ЭЛЕМЕНТ GRID

`Grid` с английского переводится как сетка. `Grid` является базовым и универсальным элементом компоновки, однако для его точной настройки требуется учитывать немало факторов.

Контейнер `Grid` объявляется следующим образом.

```
<Grid>
    <!--Содержимое-->
</Grid>
```

Рисунок 1

`Grid` имеет табличную структуру. При начальном объявлении автоматически инициируется ***одна строка*** и ***один столбец***, что в конечном счёте образует ***одну ячейку.***

Ячейка описывается определением ***строки*** и определением **столбца**, т.е. у каждой ячейки есть свойство строки и свойство столбца. Каждое из этих свойств описывается отдельно.

Как раз для определений строк и столбцов у `Grid` есть специальные ***коллекции***.

Определение строк

Для описания списка определений строк используют вложенное свойство `Grid.RowDefinitions`.

**Definitions – определения*

```
<Grid>
    <Grid.RowDefinitions>
        <!--Список определений строк-->
    </Grid.RowDefinitions>

</Grid>
```

Рисунок 2

Список `Grid.RowDefinitions` ожидает ***список определений*** по каждой ***строке***. Одна строка – одно определение. Определение строки описывается с помощью объекта `RowDefinition`.

**Definition - определение*

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>

</Grid>
```

Рисунок 3

Для тега `RowDefinition` удобно использовать сокращённую запись, так как этот объект представляет дополнительную информацию с помощью атрибутов.

Для определения `RowDefinition` можно указывать атрибут размеров ***высоты***, в том числе максимальной `MaxHeight`, и минимальной `MinHeight`. Для строки бессмысленно указывать ширину, так как за это свойство отвечает определение столбца.

Определение столбцов

Для описания списка определений столбцов используют вложенное свойство `Grid.ColumnDefinitions`.

```
<Grid>
    <Grid.ColumnDefinitions>
        <!--Определение столбцов-->
    </Grid.ColumnDefinitions>

</Grid>
```

Рисунок 4

Для определения каждого столбца указывают тег `ColumnDefinition`.

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>

</Grid>
```

Рисунок 5

Определение ***столбца*** может иметь размеры ***ширины***, в том числе минимальные и максимальные. Указание высоты столбца отсутствует, т.к. за высоту отвечает определение строки.

Позиционирование элемента в сетке

Каждый вложенный элемент имеет так называемые ***«координаты»***: ***номер строки*** и ***номер столбца*** в сетке.

Каждому вложенному в `Grid` элементу ***«прицепляется»*** свойство `Grid`, которое имеет вложенные свойства `Row` и `Column` для указания позиции.

Номер позиции отсчитывается с `0`, как в массивах.

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>

    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>

    <Button Grid.Row="1"
            Grid.Column="1"></Button>

</Grid>
```

номер соответствует определению

элемент берет данные из определения для установки размера

Рисунок 6

Разметка из этого примера будет иметь следующее графическое представление в построителе.

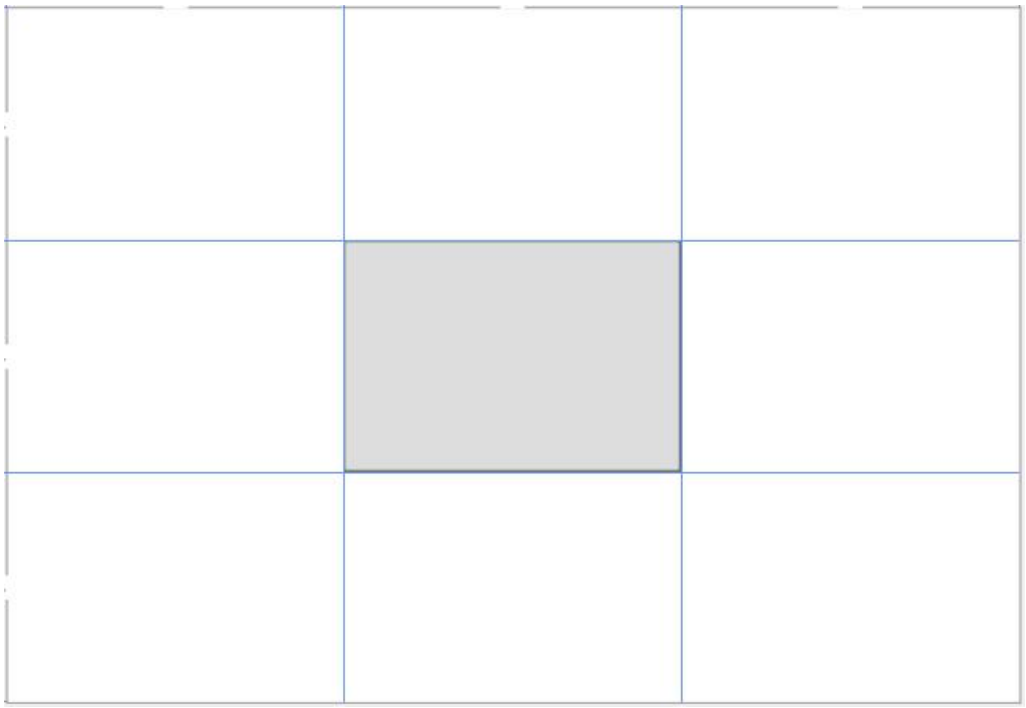


Рисунок 7

Размеры

Изначально размеры определений устанавливаются ***автоматически***, деля на ***равные части*** данную ему область *(ширину и высоту)*.

Для того, чтобы указать конкретный размер используют ***базовые свойства*** ширины для `ColumnDefinition` и базовые свойства высоты для `RowDefinition`.

Размеры относительно содержимого

Если необходимо указать то, что строка или столбец должны подстраивать свои размеры относительно содержимого, необходимо в значение размера указать `Auto`.

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition/>
    </Grid.RowDefinitions>

    <Button Height="50"
            Grid.Row="0"/>
</Grid>
```

