Review 知识点

- 输入设备
- 图形的显示设备

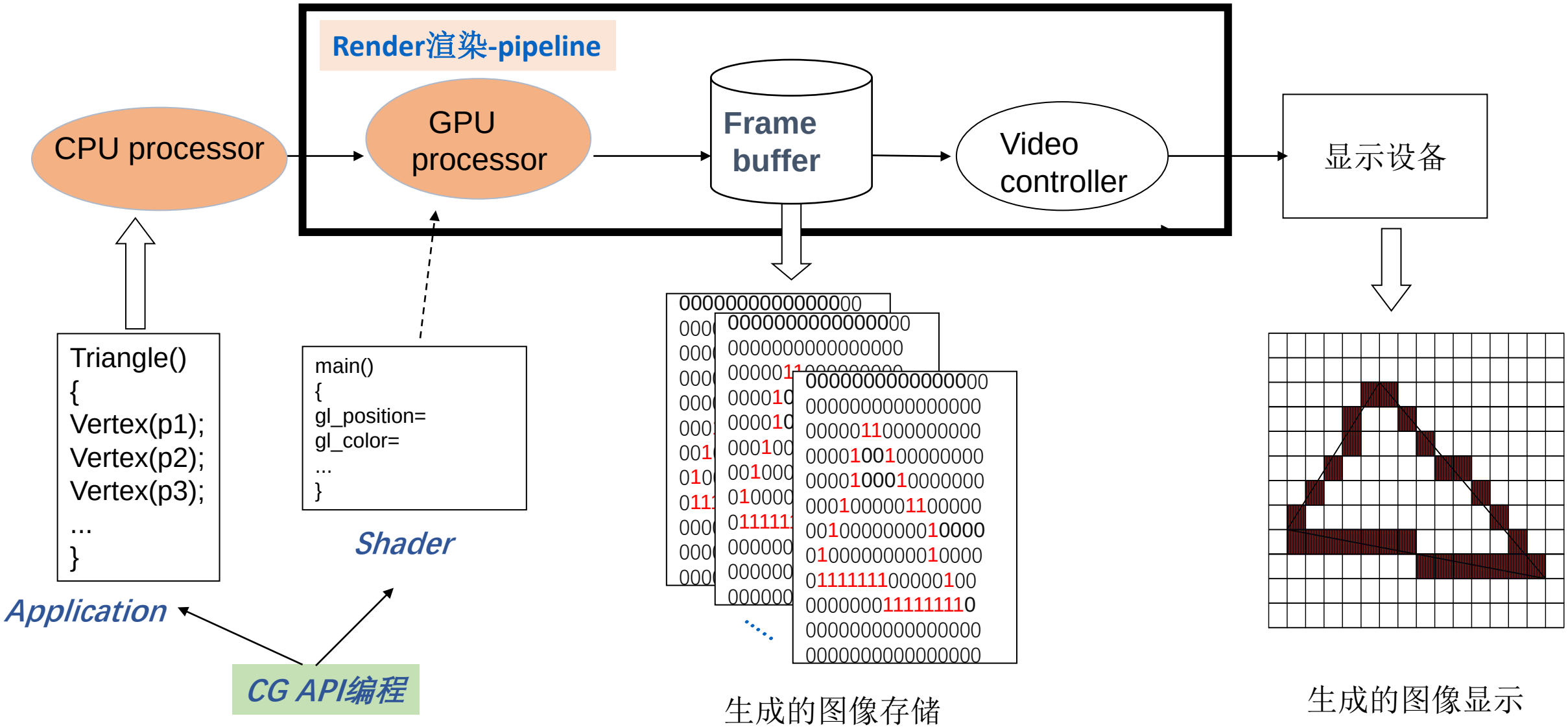
-随机扫描和光栅扫描显示器
  -发光原理，余辉时间，视觉停留时间，刷新频率，
  -光栅，帧，像素，分辨率，纵横比

- 光栅扫描系统组成（显卡主要组成）****

-视频处理器controler：将帧缓存里的像素颜色编码取出用以控制屏幕显示。
-帧缓存framebuffer：存放一帧图像的颜色编码或查找表地址，用以显示。
-显示处理器processor：将图形进行管线并行处理后生成图像数据存入帧缓存。

