Контейнеры компоновки

Box

Компонент Box является наиболее простым контейнером, позволяя позиционировать вложенное содержимое. Он представляет некоторую область экрана. Функция данного компонента принимает четыре параметра:

```
@Composable
inline fun Box(
    modifier: Modifier = Modifier,
    contentAlignment: Alignment = Alignment.TopStart,
    propagateMinConstraints: Boolean = false,
    content: BoxScope.() -> Unit
): @Composable Unit
```

- modifier: объект Modifier, который позволяет настроить внешний вид и поведение компонента с помощью модификаторов
- contentAlignment: объект Alignment, который устанавливает расположение компонента. По умолчанию имеет значение Alignment.TopStart (расположение вначале контейнера в верхнем углу)
- propagateMinConstraints: значение типа Boolean, который указывает, надо ли применять к содержимому ограничения по минимальным размерам. По умолчанию равно false (ограничения не применяются)
- content: объект интерфейса BoxScope, который представляет вложенное содержимое

Modifier

Контейнер Box может использоваться как самодостаточный компонент без какого-либо вложенного содержимого:

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.size
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Box(
                modifier = Modifier.size(300.dp, 250.dp).background(Color.Blue)
            )
        }
    }
}
```

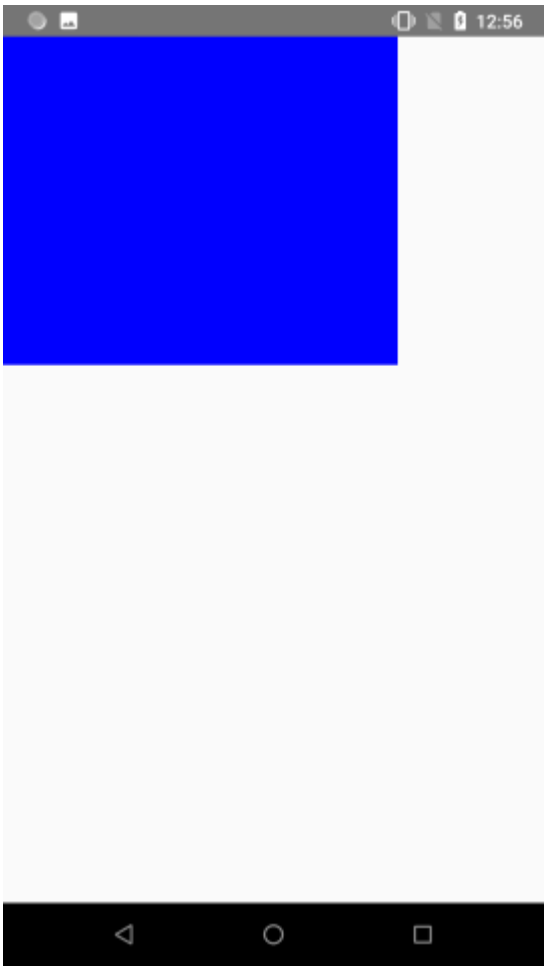
Для установки визуальных свойств объекта Box применяется свойство modifier, которое представляет объект Modifier, рассмотренный в прошлой главе. Как и для других компонентов, для компонента Box мы можем вызвать функцию size(), которая принимает ширину и длину контейнера - в данном случае ширина 300 единиц и высота 250 единиц.

```
Modifier.size(300.dp, 250.dp)
```

Эта функция опять же возвращает объект Modifier, у которого далее вызывается другая функция - background(), которая устанавливает фоновый цвет контейнера - в данном случае синий цвет или значение Color.Blue

```
Modifier.size(300.dp, 250.dp).background(Color.Blue)
```

В итоге мы увидим на экране синюю прямоугольную область размером 300 x 250.



Установка размеров Box

По умолчанию Box занимает те размеры на экране, которые необходимы, чтобы вместить содержимое. Но выше в Box никакого вложенного содержимого не было, поэтому чтобы установить размеры, применялся объект Modifier и его функция size()

```
Modifier.size(300.dp, 250.dp)
```

Если необходимо растянуть элемент по всей ширине и длине экрана, то применяется другая функция - fillMaxSize():