Рисунок 8

В отображении появится следующая разметка.

Рисунок 9

Относительные размеры

Если после указания конкретных размеров остаётся неразмеченная область, она распределяется поровну на оставшиеся ***строки/столбцы*** в определении. Если необходимо указать на то, что ***строка/столбец*** должна ***заполнять*** оставшееся ей место, в значении размеров записывают символ *****.

```
<Grid>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="20"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="20"/>
        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>

        <Button Grid.Row="1"
                Grid.Column="1"></Button>

    </Grid>
```

Рисунок 10

Пример будет отображаться следующим образом.

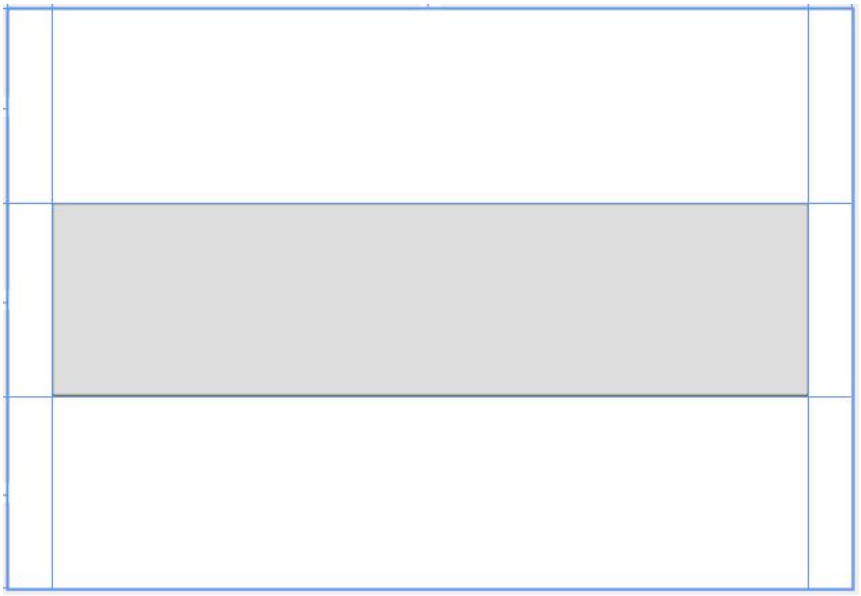


Рисунок 11

Символ ***** указывает на заполнение оставшейся части контейнера.

В случаях, когда ***строк/столбцов*** с автоматическим размером больше **1**, эти ***строки/столбцы*** делят оставшуюся область поровну.

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="20"/>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>

