

Introduction to Computer Graphics



Pipeline Architecture

Ming Zeng
Software School,
Xiamen University

Contact Information:
zengming@xmu.edu.cn

Contents

- The pipeline architecture of the graphics rendering procedures (图形渲染过程的体系结构)
- Application Programming Interface (API) 应用程序编程接口

Image Formation Revisited

(成像模型回顾)

- Can we mimic the synthetic camera model to design graphics hardware and software? (可否模拟虚拟相机)
- Application Programming Interface (API)
 - Need only specify
 - Objects (物体)
 - Materials (材质)
 - Viewer (观察者)
 - Lights (光源)
- But how is the API implemented?

Two tasks of our course

- Learn how to use a specific API (OpenGL)
如何用API来绘制图像

- Learn the implementation details behind the API

这些API又是如何实现的呢？你自己可否写一个这样的API

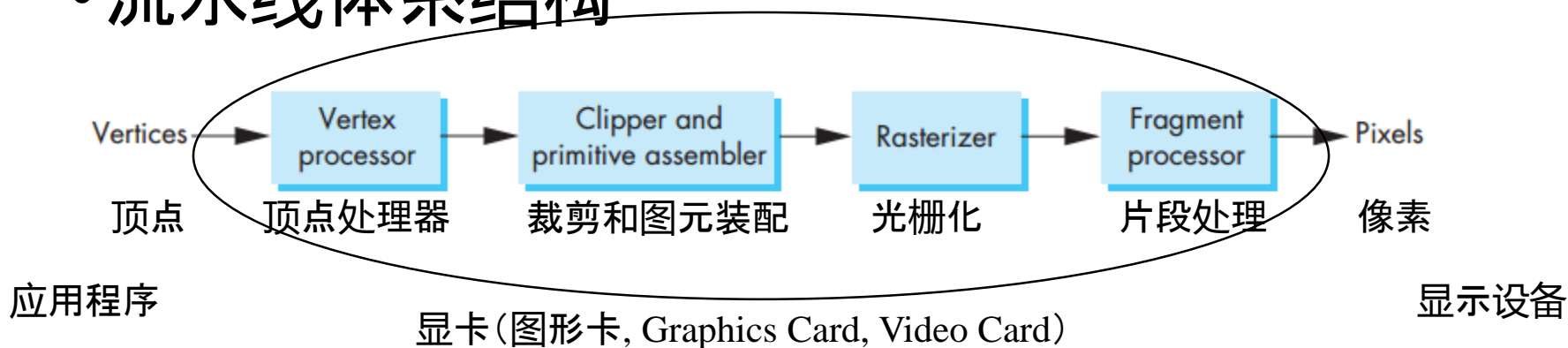
Global Illumination or Local Illumination (全局光照明 还是 局部光照明)

- Global Illumination
 - Ray Tracing 光线跟踪方法
 - Radiosity 辐射度方法
 - 计算量非常大, 目前没有硬件加速支持, 不适合实时或交互式系统
- Local Illumination 局部光照模型
 - Phong 光照模型
 - 计算量适中, 有硬件加速支持
 - 当前主流API: OpenGL, Direct3D