```
Modifier.fillMaxSize().background(Color.Blue)
```

Позиционирование внутри Box

Например, определим простейший элемент Box с вложенным элементом Text:

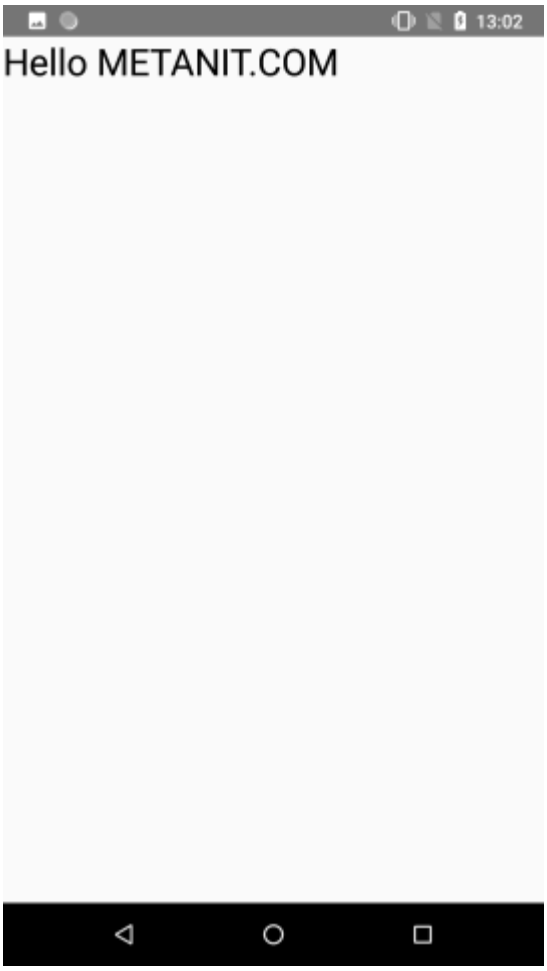
```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
```

```
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.Box
import androidx.compose.material.Text
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Box {
                Text("Hello METANIT.COM!", style = TextStyle(fontSize = 22.sp))
            }
        }
    }
}
```

Здесь мы видим, что вложенный компонент Text позиционируется в левом верхнем углу контейнера Box.



Есть три причины подобного позиционирования

- Элемент Box сам по умолчанию располагается в верхнем левом углу устройства
- Для Box не задано размеров, он будет стремиться занять пространство, необходимое для вложенного содержимого.

- Если вложенное содержимое меньше размеров контейнера Box, то для его позиционирования внутри Box будет использоваться настройка contentAlignment. А по умолчанию применяет значение Alignment.TopStart (расположение вначале контейнера в верхнем углу)

Теперь применим параметр contentAlignment для позиционирования содержимого:

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material.Text
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Box(
                contentAlignment = Alignment.Center,
                modifier = Modifier.fillMaxSize()
            )
            {
                Text("Hello METANIT.COM!", style = TextStyle(fontSize = 22.sp))
            }
        }
    }
}
```

Значение Alignment.Center указывает, что содержимое будет позиционироваться по центру как по горизонтали, так и по вертикали.

Если Box содержит несколько вложенных компонентов, то они будут накладываться друг на друга в порядке следования (последний компонент располагается поверх предыдущих)

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material.Text
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Box( modifier = Modifier.fillMaxSize(), contentAlignment =
Alignment.Center )
            {
                Box(modifier = Modifier.background(Color.Blue).size(300.dp))
```

```
                Box(modifier = Modifier.background(Color.LightGray).size(200.dp))
                Text("Hello METANIT.COM!", style = TextStyle(fontSize = 22.sp))
            }
        }
    }
}
```

Column

Контейнер Column позволяет выстроить вложенные компоненты в столбик. Функция Column принимает четыре параметра:

```
@Composable
inline fun Column(
    modifier: Modifier = Modifier,
    verticalArrangement: Arrangement.Vertical = Arrangement.Top,
    horizontalAlignment: Alignment.Horizontal = Alignment.Start,
    content: ColumnScope.() -> Unit
): @Composable Unit
```

- modifier: объект Modifier, который позволяет настроить внешний вид и поведение компонента

- verticalArrangement: объект Arrangement.Vertical, который устанавливает выравнивание компонента по вертикали. По умолчанию имеет значение Arrangement.Top (расположение в верху)
- horizontalAlignment: объект Alignment.Horizontal, который устанавливает выравнивание компонента по горизонтали. По умолчанию имеет значение Alignment.Start (расположение в начале - слева для языков с левосторонним письмом и справа для языков с правосторонним письмом)
- content: объект интерфейса BoxScope, который представляет вложенное содержимое

Так, изменим код MainActivity.kt следующим образом:

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.Column
import androidx.compose.material.Text
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Column {
                Text("Hello", style = TextStyle(fontSize = 22.sp))
                Text("World", style = TextStyle(fontSize = 22.sp))
                Text("from", style = TextStyle(fontSize = 22.sp))
                Text("Jetpack Compose", style = TextStyle(fontSize = 22.sp))
            }
        }
    }
}
```

Установка веса компонентов

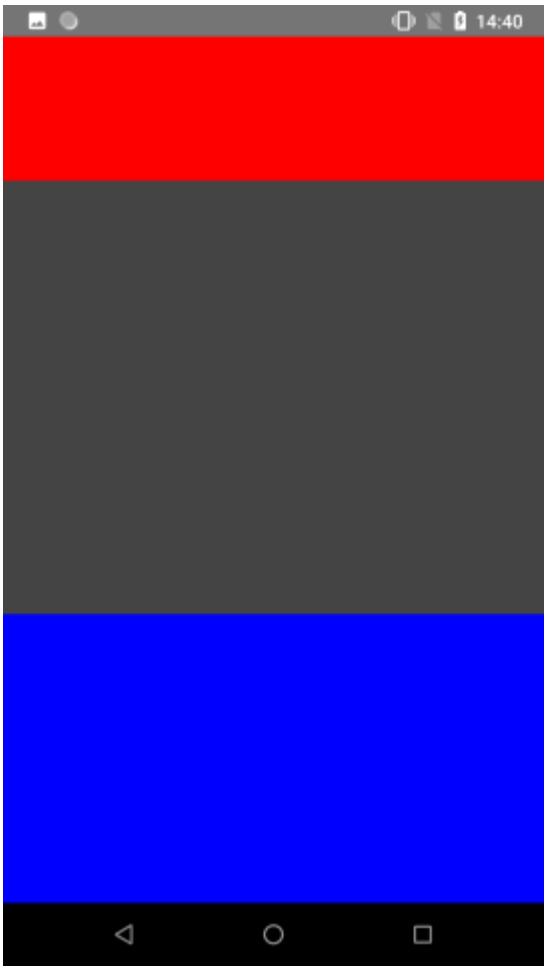
Контейнер Column позволяет назначить вложенным компонентам высоту в соответствии с их весом. Для указания веса применяется модификатор ColumnScope.weight. Стоит учитывать, что если контейнер Column обеспечивает вертикальную прокрутку или располагается в контейере, который предполагает вертикальную прокрутку, то веса компонентов игнорируются, поскольку общее пространство по вертикали условно бесконечно.

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Column()
            {
```

```
                Box(modifier =
Modifier.background(Color.Red).fillMaxWidth().weight(1f))
                Box(modifier =
Modifier.background(Color.Yellow).fillMaxWidth().weight(3f))
                Box(modifier =
Modifier.background(Color.Green).fillMaxWidth().weight(2f))
            }
        }
    }
}
```



Обратите внимание, как передается вес:

```
Modifier.background(Color.Red).fillMaxWidth().weight(1f)
```

В качестве веса в функцию weight() фактически передается доля пространства в единицах f. Так, первый элемент Box имеет вес 1f, второй - 3f, третий - 2f. Соовокупный вес всех комонентов, таким образом, будет 1f + 3f + 2f = 6f. И в итоге получится, что первый элемент получит от пространства 1f/6f часть или одну шестую. Второй элемент - 3f/6f или половину простанства, а третий - 2f/6f или одну третью пространства. Таким образом будет распределено пространство по вертикали между вложенными элементами.

Однако веса управляют только распределением пространства по вертикали, то есть для установки высоты. Ширина же устанавливается либо исходя из явно указанных значений ширины, либо исходя из вложенного содержимого. В данном случае по умолчанию вложенные элементы Box получат нулевую ширину, поэтому, чтобы их бло видно, в данном случае растягивает элементы по всей ширине с помощью модификатора fillMaxWidth().

Стоит отметить, что модификатор weight() может принимать второй параметр - булевое значение, которое указывает, будет ли ему выделяться пространство в соответствии с его весом. Если этот параметр равен false, то его вес не учитывается:

```
Box(modifier = Modifier.background(Color.Yellow).fillMaxWidth().weight(3f,
fill=false))
```

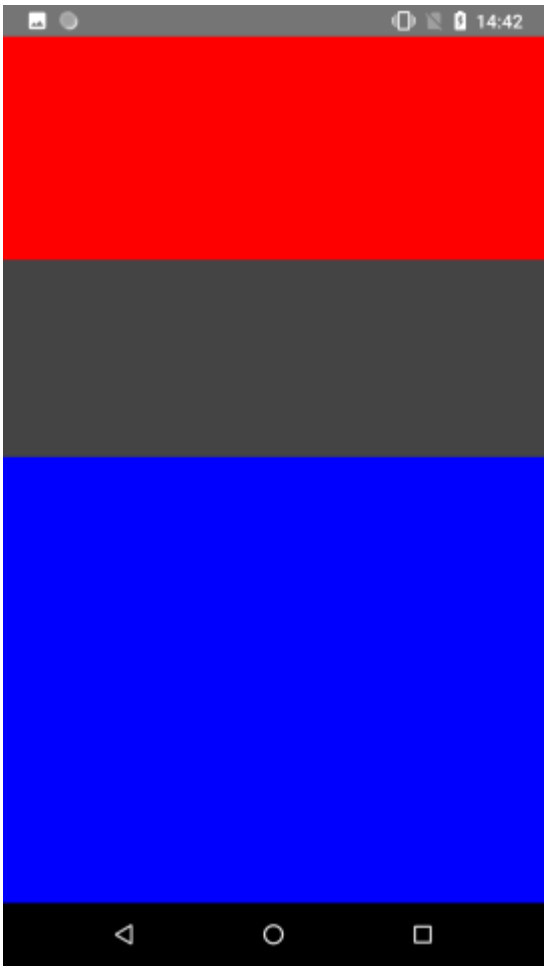
Сочетание весов и точных размеров

Если у компонента не указан вес, то контейер сначала запрашивает его предпочтительную высоту. Затем оставшее простанство распределяется между компонентами, для которых указан вес, в соответствии с их весом.

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Column()
            {
                Box(modifier =
Modifier.background(Color.Red).fillMaxWidth().weight(1f))
                Box(modifier =
Modifier.background(Color.Yellow).fillMaxWidth().height(150.dp))
                Box(modifier =
Modifier.background(Color.Green).fillMaxWidth().weight(2f))
            }
        }
    }
}
```



Здесь для второго элемента Box установлена высота в 150 единиц, а для остальных установлены веса - 1f и 2f. Таким образом, получится, что от всей длины контейера Column (а он будет растягиваться на весь экран) второй элемент получит высоту в 150 единиц. Оставшееся пространство затем будет распределено между первым и третим элементами в соответствии с их весами.

Установка размеров Column

Если ни у одного вложенного компонента НЕ указан вес, Column занимает то пространство, которое необходимо, чтобы вместить вложенные компоненты. Если подобный размер контейера нежелателен, то можно задать конкретный размер с помощью модификатора Modifier.height:

```
Column(modifier = Modifier.height(550.dp))
```

С помощью модификатора Modifier.fillMaxHeight можно растянуть контейер по всей длине экрана.

```
Column(modifier = Modifier.fillMaxSize())
```

Если как минимум для одного вложенного компонента указан вес, то в модификаторе Modifier.fillMaxHeight нет смысла. Однако в этот случае по прежнему можно использовать модификаторы Modifier.height и Modifier.size для ограничения длины контейера.

