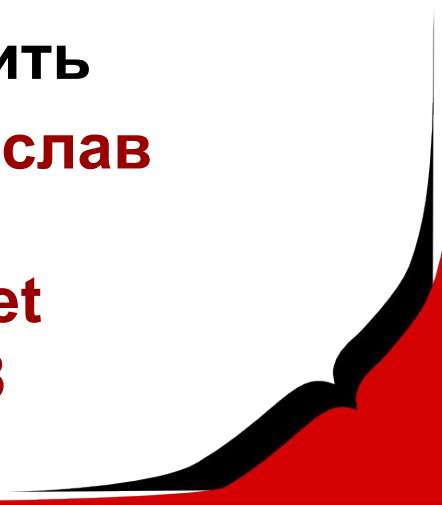

Проектування мобільних застосунків

Лекція №4. Розмітка сторінок

Заняття проводить
Лимаренко Вячеслав
Володимирович
slaw_lww@ukr.net
+38094-977-08-08



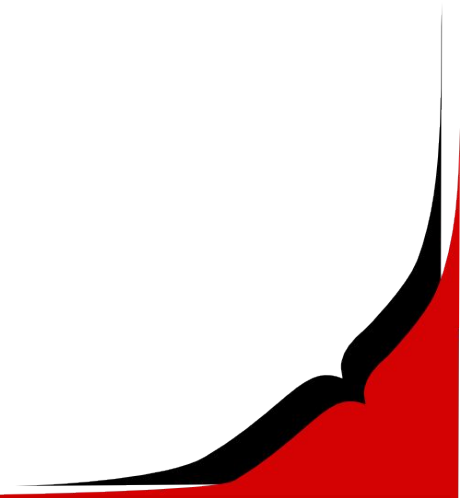
Екрани минулого

| Модель | Рік випуску | Роздільна здатність |
|----------------------------|-------------|-------------------------------|
| Nokia N95 | 2007 | 240×320, ~154 ppi |
| iPhone 1 | 2007 | 320×480, 163 ppi |
| HTC Athena | 2007 | 640×480, 160 ppi |
| iPhone 3s | 2009 | 480×320, 163 ppi |
| Sony Ericsson Satio (Idou) | 2009 | 360×640, ~210 ppi |
| Nokia N8 | 2010 | 360×640, ~210 ppi |
| iPhone 4 | 2010 | 960×640, retina 326 ppi |
| iPhone 5 | 2012 | 1,136×640, retina 326 ppi |
| iPhone 7 | 2016 | 1334×750, retina HD, 326 ppi |
| iPhone 7 plus | 2016 | 1920×1080, retina HD, 401 ppi |

Одиниці вимірювання

Еволюція

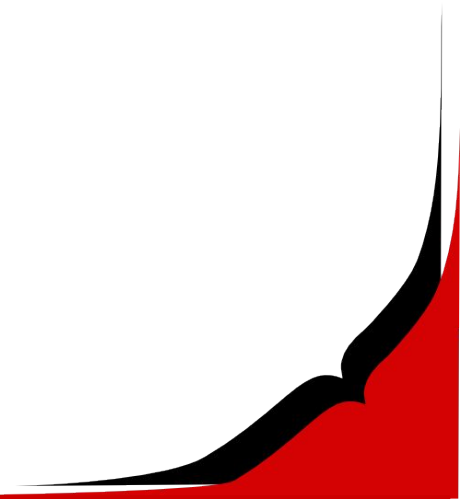
- Inches - дюйми
- Pixels – точки
- щільність пікселів
- PPI (англ. pixels per inch) - пікселі на дюйм
- розмір пікселя



Класифікація екранів

Орієнтація

- Книжкова (англ. Portrait)
- Альбомна (англ. Landscape)
- Квадратна (англ. Square)



