  在 Qt Widgets 中，我们经常使用布局管理器来管理界面上的众多 widgets 。在 Qt Quick 中其实有两套与布局管理相关的类库，一套叫作 Item Positioner ，一套叫作 Item Layout 。 Item Layout 包括 RowLayout 、 ColumnLayout 、 GridLayout，与 Qt Widgets 中的布局管理器更相近，不过这里不打算介绍它们，本文的重点是 Item Positioner 。 Qt Quick 提供这么几种常用的 Positioners ：

* anchors ，锚布局
* Row ，行布局
* Column ，列布局
* Grid ，表格布局
* Flow ，流式布局

    咱们一个一个来看。

    请给我的决赛文章《[Qt Quick 图像处理实例之美图秀秀（附源码下载）](http://vote.blog.csdn.net/Article/Details?articleid=37740583)》投票，谢谢。

    版权所有 foruok ，转载请注明出处：[http://blog.csdn.net/foruok](http://blog.csdn.net/foruok" \t "_blank) 。

**anchors(锚)布局**

    在《[Qt on Android: Qt Quick 简单教程](http://blog.csdn.net/foruok/article/details/28859415" \t "_blank)》一文中我们已经介绍过锚布局了。为了自成篇幅，这里再重复一下，不过示例会更新哦。

    anchors 提供了一种方式，让你可以通过指定一个元素与其它元素的关系来确定元素在界面中的位置。

    你可以想象一下，每个 item 都有 7 条不可见的锚线：左（left）、水平中心（horizontalCenter）、上（top）、下（bottom）、右（right）、垂直中心（verticalCenter）、基线（baseline）。看图 1 就明白了：

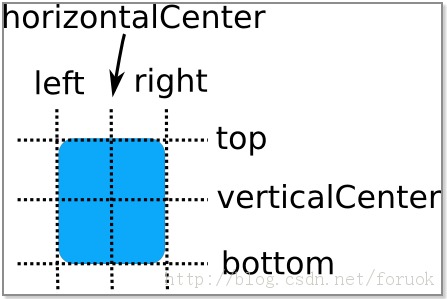


                图 1 锚线

    在图 1 中，没有标注基线，基线是用于定位文本的，你可以想象一行文字端坐基线的情景。对于没有文本的图元，baseline 和 top 一致。

    使用 anchors 布局时，除了对齐锚线，还可以在指定上（topMargin）、下（bottomMargin）、左（leftMargin）、右（rightMargin）四个边的留白。看图 2 就明白了：

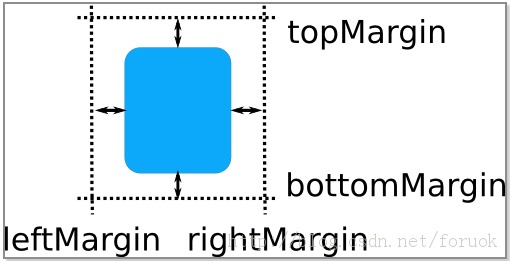


                图 2 留白

    除了图 1 和图 2 介绍的属性， anchors 还有一些属性， centerIn 表示将一个 item 居中放置到另一个 item 内， fill 表示充满某个 item ……更多请参考 Item 类的文档。

    好了，基础知识介绍完毕，看一个大而全的例子，文件名是 anchors\_layout.qml ，内容如下：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/33738227)[在CODE上查看代码片](https://code.csdn.net/snippets/412578)