    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>

    <Button Grid.Row="1"
            Grid.Column="1"></Button>

</Grid>
```

Рисунок 12

Результат примера.

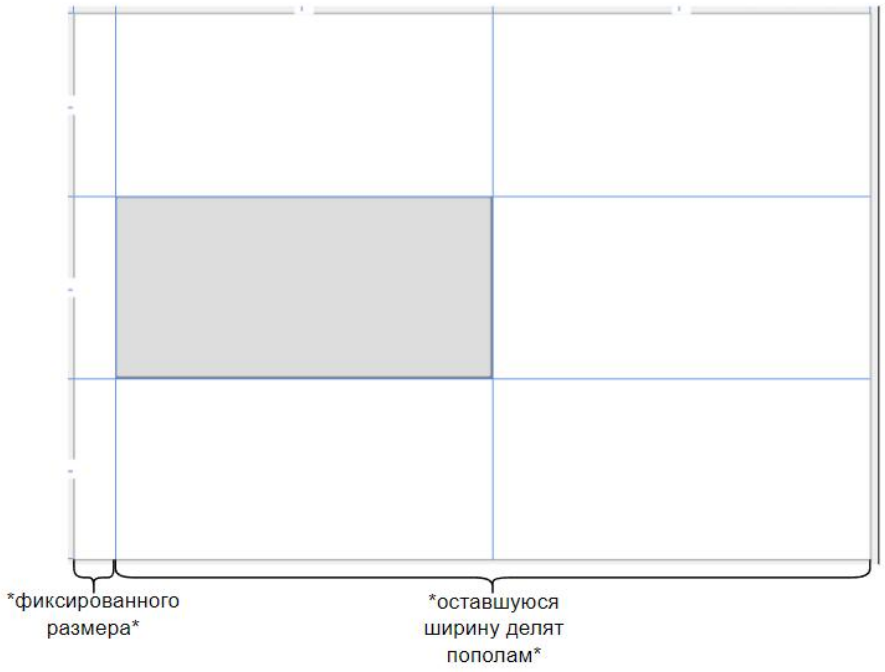


Рисунок 13

Указание пропорций

Если необходимо указать отношение размеров ***пропорционально***, указывают ***число коэффициента*** перед символом **`*`**.

Для определения пропорций сначала коэффициенты ***складываются***. Полученное число обозначает то, на сколько логических частей поделилась данная контейнеру область. После этого, каждая из отдельных ***строк/столбцов*** берет то количество частей, которое указано в ***коэффициенте***. Значение коэффициента можно указывать как целым, так и числом с дробной частью *(вещественным числом)*.

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="20"/>
        <ColumnDefinition Width="0.5*"/>
        <ColumnDefinition Width="2*"/>
    </Grid.ColumnDefinitions>

    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>

    <Button Grid.Row="1"
            Grid.Column="1"></Button>

</Grid>
```

Рисунок 14

Из примера, первая колонка будет фиксированной ширины равной **`20px`**. Оставшаяся часть сначала логически разделилась на **`2+0.5=2.5`** частей, из которых **`0.5`** частей отдаётся второму столбцу, а **`2`** третьему.

Рисунок 15

Объединение ячеек

Бывают случаи, когда необходимо указать на то, что элемент должен занимать несколько ячеек и столбцов. Объединение происходит, начиная **с текущей позиции объекта** *(координаты Grid.Row и Grid.Column)***.** Для ***столбцов*** отсчёт идёт в правую сторону, а для ***строк*** – вниз.