图形的颜色表示和存储**
  -颜色表示（黑白，灰度，彩色；颜色模型）
  -颜色存储（像素深度/精度，位平面，分辨率）
  -查色表

Render sketch map



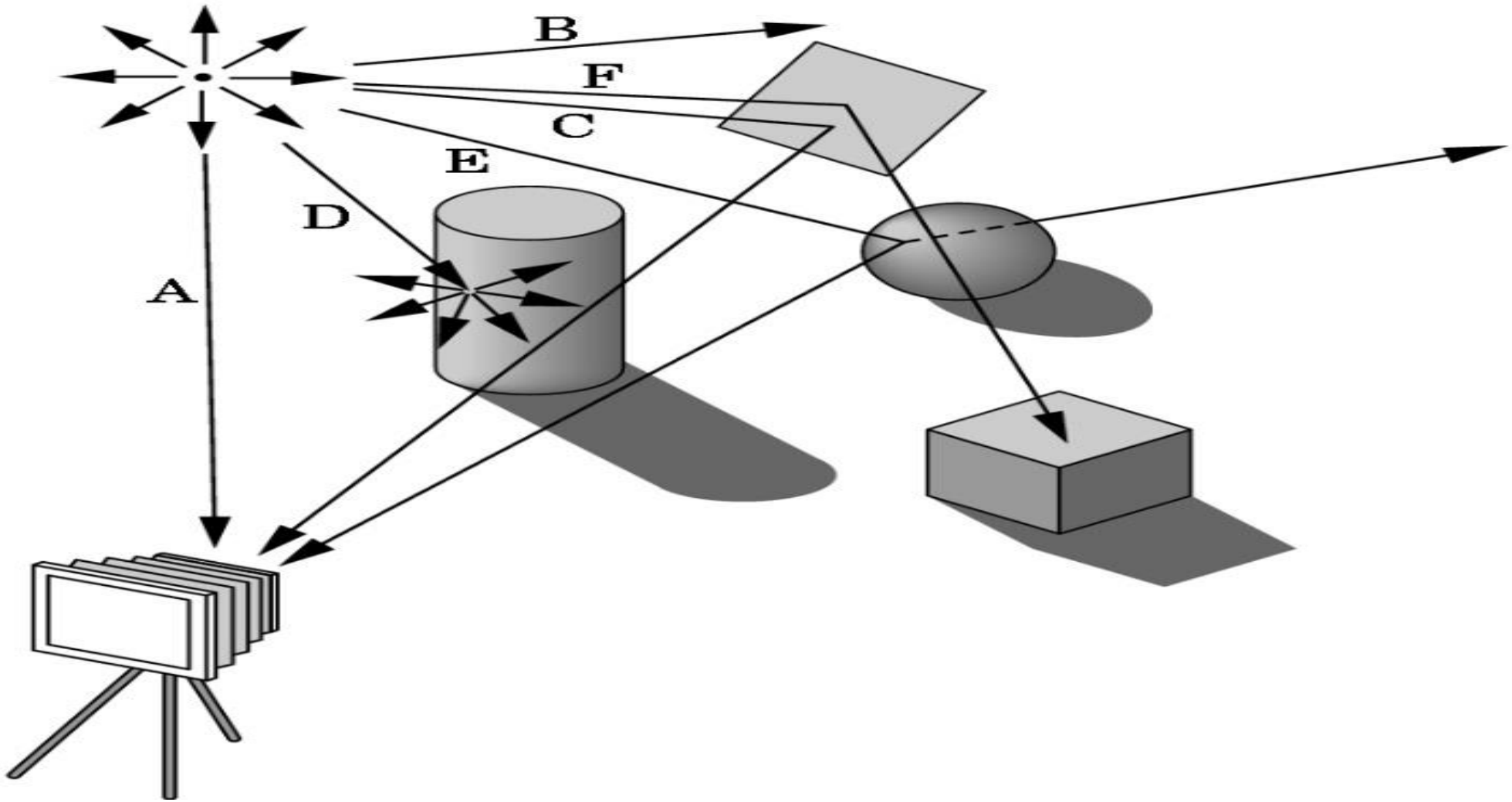
Question

1. 为了绘图需要怎样的一套工具-图形系统（硬件）？
2. 这套工具（图形系统）是如何实现绘图的（模型和框架-软件）？
3. 程序员如何使用所设计的这套计算机工具（API接口）？

Outlines

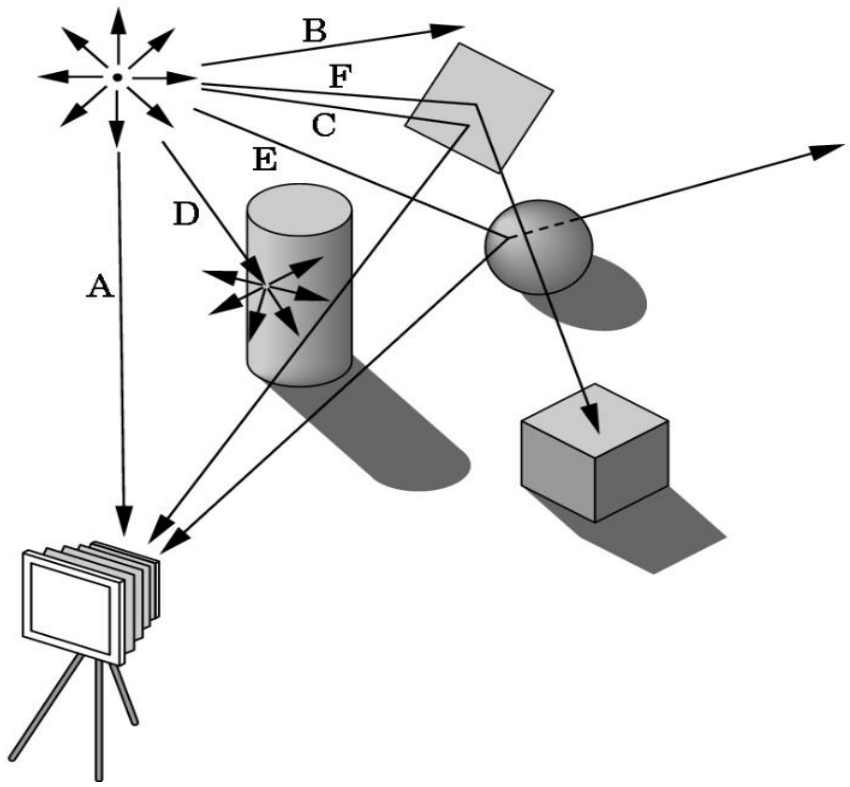
- **Imaging Principle and models(图形处理所基于的模型)**
  - Elements of Image Formation-图形生成的要素
  - Synthetic camera Model –虚拟相机模型
  - Global lighting & Local lighting Model-全局/局部光照模型
- **Imaging Implement （图形处理的实现框架）**
  - Model-Render mode 建模渲染模式
  - Architecture (render pipeline) 渲染管线
- **Imaging Implement （图形系统的编程使用）**
  - CG API

Light and Image光和像



Elements of Image Formation

- **Objects（对象）**
- **Viewer（观察者）**
- **Light source(s)（光源）**



Note:

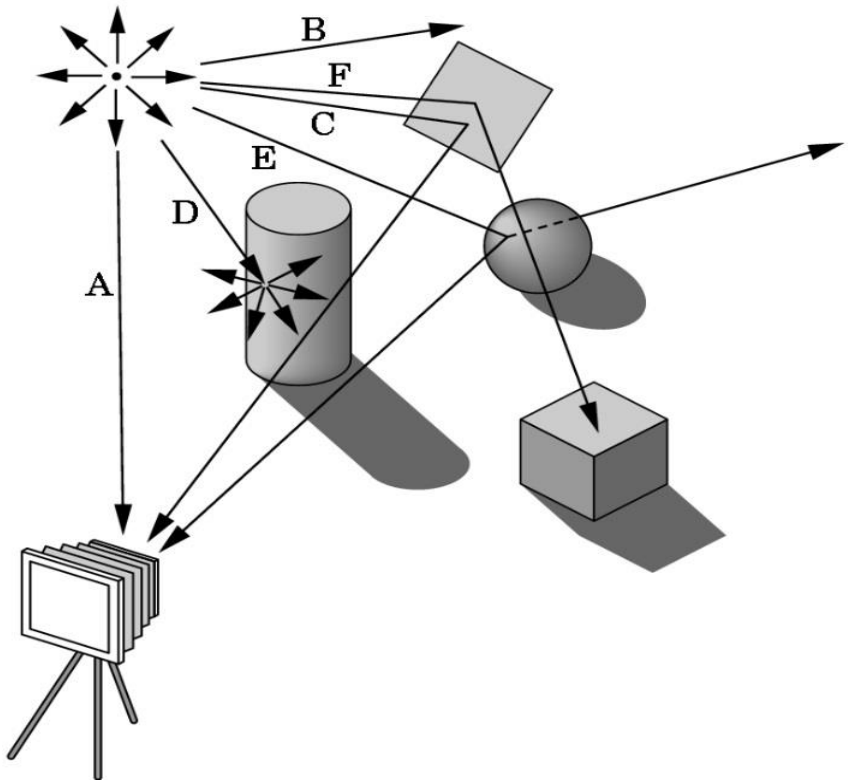
Attributes that govern how light interacts with the materials in the scene

