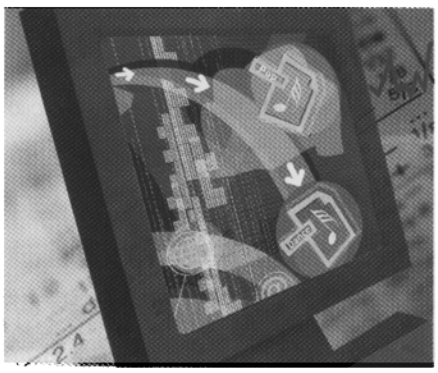
Vega环境下字符显示的实现方法

A Method to Display Character Under Vega



聂孝亮　王国良　马孝尊　汪连栋　(河南洛阳061＃516号　471003)

摘　要：Vega作为一种优秀的视景仿真软件，应用广泛，但对于字符的显示方法却没有详细说明。本文是在对该软件研究的基础上给出了西文的矢量字体、位图字体及基于OpenGL的中西文字符的显示方法，该思想方法也可用于Vega环境下其他仿真特效的实现方法中。

关键词：Vega；视景仿真；位图字体；矢量字体；OpenGL；字符显示

1 Vega简介

视景仿真系统目前在我国也已广泛应用于各种研究领域：军事演练、城市规划仿真系统、大型工程漫游系统、名胜古迹虚拟旅游系统、模拟训练系统和交互式娱乐仿真系统等。现在流行的软件有许多，MultiGen Creator&Vega、VTree、OpenGVS等。

Vega是MultiGen-Paradigm公司最主要的工业软件环境，用于实时视景仿真，虚拟现实和普通视觉应用。Vega将先进的高级仿真功能和易用工具相结合，使用户能够简单迅速地创建、编辑和运行复杂的仿真应用。Vega能显著地提高工作效率，同时大幅度减少源代码开发时间。Vega提供了许多模块，根据需要可选择不同的模块，这些模块使Vega很容易满足特殊仿真要求，例如航海、红外线、雷达、高级照明系统，动画人物，大面积地形数据库管理，CAD数据输入，DIS等等。Vega还包括完整的C语言应用程序接口，为软件开发人员提供最大限度的软件控制和灵活性，是现在比较流行的一种视景仿真软件。

2 Vega基本编程思想

2.1 Vega开发环境

Vega运行既可以运行在Windows环境下，也可以运行在UNIX环境下。在Windows环境下要求是NT内核，且显卡必须支持OpenGL。（本文的内容仅针对Windows环境下进行讨论。）

Vega提供了C语言的接口，要求以VC++5.0以上版本为基础进行二次开发，开发自己的应用软件。图1为Vega的开发环境。

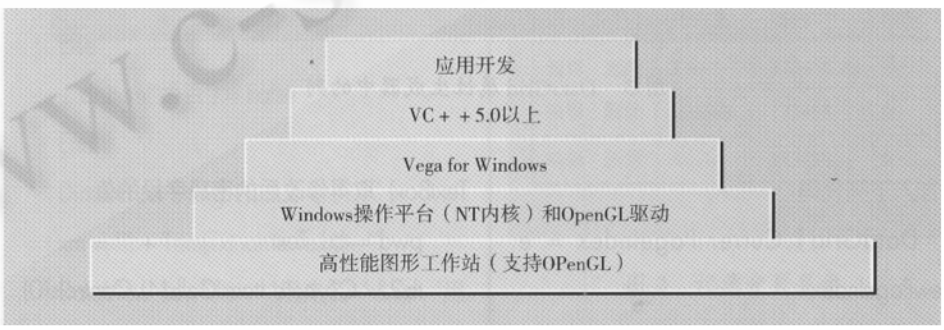


图1　Vega开发环境

2.2 VC程序基本设计方法

程序开发是采用Vega for Windows NT作为软件的支撑环境，用Visual C++ 6.0作为软件开发平台，按以下步骤创建开发自己的程序。在此不对编程方法详细描述，只做简要描述。

（1）创建自己程序的基本框架（单视图）；

（2）增加Vega提供的zsVegaView类到自己的工程；zsVegaView类在Vega安装目录下的\Paradigm\Vega\Sample\Shared下可以找到，它实现了一些基本的功能。

（3）修改自己工程的视图基类为zsVegaView；

（4）修改打开文件的参数为自己需要的参数；

（5）编译运行。

2.3 Vega 中回调函数(Callback Function)的调用方法

Vega作为一优秀的视景仿真软件，它提供了大量的API供开发者进行调用，为软件开发人员提供最大限度的软件控制和灵活性。它设计了大量的回调函数接口，以供开发者根据需要定制自己的操作，不仅有关于对Vega的基本类（Class）操作的回调函数，同时也有对各种事件(Instance)的回调函数。它们的实现方法是相同的，下面以事件的回调函数的实现方法为例进行说明。