1. **import** QtQuick 2.0
2. **import** QtQuick.Controls 1.1
4. Rectangle {
5. width: 360;
6. height: 240;
7. color: "#EEEEEE";
8. id: rootItem;
10. Text {
11. id: centerText;
12. text: "A Single Text.";
13. anchors.centerIn: parent;
14. font.pixelSize: 24;
15. font.bold: **true**;
16. }
18. **function** setTextColor(clr){
19. centerText.color = clr;
20. }
22. //color pickers look at parent's top
23. ColorPicker {
24. id: topColor1;
25. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
26. anchors.left: parent.left;
27. anchors.leftMargin: 4;
28. anchors.top: parent.top;
29. anchors.topMargin: 4;
30. onColorPicked: setTextColor(clr);
31. }
33. ColorPicker {
34. id: topColor2;
35. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
36. anchors.left: topColor1.right;
37. anchors.leftMargin: 4;
38. anchors.top: topColor1.top;
39. onColorPicked: setTextColor(clr);
40. }
42. ColorPicker {
43. id: topColor3;
44. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
45. anchors.right: parent.right;
46. anchors.rightMargin: 4;
47. anchors.top: parent.top;
48. anchors.topMargin: 4;
49. onColorPicked: setTextColor(clr);
50. }
52. ColorPicker {
53. id: topColor4;
54. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
55. anchors.right: topColor3.left;
56. anchors.rightMargin: 4;
57. anchors.top: topColor3.top;
58. onColorPicked: setTextColor(clr);
59. }
61. //color pickers sit on parent's bottom
62. ColorPicker {
63. id: bottomColor1;
64. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
65. anchors.left: parent.left;
66. anchors.leftMargin: 4;
67. anchors.bottom: parent.bottom;
68. anchors.bottomMargin: 4;
69. onColorPicked: setTextColor(clr);
70. }
72. ColorPicker {
73. id: bottomColor2;
74. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
75. anchors.left: bottomColor1.right;
76. anchors.leftMargin: 4;
77. anchors.bottom: bottomColor1.bottom;
78. onColorPicked: setTextColor(clr);
79. }
81. ColorPicker {
82. id: bottomColor3;
83. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
84. anchors.right: parent.right;
85. anchors.rightMargin: 4;
86. anchors.bottom: parent.bottom;
87. anchors.bottomMargin: 4;
88. onColorPicked: setTextColor(clr);
89. }
91. ColorPicker {
92. id: bottomColor4;
93. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
94. anchors.right: bottomColor3.left;
95. anchors.rightMargin: 4;
96. anchors.bottom: bottomColor3.bottom;
97. onColorPicked: setTextColor(clr);
98. }
100. //align to parent's left && vertical center
101. ColorPicker {
102. id: leftVCenterColor;
103. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
104. anchors.left: parent.left;
105. anchors.leftMargin: 4;
106. anchors.verticalCenter: parent.verticalCenter;
107. onColorPicked: setTextColor(clr);
108. }
110. //align to parent's right && vertical center
111. ColorPicker {
112. id: rightVCenterColor;
113. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
114. anchors.right: parent.right;
115. anchors.rightMargin: 4;
116. anchors.verticalCenter: parent.verticalCenter;
117. onColorPicked: setTextColor(clr);
118. }
120. //align to parent's top && horizontal center
121. ColorPicker {
122. id: topHCenterColor;
123. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
124. anchors.top: parent.top;
125. anchors.topMargin: 4;
126. anchors.horizontalCenter: parent.horizontalCenter;
127. onColorPicked: setTextColor(clr);
128. }
130. //align to parent's bottom && horizontal center
131. ColorPicker {
132. id: bottomHCenterColor;
133. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
134. anchors.bottom: parent.bottom;
135. anchors.bottomMargin: 4;
136. anchors.horizontalCenter: parent.horizontalCenter;
137. onColorPicked: setTextColor(clr);
138. }
139. }

    示例代码中用到的 ColorPicker 是我们在《[Qt Quick 组件与对象动态创建详解](http://blog.csdn.net/foruok/article/details/32730911" \t "_blank)》中自定义的组件，如果忘了请回头看看，要确保 ColorPicker.qml 与 anchors\_layout.qml 在一个文件夹内。 ColorPicker 定义了 colorPicked 信号，参数为 clr ，类型是 color 。我在示例代码中为每个 ColorPicker 对象创建了 onColorPicked 信号处理器，调用 setTextColor() 函数改变界面中间的文本的颜色。

    Qt 是 QML 提供的一个全局对象，提供了很多有用的方法和枚举值，这里我们使用 rgba() 函数来生成颜色值。 Math 对象是 JavaScript 语言那只对象，它的 random() 方法放回 0 到 1 之间的随机值。