Elements of Image Formation

- **Objects: need to be modeled**
  - Object properties: Position, Color
  - Texture color
  - Material properties

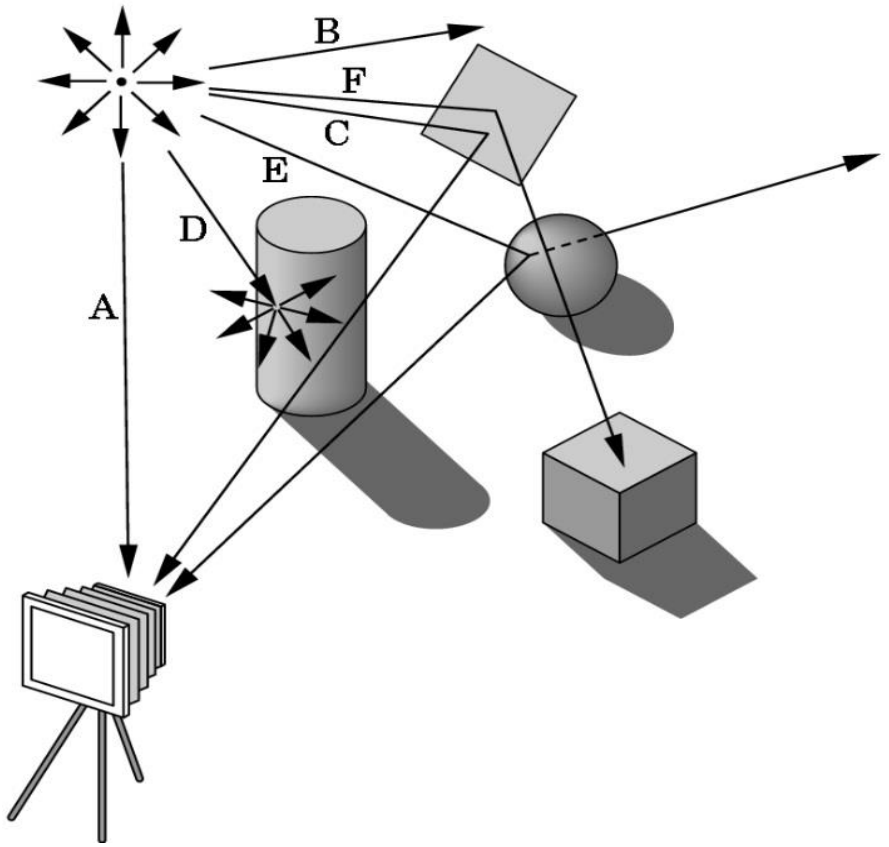
    Absorption吸收: color properties

    Scattering发散：Diffuse漫反射 or Specular镜面反射
- Light source(s)
- Viewer



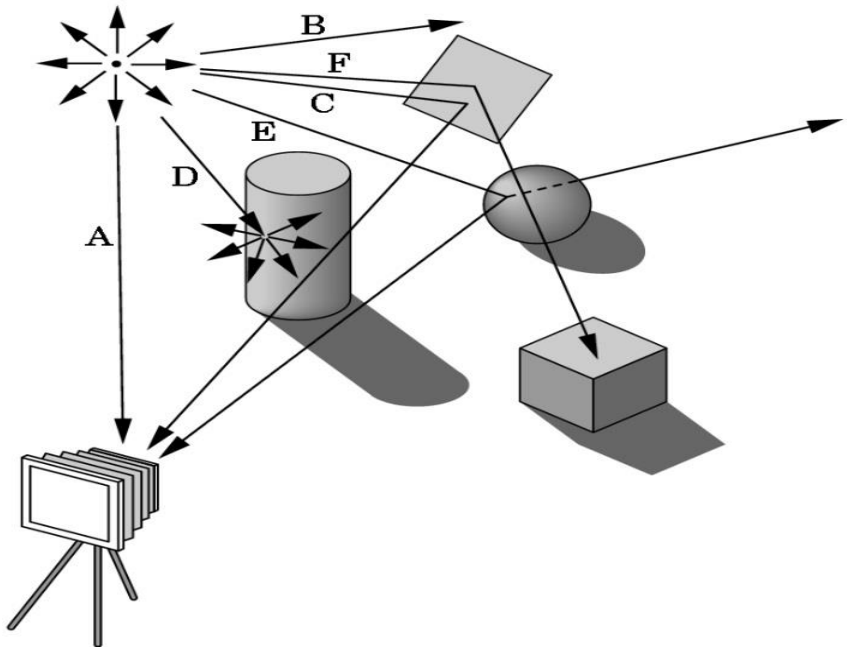
Elements of Image Formation

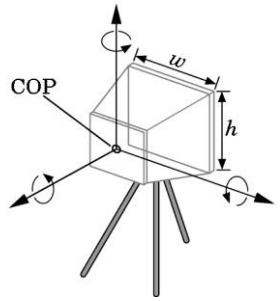
- Objects
  - Object properties （Position,color）
  - Material properties
- **Light source(s)**
  - Types of lights
    - Point sources vs distributed sources
    - Spot lights聚光灯
    - Distance: Near and far sources
    - Color properties
- Viewer



Elements of Image Formation

- Objects
- Light source(s)
- **Viewer /Camera/Eye**
  - Viewer properties
    - 透镜位置Position of center of lens
    - 透镜方向Orientation of Lens
    - 胶片尺寸Film size
    - 胶片平面朝向Orientation of film plane
    - 焦距Focal Length



