OpenGL三维图形开发环境设计

莫家庆，刘军营

(广东肇庆市西江大学电脑中心，广东肇庆 526061)

摘　要：本文介绍了在 C++ Builder 3.0 中如何利用 OpenGL 软件包建立三维图形开发环境，并给出相应的程序代码。

关键词：OpenGL；三维图形环境；C++ Builder

OpenGL 是 SGI 公司在其 GL 图形库基础上联合 IBM、微软等大公司共同开发的通用的三维图形库。OpenGL 由于杰出的跨平台性以及简捷、高效，功能完善，而成为三维图形绘制事实上的工业标准。微软自 Windows NT 3.51 和 Windows 95 开始提供对 OpenGL 的支持。在多媒体和网络迅速发展的的今天，OpenGL 的日益流行已成为一种必然。真正的可视化开发工具 C++ Builder 提供的可视化控件(VCL)和各种开发向导使得复杂的应用程序开发变得轻松自如。将两者结合起来，我们可开发出有相当水平的三维图形应用程序。

1　图形环境初始化

OpenGL 的绘图函数和 Wndows 的图形设备接口(GDI)不太一致，在调用 OpenGL 库函数进行绘图之前，需要以一定的步骤进行初始化。为此 Wndows 专门提供了一些 API 函数。

1.1 设置像素格式

OpenGL 的像素格式包含有设备绘图界面的属性，这些属性包括绘图界面是用 RGBA 模式还是颜色索引模式，像素缓存是单缓存还是双缓存，以及颜色位数、深度缓存和模版缓存所用的位数等。为此我们需定义一个 PIXELFORMHTDESCRIPTOR 的结构去描述像素格式，并对其中的一些变量设置相应值，使其满足 OpenGL 颜色模式要求。要完成此过程，我们可先调用 ChoosePixelFormat 函数选择一个最匹配的像素模式，然后再调用 SetPixelFormat 函数将其设为当前像素格式。

完成像素格式设置后，还需要创建特殊的图形描述表(Rendering Context 简写 RC)，用于处理 OpenGL 与 Windows 操作系统相关的各种信息。应用程序对所有 OpenGL 函数的调用都必须使用图形描述表，就象 Windows 编程中用 GDI 作图必须通过设备描述表(DeviceContext 简写 DC)调用相应的函数一样。建立图形描述表后，应用程序就可调用 OpenGL 函数进行绘图、变换等操作。

wglCreateContext 函数用于创建图形描述表。它以设备描述表作为参数，返回一个与设备描述表对应的图形描述表，再以此描述表为参数调用 wglMakeCurrent 函数把图形描述表设为当前描述表，完成 OpenGL 图形环境初始化。

2　程序框架的建立

选择 File 菜单下的 New 命令，C++ Builder 的可视化开发环境就自动为我们建立一个具备 Windows 基本功能的空白窗体。在此基础上作进一步处理就可使其成为 OpenGL 应用程序框架。

2.1 窗体初始化

设置调色板。OpenGL 默认支持的是真彩色，用户的显示卡如果支持 16 位高彩色或 24 位真彩色，就不必设置逻辑调色板。若只支持 256 色，使用下面的函数完成逻辑调色板的映射。

```
void TForm1::SetupLogicalPalette( )
{
 struct {
    WORD Version;
```

```
    WORD NumberOfEntries;
    PALETTEENTRY aEntries[256];
  } logicalPalette={0x300,256};
 BYTE reds[ ]={0,36,72,109,145,182,218,255};
 BYTE greens[ ]={ 0,36,72, 145,182,218,255};
 BYTE blues[ ]={0,85,170,255};
 for(int colorNum=0;colorNum<256;++colorNum)
 {
 logicalPalette.aEntries[colorNum].peRed=red
          s[colorNum&0x07];
 logicalPalette.aEntries[colorNum].peGreen=greens
          [(colorNum>>0x03)&0x07];
 logicalPalette.aEntries[colorNum].peBlues=blues[colorNum].
          peFlags=0;
  }
  m_hPalette=CreatePalette((LOGPALETTE*)&logicalPalette);
}
```

OpenGL 要求 Windows 至少设为 256 色，最好是设为 16 位高彩色或 24 位真彩色。因为在 256 色下，必须重新映射调色板，难免使效果大打折扣。