Історичні шляхи вирішення

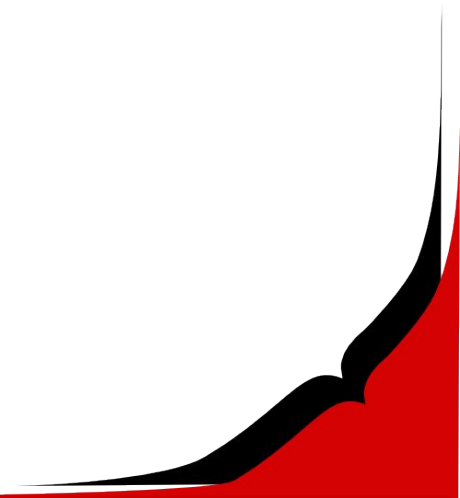
Статична розмітка

Плюси:

- мінімум часу розробника

Мінуси:

- елементи зазвичай малі
- наляпистий інтерфейс
- необхідно прокручувати екран – не все видно



Історичні шляхи вирішення

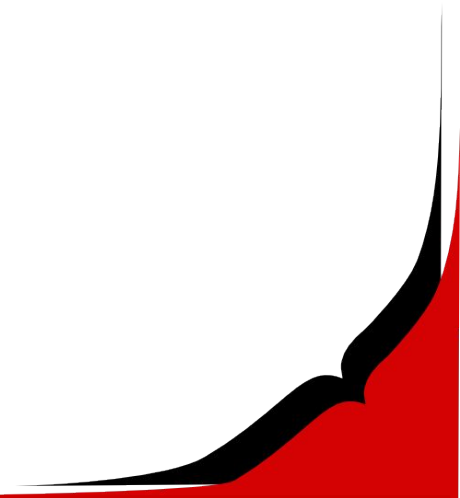
Динамічна розмітка

Плюси:

- все красиво виглядає

Мінуси:

- дуже велика складність реалізації
- дорого
- передбачити усе неможливо



Роздільні здатності

Еволюційні рішення

- для різних орієнтацій – різні розмітки
- для різних розмірів – різні розмітки

Задача

Ми знаємо, що є телефони з розширеннями:

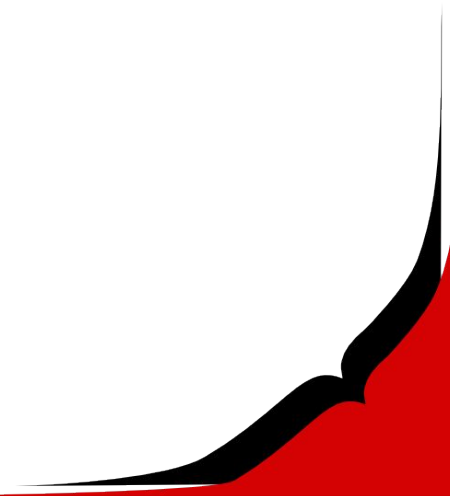
- 240×320 – можливі повороти
- 320×240 – неможливі повороти
- 240×240 – можливі повороти
- 160×240 – можливі повороти
- 480×640 – можливі повороти

Потрібно написати застосунок із 4-ма формами по 8 елементів управління на кожній

Роздільні здатності

Додамо «романтики»

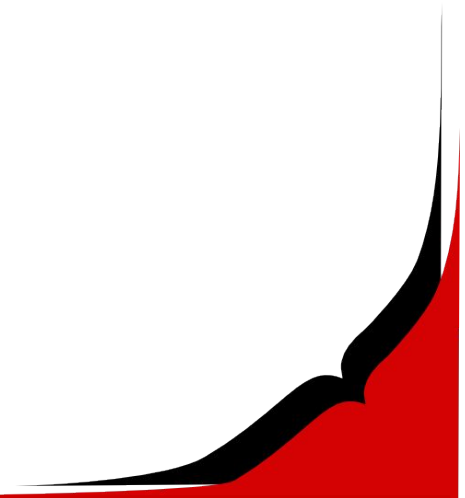
- щотижня по одному елементу на формі додають
- що два тижні по 2 елементи переміщують



Роздільні здатності

Висновки

- необхідно враховувати роздільності
- необхідно враховувати повороти
- необхідно враховувати людей з проблемами зору



Характеристики екрану

Параметри

- розмір екрану (англ. screen size) – фізичний розмір, діагональ в дюймах
- щільність екрану (англ. density) – кількість пікселів на одиницю розміру, DPI
DPI (англ. dots per inch)
- роздільна здатність (англ. resolution) – ширина та висота, пікселі
- незалежний від щільності піксель (англ. density-independent pixel, dp) – віртуальна одиниця для абстрагування від реальної щільності. Далі – **незалежний піксель**.
- пункт (англ. points, pt) – величина, що дорівнює 1/72 дюйма.
- масштабонезалежна точка (англ. scale-independent pixels, sp) – текст, цілі числа, співпадає з dp.