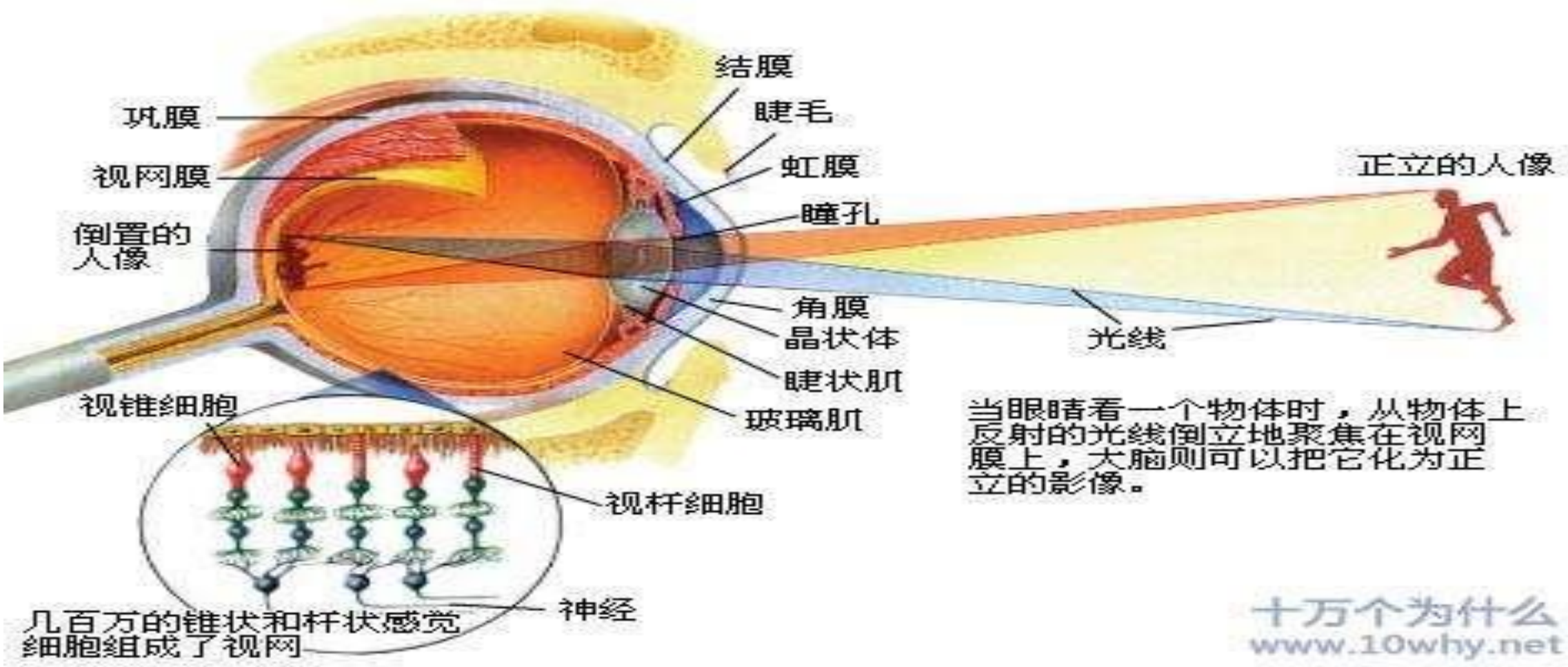


Outlines

- **Imaging Principle and models(图形处理基于的模型)**
  - Elements of Image Formation-图形生成的要素
  - Synthetic camera Model –虚拟相机模型
  - Global lighting & Local lighting Model-全局/局部光照模型
- **Imaging Implement （图形处理的实现框架）**
  - Model-Render mode 建模渲染模式
  - Architecture (render pipeline) 渲染管线
- **Imaging Implement （图形系统的使用方式）**
  - CG API

Human visual system

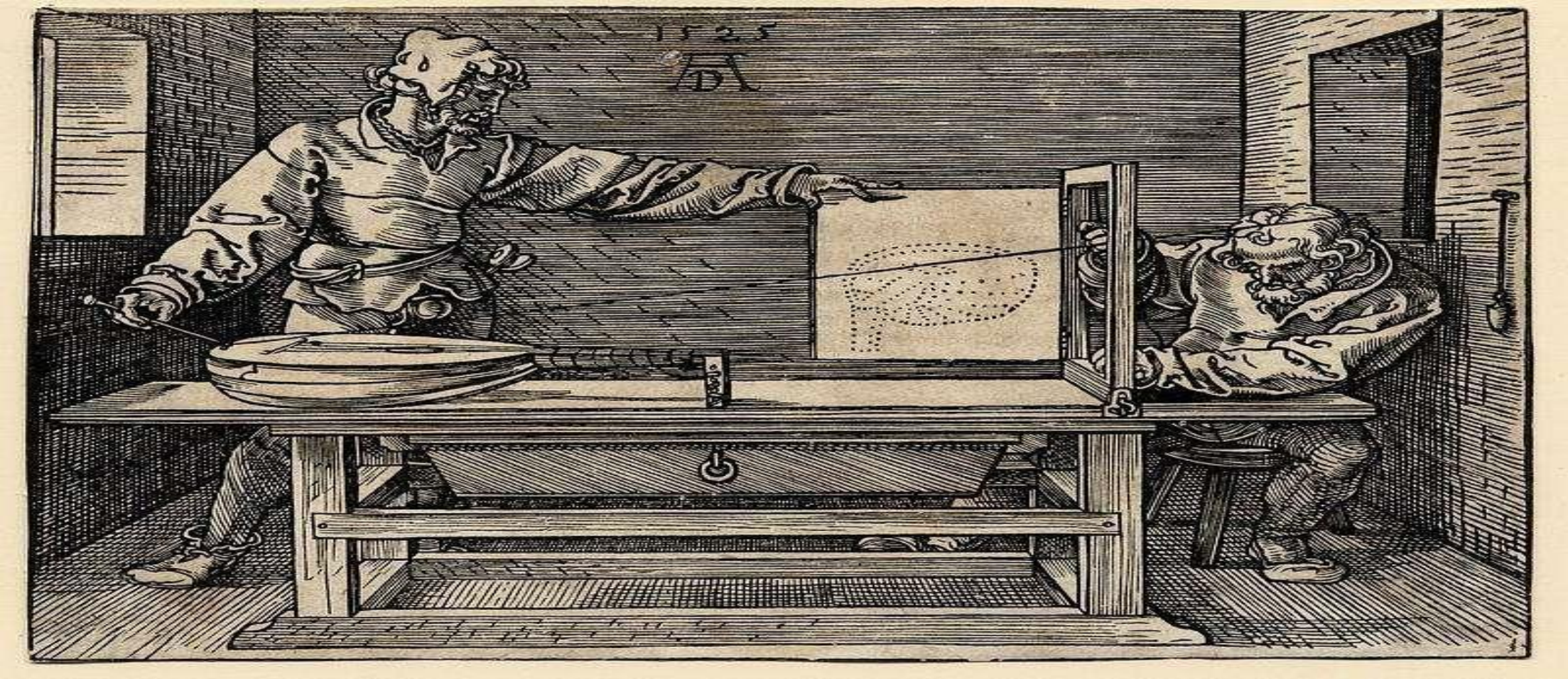
receiving light And transforming it into electrical energy light reflects from objects.接收从物体反射的光，并且将它转换成电能



