True Engineering

Основы QML

Qt Quick – технология быстрой разработки

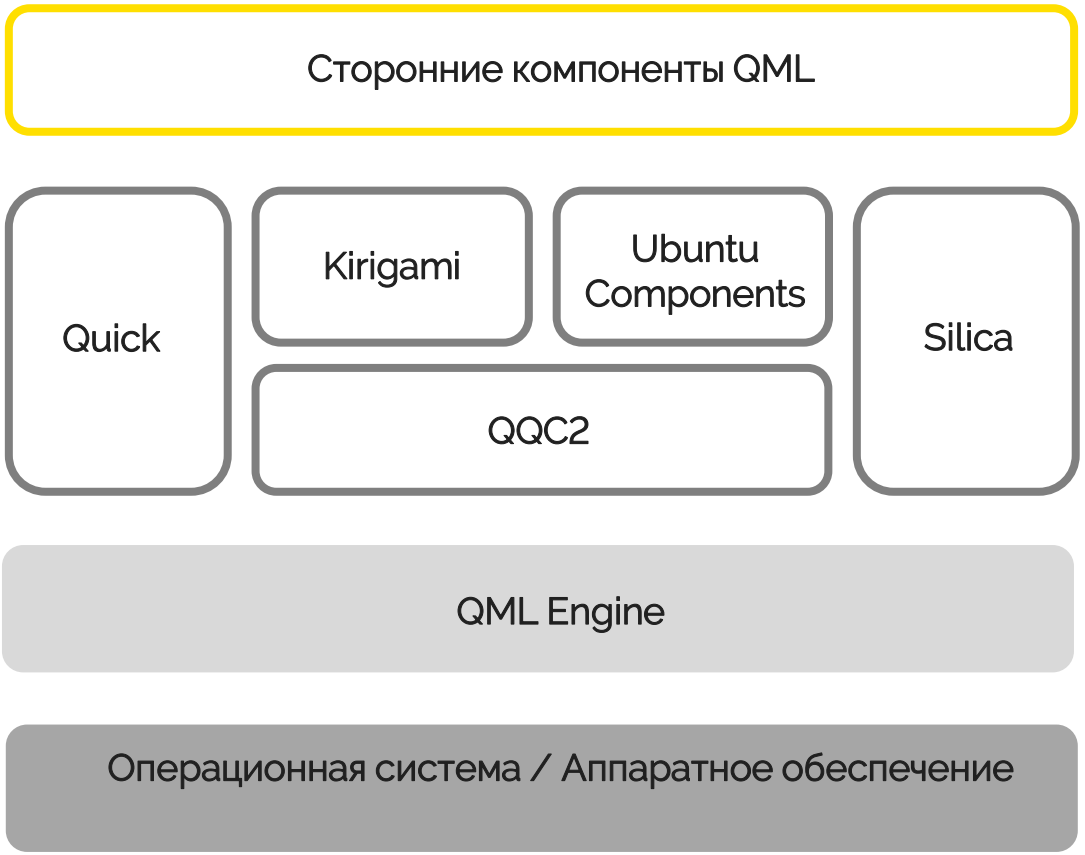
QML — язык разметки

- Декларативный
- Поддерживает вставки на «JavaScript»
- Доступны свойства объектов C++

QML Engine — интерпретатор QML

QtQuick — общие компоненты, поставляемые Qt

- import QtQuick 2.0



QML: Вложенные объекты, свойства, привязки

```
import QtQuick 2.6
import Sailfish.Silica 1.0

Page {
    Column {
        id: column
        width: parent.width
        spacing: Theme.paddingLarge
        PageHeader { title: qsTr("Page UI") }
        Label {
            x: Theme.horizontalPageMargin
            text: qsTr("Hello world!")
            color: Theme.highlightColor
            font.pixelSize: Theme.fontSizeHuge
        }
    }
}
```

Элементы компонента QML

Id — идентификатор для обращения к объекту

Properties — свойства определённых типов, обладающие названиями и значениями

Methods — исполняемый код на «JavaScript»

Signals — уведомления, испускаемые объектом

Signal Handlers — выражения или функции, инициируемые сигналами

Nested objects — вложенные объекты

Глоссарий терминов QML

Тип — базовый тип или объектный тип

Базовый тип — тип простых данных, например, int, string или bool

Объектный тип — тип, экземпляры которого могут быть созданы движком QML

Объект — экземпляр объектного типа

Компонент — шаблон для создания объекта или дерева объектов

Документ — самостоятельная часть кода QML

- Начинается с одного или нескольких операторов импорта
- Содержит одно объявление объекта верхнего уровня