DPI vs PPI

- DPI – пристрої виводу (переважно принтери)
- PPI – екрани

Характеристики екрану

Типи за щільністю

$$px = dp * (dpi / 160)$$

Типи за щільністю

- низька - *ldpi* (low) ~120dpi
- середня - *mdpi* (medium) ~160dpi
- висока - *hdpi* (high) ~240dpi
- дуже висока - *xhdpi* (extra-high) ~320dpi
- дуже дуже висока - *xxhdpi* (extra-extra-high) ~480dpi
- дуже дуже дуже висока - *xxxhdpi* (extra-extra-extra-high) ~640dpi

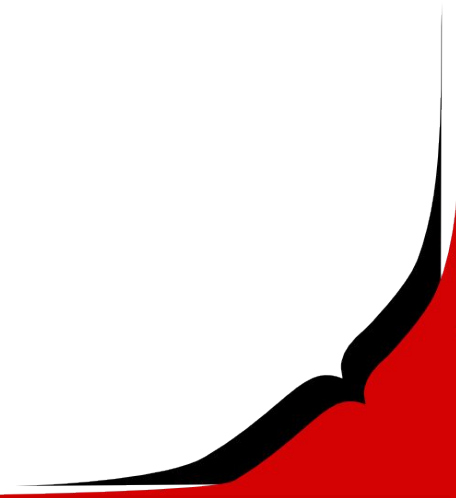
Запитання

Скільки фізичних пікселів припадає на один незалежний піксель для високої щільності?

Характеристики екрану

Задача

Дизайнер спроектував, наприклад у Photoshop, дизайн вашого застосунку. Вам необхідно реалізувати його, базуючись на вказаних параметрах. У параметрах вказано три види шрифтів на екрані 24pt, 48pt та 56pt.



Характеристики екрану

Рішення

1 inch = 72 pt.

1 px = 1 dp при 160 dpi

Визначимо розмір у дюймах на екрані:

$$S_{24} = 24\text{pt} * 1"/72\text{pt} = 1/3";$$

$$S_{48} = 48\text{pt} * 1"/72\text{pt} = 2/3";$$

$$S_{56} = 56\text{pt} * 1"/72\text{pt} = 7/9".$$

Визначимо розмір у незалежних пікселях та у звичайних пікселях, якщо у нас hdpi:

$$S'_{24} = 160\text{dp} * 1/3" \sim \mathbf{54\text{dp}} \Rightarrow 54\text{px} * 240/160 = \mathbf{81\text{px}}$$

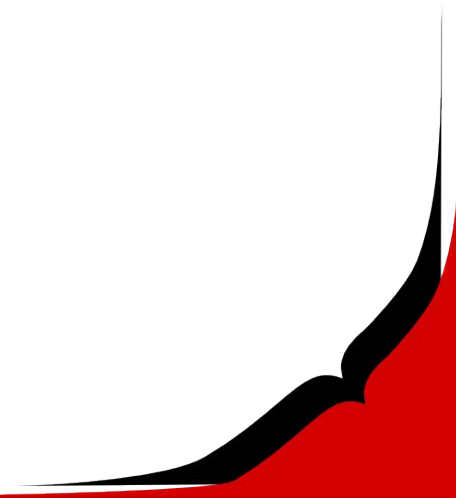
$$S'_{48} = 160\text{dp} * 2/3" \sim \mathbf{107\text{dp}} \Rightarrow 107\text{px} * 240/160 \sim \mathbf{160\text{px}}$$

$$S'_{56} = 160\text{dp} * 7/9" = \mathbf{124\text{dp}} \Rightarrow 124\text{px} * 240/160 \sim \mathbf{186\text{px}}$$

Калькулятор

Демонстрація калькулятора

- URL: <http://angrytools.com/android/pixelcalc/>



Приклад відображення елементів керування при різних розмірах екрану



Основні поняття XML (eXtensible Markup Language - «розширювана мова розмітки».)