Pipeline architecture

流水线体系

- 按照应用程序定义对象的先后顺序, 依次处理每个对象
 - 只考虑局部光照
- 流水线体系结构

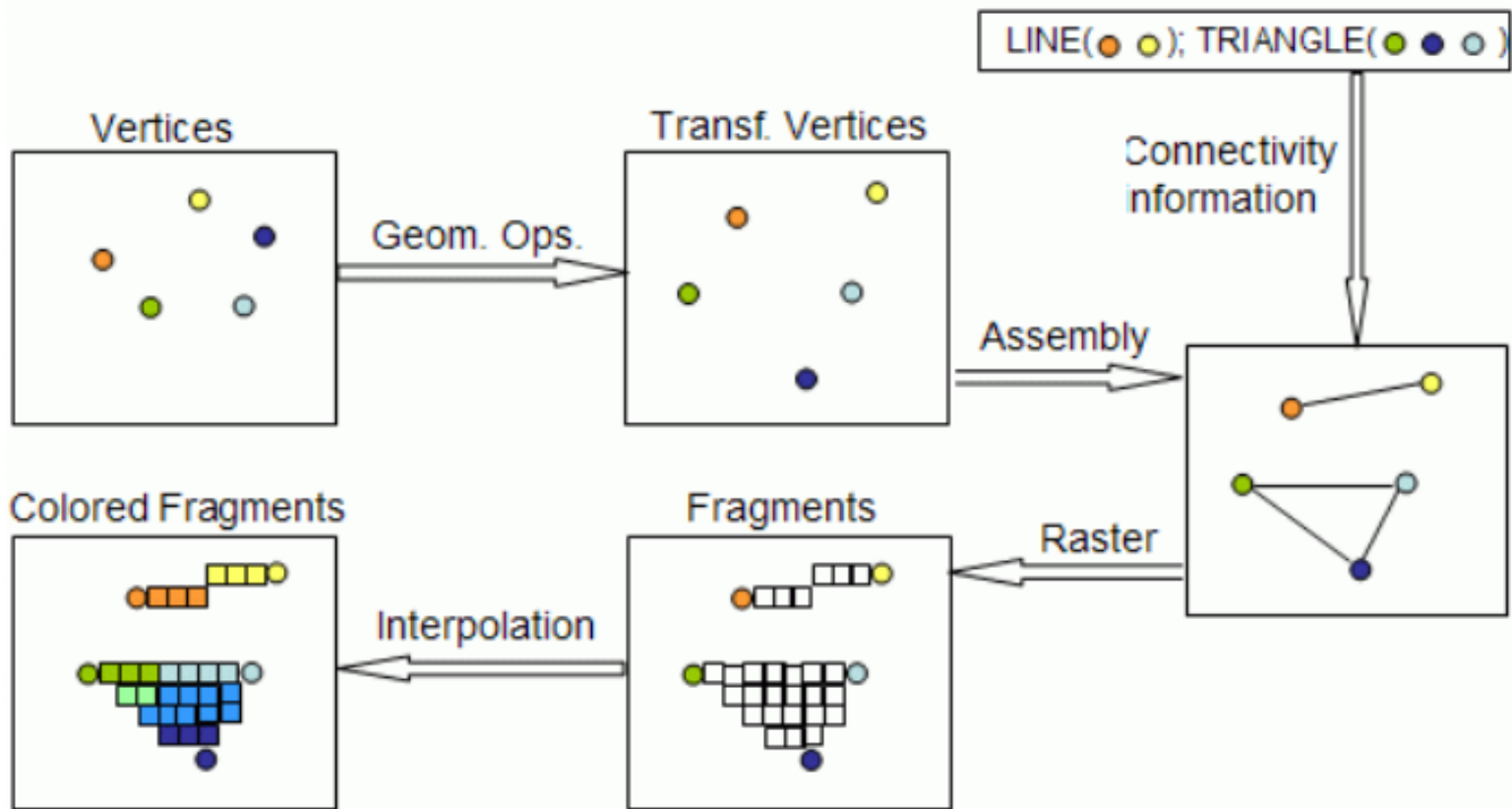


- 所有步骤都可以通过显示卡的硬件实现

Physical Approaches (基于物理的方法)

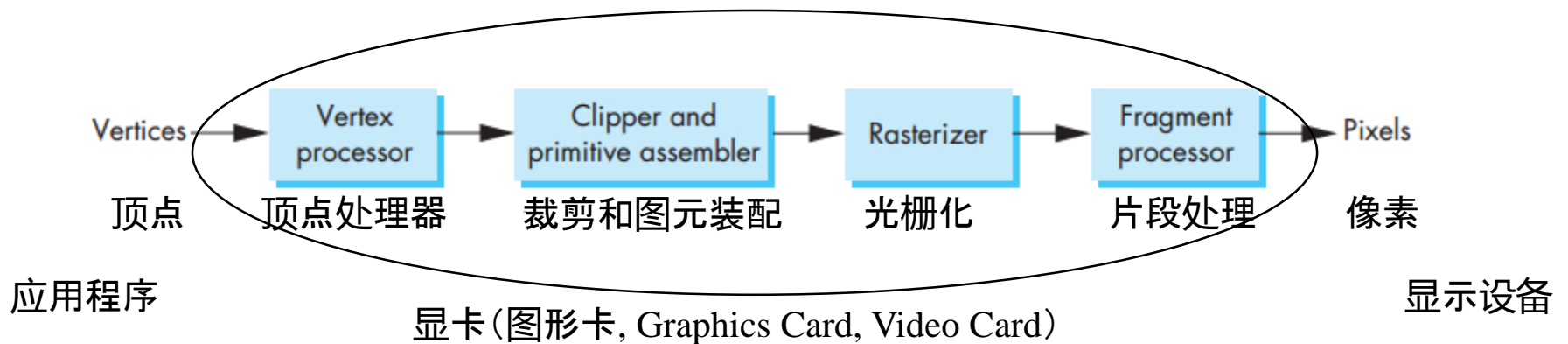
- **Ray tracing:** follow rays of light from center of projection until they either are absorbed by objects or go off to infinity
 - Can handle global effects
 - Multiple reflections
 - Translucent objects
 - Slow
 - Must have whole data base available at all times
- **Radiosity:** Energy based approach
 - Very slow

