**opengl的显示列表**

几何图形或其他opengl数据在帧的变化过程中常常保持不变。例如，一个旋转的圆环面总是由同一组三角形带组成，具有相同的顶点数据，每一帧都要通过昂贵的三角函数重新进行计算，而帧与帧之间唯一发生的变换就是模型视图矩阵。解决这种不必要的重复渲染问题的一种方法是从命令缓冲区取出一块预先计算好的数据，他们负责执行一些重复的任务，例如绘制圆环面，这块数据随时可以复制回命令缓冲区，从而节省创建这块数据所需要的大量函数调用和编译开销。这就是显示列表，可以通过glNewList和glEndList定义一个显示列表

glNewList(<unsigned integer name>, GL\_COMPILE);

// 一些opengl代码

glEndList()

GL\_COMPILE告诉opengl编译这个列表但不要执行它，我们也可能执行GL\_COMPILE\_AND\_EXECUTE，创建这个显示列表并执行其中的渲染命令，但是一般而言，显示列表是在程序的初始化阶段创建的（只使用GL\_COMPILE），以后在渲染期间再执行

**显示列表的名称可以是任何无符号整数，但是，如果两次使用了相同的值，第二个显示列表将会覆盖前一个显示列表，opengl对分配唯一的显示列表提供了内置的支持，**

Gluint glGenLists(GLsizei range)

对应的用于释放显示列表名称以及这些显示列表所分配的内存的函数是：

glDeleteLists(GLuint list, GLsizei range)

然后，我们可以用glCallList(GLuint list)命令执行一个包含了任意数量的预编译OpenGL命令的显示列表，我们也可以使用glCallLists(GLsizei n, GLenum type, const GLvoid \*lists)

第一个参数指定了lists数组列表所包含的显示列表的数量，第二个参数表示这个数组的类型，在一般情况下，它是GL\_UNSIGNED\_BYTE。它可以非常方便地用做字体显示列表的地址偏移量

注意：一旦建立了显示列表就不能修改它，因为如果显示列表可以被修改，则显示列表的搜索、内存管理的执行等开销会降低性能

采用显示列表方式绘图一般要比立即模式快，尤其是显示列表方式可以大量地提高网络性鞥，即当通过王阔发出绘图命令时，由于显示列表驻留在服务器中，因而使网络的负担减轻到最小。另外，在单用户的机器上，显示列表同样可以提高效率，因为一旦显示列表被处理成适合于图形硬件的格式，则不同的opengl实现对命令的优化程序也不同。比如glRotate，若将它置于显示列表中，则可大大提高性能，在显示列表中，它值被存储为最终的渲染矩阵。一般来说，显示列表能将许多相邻的矩阵变换结合成单个的矩阵乘法，从而加快速度。

可以在glNewList与glEndList中调用glCallList，但是不允许在两者中间再次嗲用glNewList和glEndList

**普通方式与显示列表性能比较：**

groundList = glGenLists(3);

sphereList = groundList + 1;

torusList = groundList + 2;

glNewList(sphereList, GL\_COMPILE);

gltDrawSphere(10.0f, 40, 20);

glEndList();

glNewList(torusList, GL\_COMPILE);

gltDrawTorus(35, 0.15, 61, 37);

glEndList();

调用方法：

glColor4f(1.0f, 0.0f, 0.0f, 1.0f);

int method = 0;

auto start = chrono::system\_clock::now();

for(int i = 0, size = 10000; i < size; i++)

{

if(method == 0)

{

glCallList(torusList);

}

else

{

gltDrawTorus(35, 0.15, 61, 37);

}

}

auto end = chrono::system\_clock::now();

cout << "elapse time = " << chrono::duration\_cast<chrono::milliseconds>(end - start).count() << endl;

method==0时：

method==1时：