2018-2019年度第二学期 00106501

计算机图形学

童 伟 华　管理科研楼1205室

E-mail: tongwh@ustc.edu.cn

中国科学技术大学　数学科学学院

http://math.ustc.edu.cn/

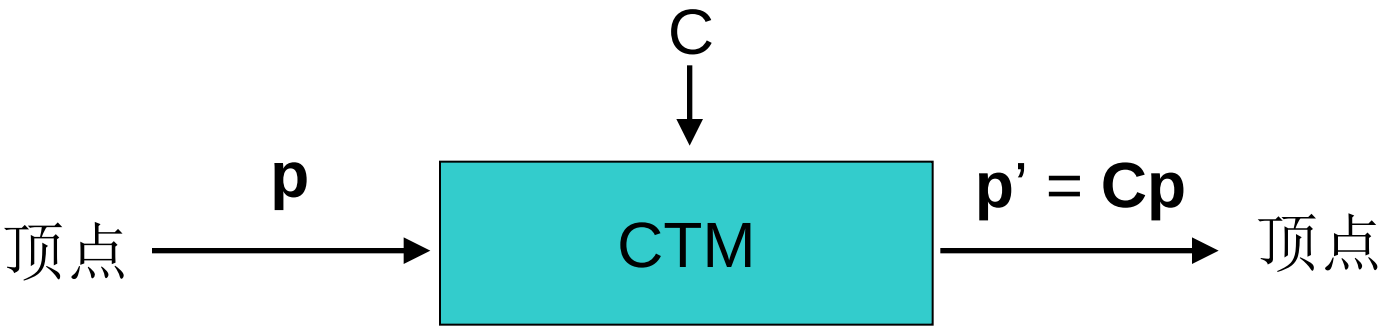
第四节 OpenGL中的变换

OpenGL中的矩阵

- 在OpenGL中矩阵是状态的一部分
- 有三种类型
  - 模型视图 (GL_MODELVIEW)
  - 投影 (GL_PROJECTION)
  - 纹理 (GL_TEXTURE) (不讲)
  - 颜色(GL_COLOR) (不讲)
- 用于操作的单组函数
- 选择所操作的对象
  - glMatrixMode(GL_MODELVIEW);
  - glMatrixMode(GL_PROJECTION);
- 注意：本节内容仅适用于Compatibility profile（与坐标变换相关的函数，在Core profile中都被废弃了，需要自己编写相关的函数或使用第三方数学库，譬如glm库或Eigen库）

当前变换矩阵（CTM）

- 从概念上说，当前变换矩阵（CTM）就是一个4x4阶的齐次坐标矩阵，它是状态的一部分，被应用到经过流水线中的所有顶点
- CTM是在应用程序中定义的，并被上载到变换单元中



CTM运算

- CTM可以被改变，改变的方法是上载一个新的CTM或者右乘一个矩阵

上载单位阵：C ← I
上载任意矩阵：C ← M
上载一个平移矩阵：C ← T
上载一个旋转矩阵：C ← R
上载一个放缩矩阵：C ← S
右乘任意矩阵：C ← CM
右乘一个平移矩阵：C ← CT
右乘一个旋转矩阵：C ← CR
右乘一个放缩矩阵：C ← CS