Graphics Pipeline (图形渲染流水线)



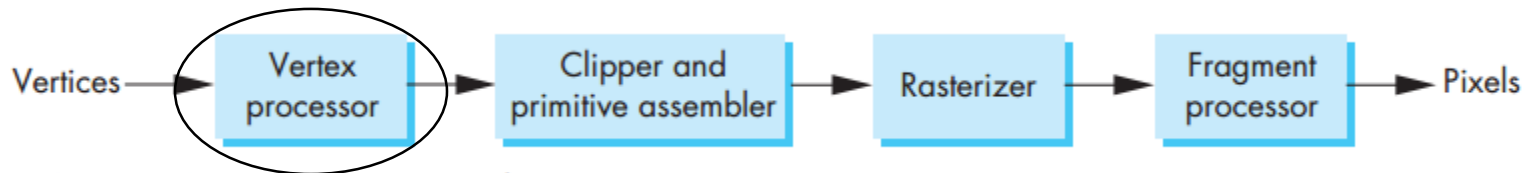
主要步骤

- 四个主要步骤：
 - 顶点处理
 - 裁剪和图元装配
 - 光栅化
 - 片段处理



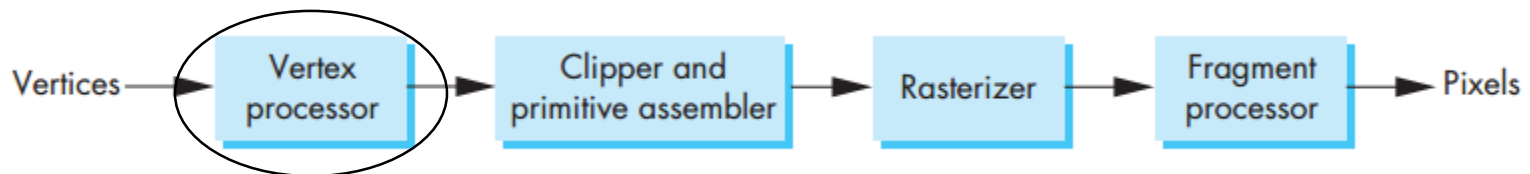
顶点处理

- 图元的类型和顶点集定义了场景的几何
 - 对象由一组图元组成, 而每个图元又包含一组顶点
- 流水线中大部分工作是把对象在一个坐标系中表示转化为另一坐标系中的表示:
 - 世界坐标系
 - 照相机(眼睛)坐标系
 - 屏幕坐标系
- 坐标的每个变换相当于一次矩阵乘法
- 顶点处理器也计算顶点的颜色



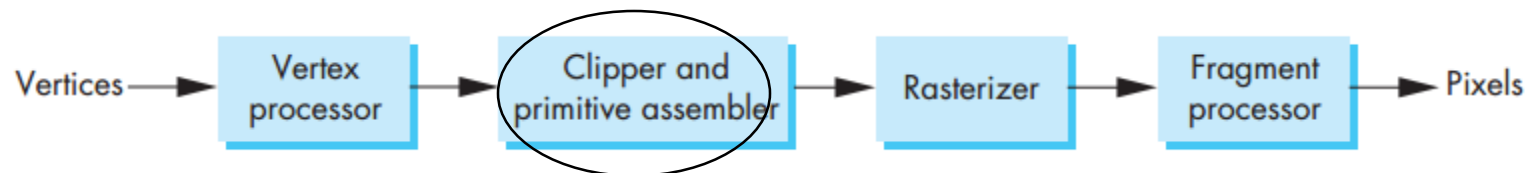
投影(Projection)