2.3.1 回调函数的定义

形式：

```
void MyFunc ( vgCommand *handle, void *my_data_ptr ) ;
```

其定义方法同C语言的方法。在此应该说明的是必须为这种形式，否则参数传递会出现错误。

2.3.2 回调函数的安装与卸载

安装：

Vega中用vgAddFunc(vgCommaon *handle, int which, vgCallBack *func, void *data)实现回调函数的安装，在主循环之前。安装之后，Vega则在需要时自动运行该函数。

卸载：

Vega中用vgDelFunc(vgCommaon *handle, int which, vgCallBack *func, void *data)实现回调函数的卸载。

3 Vega 字符显示方法

Vega作为一优秀的视景仿真软件，对于字符的显示方法给出了两种方法，一种用于显示Vector Font，另一种用于显示Windows的Bitmap Font。但都只能用于有西文字符的显示，对于中文字符的显示则不能显示，且显示方法并没有详细说明，只给出了vgDrawFont(const char *string)与vgXFontDrawString(const char *string)两个函数及其字体设置的相关函数，如果直接使用这两个函数去显示一些字符，则是不行的，需要做一些工作方可。

经分析研究，Vega提供的字体函数是基于OpenGL调用，因此必须在Vega的Draw过程实现，即增加自己的回调函数，在画（Draw）的过程中利用回调函数实现字体的显示。

一般的三维视景仿真界面上，都需要多个通道进行多视角的切换，因此应考虑字符的显示应满足不同通道的需求是最好的。这就要求响应VGCHAN_PREDRAW或VGCHAN_POSTDRAW，这两个函数都可以实现。VGCHAN_PREDRAW表示在对一个通道进行任何画之前响应该函数；VGCHAN_POSTDRAW表示在一个通道画之后响应该函数。

3.1 字符回调函数的实现方法

由于字符的显示是基于OpenGL的，同其他的OpenGL变换之前一样，一般需要先保护当前的显示状态，再进行显示，退出之前，恢复到初始状态，如下为一例子代码。

例子代码：

```
//保存当前状态
GLint zbuf;
pfPushState();
pfPushMatrix();
zbuf = glIsEnabled( GL_DEPTH_TEST );
//设置写字状态，根据你的需要进行设置
glDisable( GL_DEPTH_TEST );
glMatrixMode( GL_PROJECTION );
glLoadIdentity();
gluOrtho2D( 0.0f, 1.0f, 0.0f, 1.0f );
glMatrixMode( GL_MODELVIEW );
glLoadIdentity();
//输出字符，这是本文主要论述的内容
.....
//恢复以前状态
if ( zbuf )
glEnable( GL_DEPTH_TEST );
pfPopMatrix();
pfPopState();
```

3.2 Vector Font 显示方法

Vector Font是矢量字体，该字体随着距视点的远近，其可视的大小也随着变化，即视点距离字体位置近时，则看上去大，反之则小。由于是矢量字体，放大后不平滑。

字符大小用vgFontSize( )进行设置，位置用vgFontPos(float x, float y, float z)进行定义，用vgDrawFont( )在vgFontPos指定的位置显示vgFontSize指定大小的字符。显示的字符在XY平面内。图2中显示的"vgDrawFont TEXT"为一运行结果，其经过OpenGL变换到XZ平面内。其代码为：

```
glRotatef(90, 1, 0, 0);
glLineWidth( 3.0 );
glColor3f(1.0,1.0,1.0);
vgFontSize(0.3, 0.3);
vgFontPos(0.0f, 0.5f, 0.0f);
vgDrawFont("vgDrawFont TEXT");
vgFontPos(0.0f, 1.5f, 0.0f);
vgDrawFont("vgDrawFont TEXT");
```

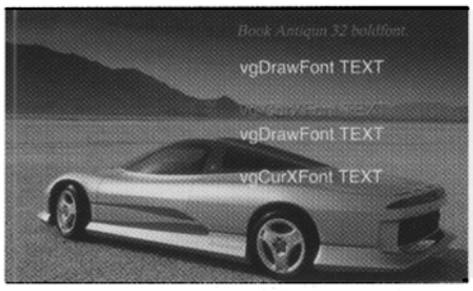


图 2　运行结果一例

3.3 Bitmap Font 显示方法

任何一种Windows中可用的字体都可用作Vega中的Bitmap Font。可设置字符的字体、大小(pixels)、字形(bold、normal、italic)。用vgLoadWinFont()调用字体设置。

为与SGI版本兼容，Vega还提供了调用X

Font的API函数，X Font是XLFD(X Logical Font Description)字符串，字符串包括14个部分，形式如"—FOUNDRY—FAMILY_NAME—WEIGHT_NAME—SLANT—normal—ADD_STYLE—POINT_SIZE—240—75—75—SPACING—120—charsets"，详细可参考相关资料。