绕固定点的旋转

- 从单位阵开始：$C \leftarrow I$

  把固定点移到原点：$C \leftarrow CT$

  旋转：$C \leftarrow CR$

  把固定点移回到原处：$C \leftarrow CT^{-1}$

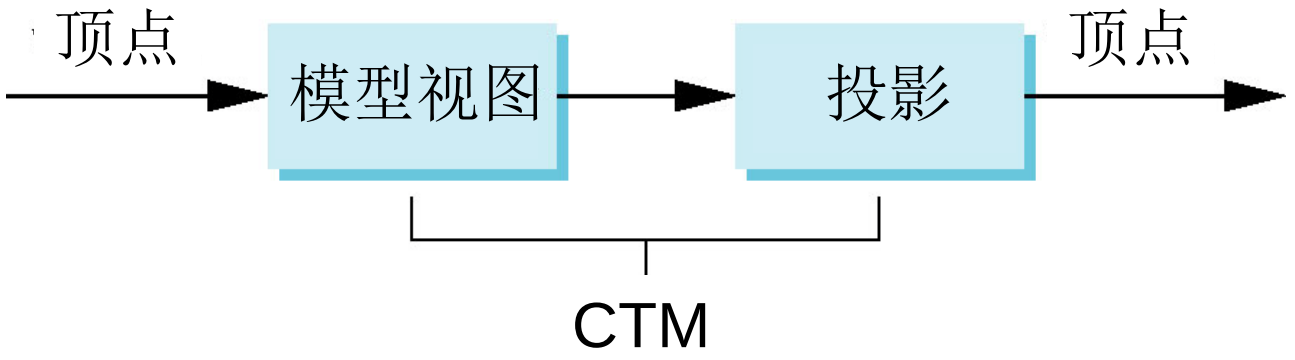
  结果：$C = TRT^{-1}$

  其中每个运算对应于程序中的一个函数调用

  注意：在程序中最后指定的运算是最先被执行的运算

在OpenGL中的CTM

- 在OpenGL的流水线中有一个模型视图矩阵和一个投影矩阵，这两个矩阵复合在一起构成CTM
- 可以通过首先设置正确的矩阵模式处理每个矩阵



旋转、平移、放缩

- 上载单位阵：

```
glLoadIdentity();
```

- 在右边相乘：

```
glRotatef(theta, vx, vy, vz);
```

  - theta以角度为单位，(vx，vy，vz)定义旋转轴

```
glTranslatef(dx, dy, dz);
glScalef(sx, sy, sz);
```

每个函数的参数还可以是d(double)类型

示例

- 固定点为(1.0, 2.0, 3.0), 绕z轴旋转30°

```
glMatrixMode(GL_MODEL_VIEW);
glLoadIdentity(); //此命令不会把投影
                  矩阵重设
glTranslated(1.0, 2.0, 3.0);
glRotated(30.0,0.0,0.0,1.0);
glTranslated(-1.0,-2.0,-3.0);
```

- 记住在程序中最后指定的矩阵是最先被执行的操作

任意矩阵

- 可以上载应用程序中定义的矩阵，或者使之与CTM相乘

```
glLoadMatrixf(m)
glMultMatrixf(m)
```

- 矩阵m是有16个元素的一维数组，其按列定义了4x4矩阵（按列主序方式存储，与C/C++的按行主序方式不同）
- 在glMultMatrixf(m)中m乘在已有矩阵的右边

矩阵堆栈

- 许多情况中需要保存变换矩阵，待稍后再用
  - 遍历层次数据结构
  - 当执行显示列表时避免状态改变
- OpenGL为每种类型的矩阵维持一个堆栈
  - 应用下述函数处理矩阵堆栈（也是由glMatrixMode设置矩阵类型）

```
glPushMatrix()
glPopMatrix()
```

读入后台矩阵

- OpenGL状态中有些信息是以矩阵形式保存的
- 可以利用查询函数读入矩阵（以及其它部分的状态）

```
glGetIntegerv
glGetFloatv
glGetBooleanv
glGetDoublev
```

- 例如，对于CTM：

```
double m[16];
glGetDoublev(GL_MODELVIEW_MATRIX,m);
```

变换的应用

- 例如：应用空闲函数旋转立方体，鼠标函数改变旋转的方向
- 从一个画立方体的程序开始（cube.c）
  - 立方体中心在原点
  - 各方向与坐标轴平行

main()

```
int main(int argc, char **argv){
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|
      GLUT_DEPTH);
  glutInitWindowSize(500, 500);
  glutCreateWindow("colorcube");
  glutReshapeFunc(myReshape);
  glutDisplayFunc(display);
  glutIdleFunc(spinCube);
  glutMouseFunc(mouse);
  glEnable(GL_DEPTH_TEST);
  glutMainLoop();
}
```

空闲与鼠标的回调函数

```
void spinCube(){
   theta[axis] += 2.0;
   if( theta[axis] > 360.0 )
       theta[axis] -= 360.0;
   glutPostRedisplay();
}

void mouse(GLint btn,GLint state,GLint x,GLint y)
{
   if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
       axis = 0;
   if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN)
       axis = 1;
   if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
       axis = 2;
}
```