Hand-painted

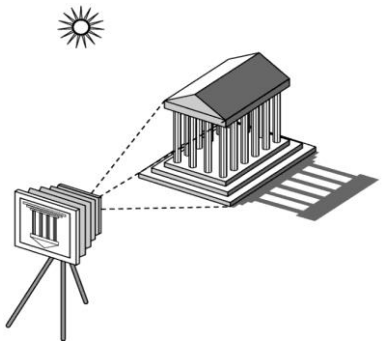
一个古老的绘制器：丢勒的”绘制引擎”

扩展资料：《计算机图形学原理及实践》约翰.F.休斯等著，彭群生等译 机械工业出版社第三章 一个古老的绘制器



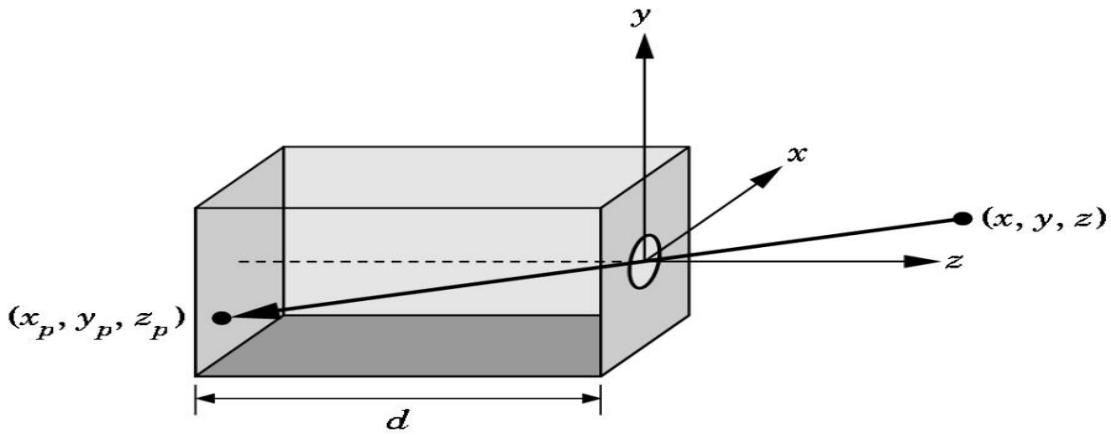
木刻画：鲁特琴图

Pinhole Camera



- **Use trigonometry（三角法） to find projection（xp,yp,zp） of point at (x,y,z)**

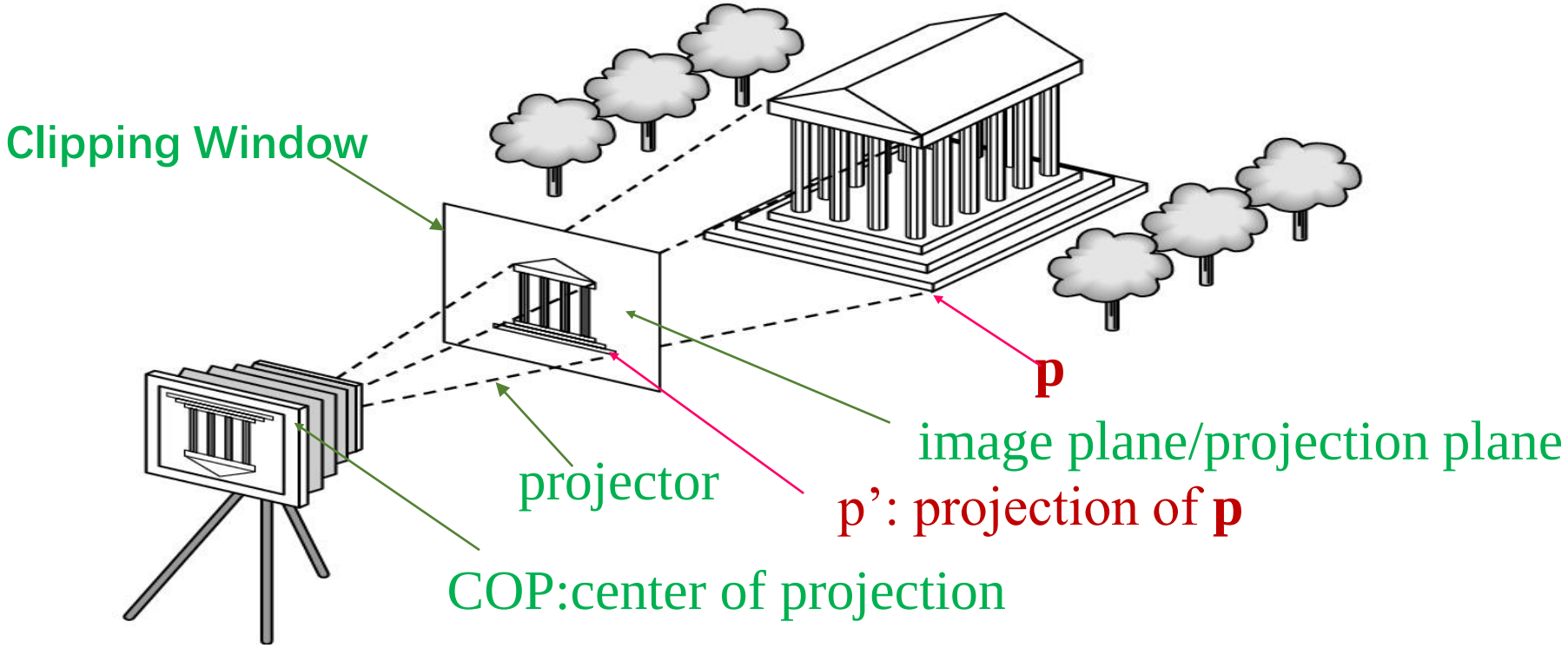
These are equations of simple perspective 利用简单的几何方法计算图像