- 把三维观察者位置与三维对象结合在一起确定二维图像的构成
 - 透视投影: 所有投影线交于投影中心
 - 平行投影: 投影线平行, 投影中心在无穷远, 用投影方向表示
- 在顶点处理步中, 对各个顶点的处理是相互独立的



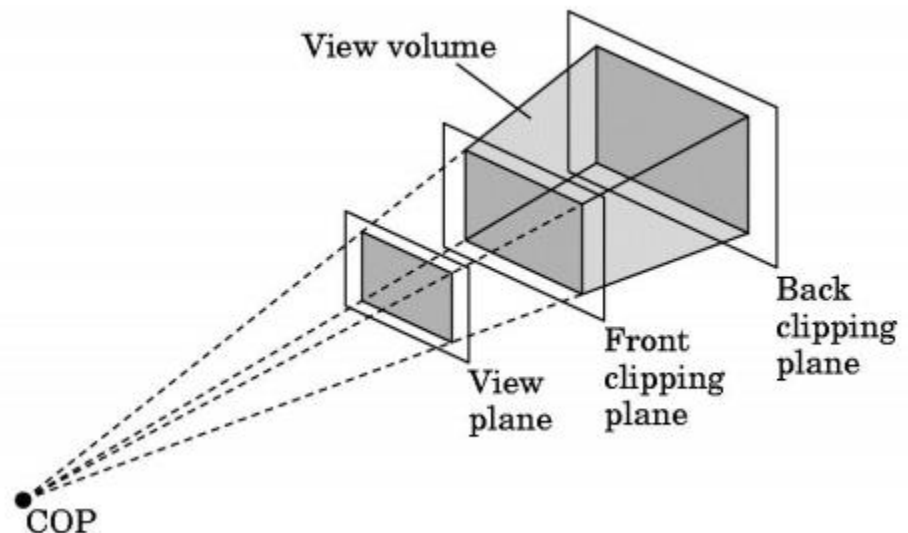
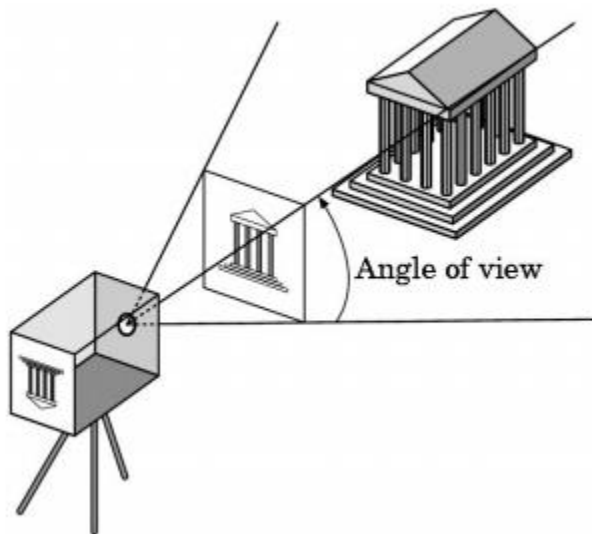
图元装配 (primitive assemble)

- 在进行裁剪和光栅化处理之前, 顶点必须组装成几何对象
 - 线段
 - 多边形
 - 曲线和曲面



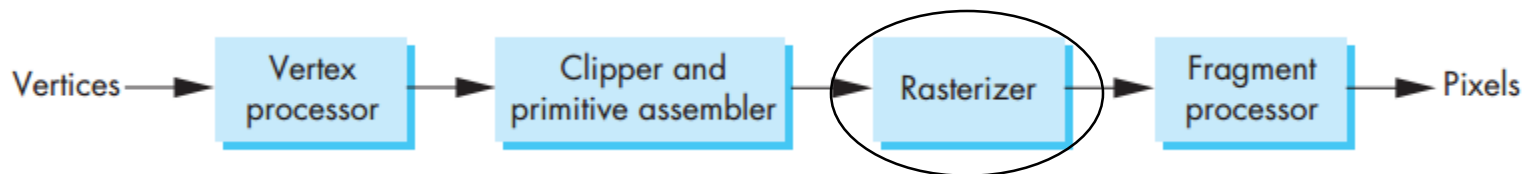
裁剪 (Clipping)