Позиционирование по вертикали и verticalArrangement

Если высота контейера Column больше суммы высот его вложенных компонентов, то для позиционирования этих компонентов может применяться параметр verticalArrangement, который может принимать следующие значения:

- Arrangement.Center: расположение по центру
- Arrangement.Bottom: расположение внизу
- Arrangement.Top: расположение вверху
- Arrangement.SpaceAround: компоненты равномерно распределяются по всей высоте с равномерными отступами между элементами, при этом отступы между первым и последним элементами и границами контейнера равен половине отступов между элементами
- Arrangement.SpaceBetween: компоненты равномерно распределяются по всей высоте с равномерными отступами между элементами, при этом первый и последний элементы прижимаются к границам контейнера
- rrangement.SpaceEvenly: компоненты равномерно распределяются по всей высоте с равномерными отступами между элементами, при этом отступы между первым и последним элементами и границами контейнера равны отступам между элементами

Например:

```
package com.example.helloapp

import android.os.Bundle
```

```
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Column(modifier = Modifier.fillMaxSize(), verticalArrangement =
Arrangement.SpaceBetween )
            {
                Box(modifier =
Modifier.background(Color.Red).fillMaxWidth().height(100.dp))
                Box(modifier =
Modifier.background(Color.Yellow).fillMaxWidth().height(100.dp))
                Box(modifier =
Modifier.background(Color.Green).fillMaxWidth().height(100.dp))
            }
        }
    }
}
```

Row

Контейнер Row располагает вложенные компоненты в строку. Функция Row принимает четыре параметра:

```
@Composable
inline fun Row(
    modifier: Modifier = Modifier,
    horizontalArrangement: Arrangement.Horizontal = Arrangement.Start,
    verticalAlignment: Alignment.Vertical = Alignment.Top,
    content: RowScope.() -> Unit
): @Composable Unit
```

- modifier: объект Modifier, который позволяет настроить внешний вид и поведение компонента
- horizontalArrangement: объект Arrangement.Horizontal, который устанавливает выравнивание компонента по горизонтали. По умолчанию имеет значение Arrangement.Start (расположение в вначале: слева для левосторонних языков и справа для правосторонних языков)
- verticalAlignment: объект Alignment.Vertical, который устанавливает выравнивание компонента по вертикали. По умолчанию имеет значение Alignment.Top (расположение вверху)
- content: объект интерфейса RowScope, который представляет вложенное содержимое

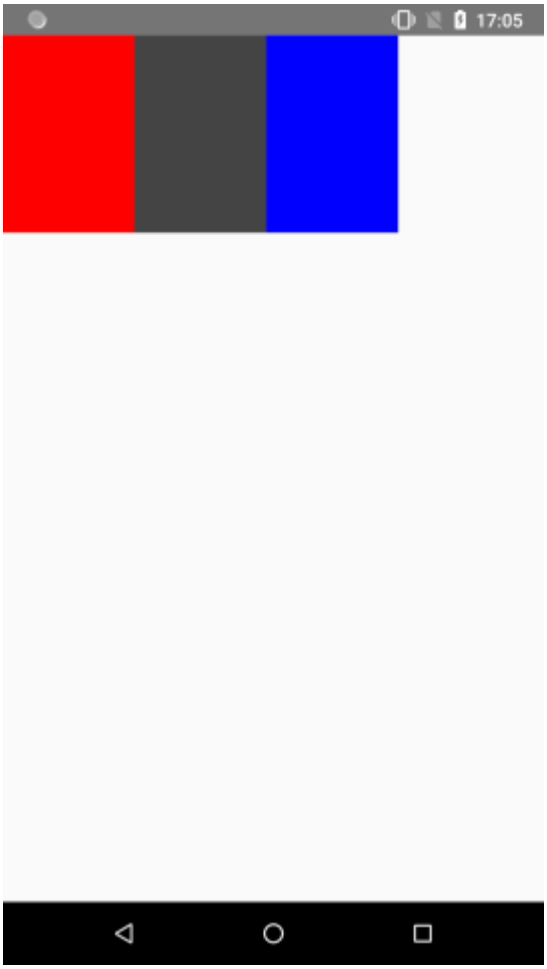
Разместим в Row несколько элементов Box:

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Row()
            {
                Box(modifier =
Modifier.background(Color.Red).height(150.dp).width(100.dp))
                Box(modifier =
Modifier.background(Color.Yellow).height(150.dp).width(100.dp))
                Box(modifier =
Modifier.background(Color.Green).height(150.dp).width(100.dp))
```