$x_p = -x/z/d$ $y_p = -y/z/d$ $z_p = d$

Synthetic Camera Model（虚拟相机模型）

- 对象上的点P 到 透视中心COP:center of projection 在一条线上
- 虚拟成像平面projection plane作为投影平面，注意：面在COP前，成像是正的!
- 投影线projector和投影面projection plane的交点p’: projection of **p**就是所成的像。
- 图像范围是受限的（取景是一个裁剪窗口Clipping Window）



Synthetic Camera Model Advantages

- Separation of objects, viewer, light sources
- Leads to fast hardware implementation（快速硬件实现）
- Leads to simple software API(简单软件接口函数)
  - Specify objects, lights, camera, attributes(Application)
  - Let implementation determine image(Pipeline)

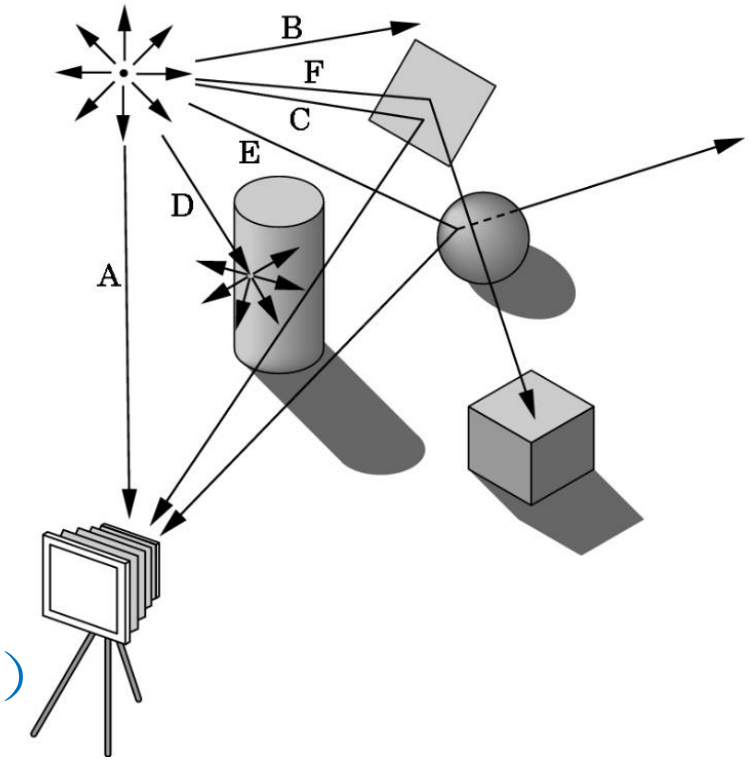
Outlines

- **Imaging Principle and models(图形处理所基于的模型)**
  - Elements of Image Formation-图形生成的要素
  - Synthetic camera Model –虚拟相机模型
  - **Global lighting & Local lighting Model-全局/局部光照模型**
- **Imaging Implement （图形处理的实现框架）**
  - Model-Render mode 建模渲染模式
  - Architecture (render pipeline) 渲染管线
- **Imaging Implement （图形系统的编程使用）**
  - CG API

Global Lighting

each ray of light may have multiple interactions with objects before being absorbed or going to infinity.

- *Some objects are blocked from light*
- *Light can reflect from object to object*
- *Some objects might be translucent（透明）*



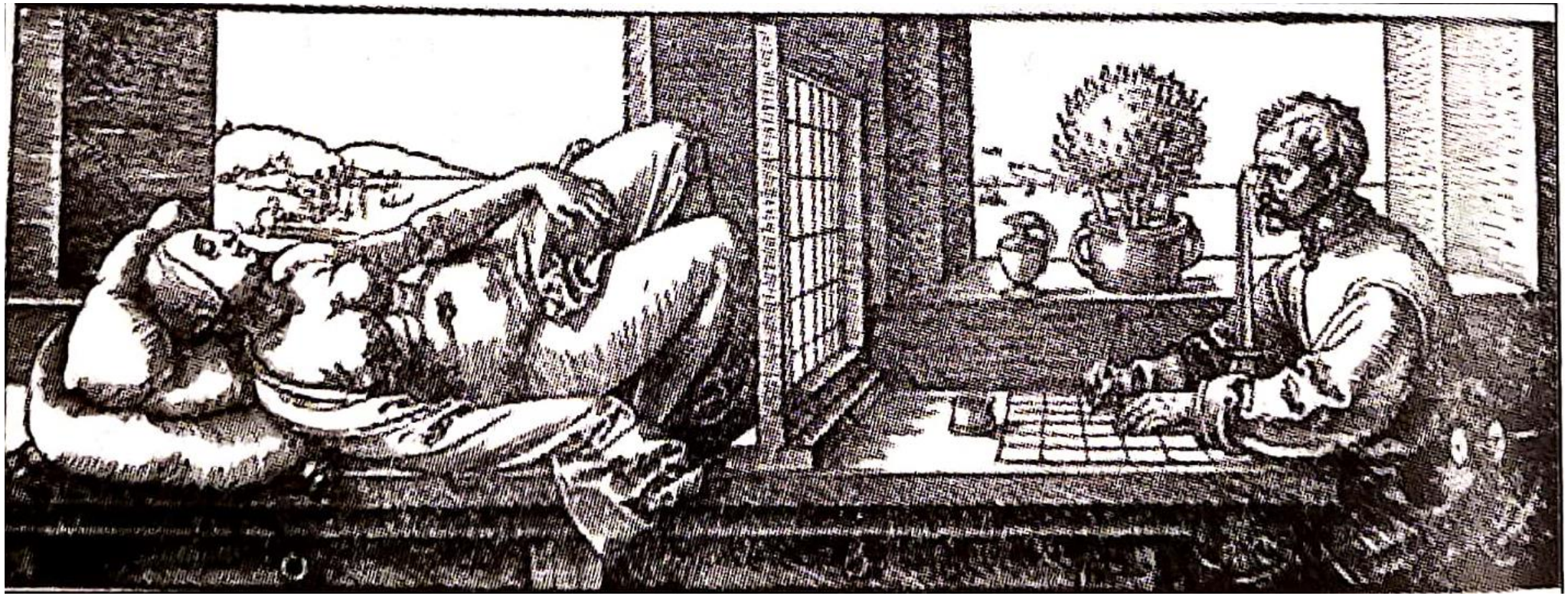
Cannot compute color or shade of each object independently（不能独立计算每个物体的颜色或阴影）

Ray Tracing and Geometric Optics

One way to form an image is

to follow rays of light from a point source

finding which rays enter the lens of the camera(eye).