    图 3 是执行 qmlscene anchors\_layout.qml 后的效果图：

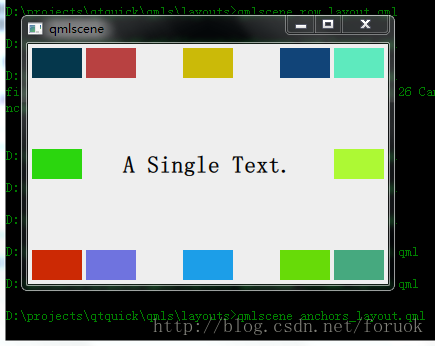


                图 3 锚布局示例效果

    锚布局是最灵活的一种 Qt Quick 布局方式，使用它你可以随意摆布界面上那些可见元素，不过，如果你的界面元素很多，它也将是代码量最大的一种布局方式。

    接下来我们看看那些传统的布局方式吧，如果你使用过 Qt Widgets ，相信会很快明白。

**行布局**

    anchors 实际上是 Item 的一个属性集，而 Row 则是一个单独的 Item ，专门用来管理其它 Item 的，后面介绍的几种布局，也是类似的。

    Row 沿着一行安置它的孩子们，在你需要水平放置一系列的 Item 时，它比锚布局更加方便。一旦你把一个 Item 交给 Row 来管理，那就不要再使用 Item 的 x 、 y 、 anchors 等属性了， Row 会安排得妥妥的。

    在一个 Row 内的 item ，可以使用 Positioner 附加属性来获知自己在 Row 中的更多位置信息。 Positioner 有 index 、 isFirstItem 、 isLastItem 三个属性。

    看我们的示例 row\_layout.qml ：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/33738227)[在CODE上查看代码片](https://code.csdn.net/snippets/412578)

1. **import** QtQuick 2.0
2. **import** QtQuick.Controls 1.1
4. Rectangle {
5. width: 360;
6. height: 240;
7. color: "#EEEEEE";
8. id: rootItem;
10. Text {
11. id: centerText;
12. text: "A Single Text.";
13. anchors.centerIn: parent;
14. font.pixelSize: 24;
15. font.bold: **true**;
16. }
18. **function** setTextColor(clr){
19. centerText.color = clr;
20. }
22. Row {
23. anchors.left: parent.left;
24. anchors.leftMargin: 4;
25. anchors.bottom: parent.bottom;
26. anchors.bottomMargin: 4;
27. spacing: 4;
29. ColorPicker {
30. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
31. onColorPicked: setTextColor(clr);
32. }
34. ColorPicker {
35. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
36. onColorPicked: setTextColor(clr);
37. }
39. ColorPicker {
40. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
41. onColorPicked: setTextColor(clr);
42. }
44. ColorPicker {
45. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
46. onColorPicked: setTextColor(clr);
47. }
48. }
49. }

    执行命令 qmlscene row\_layout.qml ，效果如图 4 所示：

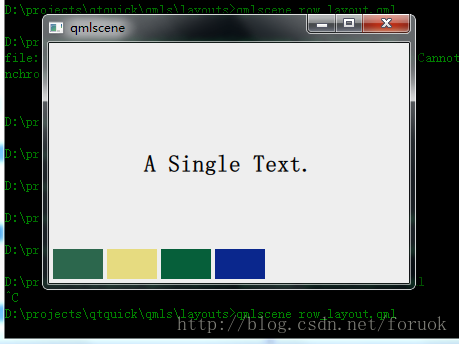


             图 4 行布局效果图

    因为 Row 本身是一个 Item ，所以你可以使用锚布局来定位一个 Row ，示例中这么做了，把 Row 放在界面的左下角。

    Row 有一个 spacing 属性，用来指定它管理的 Item 之间的间隔。还有一个 layoutDirection 属性，可以指定布局方向，取值为 Qt.LeftToRight 时从左到右放置 Item ，这是默认行为，取值为 Qt.RightToLeft 时从右向左放置 Item 。还有其它的一些属性，请参看 Qt SDK 。

**列布局**

    Column 与 Row 类似，不过是在垂直方向上安排它的子 Items 。在你需要垂直放置一系列的 Item 时，它比锚布局更加方便。

    Column 本身也是一个 Item ，可以使用 anchors 布局来决定它在父 Item 中的位置。 Column 的 spacing 属性描述子 Item 之间的间隔。

    看示例 column\_layout.qml ：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/33738227)[在CODE上查看代码片](https://code.csdn.net/snippets/412578)