```
            }
        }
    }
}
```



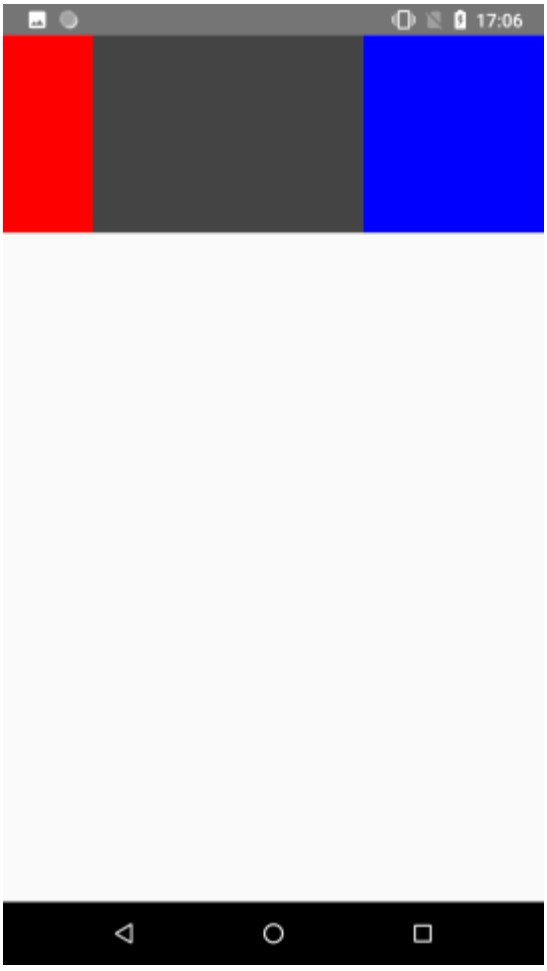
Установка веса компонентов

Контейнер Row позволяет назначить вложенным компонентам ширину в соответствии с их весом. Для указания веса применяется модификатор RowScope.weight. Стоит учитывать, что если контейнер Row обеспечивает горизонтальную прокрутку или располагается в контейере, который предполагает горизонтальную прокрутку, то веса компонентов игнорируются, поскольку общее пространство по горизонтали условно бесконечно.

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp
```

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Row()
            {
                Box(modifier =
Modifier.background(Color.Red).height(150.dp).weight(1f))
                Box(modifier =
Modifier.background(Color.Yellow).height(150.dp).weight(3f))
                Box(modifier =
Modifier.background(Color.Green).height(150.dp).weight(2f))
            }
        }
    }
}
```



Обратите внимание, как передается вес:

```
Box(modifier = Modifier.background(Color.Red).height(150.dp).weight(1f))
```

В качестве веса в функцию weight() фактически передается доля пространства в единицах f. Так, первый элемент Box имеет вес 1f, второй - 3f, третий - 2f. Соовокупный вес всех комонентов, таким образом, будет 1f + 3f + 2f = 6f. И в итоге получится, что первый элемент получит от пространства 1f/6f часть или

одну шестую. Второй элемент - 3f/6f или половину простанства, а третий - 2f/6f или одну третью пространства по горизонтали. Таким образом будет распределено пространство по горизонтали между вложенными элементами.

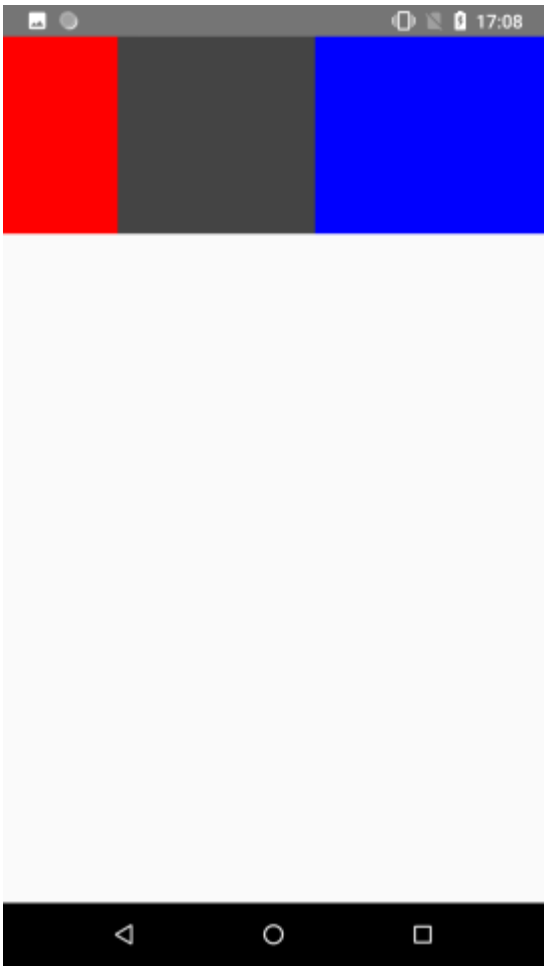
Сочетание весов и точных размеров

Если у компонента не указан вес, то контейнер сначала запрашивает его предпочтительную ширину. Затем оставшее простанство распределяется между компонентами, для которых указан вес, в соответствии с их весом.

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Row()
            {
                Box(modifier =
Modifier.background(Color.Red).height(150.dp).weight(1f))
                Box(modifier =
Modifier.background(Color.Yellow).height(150.dp).width(150.dp))
                Box(modifier =
Modifier.background(Color.Green).height(150.dp).weight(2f))
            }
        }
    }
}
```



