字体的规格和精确的文本位置

字体规格如下图所示：

em

要真正理解如何测量文本，还需要介绍另一个规格——em（或em高度）。与我们所想的相反，em高度不是M字符的高度，而是一种度量单位，定义为字体的点数。因此，如果使用12点的字体，em单位的长度就是12点。

em也用于定义em方块的高度和宽度。em方块这个术语用于印刷业，描述印刷工人设计字体的字样时使用的网格。em方块是一个方块网格，其设计单位是1em乘以1em。设计单位与任何真实的度量单位都没有内在的关系；在TrueType字体中，1em中一般有2048个设计单位。印刷工人使用许多抽象的方式在网格中布置字形，描述组成字样图形的圆弧和线条。

有时设计单位也称为字体单位、em单位或网格单位。在.NET中，它们称为设计单位。

在.NET中，在计算字体的规格时，需要使用em高度在设计单位和象素之间转换，如下面所述。

设计单位和分辨率

在GDI+（不是GDI）中，文本与分辨率是相互独立的。这就是说，用GDI+建立的窗体和窗口在所有的分辨率下看起来都是一样的。设计单位是实现这一点的关键。

有时在建立自定义控件时，让GDI+控制文本的位置就可以了。在这些情况下，仅提供了文本的边框矩形，由GDI+确定文本在该矩形中的位置。但是，在其它情况下，我们还可以精确的控制文本的位置。例如，可以让文本的基线与控制的其它一些部分对齐，或者使文本相对于一条水平线精确的垂直居中。在这两种情况下，都需要字体的详细规格。

Font Family Name       Ascent    Descent  EmHeight      Line Spacing

Courier New        1705      615        2048             2320

Arial                     1854      434        2048             2355

Times New Roman      1825      443        2048             2355

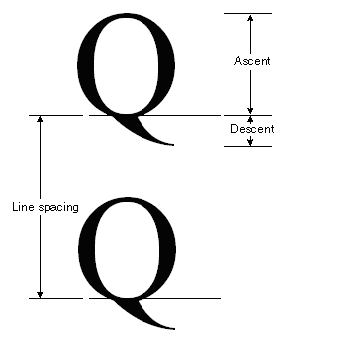
上面的表中检索出来的值用DesignUnits作为单位。可以看出，这3个TrueType字体：

都是使用2048设计单位表示1em

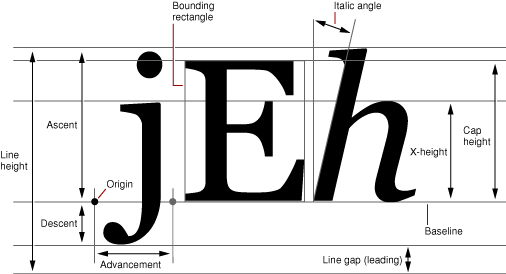
ascent大约3/4em（但每个略有不同）

descent大约1/4em（但每个略有不同）

行间距略大于1em（但每个略有不同）



From MS.



From Apple.

把字体规格转换为物理度量单位

刚才检索出来的值只取决于字体系列，与字体无关。例如，如果以设计单位作为单位，则12点Arial的字体规格与26点Arial相同。这并没有什么可奇怪的，因为在同一字体系列中，大小不同的实例都来自于同一个设计，所以这些实例都使用相同的字形，该字形是印刷工人用2048单位/em的网格设计出来的。

如果要使用这些值对文本的位置进行精确的控制，就需要通过绘制文本所用的字体系列实例的em值，把它们从DesignUnits转换为绘图表面当前的GraphicsUnit。这将把字体规格的相对尺寸（上面给定的）映射为绘图表面的实际测量尺寸。

在构建字体时，要把所构建字体的em值作为一个参数传送给Font构造函数，如下所示：

Font f = new Font(“Times New Roman”, emSize);

在默认情况下，所需的em值假定使用GraphicsUnit.Point作为单位，所以，如果需要12点的Arial字体，就可以使用下面的代码：

Font f = new Font(“Arial”, 12);

这就给出了Arial字体系列的一个实例，其em值为12点。现在不一定要使用Point作为度量单位来指定字体的em值，而可以使用任何可用的GraphicsUnit成员来构建字体的实例。而且，实例化字体后，就可以使用Font.Size属性获取字体的em值，其单位是GraphicsUnit。

现在，我们知道字体的em值可以用GraphicsUnit来衡量，也可以用DesignUnits来衡量，所以使用下面的公式，就可以轻松的把该em值从设计单位转换为GraphicsUnit：

MetricInGraphicsUnits = MetricInDesignUnits × EmSizeInGraphicsUnits / EmSizeInDesignUnits

文本的质量

要指定质量的等级，可以使用Graphics类的TextRenderingHint属性。

ClearType

使用ClearType能得到很好的反锯齿效果。但是会使得速度有所降低。

设置字体的GraphicsUnit

GDI＋的一个主要功能是允许绘制独立于设备的图形。在绘制独立于设备的图形时，最难解决的一个问题是处理各种不同的显式分辨率。我们可以使用的一个工具是Graphics.PageUnit属性。Graphics.PageUnit是绘图表面的一个度量单位——可以把这个度量单位设置为英寸、象素、显示（1/96英寸）、毫米等。

要协调图形的绘制和文本的绘制，一个简便的方法是把绘图表面的度量单位（Graphics.PageUnit）和字体的度量单位（构造函数的一个参数）设置为同一个值。这两个设置的类型都是GraphicsUnit，所以可以使用同一个GraphicsUnit枚举。

http://books.google.com/books?id=UQNmiISffeYC&pg=PA175&lpg=PA175&dq=C%23+Font+Leading&source=bl&ots=IDacxr\_L9x&sig=Opu8oU4vheGT7ZOWG417Bg6jtQg&hl=en&ei=yNFmSoDPJJCPkAXQ1Pj5Ag&sa=X&oi=book\_result&ct=result&resnum=2