1. **import** QtQuick 2.0
2. **import** QtQuick.Controls 1.1
4. Rectangle {
5. width: 360;
6. height: 240;
7. color: "#EEEEEE";
8. id: rootItem;
10. Text {
11. id: centerText;
12. text: "A Single Text.";
13. anchors.centerIn: parent;
14. font.pixelSize: 24;
15. font.bold: **true**;
16. }
18. **function** setTextColor(clr){
19. centerText.color = clr;
20. }
22. Column {
23. anchors.left: parent.left;
24. anchors.leftMargin: 4;
25. anchors.bottom: parent.bottom;
26. anchors.bottomMargin: 4;
27. spacing: 4;
29. ColorPicker {
30. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
31. onColorPicked: setTextColor(clr);
32. }
34. ColorPicker {
35. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
36. onColorPicked: setTextColor(clr);
37. }
39. ColorPicker {
40. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
41. onColorPicked: setTextColor(clr);
42. }
44. ColorPicker {
45. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
46. onColorPicked: setTextColor(clr);
47. }
48. }
49. }

    代码与 row\_layout.qml 类似，不用解释了，图 5 是执行 qmlscene column\_layout.qml 后的效果：



                图 5 列布局效果图

    与 Row 类似， Column 内的子 Item 也可以使用 Positioner 附加属性。

**表格布局**

    Grid 在一个网格上安置它的子 Items ，它会创建一个拥有很多单元格的网格，足够容纳它所有的子 Items 。 Grid 会从左到右、从上到下把它的子 items 一个一个塞到单元格里。 item 默认会被放在一个单元格左上角，(0,0) 位置。

    你可以通过 rows 和 columns 属性设定表格的行、列数。如果你不设置，默认只有四列，而行数则会根据实际的 item 数量自动计算。 rowSpacing 和 columnSpacing 指定行、列间距，单位是像素。

    Grid 的 flow 属性描述表格的流模式，可以取值 Grid.LeftToRight ，这是默认模式，从左到右一个挨一个放置 item ，一行放满再放下一行；取值为 Grid.TopToBottom 时，从上到下一个挨一个放置 item ，一列放满再放下一列。

    horizontalItemAlignment 和 verticalItemAlignment 指定单元格对齐方式。默认的单元格对齐方式和 layoutDirection 以及 flow 有关。

    先看个简单的例子， grid\_layout.qml ：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/33738227)[在CODE上查看代码片](https://code.csdn.net/snippets/412578)

1. **import** QtQuick 2.0
2. **import** QtQuick.Controls 1.1
4. Rectangle {
5. width: 360;
6. height: 240;
7. color: "#EEEEEE";
8. id: rootItem;
10. Text {
11. id: centerText;
12. text: "A Single Text.";
13. anchors.centerIn: parent;
14. font.pixelSize: 24;
15. font.bold: **true**;
16. }
18. **function** setTextColor(clr){
19. centerText.color = clr;
20. }
22. Grid {
23. anchors.left: parent.left;
24. anchors.leftMargin: 4;
25. anchors.bottom: parent.bottom;
26. anchors.bottomMargin: 4;
27. rows: 3;
28. columns: 3;
29. rowSpacing: 4;
30. columnSpacing: 4;
32. ColorPicker {
33. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
34. onColorPicked: setTextColor(clr);
35. }
37. ColorPicker {
38. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
39. onColorPicked: setTextColor(clr);
40. }
42. ColorPicker {
43. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
44. onColorPicked: setTextColor(clr);
45. }
47. ColorPicker {
48. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
49. onColorPicked: setTextColor(clr);
50. }
52. ColorPicker {
53. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
54. onColorPicked: setTextColor(clr);
55. }
57. ColorPicker {
58. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
59. onColorPicked: setTextColor(clr);
60. }
62. ColorPicker {
63. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
64. onColorPicked: setTextColor(clr);
65. }
66. }
67. }

    为了看出 flow 取值不同时的效果，我特意将行列数都设置为 3 ，创建了 7 个 ColorPicker 实例。图 6 是效果：



                图 6 表格布局，从左到右的效果

    如果设置 Grid 的 flow 属性为 Grid.TopToBottom （加一行代码 "flow: Grid.TopToBottom;"），可以看到图 7 的效果：



