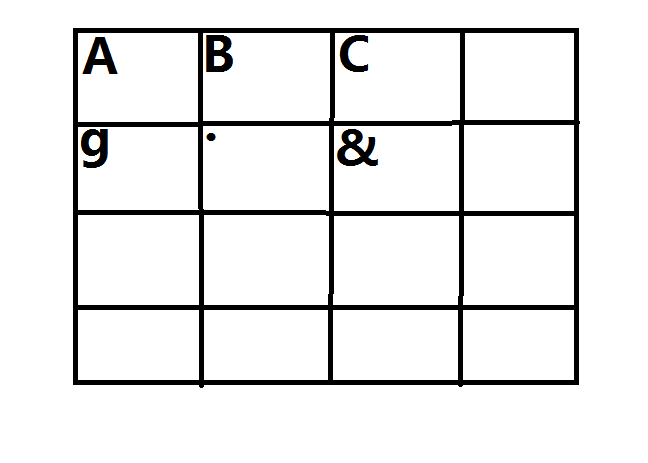
* **NoiseFontManager:**

1. (2015.11.10)因为字体和文字是个很麻烦的东西，所以引入一个fontMgr来专门管理字体的文字。首先核心是外部库FreeType 2.6.1，这个ANSI C写的外部库可以比较方便的获取字体的位图（灰度图），然后再由我来使用。FT在Set\_Pixel\_Size之后确定了字体最大边界大小，然后可以RenderToGlyph来把字体画到一个缓冲区中（缓冲区大小可能会有变化，不同的char/Wchar 大小会不同，中文比较大，Ascii比较小，而且他们之间的大小也有不同）。
2. (2015.11.10)先实现GetBitmapOfChar（bitmap大小可能变化），然后用一定的排版实现GetBitmapOfString（用户确定Width，函数来排版让每个Char完整显示，但是超过Height的部分会被截断）。
3. (2015.11.10)由于GetBitmapOfString真是慢出屎来，所以动态文本暂时只打算支持Ascii，不然每次更新文本都要调用GetBitmap就真是卡死了。静态文本直接就是一个Bitmap Texture；动态文本可以随时修改。在创建某一个字体时，fontMgr会在TextureMgr里面创建一个Ascii Bitmap Table，这个Texture装了Ascii码为0~127的char的bitmap，暂定是8行16列(16x8)，然后每个char用GraphicObject里面的Rectangle来渲染，修改文字的时候只需要修改Rectangle的纹理坐标，这样效率会高很多。在创建这个AsciiBitmapTable时，由于每个char有图像部分的实际Bounding box尺寸可能比字体尺寸要小，所以这个bitmap Table储存了每个Ascii Char的实际图像大小（fontObject&作为参数之一传到mFunction\_GetBitmapOfChar里，charSizeList会被写入尺寸列表）。如图，因为FreeType得到的像素矩阵都是字符最小尺寸(渲染出一个bounding box)，然后我又把这些小图片放到每个char对应格子的左上角，所以图片定位要小心一点。（每个Rectangle都设为字体的大小，而不是char的实际大小）。然后由于这种坑爹属性，还要在Alignment上花点功夫，太坑了



1. (2015.11.10)用FontMgr来创建StaticText或者DynamicTextA，这些装载着Text的子类内置GraphicObj。而且这些GraphicObj
2. (2015.11.19)直接在FontMgr里面内置一个TextureMananger，不要跟用户创建的材质混淆，所以StaticText的文本纹理和DynamicText的字符位图查找表都创建在fontMgr内部的TexMgr。
3. (2015.11.19)Dynamic Text每次更新都要先看看文本char个数和graphicObject里rect的个数，然后调整rect的个数来适应char个数，之后每次都更新纹理坐标（在字符位图查找表的哪一小块位图，即一个char）。还要更新rect的位置，这个要根据具体的字符来调整上下对齐的y坐标偏移。每个rect以左上角作为“光标”，每设置一个rect就移动光标，同一行光标往右跳，到行尾时光标适时跳到下一行。(2015.12.12)**超出y边界的字符会坍缩成一点但不删除（相当于不可见）**
4. (2015.11.20)有一个全局函数用于生成字符纵向对齐偏移(Y offset)，可以供GetStringBitmap和Dynamic text的对齐使用。