Тег - значення

```
<item>value</>
<item>
  <item>value1</item>
  <item>
    <item>value1</item>
    <item>value2</item>
  </item>
  <item>value1</item>
</item>
```

Атрибут

```
<item name='my name'>value</item>
```

спрощено:

```
<item name='my name'/'>
```

Різниця між XML і HTML

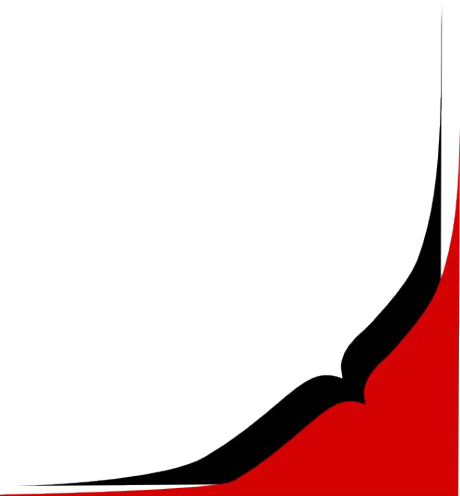
XML не є заміною HTML. Вони призначені для вирішення різних завдань: XML вирішує завдання зберігання і транспортування даних, фокусуючись на тому, що таке ці самі дані, HTML же вирішує завдання відображення даних, фокусуючись на тому, як ці дані виглядають. Таким чином, HTML піклується про відображення інформації, а XML про транспортування інформації. Теги XML не призначені. Ви повинні самі визначати потрібні теги.

ViewGroup

ViewGroup - розмітка, яка дозволяє розташувати один або кілька View.

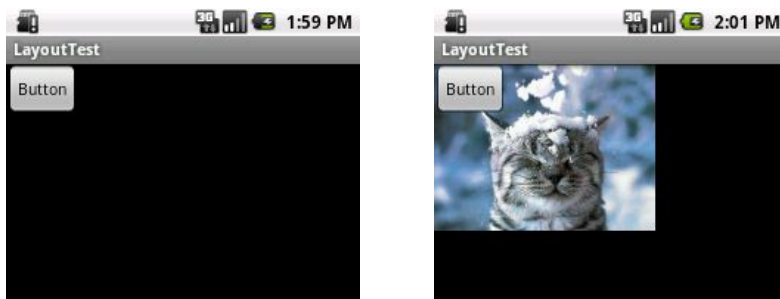
Стандартні типи ViewGroups:

- FrameLayout;
- LinearLayout;
- TableLayout;
- RelativeLayout;
- GridLayout (переважніше ніж TableLayout);
- SwipeRefreshLayout;
- ConstraintLayout;
- CoordinatorLayout.

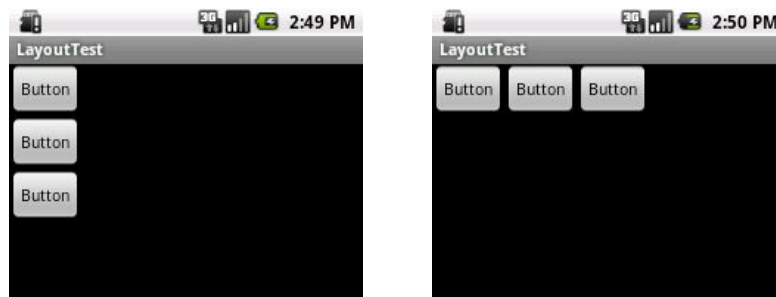


Приклади Layout (розмітки)

FrameLayout - розмітка для відображення одного елемента. Дочірні View або ViewGroup в FrameLayout вирівнюються по верхньому лівому кутку. Розмітка може містити кілька елементів, але тоді вони будуть перекривати один одного