Для указания количества столбцов, которые нужно объединить, начиная с текущей позиции и вправо, используют свойство `Grid.ColumnSpan`.

Для указания количества строк, которые нужно объединить, начиная с текущей позиции и вниз, используют свойство `Grid.RowSpan`.

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="20"/>
        <ColumnDefinition Width="0.5*"/>
        <ColumnDefinition Width="2*"/>
    </Grid.ColumnDefinitions>

    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>

    <Button Grid.Row="1"
            Grid.Column="1"
            Grid.RowSpan="2"
            Grid.ColumnSpan="2"></Button>

</Grid>
```

Рисунок 16

Пример будет выглядеть так.

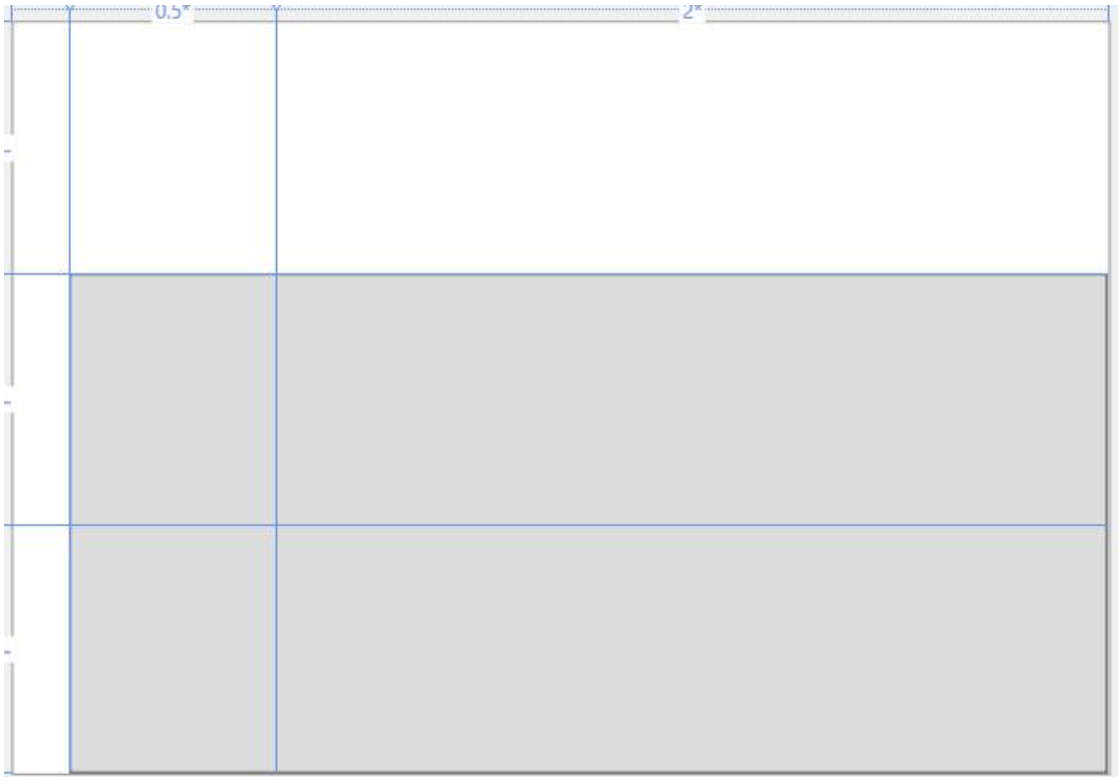


Рисунок 17

Размерная группа

Иногда необходимо сделать так, чтобы строки или столбцы имели ***одинаковые размеры***. В таких случаях объявляют имя размерной группы. Причём указать одинаковый размер можно не только для одного контейнера.

Для указания размерной группы объявляют свойство `SharedSizeGroup`, значением которого может быть любое имя.

Дальше, такое же имя можно указать любому другому столбцу или строке (смотря для чего применяется), который должен принимать общий размер. Получается, что ячейки/столбцы с одинаковым именем `SharedSizeGroup` ***синхронизируются***.

Для того чтобы использовать синхронизацию, в родительском контейнере необходимо указать свойство `Grid.IsSharedSizeScope` равным `true`. Если попытаться перевести на русский название атрибута, получится фраза *«Имеет ли контейнер контейнеры Grid с размерной группой?»*.

Размер группы определяется относительно большего значения.