Связывание (binding) — выражение JS, «привязанное» к свойству

Базовые типы QML

bool — логическое значение (true или false)

int — целое число

real — число с плавающей точкой

double — число с плавающей точкой, хранимое с двойной точностью

string — строка текста в свободной форме

url — ссылка на ресурсы

list — список объектов QML

enumeration — именованное перечисление

var — общий тип

Базовые типы QtQuick

date — дата

point — точка со свойствами x и y

rect — прямоугольник со свойствами x, y, width и height

size — размер со свойствами width и height

color — цвет в формате ARGB

font — шрифт со свойствами QFont

matrix4x4 — матрица с 4 строками и 4 столбцами

quaternion — кватернион со свойствами scalar, x, y и z

vector2d — вектор со свойствами x и y

vector3d — вектор со свойствами x, y и z

vector4d — вектор со свойствами x, y, z и w

Основные визуальные типы

Item — базовый тип

Rectangle — закрашенный прямоугольник с рамкой

Text — текст в заданном формате

Image — картинка

Column — вертикальный контейнер

Row — горизонтальный контейне

Flow — контейнер с построчным размещением

Grid — табличный контейнер

Canvas — элемент для рисования с помощью JavaScript

Item – базовый визуальный тип

parent : **Item** — родительский элемент

children : **list<Item>** — список потомков

resources : **list<Object>** — список ресурсов

data : **list<Object>** — потомки и ресурсы

x, **y** : **int** — позиция элемента

z : **int** — порядок отрисовки

width, **height** : **int** — размер элемента

visible : **bool** — отображен ли элемент

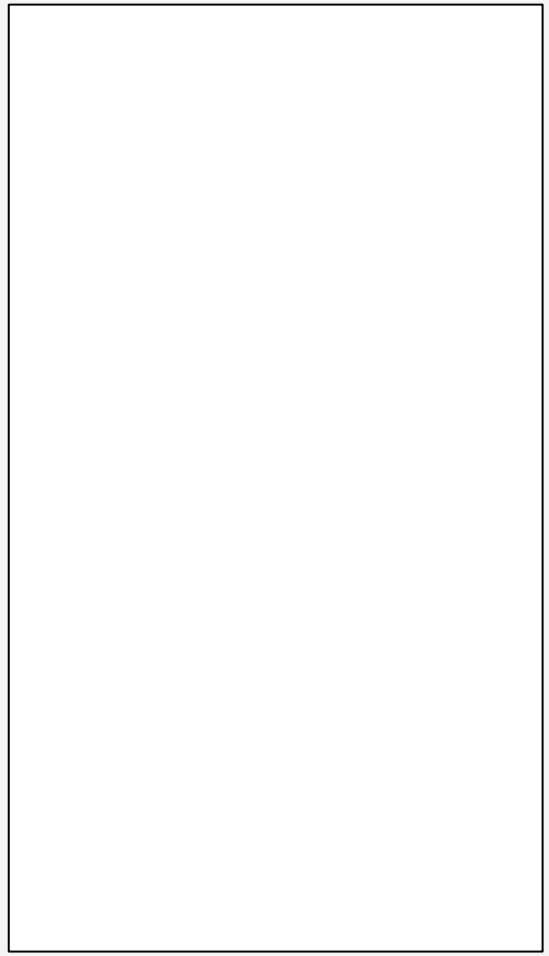
anchors — относительное позиционирование

clip : **bool** — включена ли обрезка

opacity : **real** — непрозрачность (от 0 до 1)

Простой пример QML-кода

```
import QtQuick 2.0

Item {
    width: 720
    height: 1280
}
```