- 真正的照相机不能“看到”整个世界，图形学中的虚拟照相机也只能看到世界的一部分
 - 不在视景体中的对象要从场景中裁剪掉
- 裁剪必须针对逐个图元进行



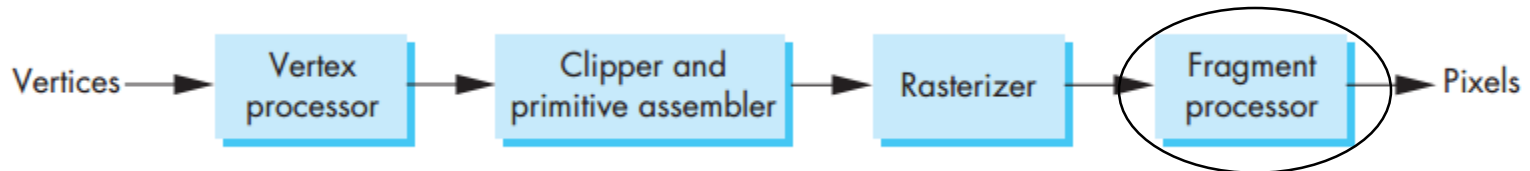
光栅化(Rasterization)

- 如果一个对象不被裁掉，那么在帧缓冲区中相应的像素就必须被赋予颜色
- 光栅化程序为每个图元生成一组片段
- 片段是“潜在的像素”
 - 在帧缓冲区中有一个位置
 - 具有颜色和深度属性
- 光栅化程序在对象上对顶点的属性进行插值得到片段的属性