Physical Approaches

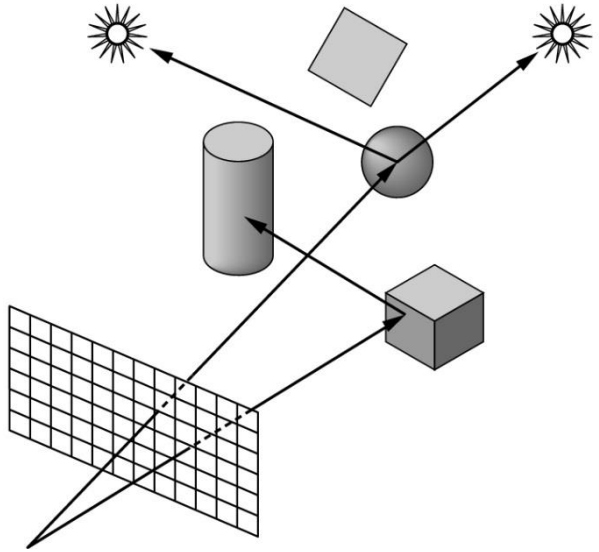
- **Ray tracing**: follow rays of light from center of projection until they either are absorbed by objects or go off to infinity

光线追踪：从投影中心开始追踪光线，直到光线被物体吸收或无限远

  - Can handle global effects
    - Multiple reflections
    - Translucent objects

Possible and is actually simple for simple objects such as polygons and quadrics with simple point sources. In principle, can produce global lighting effects such as shadows and multiple reflections

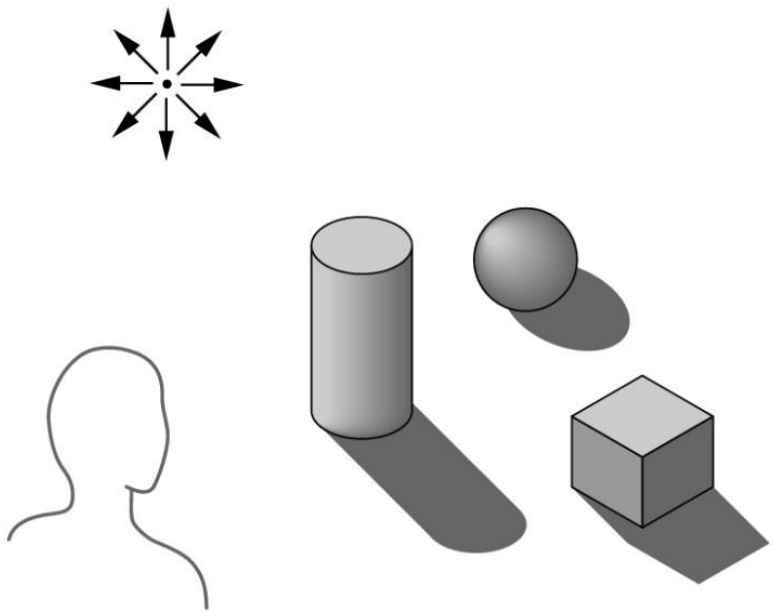
对于简单的物体（例如多边形和具有简单点源的二次曲面），光线跟踪算法是可能的而且实际上是简单的。原则上，可以产生全局照明效果，如阴影和多次反射。



Local Lighting model

不考虑物体之间的光线交互，每个物体都是独立于其它物体而单独成像

- 局部光照计算中，物体相互影响效果如阴影，透明等通过其它方法解决！



Why not ray tracing?

- Ray tracing seems more physically based so why don't we use it to design a graphics system?

  为什么不用光线跟踪模型设计图形系统的成像计算？

  but ray tracing is slow and not well-suited for interactive applications

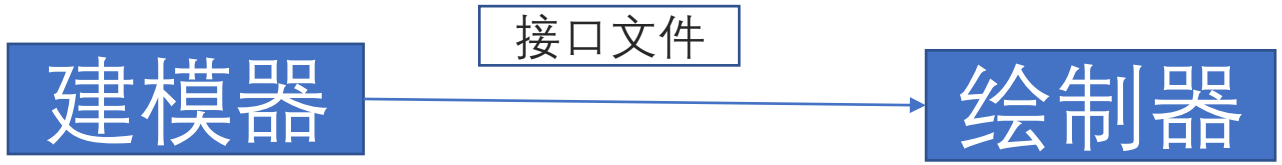
  光线跟踪是计算复杂速度慢的，不太适合复杂实时互动应用（如游戏）

  - Slow（硬件限制，现已有突破）
  - Must have whole data base available at all times （考虑场景整体数据）

Outlines

- **Imaging Principle and models(图形处理所基于的模型)**
  - Elements of Image Formation-图形生成的要素
  - Synthetic camera Model –虚拟相机模型
  - **Global lighting & Local lighting Model-全局/局部光照模型**
- **Imaging Implement （图形处理的实现框架）**
  - Model-Render mode 建模渲染模式
  - Architecture (render pipeline) 渲染管线(固定和可编程)
- **Imaging Implement （图形系统的编程使用）**
  - CG API