```
<StackPanel Grid.IsSharedSizeScope="True">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"
                              SharedSizeGroup="Btn"/>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>

        <Button Height="50"
                Grid.Row="0"/>
        <Border Background="LightBlue"
                Grid.Column="1"/>
        <Border Background="LightGray"
                Grid.Column="2"/>
    </Grid>
    <TextBlock Text="Разделительный текст"/>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"
                              SharedSizeGroup="Btn"/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>

        <Button Height="50"
                Width="50"
                Grid.Row="0"/>
        <Border Background="Gray"
                Grid.Column="1"/>
    </Grid>
</StackPanel>
```

Рисунок 18

Результат кода из примера.

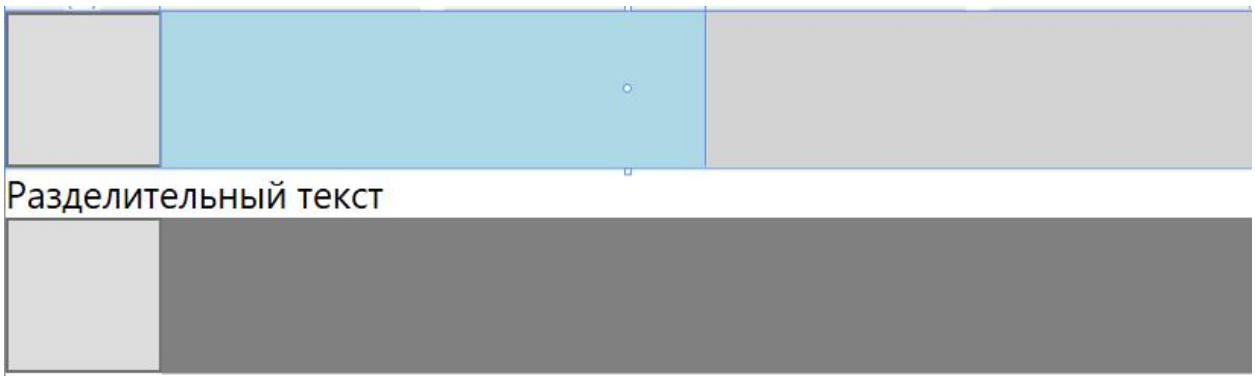


Рисунок 19

Элемент GridSplitter

Ещё одно удобное свойство `Grid` – поддержка ***разделителей***. С их помощью можно изменять ширину или высоту ячейки с помощью ***перетаскивания мыши***. Такой элемент удобно использовать для панелей инструментов, которые можно скрывать, перетаскивая разделитель.

Как следует из названия, работает только с `Grid`.

Стоит заметить, что `GridSplitter` – это ***отдельный элемент***, который может иметь размеры, цвет и позицию и т.п.. Для ячейки он как объект-прилипала:

- используется только в той ячейке, в которой находится
- двигает только ту сторону, по которой выравнивается.

Чтобы присвоить разделитель с наложением, необходимо добавить его ***в ячейку***.

У `GridSplitter` нет автоматически заданного размера, поэтому необходимо указывать их самостоятельно.

Если необходимо сделать вертикальную линию, можно указать только ширину `Width`, а для автоматического определения высоты используют выравнивание по высоте `VerticalAlignment`.

Так же нужно учитывать распределение по слоям. Чтобы его было видно, необходимо расположить его поверх всех остальных слоёв.

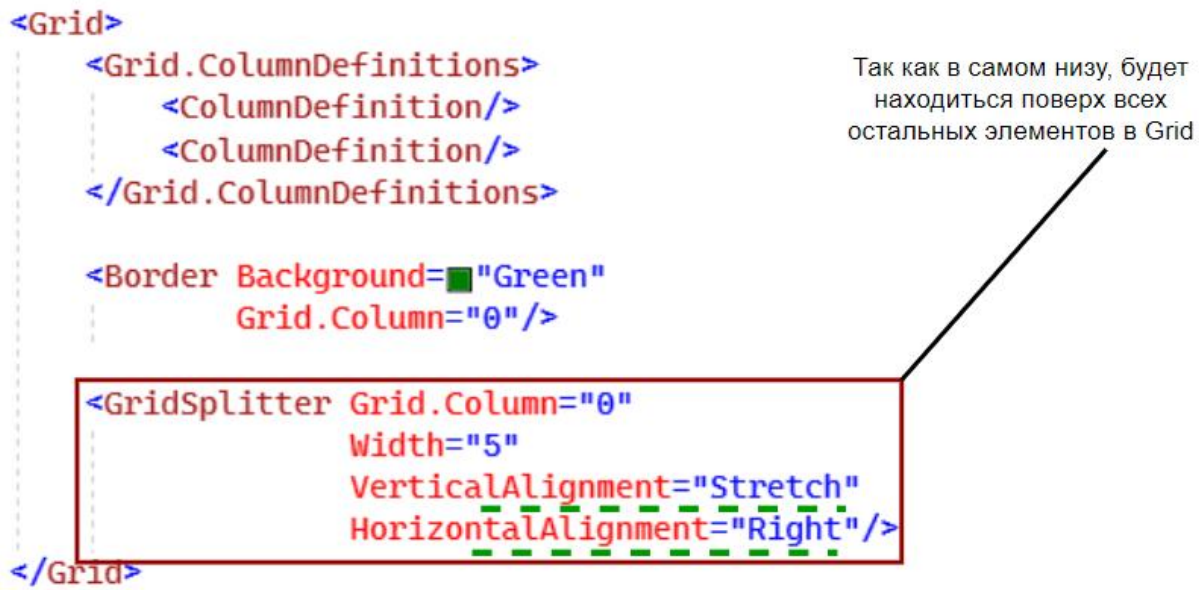


Рисунок 20

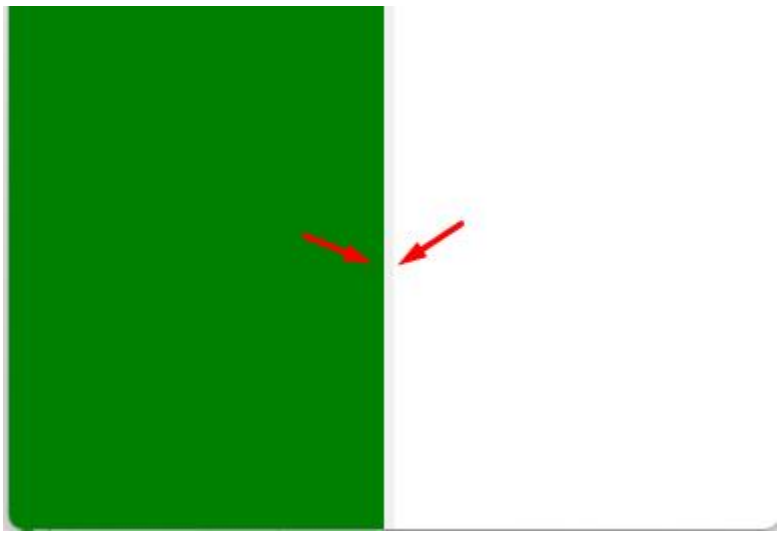


Рисунок 21

Такая запись не всегда удобна: линия может загораживать важную информацию в ячейке, в которой он находится.

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>

    <Border Background="Green"
            Grid.Column="0"/>

    <TextBlock Text="Мой важный текст"
               Foreground="White"
               HorizontalAlignment="Right"/>

    <GridSplitter Grid.Column="0"
                  Width="5"
                  VerticalAlignment="Stretch"
                  HorizontalAlignment="Right"/>
</Grid>
```