                图 7 表格布局，从上到下的效果

    调整 Grid 的其它属性会带来什么样的变化呢？请你试试吧。

**流布局**

    Flow 其实和 Grid 类似，不同之处是它没有显式的行、列数，它会计算自身尺寸和子 item 尺寸来根据需要折行。它的 flow 属性，默认取值 Flow.LeftToRight ，从左到右安排 item ，直到 Flow 本身的宽度被超出时折行；当 flow 取值 Flow.TopToBottom 时，从上到下安排 item ，直到 Flow 本身的高度被超出时开始在下一列上安排 item 。

    spacing 属性描述 item 之间的间隔。

    看个示例， flow\_layout.qml ：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/33738227)[在CODE上查看代码片](https://code.csdn.net/snippets/412578)

1. **import** QtQuick 2.0
2. **import** QtQuick.Controls 1.1
4. Rectangle {
5. width: 360;
6. height: 240;
7. color: "#EEEEEE";
8. id: rootItem;
10. Text {
11. id: centerText;
12. text: "A Single Text.";
13. anchors.horizontalCenter: parent.horizontalCenter;
14. anchors.top: parent.top;
15. font.pixelSize: 24;
16. font.bold: **true**;
17. }
19. **function** setTextColor(clr){
20. centerText.color = clr;
21. }
23. Flow {
24. anchors.left: parent.left;
25. anchors.leftMargin: 4;
26. anchors.bottom: parent.bottom;
27. anchors.bottomMargin: 4;
28. width: 280;
29. height: 130;
30. spacing: 4;
32. ColorPicker {
33. width: 80;
34. height: 20;
35. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
36. onColorPicked: setTextColor(clr);
37. }
39. ColorPicker {
40. width: 100;
41. height: 40;
42. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
43. onColorPicked: setTextColor(clr);
44. }
46. ColorPicker {
47. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
48. onColorPicked: setTextColor(clr);
49. }
51. ColorPicker {
52. width: 80;
53. height: 25;
54. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
55. onColorPicked: setTextColor(clr);
56. }
58. ColorPicker {
59. width: 35;
60. height: 35;
61. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
62. onColorPicked: setTextColor(clr);
63. }
65. ColorPicker {
66. width: 20;
67. height: 80;
68. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
69. onColorPicked: setTextColor(clr);
70. }
72. ColorPicker {
73. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
74. onColorPicked: setTextColor(clr);
75. }
76. }
77. }

    我改变了 ColorPicker 实例的大小，以便观察 Flow 布局的特点：根据自身宽高是否被 item 超出而自动折行。图 8 是 flow 为 LeftToRight （代码中未设置 flow 属性，默认值是 LeftToRight）时的效果:



                图 8 流布局，从左到右效果

    修改下代码，在 Flow 对象生命内添加 "flow: Flow.TopToBottom;" 这行代码，再次执行 qmlscene flow\_layout.qml ，效果如图 9 所示：

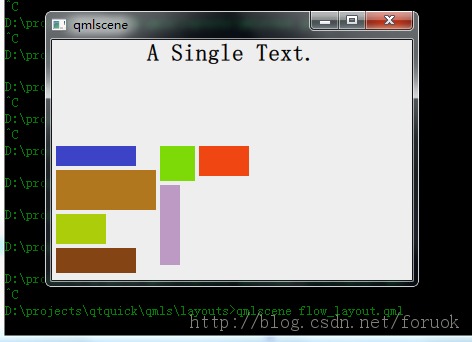


                图 9 流布局，从上到下效果

    如你所见，效果大大不同。

    其实可以把流布局想象成英文文字排版系统，一个 item 对应一个单词，横版模式时，从左到右，一行一行安排单词的位置，当接近一行的宽度时，如果下一个单词摆上去就会超出行宽，那就把这个单词放到下一行上，继续排排排……；竖版模式也是类似的……也许你看过竖版书，很容易理解这件事情。

    常见布局介绍完了，在学习 Qt Widgets 的布局管理器时，我们知道布局可以嵌套，比如我经常拿 QVBoxLayout 和 QHBoxLayout 嵌套来完成一些界面的布局。那 Qt Quick 中的布局是否可以嵌套呢？

**嵌套布局**

    Qt Quick 中布局是可以嵌套的，比如 Row 和 Column 可以相互嵌套来实现 Grid 布局的效果。

    看下 nested\_layout.qml ：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/33738227)[在CODE上查看代码片](https://code.csdn.net/snippets/412578)