2.2 图形环境初始化

```
  void __fastcall TForm1::FormCreate(Tobject *Sender)
{
  PIXELFORMATDESCRIPTOR pfd=
  { sizeof(PIXELFORMATDESCRIPTOR),//结构大小
    1,                  //版本号
    PFD_DRAW_TO_WINDOW| //属性标志
    PFD_SUPPORT_OPENGL|
    PFD_DOUBLEBUFFER,
    PED_TYPE_RGBA,
    24,           //24 位颜色
    0,0,0,0,0,0,      //忽略的颜色
    0,0,0,0,0,0,0,0,      //无 alpha 缓存
    32,           //32 位深度缓存
    0,0,          //无 stencil 或 aux 缓存
    PFD_MAIN_PLANE, //主层
    0,          //保留位
    0,0,0          //忽略的层掩码
  };
  HDC hDC=GetDc(this->Handle);
  int pixelformat;
  if((pixelformat=ChoosePixelFormat(hDC,&pfd))==0)
  {
    MessageBox(NULL,"ChoosePixelFormat Fail!","Error",
        MB_OK);
    reture false;
  }
  if((SetPixelFormat(hDC,pixelformat,&pfd)==false)
  {
    MessageBox(NULL,"SetupPixelFormat Fail!","Error",
              MB_OK);
    reture false;
  }
  DescribePixelFormat(hDC,pixelformat,sizeof(pfd),&pfd);
  if(pfd.dwFlags&PFD_NEED_PALETTE)
     SetupLogicalPalette( );
  m_hRC=wglCreateContext(hDC);
  glClearColor(0.8f,0.8f,0.8f,0.0f);
  glClear(GL_COLOR_BUFFER_BIT);
  glShadeModel(GL_SMOOTH);
  ReleaseDC(this->Habdle);
}
```

2.3 OnResize 事件

当窗体大小改变时，需要重新定义视口大小和投影变换矩阵，使窗体内的图形也随之改变。事件内容如下：

```
  void __fastcall TForm1::FormResize(TObject *Sender)
 {
   HDC hDC=GetDC(this->Handle);
   wglMakeCurrent(hDC,m_hRC);
    this ->ClientHeight =(this ->ClientHeight ==0)?1:this ->
ClientHeight;
   glViewport(0,0,this->ClientWidth,this->ClientHeight);
   glMatrixMode(GL_MODELVIEW);
   glLocalIndentify( );
   ReleaseDC(this->Handle);
}
```

2.4 OnPaint 事件

```
 void __fastcall TForm1::FormPaint(TObject *Sender)
 {
   HDC hDC=GetDC(this->Handle);
   if(m_Palette)
   {
   SelectPalette(hDC,m_Palette,false);
   RealizePalette(hDC);
   }
   wglMakeCurrent(hDC,m_hRC);
   DrawWithOpenGL( );
   wglMakeCurrent(hDC,NULL);
   SwapBuffers(hDC);
```

```
  ReleaseDC(this->Handle,hDC);
}
```

上述语句完成了对窗体重画时的响应。其中DrawWithOpenGL ( ) 函数是程序中的实际作图函数,它根据软件的具体任务而编制。SwapBuffers( ) 函数用于在绘图时使用双缓冲。双缓冲分为前台缓冲和后台缓冲,前台缓冲用于显示当前的一帧图形,后台缓冲用于绘制准备显示的下一帧图形。绘制结束后,将两个缓冲交换,后台缓冲的内容显示出来,而原来的前台缓冲则用于绘制下一帧图形。使用双缓冲技术可充分利用 Windows 的多任务能力,加快图形的显示。

2.5 OnDestroy 事件

关闭程序时应该将当前使用的图形描述表清空,并删除所建立的图形描述表。事件内容如下:

```
void __fastcall TForm1::FormDestroy(TObject *Sender)
{
  wglDeleteContext(m_hRC);
  DeleteObject(m_hPalette);
}
```

如果程序只使用一个图形描述表,使用上述方法即可。如果使用了多个图形描述表,在程序中应该建立相应的数据结构以记录所建立的图形描述表,关闭程序就可以根据该结构将其一一删除。

3 结 语

本文给出了在 C++ Builder 中建立 OpenGL 图形开发环境的一种方法。希望本文的探索对那些想使用 OpenGL 进行三维图形开发的读者有所启发。

参考文献

[1] 李智慧等. 跨越 C++ Builder.成都:四川大学出版社,1998.

[2] 雷霖等. Borland C++ Builder 使用与开发指南.北京:人民邮电出版社.

[3] 蔡茂.Windows 95/NT 下 OpenGL 编程原理.计算机世界,1999(32).

(收稿日期:1999-04-30)

(上接第 7 页)

Design & Built Neural Network Evaluation Decision Support System

LIU Xiao

(Economic Information Management Department JiNan University, guangzhou 510632, China)

Abstract: This paper applies three-layer feed-foreword neural network to decision problem as a new evaluation model because of its features and potential advantage ,integrate ORACLE relationship database and high programming language C++ respective advantage to built Neural Network Evaluation Decision Support System (NNEDSS).

Key words: Neural Network; Evaluation; Decision Support System