Рисунок 22

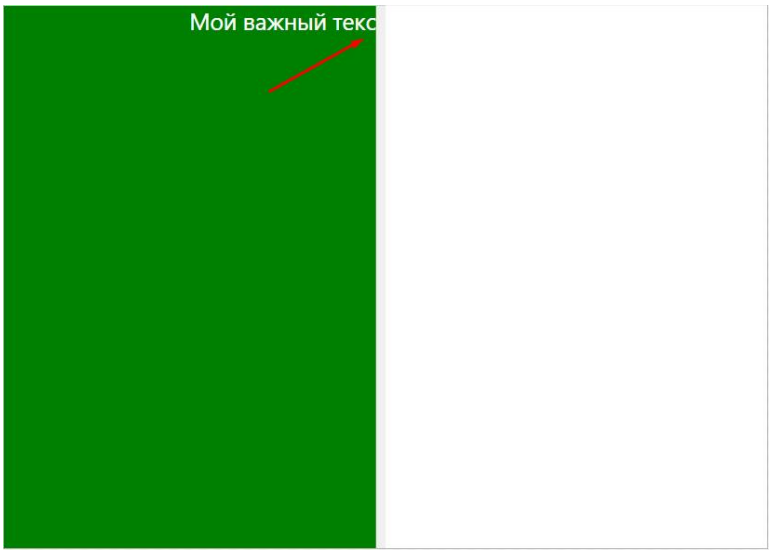


Рисунок 23

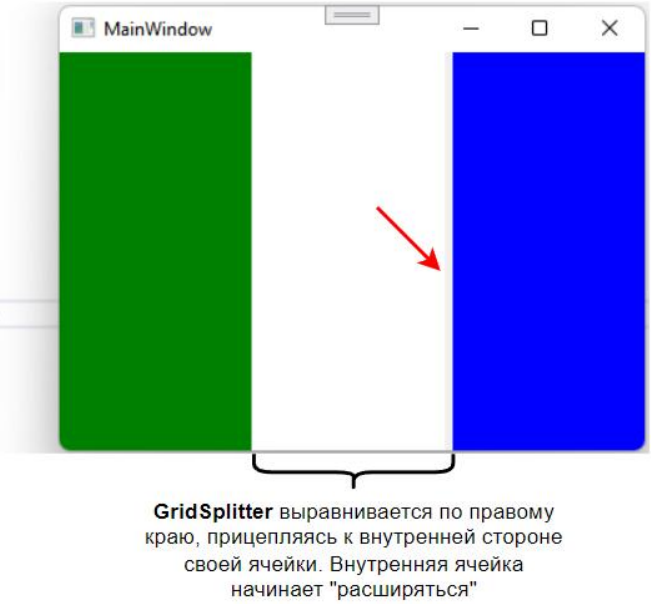
Чтобы такого не происходило, под **GridSplitter** выделяют отдельную ячейку. Чтобы **GridSplitter** мог изменять размеры обеих ячеек, между которых он находится, необходимо выровнять его по центру.

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>

    <Border Background="Green"/>

    <GridSplitter Grid.Column="1"
                  Width="5"
                  Grid.ColumnSpan="1"
                  VerticalAlignment="Stretch"
                  HorizontalAlignment="Center"/>
</Grid>
```

Рисунок 24

Если в этом примере установить выравнивание по левому или правому краю, то **GridSplitter** будет изменять размер ячейки, в которой находится. Учитывая, что под него выделялась отдельная ячейка, получим странный эффект.

```
<Border Background="Green"/>

<GridSplitter Grid.Column="1"
              Width="5"
              Grid.ColumnSpan="1"
              VerticalAlignment="Stretch"
              HorizontalAlignment="Right"/>
<Border Background="Blue"
        Grid.Column="2"/>
```



GridSplitter выравнивается по правому краю, прицепляясь к внутренней стороне своей ячейки. Внутренняя ячейка начинает "расширяться"

Рисунок 25

Если на фон Window поставить картинку, пример станет более наглядным :-)

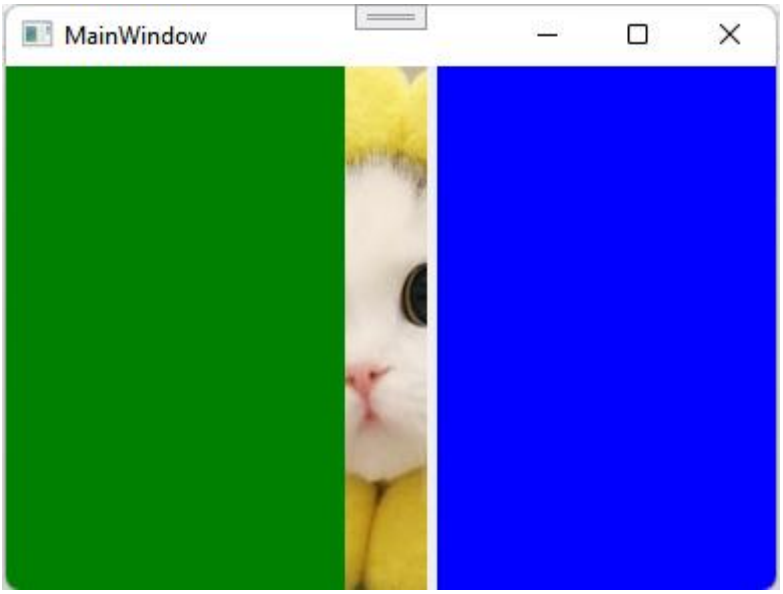


Рисунок 26

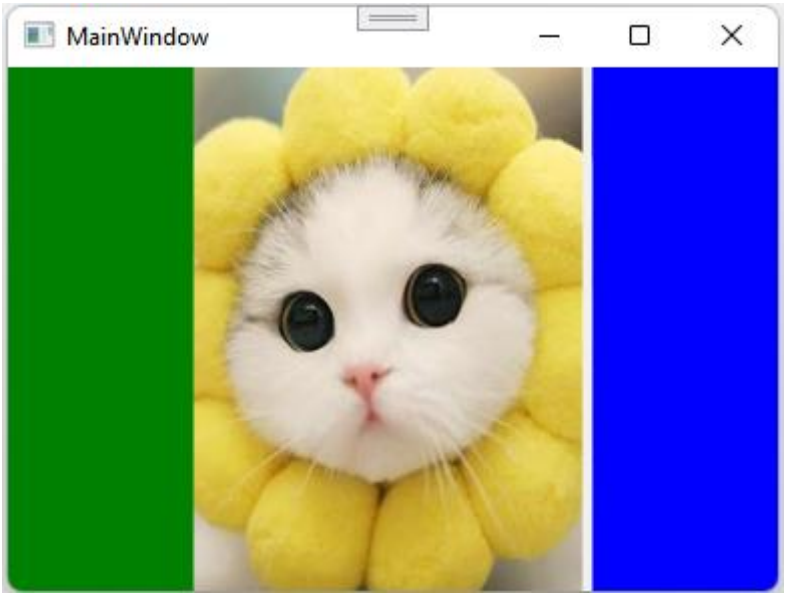


Рисунок 27

UniformGrid

Контейнер **`UniformGrid`** призван создавать ***автоматическую сетку*** для находящихся внутри элементов. У элементов будут ***отсутствовать*** прикрепляемые свойства указания номера строки и колонки.

Количество столбцов указывают в свойстве **`Columns`**. Количество строк указывают в свойстве **`Rows`**.