Rectangle – закрашенный прямоугольник с рамкой

color : **color** — цвет для заполнения

radius : **real** — радиус скругления углов

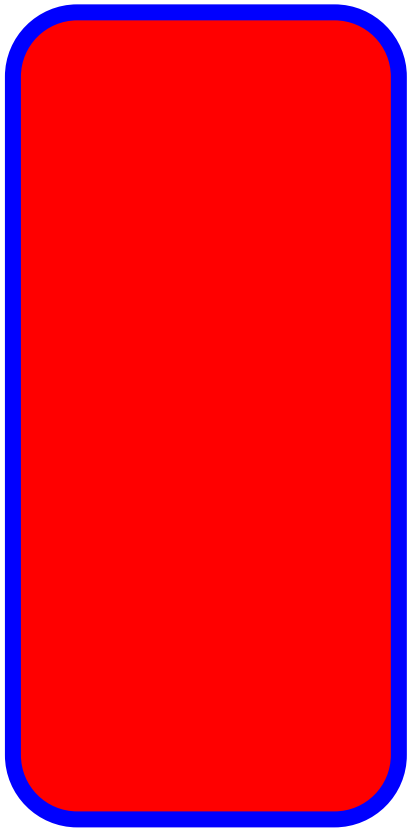
border — свойства рамки

- **width** : **int** — ширина рамки
- **color** : **color** — цвет рамки

gradient : **Gradient** — градиент для заполнения

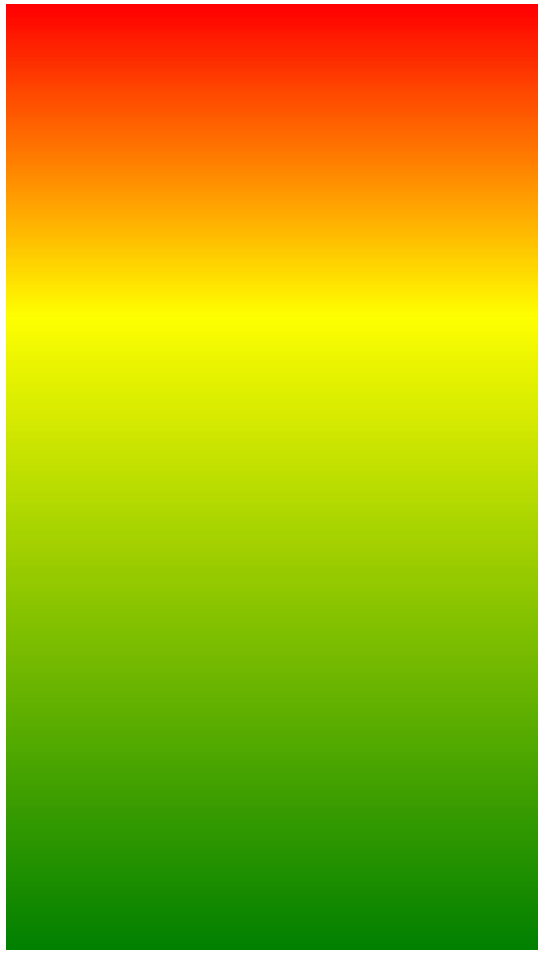
Пример прямоугольника

```
Rectangle {
    x: 100
    y: 100
    width: parent.width - 200
    height: parent.height - 200
    color: "#FF0000"
    radius: 60
    border { color: "blue"; width: 20 }
}
```



Пример с градиентом

```
Rectangle {
    width: parent.width
    height: parent.height
    gradient: Gradient {
        GradientStop { position: 0.00;
                       color: "red" }
        GradientStop { position: 0.33;
                       color: "yellow" }
        GradientStop { position: 1.00;
                       color: "green" }
    }
}
```



Text – текст в заданном формате

text : **string** — отображаемый текст

color : **color** — цвет текста

linkColor : **color** — цвет ссылок в тексте

font : **font** — свойства шрифта

textFormat : **enumeration** — способ отображения

- Text.AutoText, Text.PlainText,
- Text.StyledText, Text.RichText

horizontalAlignment : **enumeration** — горизонтальное выравнивание

verticalAlignment : **enumeration** — вертикальное выравнивание

lineCount : **int** — число отображаемых строк

contentWidth : **real** — ширина всего текста

contentHeight : **real** — высота всего текста

Примерыформатированного текста

```
Text {
    y: 0; width: parent.width
    text: "<b>Hello</b> <i>World!</i>"
    color: "red"
    font.pointSize: 48
}
Text {
    y: 200; width: parent.width
    text: "<b>Hello</b> <i>World!</i>"
    color: "green"
    font { pointSize: 48; underline: true }
    textFormat: Text.RichText
}
Text {
    y: 400; width: parent.width
    text: "<b>Hello</b> <i>World!</i>"
    color: "blue"
    font { pointSize: 32; bold: true }
    textFormat: Text.PlainText
}
```

Hello *World!*

Hello *World!*

** Hello World!**

Image - картинка

source : **url** — источник изображения

sourceSize : **QSize** — загружаемый размер изображения

paintedWidth, **paintedHeight** : **real** — отображаемый размер

asynchous : **bool** — загружать ли в отдельном потоке

progress : **real** — прогресс загрузки (от 0 до 1)

autoTransform : **bool** — применять ли трансформацию из EXIF

horizontalAlignment : **enumeration**

verticalAlignment : **enumeration**

Пример Image

```
Image {
    width: parent.width; height: parent.height
    source: "aurora.svg"
}
```

fillMode : enumeration

- **Image.Stretch** — растянуть по элементу
- **Image.PreserveAspectFit** — вписать без обрезки
- **Image.PreserveAspectCrop** — вписать с обрезкой
- **Image.Tile** — дублировать горизонтально и вертикально
- **Image.TileHorizontally** — растянуть вертикально, дублировать горизонтально
- **Image.TileVertically** — растянуть горизонтально, дублировать вертикально
- **Image.Pad** — не масштабировать

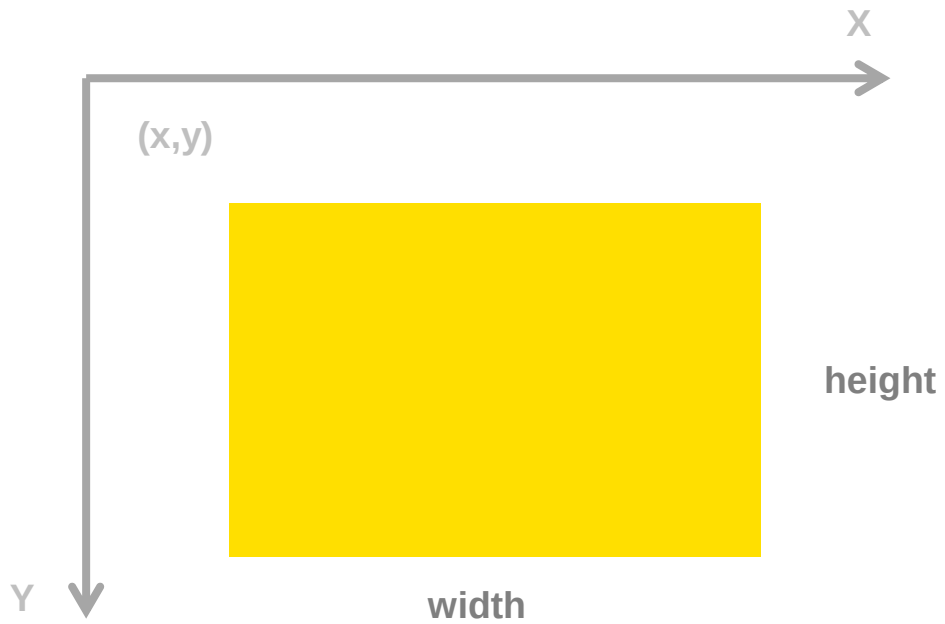
Способы позиционирования элементов

- Вручную — используя x, y, width и height
- Привязка с помощью anchors
- Контейнеры
- Компоновки