LinearLayout - розмітка для відображення одного або декількох елементів в одну лінію, горизонтально або вертикально. Для вибору орієнтації використовується атрибут `android:orientation` з двома можливими значеннями «horizontal» і «vertical»

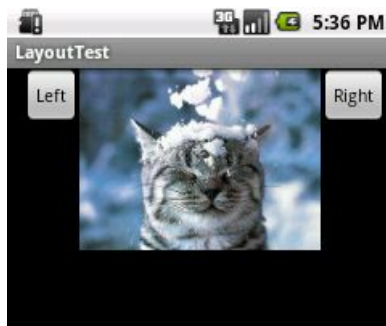


Приклади Layout (розмітки)

TableLayout - розмітка для розташування елементів у вигляді таблиці. Ряди задаються в xml за допомогою тега TableRow, а осередки в кожному ряду створюються автоматично для кожного елемента. Кількість колонок в таблиці дорівнюватиме максимальній кількості елементів в рядах. Ширина колонки визначається по самому широкому елементу в ній.

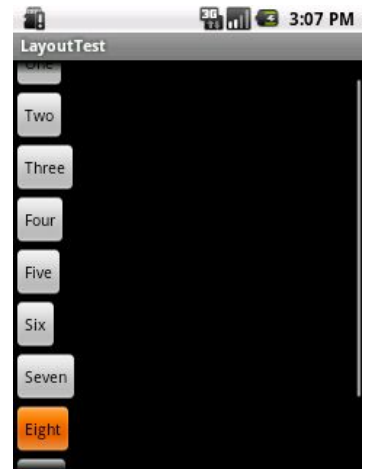


RelativeLayout - розмітка для розташування елементів щодо одного з батьків або один одного. Елементи починають розташовуватися в зазначеному порядку, тому необхідно щоб елемент був описаний до того, як інший елемент буде на нього посилатися.



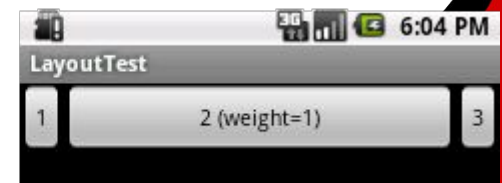
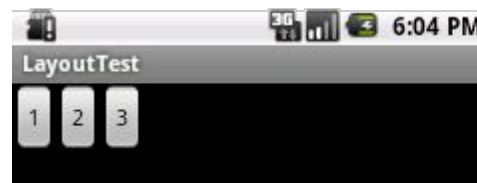
Приклади Layout (розмітки)

ScrollView - є спадкоємцем класу `FrameLayout`. Відмінністю є те, що він дозволяє прокручувати елементи, якщо вони займають більше місця ніж фізичний розмір екрану. У `ScrollView` також може міститися лише один елемент (`View` або `ViewGroup`), найчастіше використовується `LinearLayout`, в який вкладено декілька елементів. `ScrollView` підтримує тільки вертикальну прокрутку.



Якщо атрибут `layout_weight` поставити тільки у одного елемента, то він займе максимум вільного простору.

Так на першому скріншоті у трьох кнопок `android:layout_width="wrap_content"` і не заданий `weight`. На другому скріншоті у другій кнопки заданий `android:layout_weight="1"`



Linear Layout

Призначення

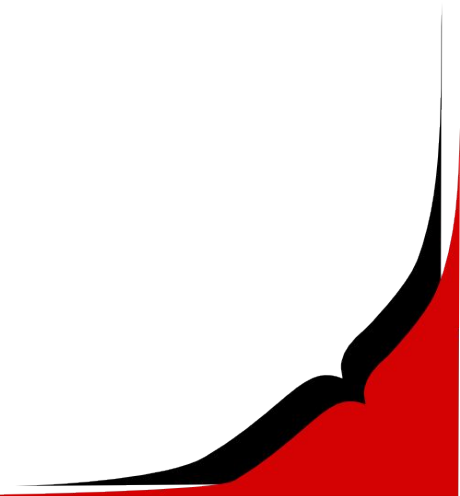
- однонапрямне розміщення елементів
- напрямок – горизонтально *android:orientation="horizontal"*
- напрямок – вертикально *android:orientation="vertical"*