1. **import** QtQuick 2.0
2. **import** QtQuick.Controls 1.1
4. Rectangle {
5. width: 360;
6. height: 240;
7. color: "#EEEEEE";
8. id: rootItem;
10. Text {
11. id: centerText;
12. text: "A Single Text.";
13. anchors.centerIn: parent;
14. font.pixelSize: 24;
15. font.bold: **true**;
16. }
18. **function** setTextColor(clr){
19. centerText.color = clr;
20. }
22. Row {
23. anchors.left: parent.left;
24. anchors.leftMargin: 4;
25. anchors.bottom: parent.bottom;
26. anchors.bottomMargin: 4;
27. spacing: 4;
29. Column {
30. spacing: 4;
31. ColorPicker {
32. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
33. onColorPicked: setTextColor(clr);
34. }
36. ColorPicker {
37. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
38. onColorPicked: setTextColor(clr);
39. }
40. }
42. Column {
43. spacing: 4;
44. ColorPicker {
45. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
46. onColorPicked: setTextColor(clr);
47. }
49. ColorPicker {
50. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
51. onColorPicked: setTextColor(clr);
52. }
53. }
55. Column {
56. spacing: 4;
57. ColorPicker {
58. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
59. onColorPicked: setTextColor(clr);
60. }
62. ColorPicker {
63. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
64. onColorPicked: setTextColor(clr);
65. }
66. }
67. }
68. }

    我在一个 Row 内嵌套了 3 个 Column ，实现了 2x3 的表格布局。执行 qmlscene nested\_layout.qml ，可以看到图 10 ：

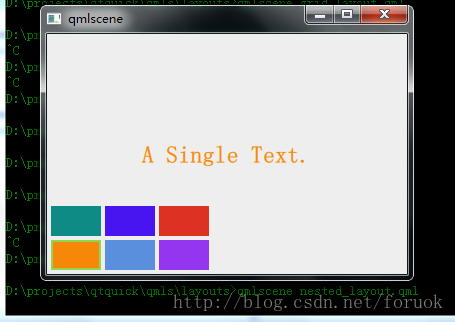


                图 10 Row 与 Column 嵌套

    嵌套时，比如放在 Row 内的 Column ，其实对于 Row 来讲和其它非布局类的 item 一样，没什么区别。

    我们修改一下上面的代码，嵌套两个 Column ，让其它两个 ColorPicker 实例与 Column 处在同一层级。代码如下：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/33738227)[在CODE上查看代码片](https://code.csdn.net/snippets/412578)

1. **import** QtQuick 2.0
2. **import** QtQuick.Controls 1.1
4. Rectangle {
5. width: 360;
6. height: 240;
7. color: "#EEEEEE";
8. id: rootItem;
10. Text {
11. id: centerText;
12. text: "A Single Text.";
13. anchors.centerIn: parent;
14. font.pixelSize: 24;
15. font.bold: **true**;
16. }
18. **function** setTextColor(clr){
19. centerText.color = clr;
20. }
22. Row {
23. anchors.left: parent.left;
24. anchors.leftMargin: 4;
25. anchors.bottom: parent.bottom;
26. anchors.bottomMargin: 4;
27. spacing: 4;
29. Column {
30. spacing: 4;
31. ColorPicker {
32. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
33. onColorPicked: setTextColor(clr);
34. }
36. ColorPicker {
37. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
38. onColorPicked: setTextColor(clr);
39. }
40. }
42. Column {
43. spacing: 4;
44. ColorPicker {
45. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
46. onColorPicked: setTextColor(clr);
47. }
49. ColorPicker {
50. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
51. onColorPicked: setTextColor(clr);
52. }
53. }
55. //Column {
56. //    spacing: 4;
57. ColorPicker {
58. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
59. onColorPicked: setTextColor(clr);
60. }
62. ColorPicker {
63. color: Qt.rgba(Math.random(), Math.random(), Math.random(), 1.0);
64. onColorPicked: setTextColor(clr);
65. }
66. //}
67. }
68. }

    注意代码中我只是把最后一个 Column 对象声明给注释掉了，图 11 是运行效果：



                图 11 嵌套布局

    可能你想到了，锚布局无法和其它布局嵌套……你可以试一下。

    好啦，嵌套布局就介绍到这里，感兴趣的话请自行实验其它布局之间的嵌套效果。

    版权所有 foruok ，转载请注明出处：<http://blog.csdn.net/foruok> 。