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/

第五节 场景图

直接模型图形的局限

- 当在应用程序中定义一个几何对象时，在代码被执行后，对象就进入流水线过程
- 然后它就会从图形系统中消失了
- 为了重新绘制同样的或者改变了的对象，那么就需要重新执行代码
- 显示列表功能或缓冲区对象只是对这个问题进行了部分解决

OpenGL与对象

- OpenGL缺少面向对象的功能
- 例如，考虑一个绿球
  - 可以用多边形建立它的模型，也可以用OpenGL提供的二次曲面功能建模
  - 它的颜色是由OpenGL状态确定的，这不是对象的一种属性
- 这不符合物理对象的观念
- 可以利用面向对象语言和技术建立起更好的对象代码

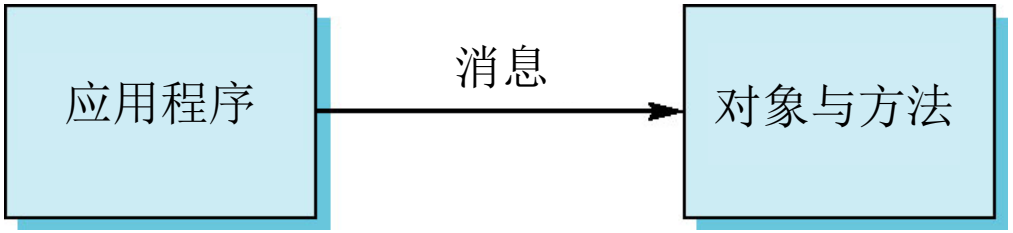
强制程序模式

- 例：旋转立方体

- 旋转函数必须知道立方体的表示方式
  - 顶点列表
  - 边表

面向对象的程序模式

- 在这种模式中，表示是与对象存贮在一起的



- 应用程序发送消息给对象
- 对象中包含函数（也称为方法）可以自己变换对象

C/C++

- 可以利用C的struct类型建立对象
- C++提供了更好的功能支持
  - 可以应用class结构
  - 可以利用类中的public, private和protected对实现进行必要的隐藏
  - 也可以利用友元标识使得类可以彼此访问

立方体对象

- 假设我们要创建一个简单的立方体对象，可以对它进行放缩、定向、定位，并直接利用代码设置它的颜色，例如

```
cube mycube;
mycube.color[0] = 1.0;
mycube.color[1] = mycube.color[2] = 0.0;
mycube.matrix[0][0]=……
```

立方体对象的函数

- 我们也希望具有作用在立方体上面的函数，例如：
  - `mycube.translate(1.0, 0.0, 0.0);`
  - `mycube.rotate(theta, 1.0,0.0,0.0);`
  - `setcolor(mycube,1.0,0.0,0.0);`
- 也有方法显示立方体
  - `mycube.render();`

建立立方体对象

```
class cube {
  public:
     float color[3];
     float matrix[4][4];
     // public 方法

  private:
     // 实现
}
```

实现

- 可以在private部分进行任何实现，例如应用顶点列表
- private部分可以访问public成员，类方法的实现可以应用任何实现，而不需要把它们变为可见的
- 显示方法需要一些技巧，但它会调用标准OpenGL的绘图函数，例如glVertex

其它对象

- 其它对象具有几何特征
  - 照相机
  - 光源
- 但我们也应当能够包含非几何对象
  - 材料
  - 颜色
  - 变换（矩阵）

应用程序代码

```
cube mycube;
material plastic;
mycube.setMaterial(plastic);

camera frontView;
frontView.position(x,y,z);
```