Основні атрибути

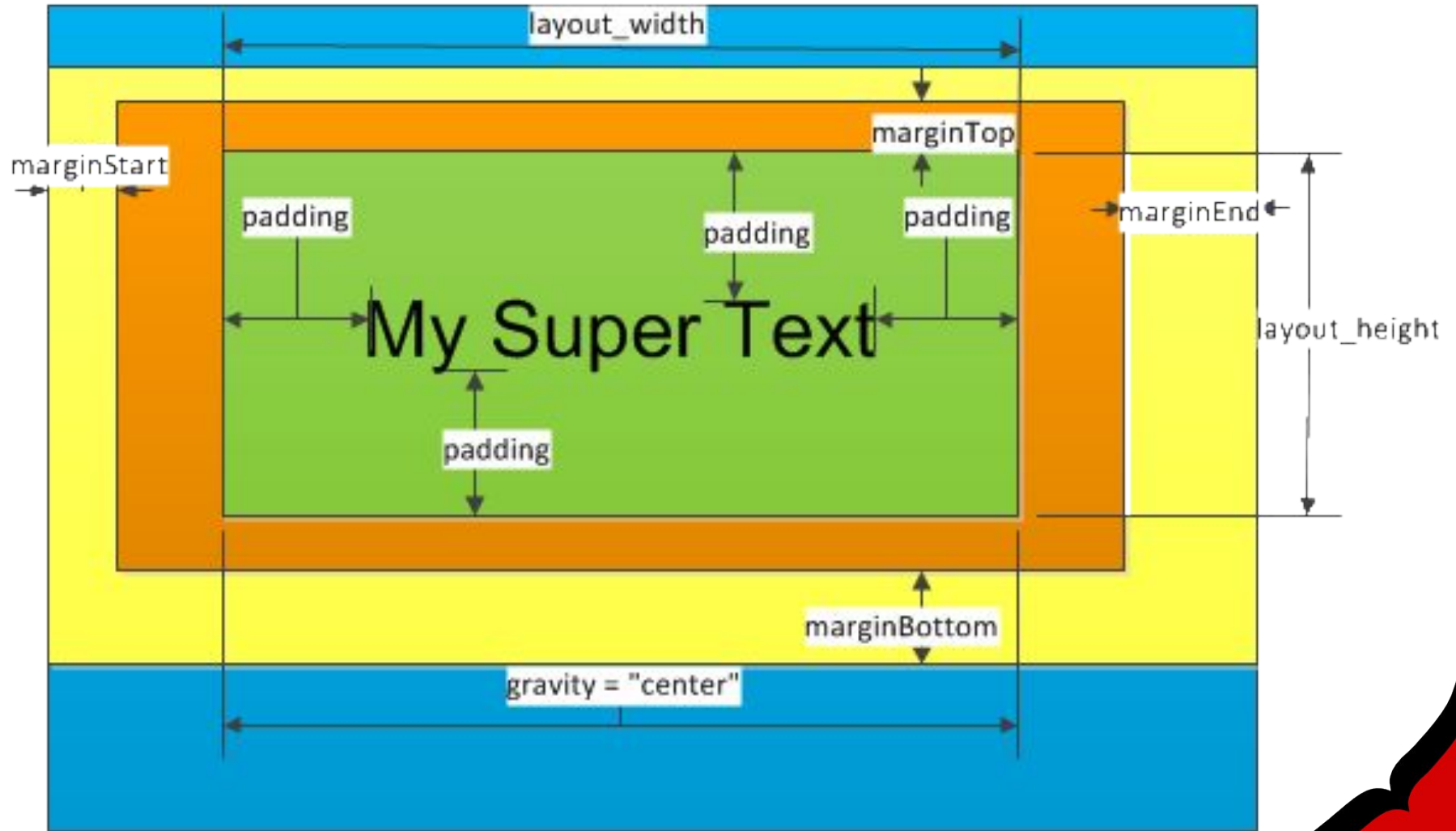
- `layout_width`, `layout_height`
- `gravity` – розміщення вмісту всередині елемента
- `layout_gravity` – розміщення всередині контейнера