Позиционирование вручную

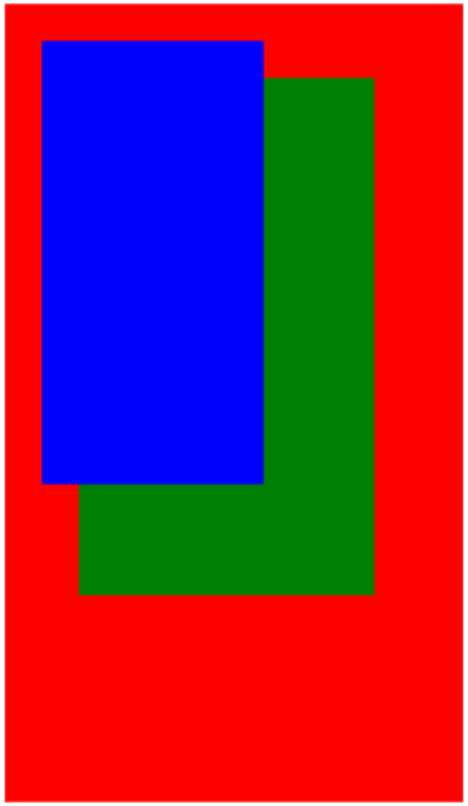
x, **y** : **int** — координаты относительно родителя

width, **height** : **int** — размеры



Пример размещения вручную

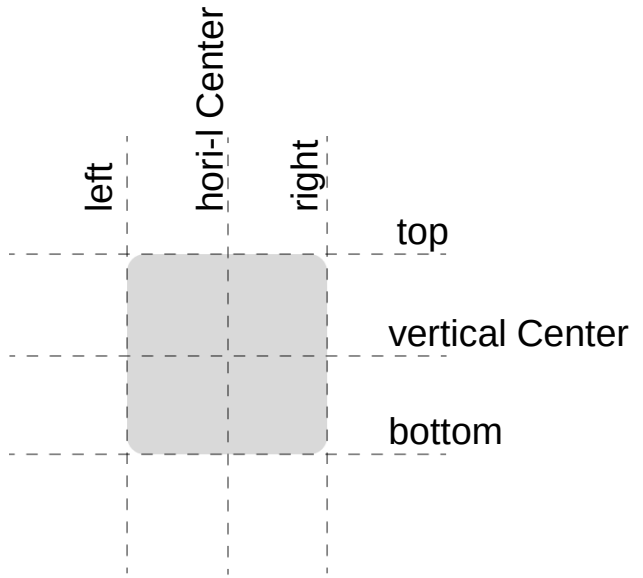
```
Rectangle {
    color: "red"
    x: 50; y: 100
    width: parent.width - 2 * x
    height: parent.height - 2 * y
    Rectangle {
        color: "green"
        x: 100; y: 100
        width: 400; height: 700
        Rectangle {
            color: "blue"
            x: -50; y: -50
            width: 300; height: 600
        }
    }
}
```

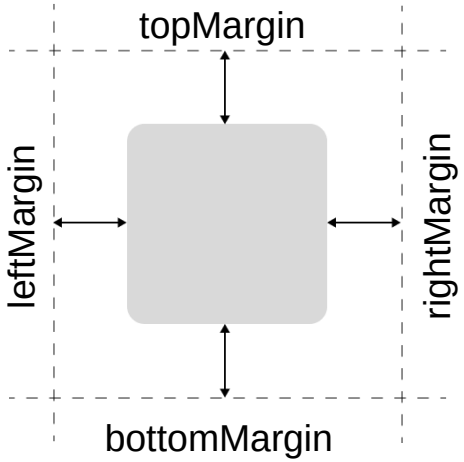


Использование якорей

Применяются для позиционирования относительно родителя или сайблингов **anchors**