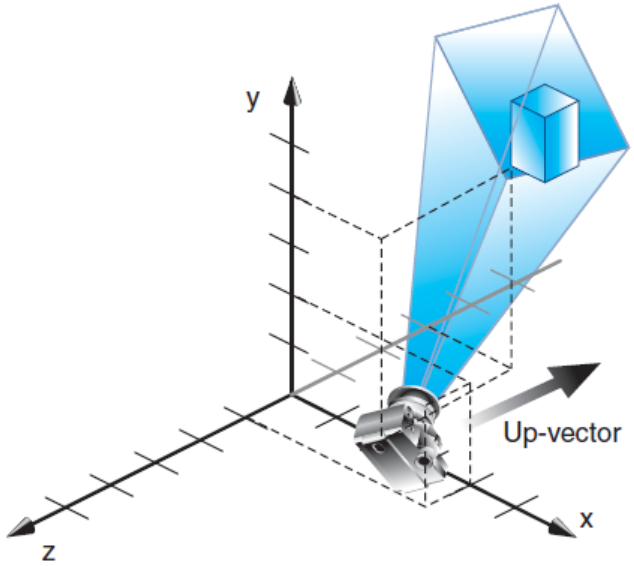
显示回调函数

```
void display(void){
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glRotatef(theta[0],1.0, 0.0, 0.0);
   glRotatef(theta[1],0.0, 1.0, 0.0);
   glRotatef(theta[2],0.0, 0.0, 1.0);
   colorcube();
   glFlush();
   glutSwapBuffers();
}
```

注意：由于回调函数的参数形式是固定的，类似于theta和axis等变量必须定义为全局变换照相机信息在标准的形状改变回调函数中定义

模型视图矩阵的应用

- 在OpenGL中模型视图矩阵可以用来
  - 定位照相机
    - 可以利用旋转和平移，但通常采用gluLookat()会更简单
  - 建立对象模型
- 投影矩阵用来定义视景体以及选择照相机镜头
- 虽然模型视图矩阵和投影矩阵使用同样的函数来进行操作，但需注意累加的改变总是右乘到CTM上



光滑旋转

- 从实际的标准来看，经常需要通过变换来光滑移动和旋转一个对象
  - 问题：给出一个模型视图矩阵列$\mathbf{M}_0, \mathbf{M}_1, \ldots, \mathbf{M}_n$使得当把它们作用到一个或多个对象上时，得到光滑的变换效果
- 在定向对象时，可以利用一个事实，那就是任一旋转对应着球面上的大圆的一部分
  - 求出旋转轴与角度
  - 教材：虚拟跟踪球

旋转累加

- 考虑两种途径
  - 对于一组旋转矩阵$\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_n$，求出每个的Euler角，从而应用$\mathbf{R}_i = \mathbf{R}_{iz}\mathbf{R}_{iy}\mathbf{R}_{ix}$
    - 不是很有效
  - 利用最终的位置确定旋转轴与旋转角度，从而递增旋转角
- 四元数方法比上述两种方法都有效

Hamilton, W. R.

- 1805 - 1865
- 爱尔兰数学家
- 小时学习14种语言，17岁自学数学
- 四元数公式刻在Brougham桥的石头上

四元数

- 把虚数从二维推广到三维
- 需要一个实部和三个虚部i, j, k

$$q = q_0 + q_1 i + q_2 j + q_3 k = (q_0, Q)$$

- 运算性质：
  - $i^2 = j^2 = k^2 = ijk = -1$
  - 加法：对应分量相加
  - 乘法：$pq = (p_0 q_0 - P \cdot Q, q_0 P + p_0 Q + P \times Q)$
  - 乘法单位元(1, 0)
  - 乘法逆元
  - 长度：四分量平方和的平方根

四元数与旋转

- 点：p = (0, Q)
- 定义：r = (cosθ/2, sin θ/2 v)，其中v为一个单位向量
- rpr^{-1}也是一个点，是点Q绕方向v旋转θ角后的位置

$$\mathbf{p}' = \cos^2\frac{\theta}{2}\mathbf{p} + \sin^2\frac{\theta}{2}(\mathbf{p}\times\mathbf{v})\mathbf{v} + 2\sin\frac{\theta}{2}\cos\frac{\theta}{2}(\mathbf{v}\times\mathbf{p}) - \sin\frac{\theta}{2}(\mathbf{v}\times\mathbf{p})\times\mathbf{v}$$

用四元数定义旋转

- 四元数可以表示在球面上的光滑旋转，而且非常有效
- 处理过程为
  - 模型视图矩阵 → 四元数
  - 用四元数进行运算
  - 四元数 → 模型视图矩阵

界面

- 交互计算机图形学的一个主要问题就是如何应用二维的设备（如鼠标）控制三维的对象
  - 例如：如何构造一个实例矩阵？
- 替代方式
  - 虚拟跟踪球
  - 三维输入设备，如：空间球（spaceball）
  - 应用屏幕区域：根据不同的鼠标按钮状态，利用到中心点的距离控制角度、位置、放缩

Thanks for your attention!