Здесь для второго элемента Box установлена ширина в 150 единиц, а для остальных установлены веса - 1f и 2f. Таким образом, получится, что от всей ширины контейера Row (а он будет растягиваться на весь экран) второй элемент получит ширину в 150 единиц. Оставшееся пространство затем будет распределено между первым и третим элементами в соответствии с их весами.

Установка размеров Row

Если ни у одного вложенного компонента НЕ указан вес, Row занимает то пространство, которое необходимо, чтобы вместить вложенные компоненты. Если подобный размер контейера не желателен, то можно задать конкретный размер с помощью модификатора Modifier.width:

```
Row(modifier = Modifier.width(350.dp))
```

С помощью модификатора Modifier.fillMaxWidth можно растянуть контейер по всей длине экрана.

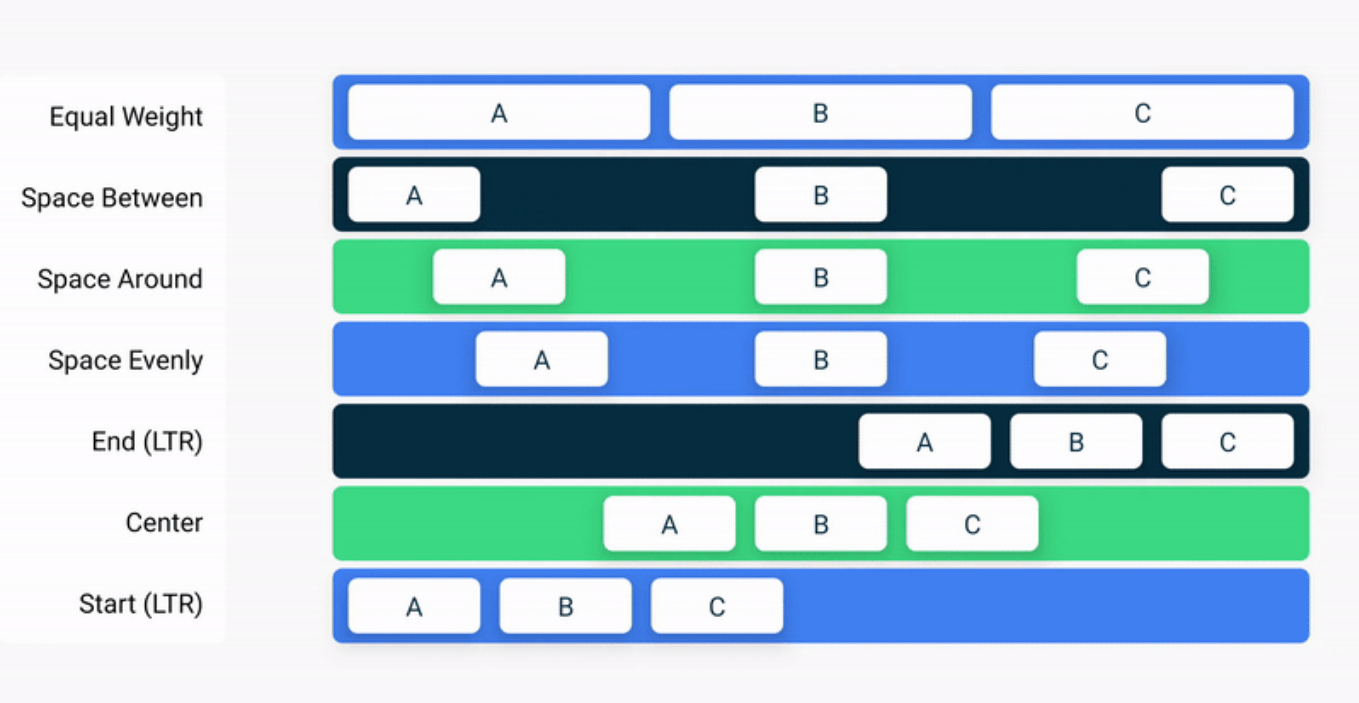
```
Row(modifier = Modifier.fillMaxWidth())
```