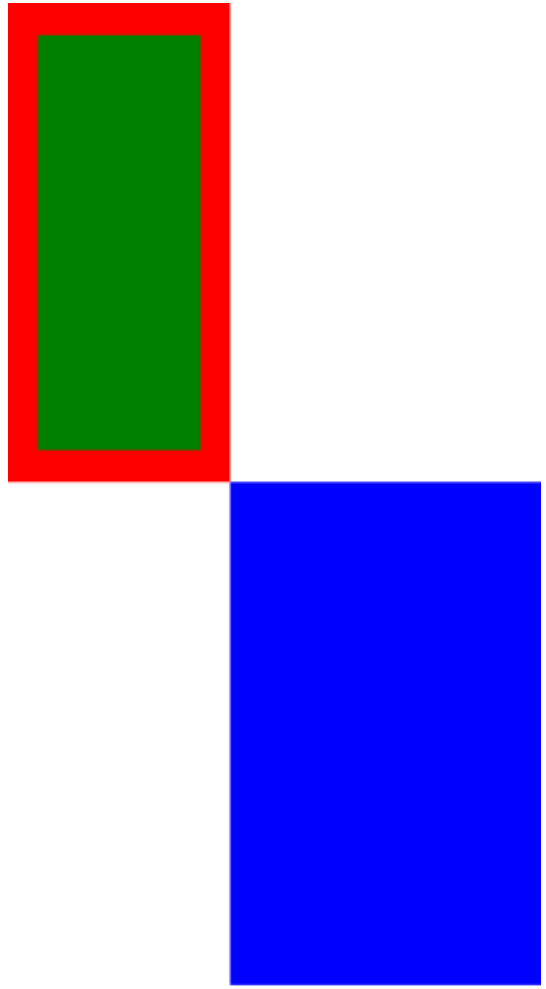
- **top**, **bottom**, **left**, **right** : **AnchorLine** — привязки границ
- **horizontalCenter**, **verticalCenter** : **AnchorLine** — привязки центральных линий
- **baseline** : **AnchorLine** — базовая линия текста
- **margins** : **real** — отступы у границ
- **topMargin**, **bottomMargin**, **leftMargin**, **rightMargin** : **real** — отступы у конкретных границ
- **fill** : **Item** — заполнить заданный элемент
- **centerIn** : **Item** — центрировать в заданном элементе





Пример использования якорей

```
Rectangle {
    id: redRect
    color: "red"
    width: 300; height: 600

    Rectangle {
        color: "green"
        anchors { fill: parent; margins: 40 }
    }
}
Rectangle {
    color: "blue"
    anchors {
        top: redRect.bottom
        bottom: parent.bottom
        bottomMargin: 50
        left: redRect.right
        right: parent.right
    }
}
```



Контейнеры

Column, **Row** — разместить в столбец или строку

- **spacing** : **real** — расстояние в пикселях между объектами
- **padding** : **real** — отступы вокруг содержимого
- **topPadding**, **bottomPadding**, **leftPadding**, **rightPadding** : **real** — отступы

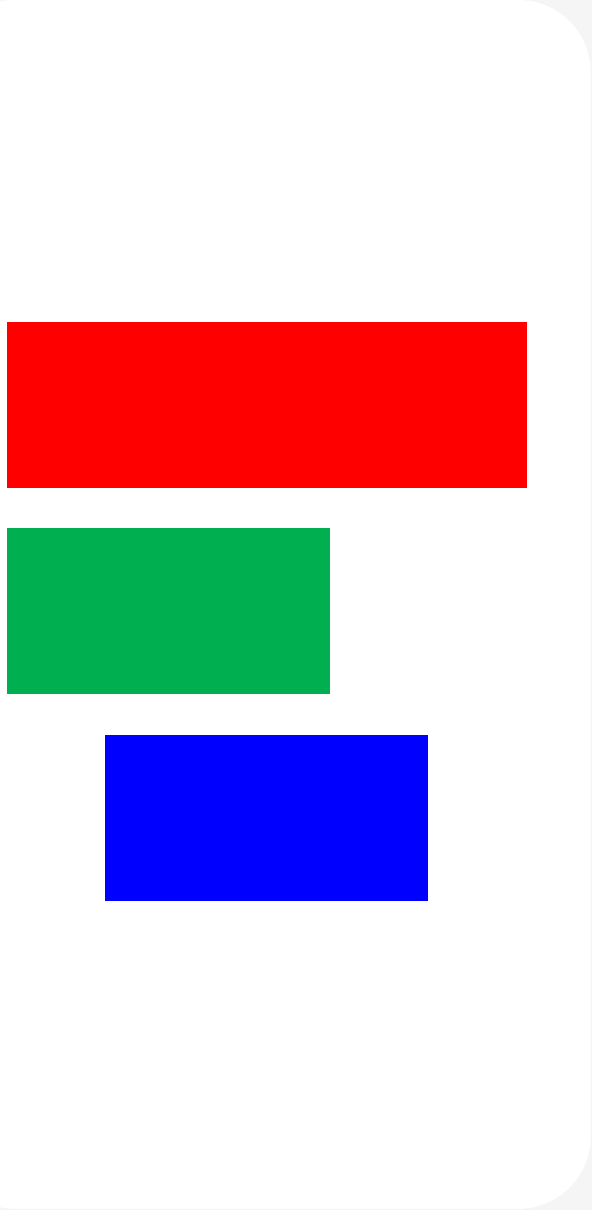
Flow — разместить один за другим, перенося, если нужно **flow** : **enumeration** — направление заполнения

Grid — разместить в таблицу

- **columns**, **rows** : **int** — количество столбцов и строк
- **columnSpacing**, **rowSpacing** : **real** — расстояние между столбцами и строками
- **flow** : **enumeration** — направление заполнения

