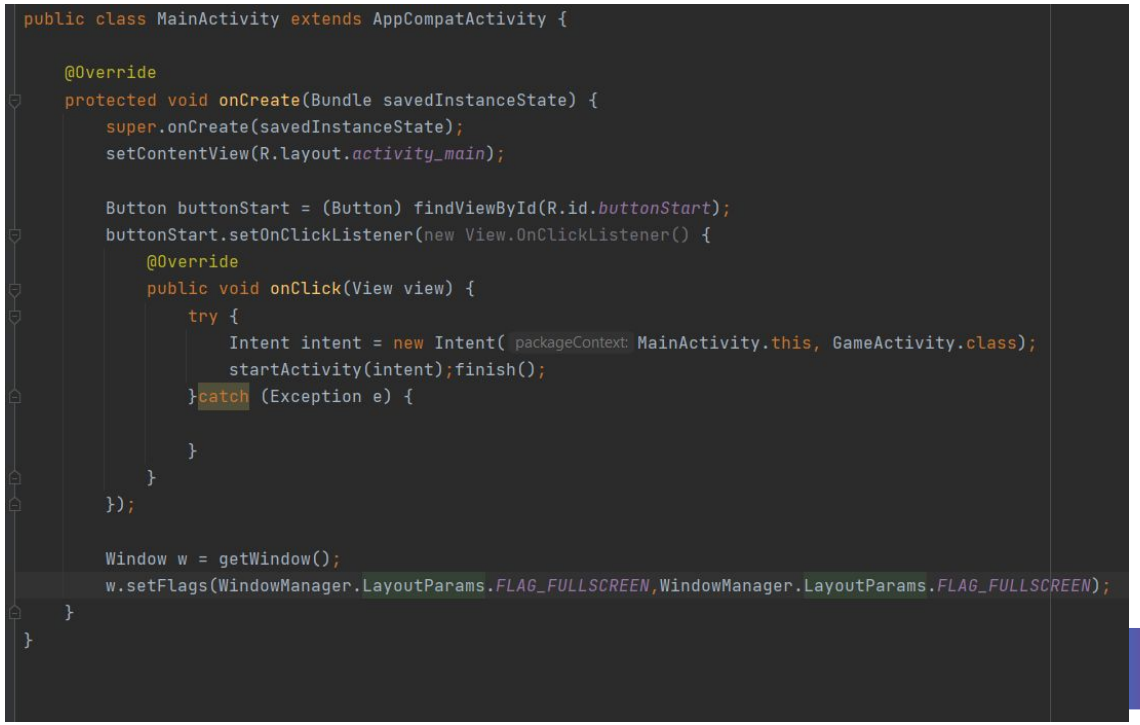
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.quiz">

    <application
        android:allowBackup="true"
        android:icon="@drawable/main_img"
        android:label="Quiz"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/style_name">
        <activity
            android:name=".MainActivity"
            android:screenOrientation="portrait"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

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

        <activity
            android:name=".GameActivity"
            android:screenOrientation="portrait"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

# Переход из главного меню в Уровням

- Откройте MainActivity.java
- Запишем код, который дает перейти на другой уровней при нажатии кнопки с обработчиком ошибок.

```
Button buttonStart = (Button) findViewById(R.id.buttonStart);
        buttonStart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                try {
                    Intent intent = new Intent(MainActivity.this,
GameActivity.class);
                    startActivity(intent);finish();
                }catch (Exception e) {


                }
            }
        });
```



```
public class MainActivity extends AppCompatActivity {

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

        Button buttonStart = (Button) findViewById(R.id.buttonStart);
        buttonStart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                try {
                    Intent intent = new Intent( packageContext: MainActivity.this, GameActivity.class);
                    startActivity(intent);finish();
                }catch (Exception e) {

                }
            }
        });

        Window w = getWindow();
        w.setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParams.FLAG_FULLSCREEN);
    }
}
```

# Переход из главного меню к Уровням

- Откройте GameActivity.java
- Запишем код, который дает перейти на другой уровней при нажатии кнопки с обработчиком ошибок.

```
Button buttonBack = (Button) findViewById(R.id.button_back);
        buttonBack.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                try {
                    Intent intent = new Intent(GameActivity.this,
MainActivity.class);
                    startActivity(intent);finish();
                }catch (Exception e) {


                }
            }
        });
```

# Системная кнопка Назад

- Откройте GameActivity.java
- Запишем код в самый конец перед закрывающейся фигурной скобкой }.

```
@Override
   public void onBackPressed() {
       try {
           Intent intent = new Intent(GameActivity.this,
MainActivity.class);
           startActivity(intent);finish();
       }catch (Exception e) {


       }
   }
```

# Подведем итоги

- Кнопка «Назад»
- Стиль для кнопки
- Верстка выбора уровня
- Переход от одного уровня к другому