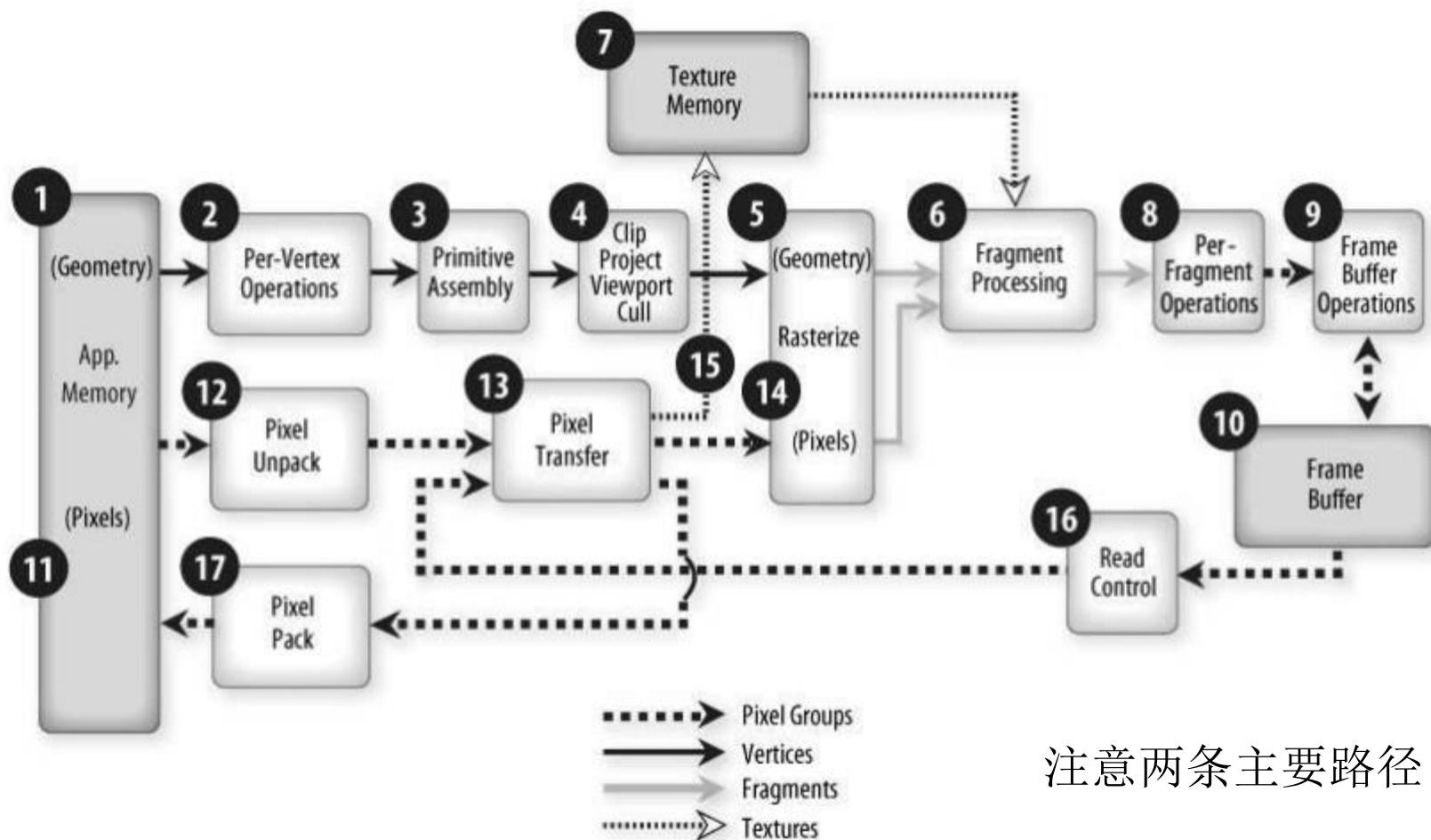


片段处理(Fragment Processing)

- 对片段进行处理，以确定帧缓冲区中相应像素的颜色
- 颜色可以由纹理映射确定，也可以由顶点颜色插值得到
- 片段可能被离照相机更近的其他片段挡住
 - 隐藏面消除