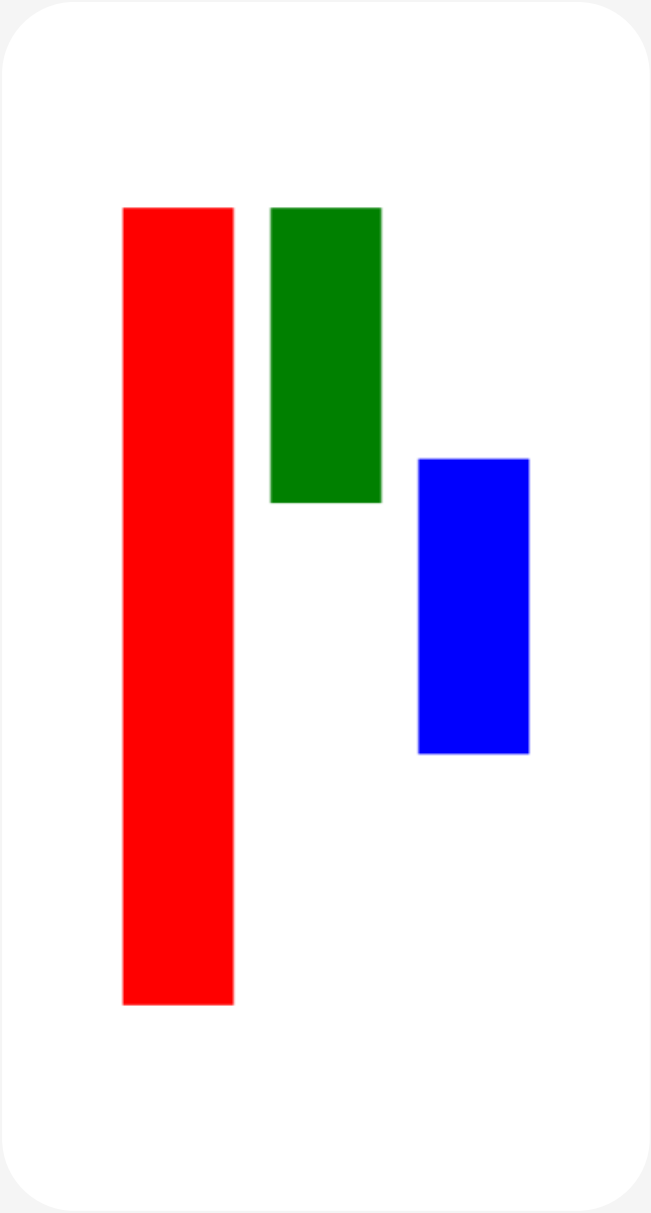
Пример вертикального контейнера

```
Column {
    anchors.centerIn: parent
    width: parent.width - 100
    spacing: 50
    Rectangle {
        color: "red"
        width: parent.width; height: 200
    }
    Rectangle {
        color: "green"
        width: 400; height: 200
    }
    Rectangle {
        color: "blue"
        width: 400; height: 200
        anchors.horizontalCenter: parent.horizontalCenter
    }
}
```



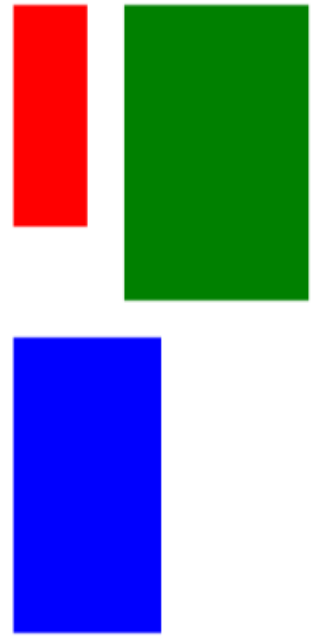
Пример горизонтального контейнера

```
Row {
    anchors.centerIn: parent
    height: parent.height - 200
    spacing: 50
    Rectangle {
        color: "red"
        width: 150; height: parent.height
    }
    Rectangle {
        color: "green"
        width: 150; height: 400
    }
    Rectangle {
        color: "blue"
        width: 150; height: 400
        anchors.verticalCenter: parent.verticalCenter
    }
}
```



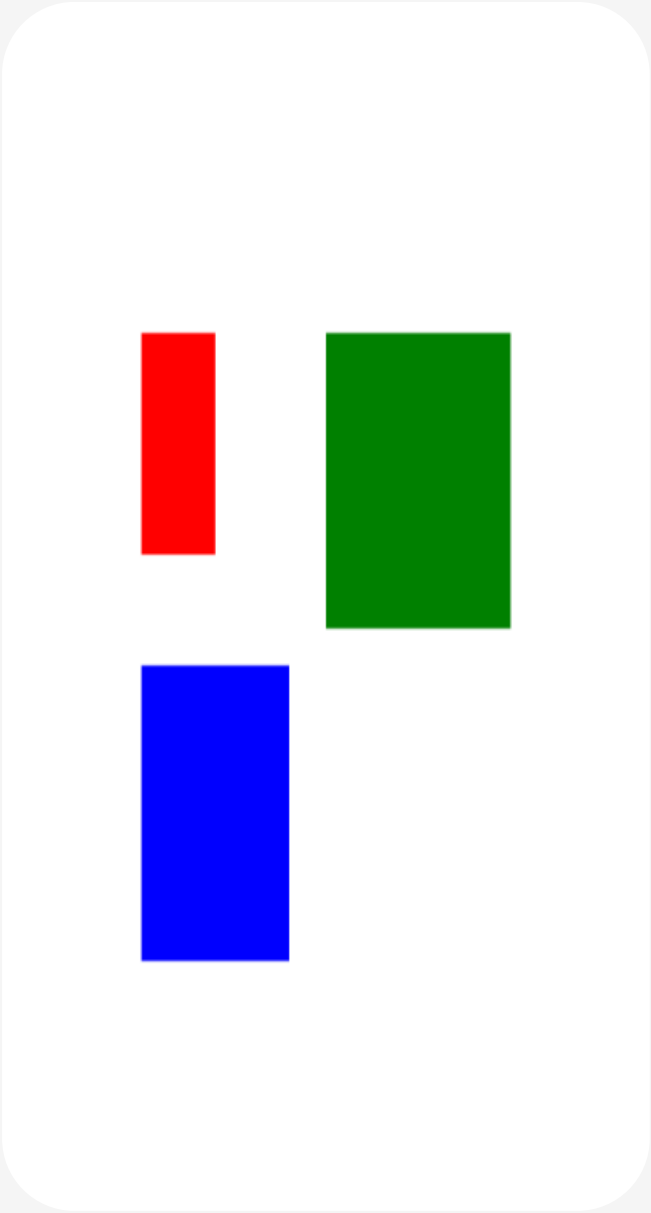
Установочные пакеты

```
Flow {
    width: 500
    spacing: 50


    Rectangle {
        color: "red"
        width: 100; height: 300
    }
    Rectangle {
        color: "green"
        width: 250; height: 400
    }
    Rectangle {
        color: "blue"
        width: 200; height: 400
    }
}
```