```
<UniformGrid Columns="3"
             Rows="2">
    <Button/>
    <Button/>
    <Button/>
    <Button/>
    <Button/>
    <Button/>
</UniformGrid>
```

Рисунок 28

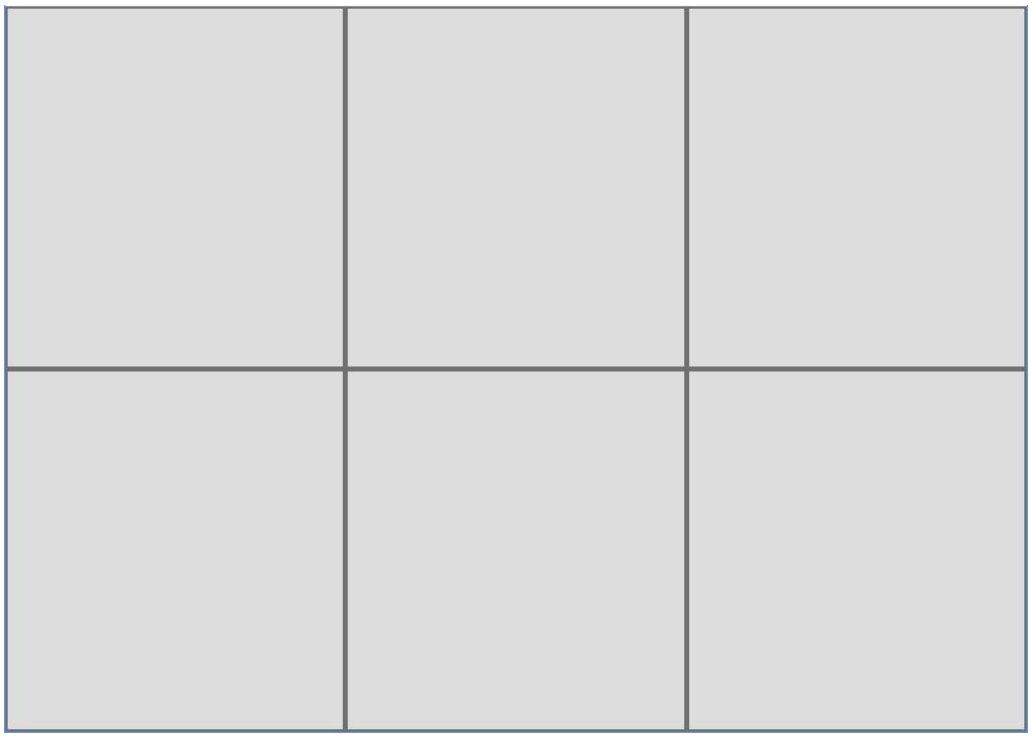


Рисунок 29

Вложенные элементы заполняются в порядке чтения: ***слева направо*** и ***сверху вниз***.

Если их не указывать, `UniformGrid` ***автоматически*** подберёт ***одинаковое*** число для строк и столбцов (квадратная матрица - x^2).