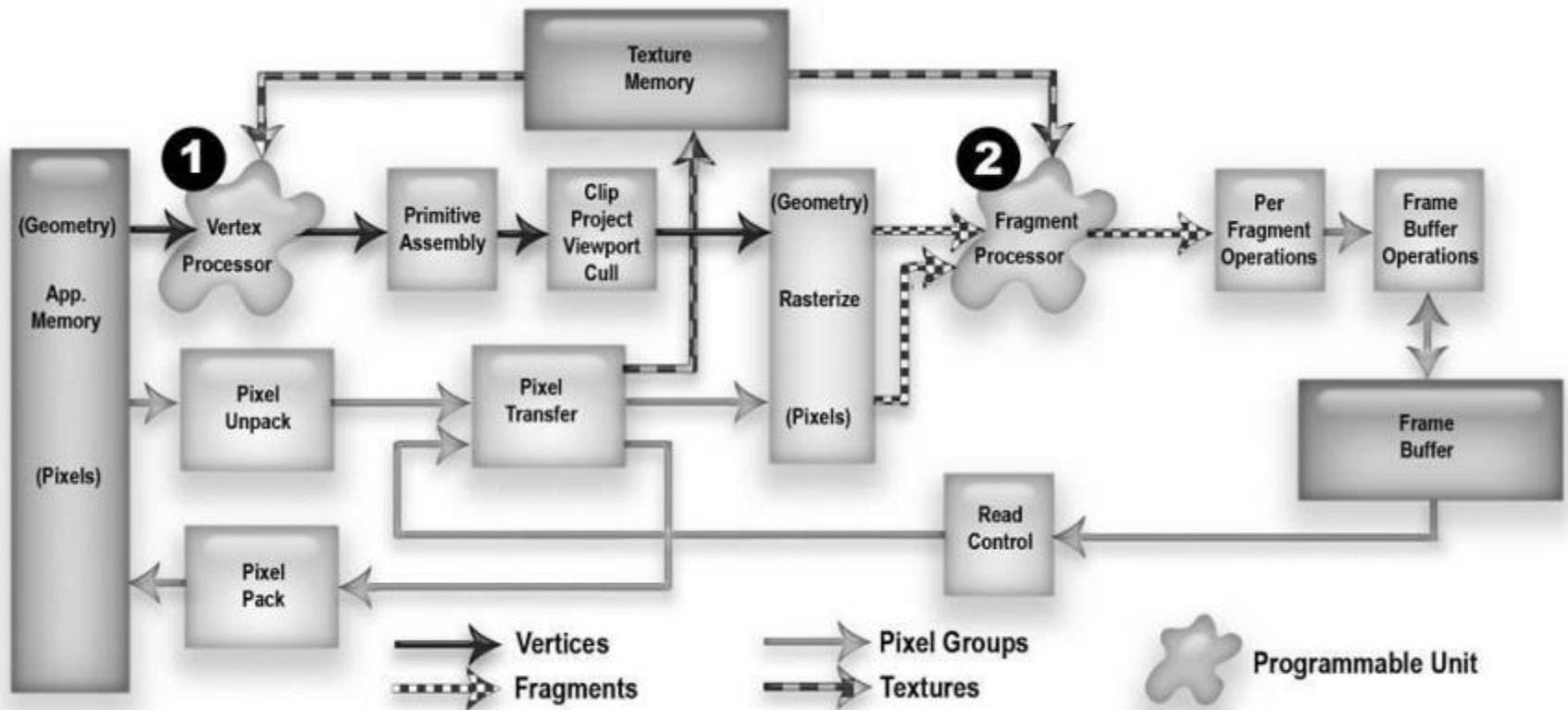


固定功能流水线(fixed function pipeline)



注意两条主要路径

可编程流水线 (Programmable Pipeline)

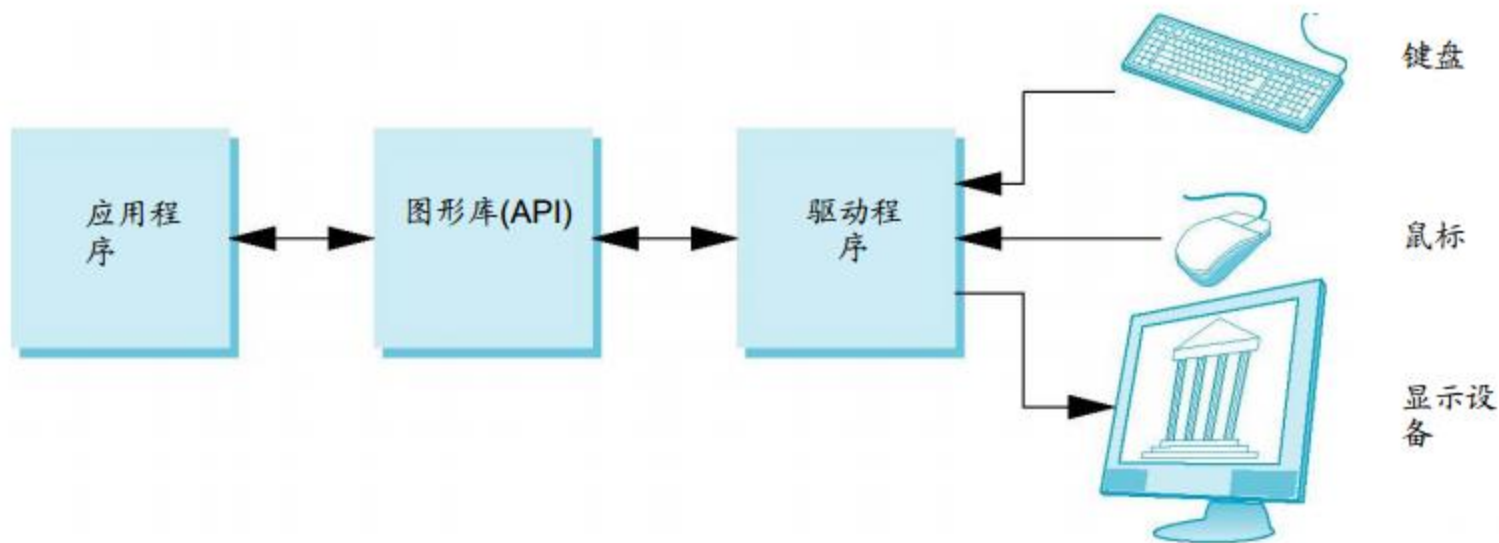


注意两条主要路径

编程接口

(Application Programming Interface, API)

