文件txt

1.基本阅读功能：翻页，跳转，自动翻行，自动翻页，保留上一页的最后一行

2.背景更换，字体设置，书签

3.编辑功能：双击进入编辑，两种选择文字的方式：传统，插入起点，插入终点

4.自动记录上次阅读的位置

5.浏览方式：单页，双页

6.界面主题

界面设计

1.缩略图

单击打开一个新窗口进行阅读。

缩略图中显示上次阅读位置的上下文，鼠标右击查看文件信息

2.列表

缩略图中显示上次阅读位置的上下文，鼠标右击查看文件信息

构造一个虚拟目录，所有载入的书籍都放在该目录下显示。并且允许设置一个常用文件夹，作为打开文件时的起始目录。

对于预览图，有两种显示方案，一是固定块的大小，块的数量随窗口的变化而增减，这会导致块在小窗口下看着合适但是在大窗口下偏小，而且数量偏多；块的数量不变，块的大小随窗口的变化而增减，这回导致块在大窗口下显示偏大而又偏少。应该二者结合，也就是说在合适的窗口面积下块具有合适的尺寸，但是这个实现比较麻烦，而且不同的人在相同面积的觉得适宜的块的大小是有所差别的。所以还是暂且固定块的大小，也就是说允许用户更改块的大小。

一直横向或竖向拉长会怎么样，由于受到高度/宽度的限制，只能增加数量。

对于是否将块的区域存入块中。由于不是固定的每行多少块，每列多少个块，所以一定得先计算出该页面的块的数量，精确至行和列，否则的话岂不很容易就画到窗口外面去了。由于计算了列数和块数，所以不用将块的区域存入块，直接计算出块的索引。特殊情况，窗口没有排满，单击的是空白如何处理？如果在计算出块之后，访问CBlockManager验证是否存在该块(直接判断是否大于块数就行了)。因此要求块是按顺序排列的，否则怎么根据行坐标和列坐标计算索引。

如果是在滚动视图中，由鼠标位置计算索引是不合适的。块不是规则的排放在窗口中，有的露出一半，有的露出一丁点，这样的话让块记录区域比较好。(??分析可能有问题，有待研究滚动视图，如果是根据画布的坐标来判断，照样可以计算索引的。Windows资源管理器应该是这样做的，在块中判断效率低??)

1.系统内部应该使用Unicode编码，否则的话，如果用ANSI，当读取到Unicode编码时，会有不少字符无法显示。

因此使用C语言函数对文件进行操作，CFile读取之后放进宽字符里了。

前者的过程为：

文件内容是宽字符：直接用

文件内容ANSI字符：wchar\_t提取得到char系统函数转换得到wchar\_t

后者的过程为：

文件内容是宽字符：char提取构造得到wchar\_t

文件内容是ANSI字符：char通过转换函数转换得到wchar\_t

CBlockManager：任务是管理在窗口中显示的块，如果显示的块数量增加，那么链表中的块的数量也得增加。块的显示工作交给对话框，对话框记载阵列和尺寸，同时根据阵列和尺寸判断鼠标是否在块上。或许交给CBlockManager更好看些，绘制块，鼠标在块上。

一个文件一个块还是多个文件公用一个块？且看绘图工作：绘图工作交给对话框。对话框绘制好块之后，通过索引找到块的内容并显示。如果有多页的话，将页数加入计算即可。如果不使用多页的话，用滚动条，那么就不用将页数加入计算，还是那样直接。因此不需要考虑这样的问题，除非块中要包含区域，但是块中包含的区域没有任何使用价值，因为是通过直接计算判断是否在块中的。

块的阵列问题：当窗口大小更改时，是让窗口的Margin变化还是让块之间的Gap更改呢？应该让Gap更改比较好。Margin带来的问题当图片很大时，留下的Margin会很大，看着不怎么舒服，而且计算Margin时，由于给定了最小值nMin，那么会出现nMin+Margin>块的cx/cy的情况，看起来就是分明可以容纳一个新块却不放进去。如果会标识出Margin从而将空白和Margin区分开来就不会产生这种视觉问题，Margin也就可以(不过Gap还是首选)。

存储配置信息，导入配置信息

存储文件信息，导入文件信息

在块中的显示：

1.在块中的显示文件名、预览

2.设计块的样式。底面是纯画刷，不用图片了。

鼠标对块的操作：

1.鼠标在块的上方和选中块

2.鼠标单击块，弹出窗口

文件打开：

1.如果文件已存在，提示是否添加。

2.文件数增加，记录已选择块的数组和记录已打开窗口的数组也得增加。

文本显示：

1.显示文字。读取文件的策略：先读取一段，该段读完了再读取下一段。一个汉字是2个字节，1024也就512个字1KB内存。word中A4纸一页大概1200字。缓存至少得有10页。设置字体，文字间距(暂不支持)，行距。设置文字颜色，窗口背景。