定义	取值
FOUNDRY	dt
FAMILY_NAME	Song或Kai
WEIGHT_NAME	medium或bold
SLANT	r或i（斜体）
ADD_STYLE	sans或serif
POINT_SIZE	点阵大小
SPACING	p或m
charsets	ISO8859—1 JISX0208.1983 JISX0201.1976 GB2312—1980.0

该函数显示的字体是在XZ平面内，其大小不随视点的远近而变化，是字体定义的大小，只要在视野范围之内，其大小是不变的，且一直可见。其运行结果如图2中的部分。

```
//设置32大小的字体，XLFD
vgCurXFont(vgLoadXFont( "-dt-application-bold-r-normal-sans-32-75-75-75-m-120-ISO8859-1" ));
glColor3f(1.0,0.0,0.0);
glRasterPos3f(0.0,0.0,1.0);
vgXFontDrawString("vgCurXFont TEXT");
//设置32大小的斜体字，windows的Book Antiqua字体
glRasterPos3f(0.0,0.0,2.1);
vgCurXFont(vgLoadWinFont( "Book Antiqua",32,FW_BOLD,1));
vgXFontDrawString("Book Antiqua 32 bold font.");
```

3.4 汉字显示方法

上述的方法仅仅能实现ASCII字符的显示问题，对于中文字符却不能实现。由于Vega中ASCII字符的显示是通过OpenGL实现的，而中文字符可以用OpenGL进行显示，因此应该可以通过OpenGL功能实现汉字在Vega中的输出。

其显示原理是建立所要显示字符的显示列表，然后调用显示列表画出所要显示的字符。这种显示方法其位置的定义与OpenGL的定义方法一致，其字体的着色、放大、缩小、移动等都须用OpenGL的方法实现。

该方法既可以显示西文字符，也可以显示中文字符，由于wglUseFontOutlines所限，只能显示TrueType字体，不支持笔划字体和光栅字体，但显示的字符比较美观，图3为一运行结果。以下为主要代码。

```
//建列表，生成字符列表
    GLYPHMETRICSFLOAT agmf[128];
    unsigned int ich=0, j=0;
    char COText[128], cch;
    BYTE FTextList[128];
    const GLuint FListBase=1000;
    unsigned int i=0;

    glPushMatrix();
    strcpy(COText, "字1测试");
    while( i<strlen(COText) )
    {
        if (IsDBCSLeadByte(COText[i]))
//判断是否为双字节
        {
            ich=COText[i];
            ich=(ich<<8)+256; //256为汉字内码"偏移量"
            ich=ich+COText[i+1];
            i++;i++;
            wglUseFontOutlines(wglGetCurrentDC(), ich, 1,
                FListBase+j,     //显示列表的基数
                0.0f,    0.0f,
WGL_FONT_POLYGONS,     //指定显示列表线段或多边形
                &agmf[j];     //接受字符的地址
                FTextList[j]=j;
                j++;
        }
        else
        {
            cch=COText[i];
            i++;
            wglUseFontOutlines(wglGetCurrentDC() cch, 1,
                FListBase+j,     //显示列表的基数
                0.0f,    0.0f,
WGL_FONT_POLYGONS,     //指定显示列表线段或多边形
                &agmf[j]     ;//接受字符的地址
                FTextList[j]=j;
                j++;
        }
    }

    glTranslatef(fontstring_x, fontstring_y, 0); //要显示的文本位置
    glScalef(fontstring_size, fontstring_size, 1); //要显示的文本大小
    glListBase(FListBase);
  glCallLists(strlen(COText), GL_UNSIGNED_BYTE ,&(FTextList));
    glPopMatrix();                    //end draw
```

下转第71页 >>

图 3　中文运行结果

4 结束语

对于字符位置的定位问题，比较简单，利用vgGetChanWorldToScreen()与vgGetChanScreenToWorld()两个函数即可实现屏幕坐标与真实场景中坐标的关联，在此不详细介绍，可参见相应的文档。

本文对Vega环境下字符的显示方法进行了详细介绍，由于是基于OpenGl实现的，因此对于其他的需OpenGl实现的功能都可以通过这种方法实现，譬如Vega环境下雪的特效的实现即可以，只不过相对麻烦一些。因此相信本文对Vega环境下视景仿真软件的开发有很大的参考作用。

参考文献

1 Vega Programmer's Guide(Version 3.6)。
2 向世明，图形图像动画专家OpenGL编程与实例，1999年9月。
3 王华等，Visual C++ 6.0编程实例与技巧，机械工业出版社，1999年4月。
4 Common Desktop Environment: Internationalization Programmer's Guide Sun Microsystems, Inc. February 2000。