- 应用程序设计人员是通过软件接口接触图形系统, 这个接口就是应用编程接口(API)



三维API的构成

- 函数：定义生成一幅图像所需要的内容
 - 对象
 - 观察者
 - 光源
 - 材料属性
- 其它信息
 - 从鼠标和键盘等设备获取输入系统的能力

对象的定义

- 绝大多数API支持有限的基本几何对象, 例如:
 - 点 points (零维对象)
 - 线段 line segments (一维对象)
 - 多边形 polygons (二维对象)
 - 某些曲线和曲面
 - 二次曲面 quadrics
 - 多项式参数曲面
- 所有基本形状都是通过空间中的位置或顶点 (vertices) 定义的。

例如 (OpenGL)

对象类型

```
glBegin(GL_POLYGON)
```

```
    glVertex3f(0.0, 0.0, 0.0);
```

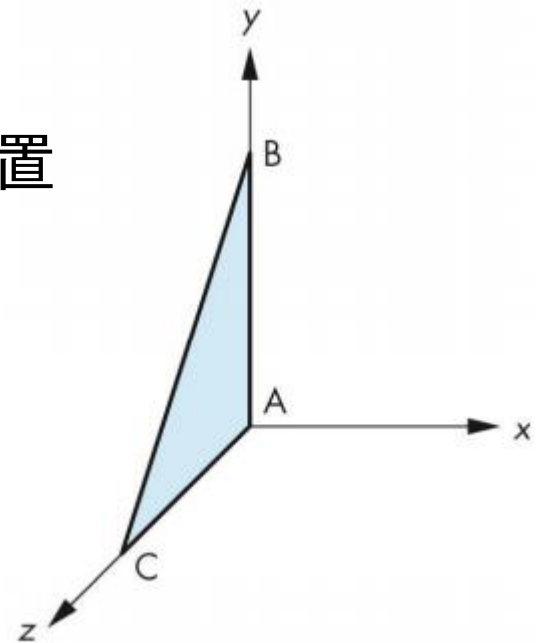
```
    glVertex3f(0.0, 1.0, 0.0);
```

```
    glVertex3f(0.0, 0.0, 1.0);
```

```
glEnd( );
```

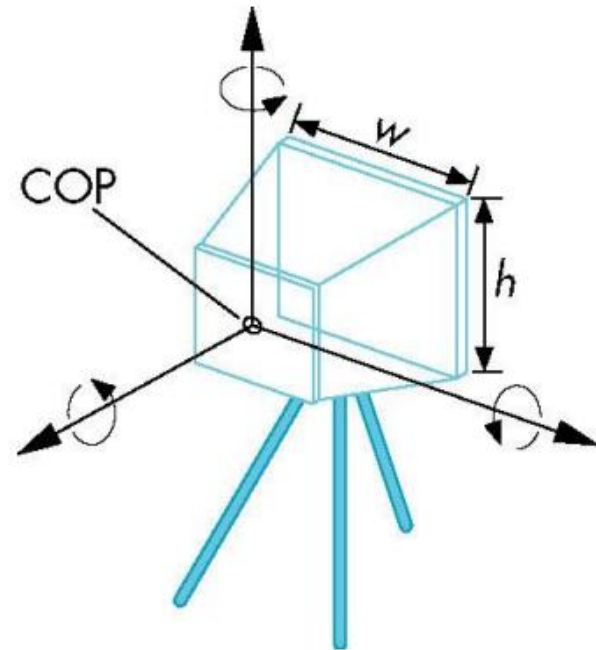
定义结束

顶点位置



照相机的指定

- 六个自由度 (6 degrees of freedom, DOF)
 - 镜头中心的位置, 即投影中心(COP)
 - 方向
- 镜头、焦距
- 胶卷尺寸



光源与材质(light sources and materials)

- 光源类型 (light type)
 - 点光源与分布式光源
 - 聚光灯
 - 远光源与近光源
 - 光源的颜色属性
- 材料属性 (materials)
 - 吸收性: 颜色属性
 - 反射性: 漫反射、镜面

本节小结

- 计算机图形系统的组成
- 虚拟照相机模型
- 图形处理流水线体系结构
- 图形API的构成