```
<UniformGrid>
    <Button/>
    <Button/>
    <Button/>
    <Button/>
    <Button/>
    <Button/>
</UniformGrid>
```

Рисунок 30

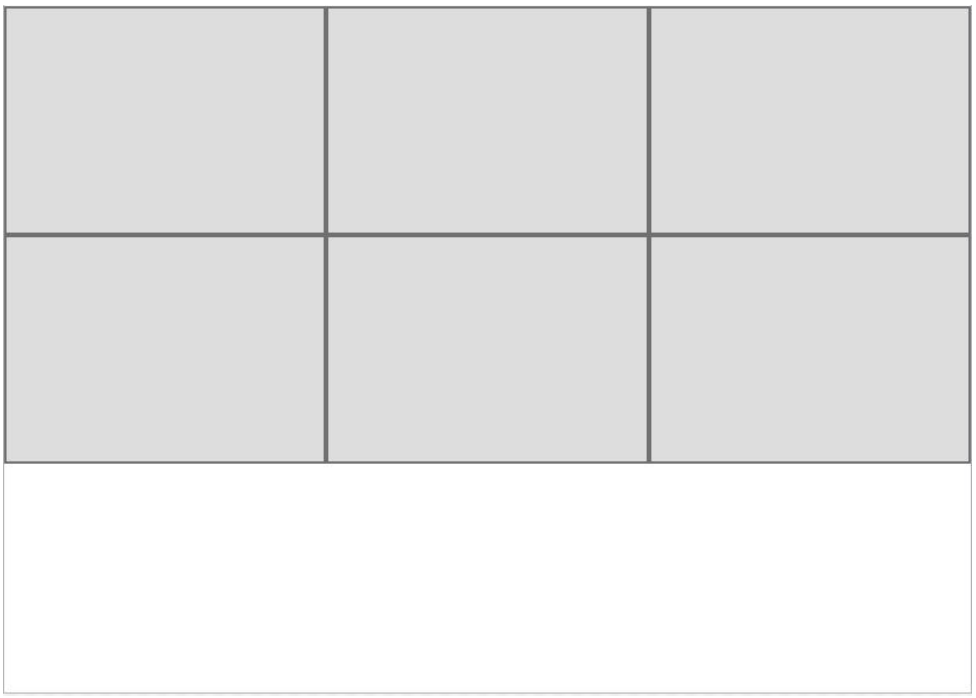


Рисунок 31

Его применение удобно только в тех случаях, когда имеется небольшое количество элементов. Этот контейнер так же удобно использовать для упорядочивания элементов в одной строке или оном столбце.

Посмотрите разницу.

```
<UniformGrid Columns="3">
    <Button/>
    <Button/>
    <Button/>
</UniformGrid>
```

Рисунок 32

И

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>

    <Button Grid.Column="0"/>
    <Button Grid.Column="1"/>
    <Button Grid.Column="2"/>
</Grid>
```

Рисунок 33

Дают одинаковый результат.

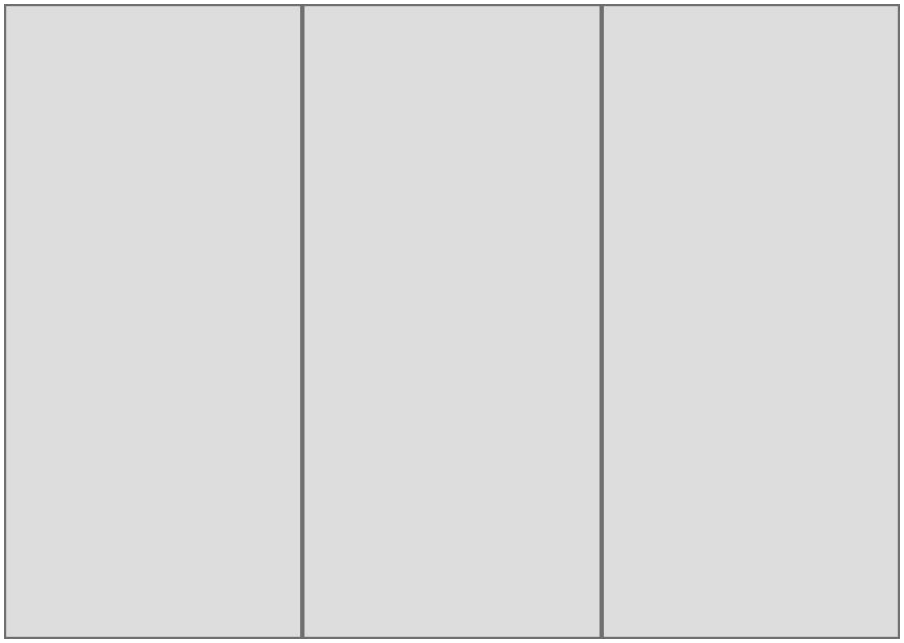


Рисунок 34