# 直线图元 QGraphicsLineItem

# 矩形图元 QGraphicsRectItem

# 椭圆图元 QGraphicsEllipseItem

# 图片图元 QGraphicsPixmapItem

# 文本图元 QGraphicsTextItem

# 路径图元 QGraphicsPathItem

1. 实例化一个QGraphicsScene场景，

setSceneRect(x, y, w, h)方法来设置**场景坐标原点和大小**

1. boundingRect() 定义了一个item的外部边缘的形状为矩形。（非item的形状）

shape() 定义了item形状

line() 返回一条直线的位置

pen() 画笔

line().p1() 直线起始位置

line().p2() 直线结束位置

QSizeF 定义了二维平面上的物体

1. QGraphicsLineItem setline(x1,y1 x2,y2 ) or setLine (QLineF())

Sets the item's line to be the line between (x1, y1) and (x2, y2).

1. QRectF::normalized()

标准化，返回一个没有负数值得长度和宽度

1. qt中坐标系 右为x正轴。 下为y正轴

形状

低可信度描述已自动生成

图形视图提供了三种坐标系之间相互转换的函数，

以及图元和图元之间的转换函数

表格

描述已自动生成

图片包含 日程表

描述已自动生成

QGraphicsItem

是QGraphicScene中所有item的基类

它提供了一个轻量级的基础，用于编写自定义item：

定义item的几何形状，碰撞检测，绘制实现，以及通过其时间处理程序进行item的交互。

QGraphicsItem是Qt图形识图框架的一部分。

常见的 QGI

文本, 信件

描述已自动生成

QGraphicsLayoutItem：

setGeometry()

boundingRect()

sizeHint()： size缺省时采取的默认大小 Hint，如果没有设置则按项的内容计算项大小，确保项的内容在对应表头方向完整显示

LineNumberfield:

我们在codeeditor(LineNumber) 中保留一个单独的小部件 并在其上绘制行号。

QPlainTextEdit 继承自QAbstractScrollArea， 并且 编辑操作 在它的viewport的margins里面。 我们正是在viewpot左边的margin设置一片区域并且设置该margin合适的size来让我们绘制行号。

对于editing code，我们优先选择QPlainTextEdit而不是QTextEdit 因为前者对plain text 做了特别的优化。

QPlainText让我们添加选择可以让用户使用键盘和鼠标进行选择。

我们利用这一功能来对当前行进行高亮显示

LineNumberArea类

我们利用这个widget描绘行号，并且把它放到 viewport的左边缘区域。

我们是利用QPlaintext来描绘这个行号区域，我们还要用到文本区域，所以简单起见，就在文本区域 描绘行号。 并且这个区域还要求 文本去计算 sizeHint

注意，我们完全可以不用定义LineNumberArea 类，而是在文本区直接描绘行号，但是，QWidget类帮助我们 将内容自动设置滚动，并且，为了文本区的可拓展性比如设置断点，和其他代码编辑器的功能，单独设置一个类是一个正确的选择。 该控件之后可以帮助处理鼠标事件。

在文本区中，我们在类LineNumberArea resize并且描绘了lineNumber。 当文本区行数发生变化时 和当文本区viewport 折叠滚动时，我们需要这么做。 并且，文本区窗口尺寸发生变化时也需要这么做。我们通过调用 updateLineNumberWidth() 和 updateLineNumberArea() 方法来实现。

不论何时， 当光标位置发生变化时，我们要高亮当前的行，这部分由 highlightCurrentLine实现。

CodeEditor 类实现：

updateRequest()

当文本区域需要对某个矩形区域进行更新时。如果文本是滚动的，rect会覆盖整个viewport区域。 如果文本是垂直滚动的，dy 会存viewport滚动了多少 pixels数量

这个signal的目的是支持 纯文本编辑器的子类 中其他的widget 例如 显示行号，断点，以及其他额外的信息。

构造函数中，我们将信号和slot连接起来， 很有必要算出行号宽度，并且在编辑器刚被创建的时候高亮第一行。

LineNumberAreaWidth 函数计算了LineNumberArea widget的宽度，我们

更新行号区域的宽度，可以直接调用QAbstractScrollArea::setViewportMargins()方法

resizeEvent()

当编辑器大小改变时，我们同样需要重新调整行号区域的大小

当光标位置改变，我们高亮当前行，比如，包含了光标的行

QPlainTextEdit 提供了 同时多个选择的 可能性，我们可以设置我们选中的字符格式通过QTextCharFormat 。 我们清除光标选择 在设置新的 QPlainTextEdit::ExtraSelection之前

或者当鼠标选择多行时，可以同时高亮

用文本光标进行选择。 当调用 FullWidthSelection属性时，当前光标文本行将会被选择，如果你指向选择 一行文本的一部分，你应该用QTextCursor. movePositon() 进行移动到另外一个位置，这个位置是通过setPosition方法设置的。

LineNumberAreaPaintEvent 是被LineNumberArea调用，一旦它收到 paint event

我们先循环所有可见的行，并且 描绘出行号在每行多余的区域。注意，每一行的纯文本都由一个QTextBlock组成，但是，如果，开启了换行line wrapping，那么一行可能横跨好几行

在循环迭代中，我们先获取第一行的 顶端和顶端和底部的 y轴坐标，并且根据当前行的高度调整这些值 。

注意，我们检查是否 当前行是 visible 初次之外，还要检查是否处于当前的viewport里，

比如 一行可能被编辑器里的某个窗口遮挡隐藏住。

绘图事件是重新绘制全部 或 部分小部件的请求。引起原因：

repaint 或 update被调用。 小部件被遮盖了，然后现在以及被发现了等等

Qt还试图通过将多个绘画事件合并为一个来加速绘画。当多次调用update()或窗口系统发送多个绘画事件时，Qt会将这些事件合并为一个具有较大区域的事件。 repaint()函数不允许这种优化，所以我们建议尽可能使用update()。

Qwidget :: isvisible()

scroll()

将 widget和它所属的x y坐标轴 放入scroll当中

scroll后，widgets会接收 paint event 来repaint 那些需要paint的area

viewport() ： 滚动条 的可使部分窗口内容

update() 更新控件

sizeHint

blockBoundingGeometry return type QRectF

返回 text block 的内容坐标系中的矩形边界， 可以用contentOffset() 方法将矩形转换成viewport上面的坐标

contentOffset ： 返回内容在viewport上面的原始坐标

始终是第一个可视化内容行的坐上角的坐标。

QPlainTextEdit 的文字内容以 QTextDocument 类型存储，函数 document() 返回这个文档对象的指针。QTextDocument 是内存中的文本对象，以文本块的方式存储，一个文本块就是一个段落，每个段落以回车符结束。 blockCount() : 获得文本块个数。

总结：

软件设计并不简单，看似一个简单的功能，后面可能涉及的原理很深，实现起来相当的复杂。

尽量设计时要做模块化设计，这样利于拓展和维护。

学习的途径： 官方文档， source code ，print

ppt思路：

开头： 列出几张 优秀的软件 可视化的方式。