Спеціальний розмір

- `match_parent` – по розміру контейнера
- `wrap_content` – по розміру вмісту



Структура элемента



Relative Layout

Призначення

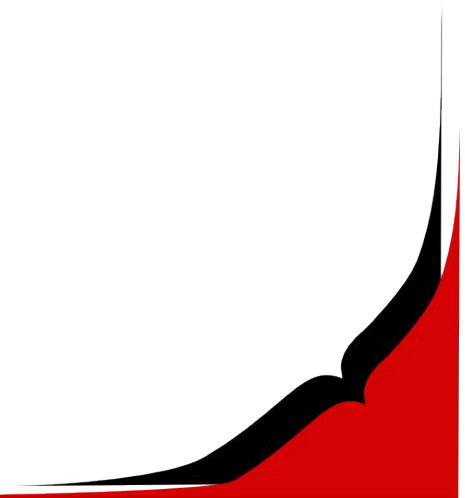
- відносне розміщення
- складні розміщення

Проблеми

- елемент може бути розміщений відносно вже оголошеного

Основні атрибути

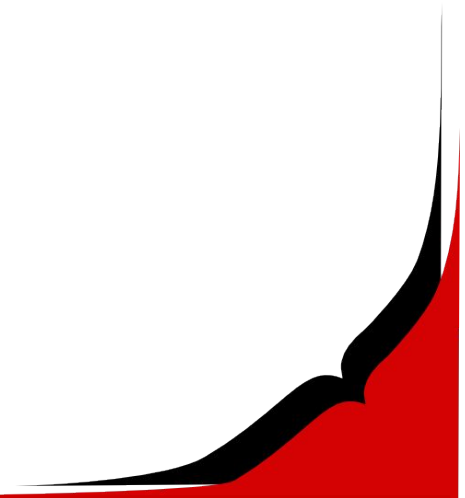
- top, bottom, left, right, center (boolean)
- center_vertical, center_horizontal (boolean)
- start, end (boolean)



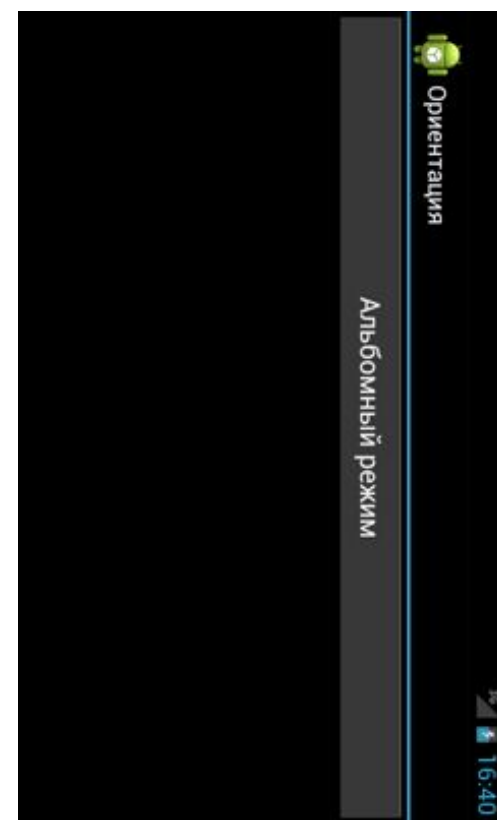
Relative Layout

Відносно сусідів

- `layout_toEndOf`, `layout_toLeftOf`, `layout_toRightOf`, `layout_toStartOf`
- `layout_alignEnd` , `layout_alignLeft`, `layout_alignRight`, `layout_alignStart`,
`layout_alignTop`
- `layout_alignParentBottom`, `layout_alignParentEnd`, `layout_alignParentLeft`,
`layout_alignParentRight`, `layout_alignParentStart`, `layout_alignParentTop`



Типові помилки



Дякую за увагу

