# Qt 之手动布局

# 

# 简述

手动布局，可以实现和水平布局、垂直布局、网格布局等相同的效果，也可实现属于自己的自定义布局，当窗体缩放时，控件可以随之变化。

其对于坐标系的建立有严格要求，纯代码思维，使用复杂，不易维护，所以一般不建议使用。

下面我们以简单的例子来讲解如何使用。

# 事件

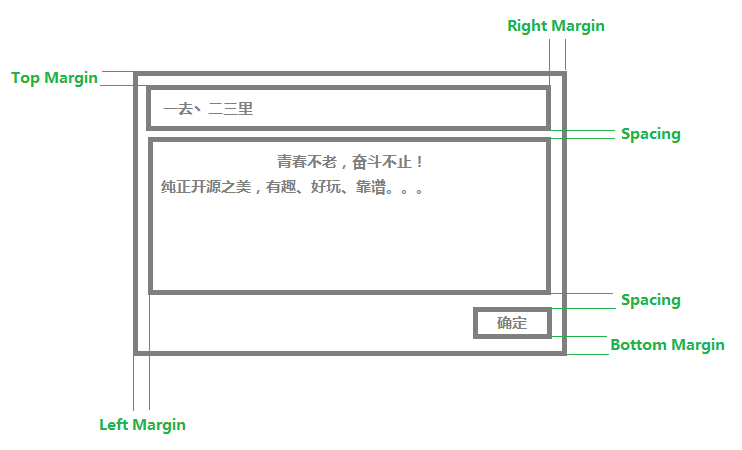
在[Qt之布局管理器](http://blog.csdn.net/liang19890820/article/details/51517002)一节中，我们介绍了对于手动布局的实现思路。

通过QWidget::resizeEvent()来计算所需要分配的大小，并且给每个子控件调用setGeometry()。

# 垂直布局

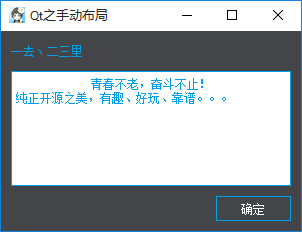
## 简略图

我们先简单看一个简略图，是一个一个垂直布局的界面，包含各个部位的组成以及大小限制。



其中Left Margin、Top Margin、Right Margin、Bottom Margin分别指各个控件距离窗体的左、上、右、下外边距，Spacing指控件之间的间距。

## 效果



## 源码

void MainWindow::resizeEvent(QResizeEvent \*event)

{

QWidget::resizeEvent(event);

int nSpacing = 10;

int nLeft = 10;

int nTop = 10;

int nRight = 10;

int nBottom = 10;

*// 标签位置、大小*

m\_pLabel->setGeometry(nLeft, nTop, width() - nLeft - nRight, 20);

*// 按钮大小*

m\_pButton->setFixedSize(75, 25);

*// 设置按钮位置*

m\_pButton->move(width() - m\_pButton->width() - nRight, height() - nBottom - m\_pButton->height());

*// 中央窗体位置、大小*

m\_pCentralWidget->setGeometry(nLeft, nTop + nSpacing + m\_pLabel->height(),

width() - nLeft - nRight, height() - nTop - 2 \*nSpacing - m\_pLabel->height() - m\_pButton->height() - nBottom);

}

标签和中央窗体的宽度均为：width() - nLeft - nRight，即：窗体宽度 - 左边距 - 右边距。

标签的坐标为QPoint(nLeft, nTop)，即：x为左边距，y为上边距。

中央窗体的坐标为：QPoint(nLeft, nTop + nSpacing + m\_pLabel->height())，即：x：左边距，y：上边距 + 间距 + 标签高度。

中央窗体的高度为：height() - nTop - 2 nSpacing - m\_pLabel->height() - m\_pButton->height() - nBottom。即：窗体高度 - 上边距 - 标签高度 - 按钮高度 - 下边距 - 2 间距。

按钮大小为：QSize(75, 25)，即：width：75，height：25。

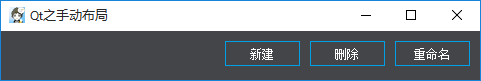
按钮位置：QPoint(width() - m\_pButton->width() - nRight, height() - nBottom - m\_pButton->height())。即：x：窗体宽度 - 按钮宽度 - 右边距，y：窗体高度 - 按钮高度 - 下边距。

# 水平布局

实现思路不再过多讲解，参考垂直布局。

## 效果





## 源码

void MainWindow::resizeEvent(QResizeEvent \*event)

{

QWidget::resizeEvent(event);

int nSpacing = 10;

int nLeft = 10;

int nTop = 10;

int nRight = 10;

int nBottom = 10;

m\_pLeftButton->setFixedSize(75, 25);

m\_pCentralButton->setFixedSize(75, 25);

m\_pRightButton->setFixedSize(75, 25);

*// 居左*

*//m\_pLeftButton->move(nLeft, nTop);*

*//m\_pCentralButton->move(nLeft + m\_pLeftButton->width() + nSpacing, nTop);*

*//m\_pRightButton->move(nLeft + m\_pLeftButton->width() + m\_pCentralButton->width() + 2 \* nSpacing, nTop);*

*// 居右*

m\_pRightButton->move(width() - m\_pRightButton->width() - nRight, nTop);

m\_pCentralButton->move(width() - m\_pCentralButton->width() - m\_pRightButton->width() - nSpacing - nRight, nTop);

m\_pLeftButton->move(width() - m\_pLeftButton->width() - m\_pCentralButton->width() - m\_pRightButton->width() - 2 \* nSpacing - nRight, nTop);

}

# 

# 总结

好了，关于网格布局或其他复杂布局就不再一一介绍了，主要计算好各个控件的相对坐标即可。

有人肯定会有疑惑，为什么不使用setGeometry(10， 10， 100， 75)，而必须通过resizeEvent()来计算坐标呢？

主要区别：

* setGeometry(10， 10， 100， 75)采用绝对坐标和位置，适用于窗体大小固定的情况。一旦大小发生变化，则无计可施。
* resizeEvent()计算坐标属于相对位置，无论窗体如何变化，都可以适应其在窗体中的布局。