Если как минимум для одного вложенного компонента указан вес, то в модификаторе Modifier.fillMaxWidth нет смысла. Однако в этот случае по прежнему можно использовать модификаторы Modifier.width и Modifier.size для ограничения ширины контейера.

Позиционирование по горизонтали и horizontalArrangement

Если ширина контейера Row больше суммы ширин его вложенных компонентов, то для позиционирования этих компонентов по горизонтали может применяться параметр horizontalArrangement, который может принимать следующие значения:

- Arrangement.Center: расположение по центру
- Arrangement.End: расположение в конце (справа - для левосторонних языков, слева - для правосторонних языков)
- Arrangement.Start: расположение в начале (слева - для левосторонних языков, справа - для правосторонних языков)
- Arrangement.SpaceAround: компоненты равномерно распределяются по всей ширине с равномерными отступами между элементами, при этом отступы между первым и последним элементами и границами контейнера равен половине отступов между элементами
- Arrangement.SpaceBetween: компоненты равномерно распределяются по всей ширине с равномерными отступами между элементами, при этом первый и последний элементы прижимаются к границам контейнера
- Arrangement.SpaceEvenly: компоненты равномерно распределяются по всей ширине с равномерными отступами между элементами, при этом отступы между первым и последним элементами и границами контейнера равны отступам между элементами

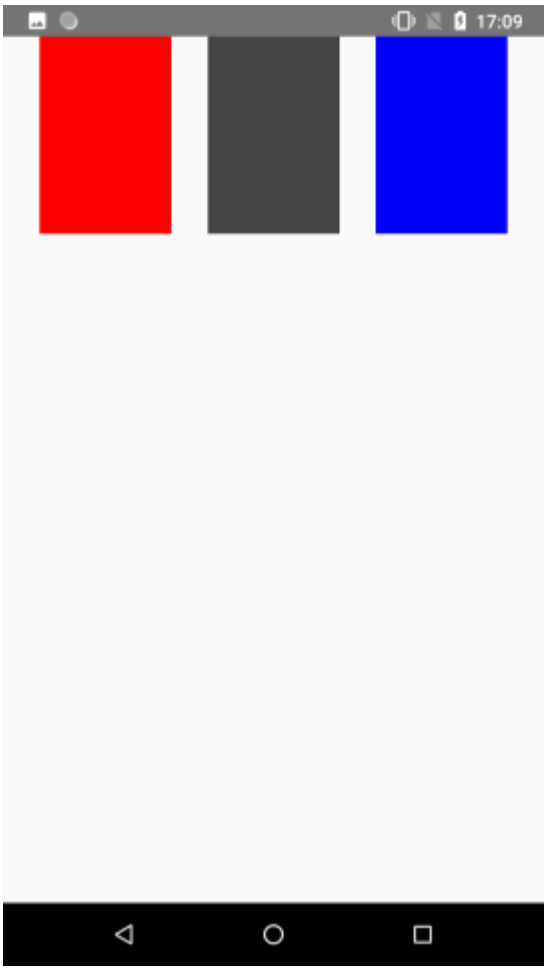


Например:

```
package com.example.helloapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Row(modifier = Modifier.fillMaxWidth(), horizontalArrangement =
Arrangement.SpaceEvenly)
            {
                Box(modifier =
Modifier.background(Color.Red).height(150.dp).width(100.dp))
                Box(modifier =
Modifier.background(Color.Yellow).height(150.dp).width(100.dp))
                Box(modifier =
Modifier.background(Color.Green).height(150.dp).width(100.dp))
            }
        }
    }
}
```


17:09