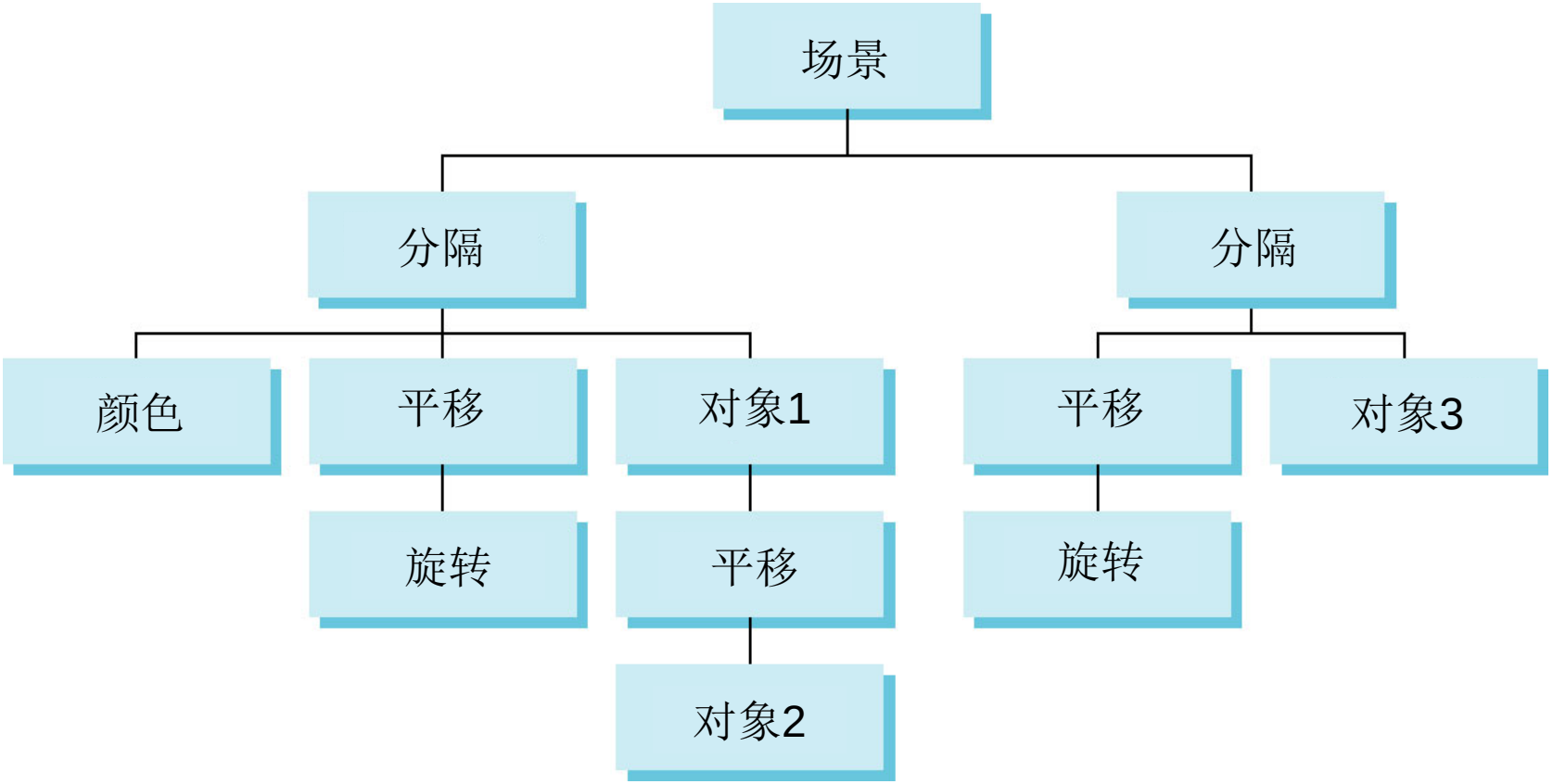
光源对象

```
class light { //与Phong模型匹配
   public:
       bool type; //正交或透视
       bool near;
       float position[3];
       float orientation[3];
       float specular[3];
       float diffuse[3];
       float ambient[3];
}
```

场景描述

- 重新考虑示意图模型，可见
  - 可以用树结构或者等价的代码描述模型
  - 可以编写出一般的遍历代码从而用于显示
- 如果可以用C++对象表示场景中所有的成员（照相机、光源、材料、几何体），我们应当能够在一个树结构中列出它们
  - 从而可以应用遍历算法显示场景

场景图



遍历

```
glPushAttrib
glPushMatrix
glColor
glTranslate
glRotate
对象1
glTranslate
对象2
glPopMatrix
glPopAttrib
……
```

分隔节点

- 需要用它隔离状态改变
  - 等价于OpenGL中的Push/Pop
- 注意，与示意图模型中一样
  - 可以编写一个适用范围广泛的遍历算法
  - 遍历的顺序是相当重要的
    - 如果不应用分隔节点，状态改变会扩散

场景图（Scene graph）

- **场景图：描述图形场景的数据结构**
  - 逻辑结构
  - 空间结构
- **场景图通常用图或树结构表示**
  - 节点（node）
  - 孩子（children）
  - 组（group）
- **场景图主要因素：内存有效性**
  - 实例化（instanced），譬如场景中有许多骑士，这些骑士可以用同一个几何模型表示，只是实例化的参数不同

场景图的国际规范

- PHIGS：第一个商业规范，ANSI standard，1988
- X3D：基于XML的网络图形表示规范，ISO standard，2010
  - 前身为VRML（Virtual Reality Modeling Language）
- 目前，工业界并无统一的场景描述规范，处于一个无序状态
  - 场景文件的转换成为一个难题
  - 目前，大多数场景图API所支持的基本元素与图形系统的功能匹配，因此主要建立在OpenGL或者DirectX之上

场景图的API

- Open Inventor：开源软件库，SGI公司
  - C++面向对象的软件开发库
  - 基于OpenGL
  - 跨平台（支持Unix/Linux、Windows等）
  - 目前已经停止开发
  - http://oss.sgi.com/projects/inventor/

场景图的API

- OpenSceneGraph：开源软件
  - C++面向对象的软件开发库
  - 基于OpenGL
  - 跨平台（支持Unix/Linux、Windows等）
  - 支持CPU与GPU之间的负载平衡
  - 支持遮挡剔除（occlusion culling）
  - 支持层次细节（level of detail）
  - 遍历过程
    - 第一次遍历处理场景更新：响应鼠标操作
    - 第二次遍历处理几何对象：遮挡剔除、半透明绘制、层次细节、包围盒计算等
    - 第三次遍历调用OpenGL进行绘制
  - 目前仍然很活跃，推荐使用
  - http://www.openscenegraph.org/projects/osg

场景图的API

- Java 3D：开源软件，Oracle公司
  - Java面向对象的软件开发库
  - 网络图形编程API
  - 手持设备编程API（譬如手机游戏）
- JOGL：the Java™ Binding for the OpenGL® API

场景图的网络规范X3D

- X3D：基于XML的网络图形表示规范，ISO standard，2010
  - 国际规范，但不是API（注意区别）
  - 支持分布式数据
  - 与OpenGL关系密切

Thanks for your attention!