Пример сетки

```
Grid {
    width: 500
    spacing: 50
    columns: 2

    Rectangle {
        color: "red"
        width: 100; height: 300
    }
    Rectangle {
        color: "green"
        width: 250; height: 400
    }
    Rectangle {
        color: "blue"
        width: 200; height: 400
    }
}
```



Компоновки

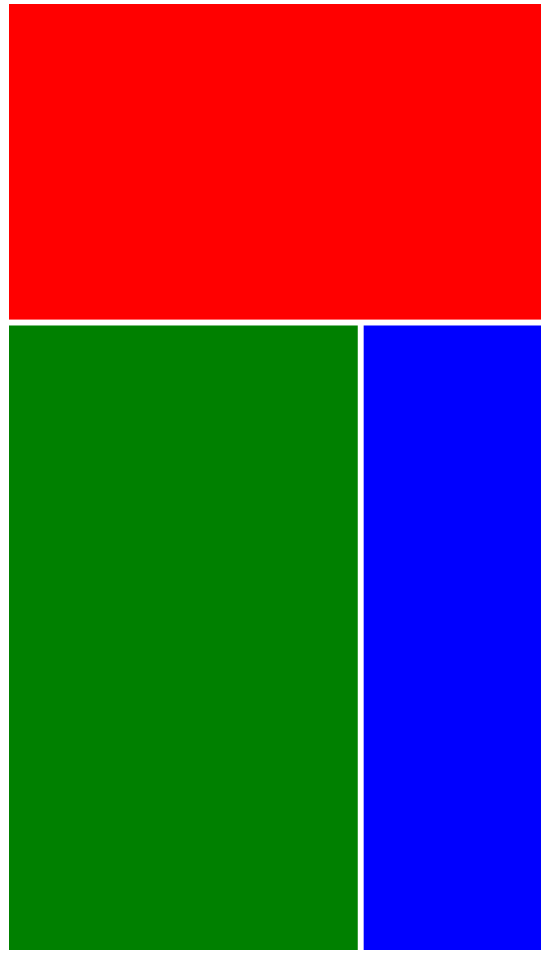
- import QtQuick.Layouts 1.3
- **ColumnLayout** — вертикальный столбец
- **RowLayout** — горизонтальная строка
- **GridLayout** — таблица
- **StackLayout** — отображение верхнего элемента в стеке

Присоединённые свойства:

- **Layout.fillHeight, Layout.fillWidth : bool** — растяжение
- **Layout.margins, Layout.leftMargin, Layout.rightMargin, Layout.topMargin, Layout.bottomMargin : real** — отступы
- **Layout.rowSpan, Layout.columnSpan : int** — объединения ячеек
- **Layout.alignment : Qt.Alignment** — отступы внутри ячейки

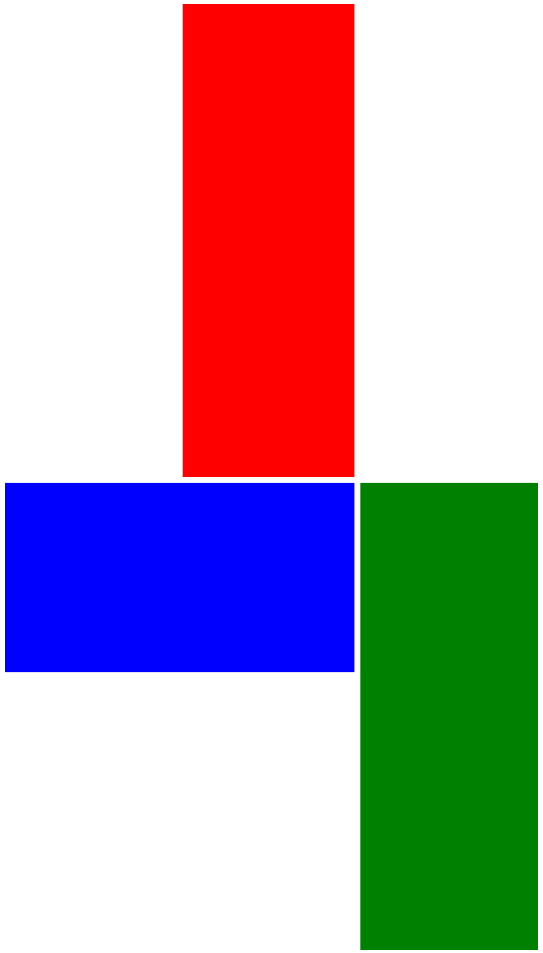
Пример компоновки

```
ColumnLayout {
    Rectangle {
        color: "red"
        Layout.fillWidth: true
        Layout.minimumHeight: parent.height / 3
    }
    RowLayout {
        Rectangle {
            color: "green"
            Layout.fillWidth: true
            Layout.fillHeight: true
        }
        Rectangle {
            color: "blue"
            Layout.preferredWidth: parent.width / 3
            Layout.fillHeight: true
        }
    }
}
```