2.管理打开的窗口

设计在窗口中显示的块，以及鼠标经过块的效果。

阅读：

1.分段处理。将文件分成若干个块，记录块的当前块的位置，和当前阅读位置的字符数组索引。如果要进入下一个块，需手动进行载入。当前阅读位置是当前页面的第一行的第一个字符的索引。将块的大小交个用户自定义是没有多大意义的，在无缝阅读下，根本不需要这样的设定，在这样的块式阅读下，只能增加程序对内存的操作。分块好处是拖动滚动条时文件读写少，滚动条拖动也方便。缺点就是不能无缝，内存浪费严重。

如果要使用无缝式阅读的话，显示文档的指针得由自己控制，指针从一个循环队列中读取字符。但是这样的无缝式阅读意味着将整个文件都放在一个窗口中，拖动滚动条就对大文件就不方便了。但是如果使用字节定位的话，不会存在空间的浪费的问题。

记录阅读位置不是用的文件中的位置，而是字符位置。文件位置虽然精确表现当前的浏览进度，但是存和字符位置之间转换的麻烦（因为根据窗口显示的状况获取的位置是基于字符数组的），这个麻烦主要是对ANSI汉字而言的。

2.进度显示条。不能精确计算。

3.块指示条，从Edit发送消息更新Viewer的进度

4.书签。CBlock中增加一个10个元素的CPoint数组， (-1,-1)表示空。暂时不进行实现，书签中的文字预览比较多，得重新设计存储文件。

收尾：

1.将块的内容预存至图像

2.块的移动和删除。移动暂不予以实现，只能提供点击移动效果。

3.设置保存

4.更新预览

文件分块不过是为了减少对磁盘的访问次数和占用过多的内存。一个块最小也得200KB，在这种情况下拖动滚动条能方便到哪儿去。

滚动条带来的问题就是在不分块的情况下，频繁的读取磁盘。也就是说也保持分块，从而避免此问题。但是在分块的情况下，查找操作就不好办了。

最终解决方案：第一使用分块(因此滚动条不是问题)，第二使用滚动条，第三，不提供编辑功能（不能将文件全部载入内存），第四，提供查找功能，第五，使用字符定位（使用字节定位的话，还得转换为字符，并且字符定位同样能够一定的准确度）。不能将所有文件一次性载入，太浪费了，正常阅读的话根本用不上。分块下的无缝阅读，首先就是计算滚动条真实的长度麻烦，然后就是滚动条频繁拖动找出磁盘访问数增加，CPU计算任务也繁重。

Viewer应该重新初始化再利用，而不是销毁。可以一个Viewer给多个对象用。

由于不是字节定位，说以Unicode和ANSI字符计算差了一半(纯中文)

块的右下角位置点击删除菜单不响应。

由于RichEdit的LineIndex获得的字符索引不是相对于数组的，所以Preview出错，等设计好了文本显示类应该可以解决。

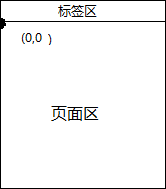
工作继续：

1.预览MouseOver,MouseClick,InterChange,MoveLocation

--------------------------------------------------------------------------------------------

标签页浏览：不采用CMFCTabCtrl，因为自定义比较困难，导致不能融入界面主题中。并不使用CMFCTabCtrl的结构，对话框主窗口中的内容太复杂了很难分离出去。将标签封装为一个窗口，放在上方，主窗口的绘制从窗口下方开始。对话框主窗口被分成了两个部分：标签区和页面区。OnPaint中不对标签区进行绘制。

首先OnPaint中将原界面下移（绘图时是相对于页面区左上角(0,0)画的），其他地方在处理时则需要在竖直方向上减去下移量，坐标就是相对于页面区左上角的了（比如判断鼠标所指方块需要在竖直方向上减掉下移量）。



待续工作：

1.主对话框中的缩略图，需要挪动至CThumbWnd

2.标签需要添加关闭按钮，标签页的切换由标签栏实现，标签栏执行的关闭的操作是删除窗口的记录、隐藏窗口、发送窗口指针给父窗口进行处理。

3.标签、主窗口、阅读窗口的主题将公用，不再分开

4.