



# 第10篇 (2D绘图) 绘制文字

代码地址：<https://github.com/Lornatang/QtStartQuicklyTutorial/tree/main/Painter01>

## 目录

目录

一、基本绘制

二、控制文字的位置

三、使用字体

## 一、基本绘制

我们接着在上一节的项目上进行讲解，首先将 `paintEvent()` 函数更改如下：

```
void MainWindow::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    painter.drawText(100, 100, "qter.org_yafeilinux");
}
```

这样就在 `(100, 100)` 的位置绘制了一个字符串。效果如下图所示。



## 二、控制文字的位置

1. 我们先到QPainter的帮助文档页面，然后查看 `drawText()` 函数的重载形式，找到：  
`QPainter::drawText ( const QRectF &rectangle, int flags, const QString &text, QRectF * boundingRect = 0 )`，如下图所示。

```
void QPainter::drawText ( const QRectF &rectangle, int flags, const QString &text, QRectF * boundingRect = 0 )
```

This is an overloaded function.

Draws the given *text* within the provided *rectangle*.



它的第一个参数指定了绘制文字所在的矩形；第二个参数指定了文字在矩形中的对齐方式，它由 `Qt::AlignmentFlag` 枚举变量进行定义，不同对齐方式也可以使用 `|` 操作符同时使用，这里还可以使用 `Qt::TextFlag` 定义的其他一些标志，比如自动换行等；第三个参数就是所要绘制的文字，这里可以使用 `\n` 来实现换行；第四个参数一般不用设置。

2. 下面我们来看一个例子。为了更明显的看到文字在指定矩形中的位置，我们绘制出这个矩形。将 `paintEvent()` 函数更改如下：

```
void MainWindow::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    //设置一个矩形
    QRectF rect(20, 20, 300, 200);
    //为了更直观地看到字体的位置，我们绘制出这个矩形
    painter.drawRect(rect);
    painter.setPen(QColor(Qt::red));
    //我们这里先让字体水平居中
    painter.drawText(rect, Qt::AlignHCenter, "yafeilinux");
}
```

现在运行程序，效果如下图所示。



可用的对齐方式如下图所示。

- Qt::AlignLeft
- Qt::AlignRight
- Qt::AlignHCenter
- Qt::AlignJustify
- Qt::AlignTop
- Qt::AlignBottom
- Qt::AlignVCenter
- Qt::AlignCenter
- Qt::TextDontClip
- Qt::TextSingleLine
- Qt::TextExpandTabs
- Qt::TextShowMnemonic
- Qt::TextWordWrap
- Qt::TextIncludeTrailingSpaces

### 三、使用字体

为了绘制漂亮的文字，可以使用 `QFont` 类来设置文字字体。大家也可以先在帮助文档中查看该类的介绍。下面将最常用的一些设置进行演示。

在 `paintEvent()` 函数中继续添加如下代码：

```
QFont font("宋体", 15, QFont::Bold, true);
//设置下划线
font.setUnderline(true);
//设置上划线
font.setOverline(true);
//设置字母大小写
font.setCapitalization(QFont::SmallCaps);
//设置字符间的间距
font.setLetterSpacing(QFont::AbsoluteSpacing, 10);
//使用字体
painter.setFont(font);
painter.setPen(Qt::green);
painter.drawText(120, 80, tr("yafeilinux"));
painter.translate(50, 50);
painter.rotate(90);
painter.drawText(0, 0, tr("helloqt"));
```

这里创建了 `QFont` 字体对象，使用的构造函数为 `QFont::QFont ( const QString & family, int pointSize = -1, int weight = -1, bool italic = false )`，第一个参数设置字体的 `family` 属性，这里使用的字体族为宋体，可以使用 `QFontDatabase` 类来获取所支持的

所有字体；第二个参数是点大小，默认大小为12；第三个参数为 `weight` 属性，这里使用了粗体；最后一个属性设置是否使用斜体。然后我们又使用了其他几个函数来设置字体的格式，最后调用 `setFont()` 函数来使用该字体，并使用 `drawText()` 函数的另一种重载形式在点 `(120, 80)` 绘制了文字。后面又将坐标系统平移并旋转，然后再次绘制了文字。运行程序，效果如下图所示。