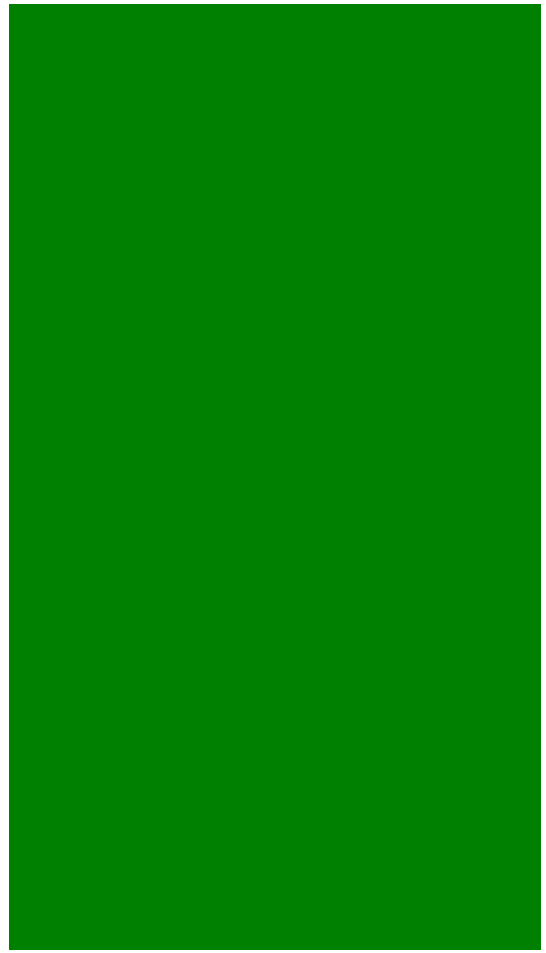
Пример компоновки таблицы

```
GridLayout {
    columns: 3; layoutDirection: Qt.RightToLeft
    Rectangle { color: "red"
        Layout.column: 1
        Layout.fillWidth: true
        Layout.minimumHeight: parent.height / 2
    }
    Rectangle { color: "green"
        Layout.row: 1
        Layout.rowSpan: 2
        Layout.fillHeight: true
        Layout.preferredWidth: parent.width / 3
    }
    Rectangle { color: "blue"
        Layout.columnSpan: 2
        Layout.fillWidth: true
        Layout.preferredHeight: parent.height / 5
    }
}
```



Пример компоновки стеком

```
StackLayout {
    currentIndex: count - 2

    Rectangle {
        color: "red"
    }
    Rectangle {
        color: "green"
    }
    Rectangle {
        color: "blue"
    }
}
```



Порядок отрисовки

Объекты отрисовываются по иерархии

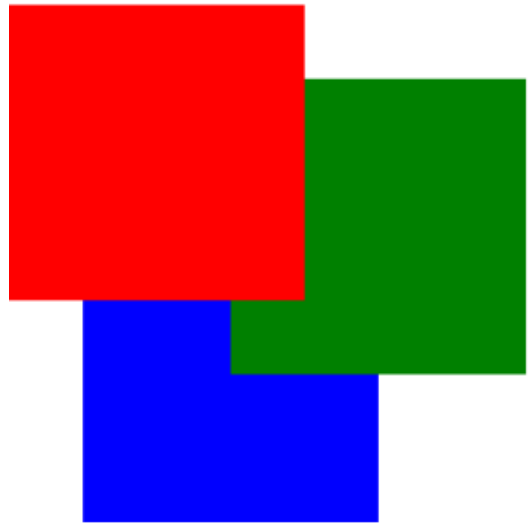
- Текущий объект
- 1й потомок с вложенными объектами
- 2й потомок с вложенными объектами
- …

z : int — поменять порядок соседей

- Значение по умолчанию: 0, может быть отрицательным
- Объект с меньшим z отрисовывается раньше
- Если z равны, то действует порядок иерархии

Пример Z-координаты

```
Rectangle {
    color: "red"
    y: 200
    width: 400; height: 400
    z: 1

    Rectangle {
        color: "green"
        x: 300; y: 100
        width: 400; height: 400
        z: -1
    }
}

Rectangle {
    color: "blue"
    x: 100; y: 500
    width: 400; height: 400
}
```



True Engineering

True Engineering

630128, г. Новосибирск,
ул. Кутателадзе, 4г

(383) 363-33-51, 363-33-50
info@trueengineering.ru
trueengineering.ru

Новосибирский Государственный Университет