Image Formation Implement



Modeling建模+Rendering渲染

- 如何实现“建模”：
  导入专用软件(Maya,3Dmax…)生成的模型数据，或者自编程创建模型

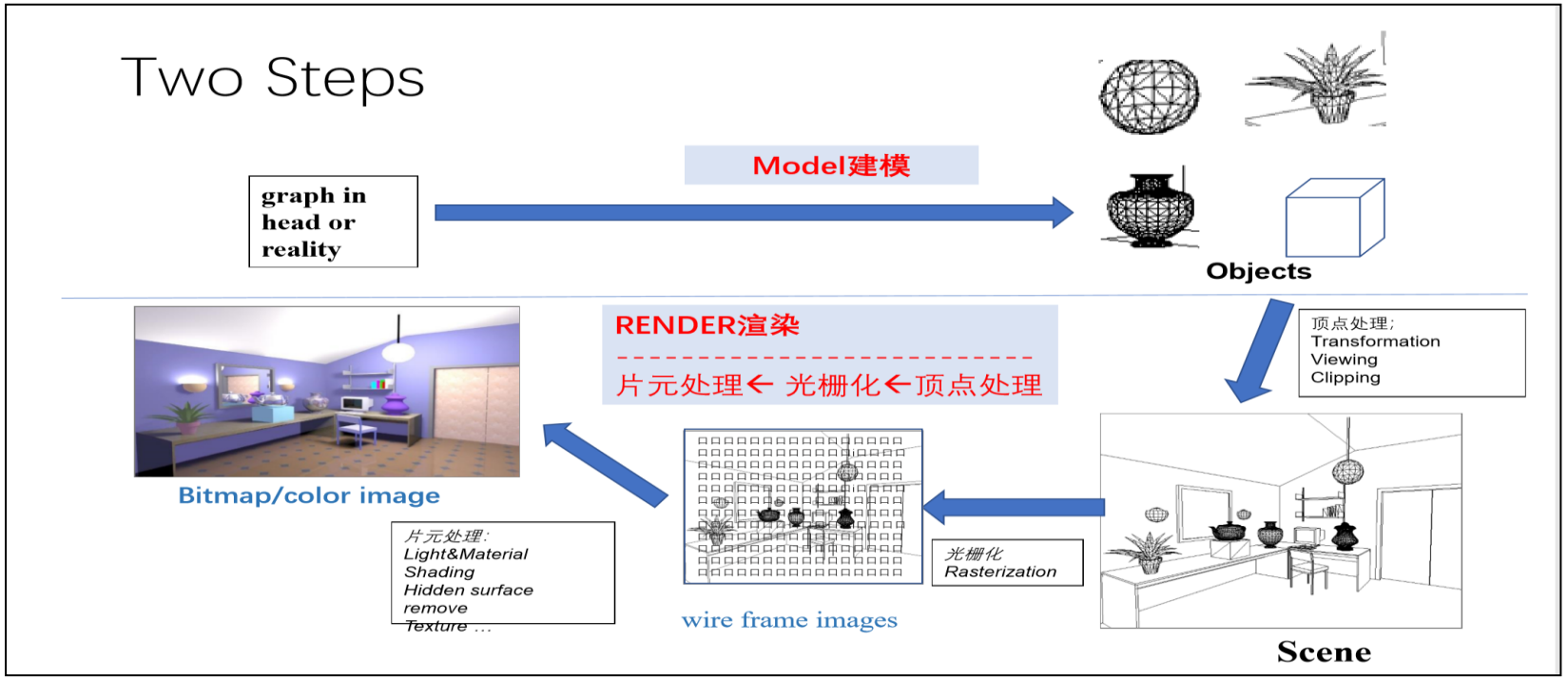
- 如何实现“渲染”：

基于模型：Synthetic Camera Model +Local Lighting Model

实现框架：Programmable Pipeline

Model-Render Mode

- **Modeling（建模器）：完成模型顶点计算工作**。描述要绘制的对象属性，材质，光源，观察者等的数据结构。
- **Rendering（绘制器）：完成图形到图像的计算工作。**Pipeline architecture



Outlines

- **Imaging Principle and models(图形处理所基于的模型)**
  - Elements of Image Formation-图形生成的要素
  - Synthetic camera Model –虚拟相机模型
  - **Global lighting & Local lighting Model-全局/局部光照模型**

- **Imaging Implement （图形处理的实现框架）**
  - Model-Render mode 建模渲染模式
  - **Architecture (render pipeline) 渲染管线(固定和可编程)**

- **Imaging Implement （图形系统的编程使用）**
  - CG API

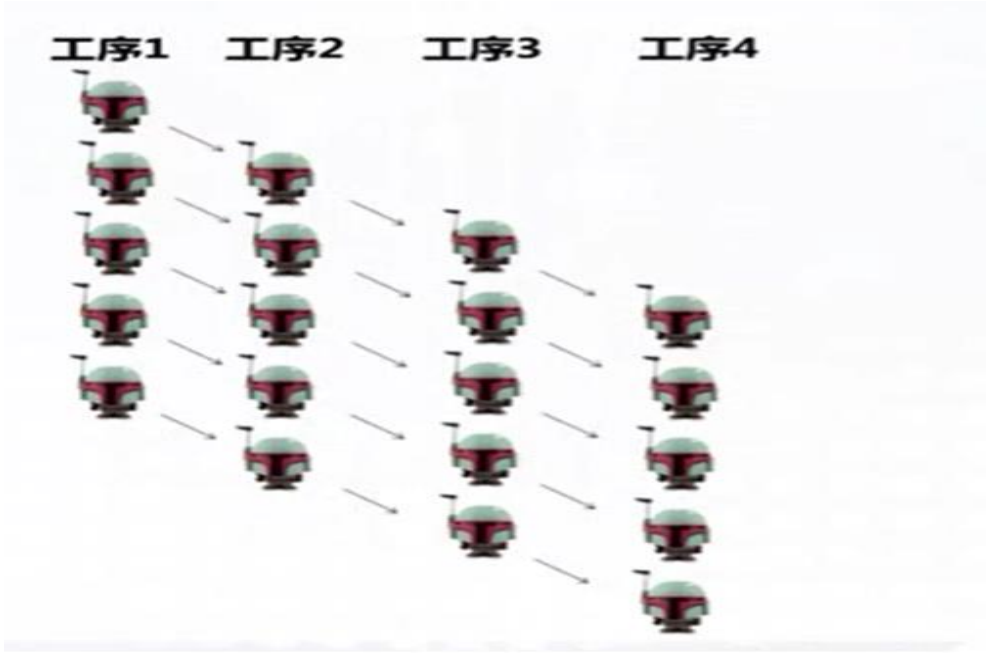
Pipeline architecture

- 流水线：将复杂处理分成多个工序。（采用VLSI电路可构建流水线体系结构）
- 并行计算(Parallel Computing)：是指同时使用多种计算资源解决计算问题的过程，是提高计算机系统计算速度和处理能力的一种有效手段。分时间和空间的并行，流水线是时间并行。

图形处理需以“相同方式”处理“数量巨大”的顶点和像素

所以采用了基于流水线的并行处理方式，

可以极大提高系统的吞吐量~提高效率~。



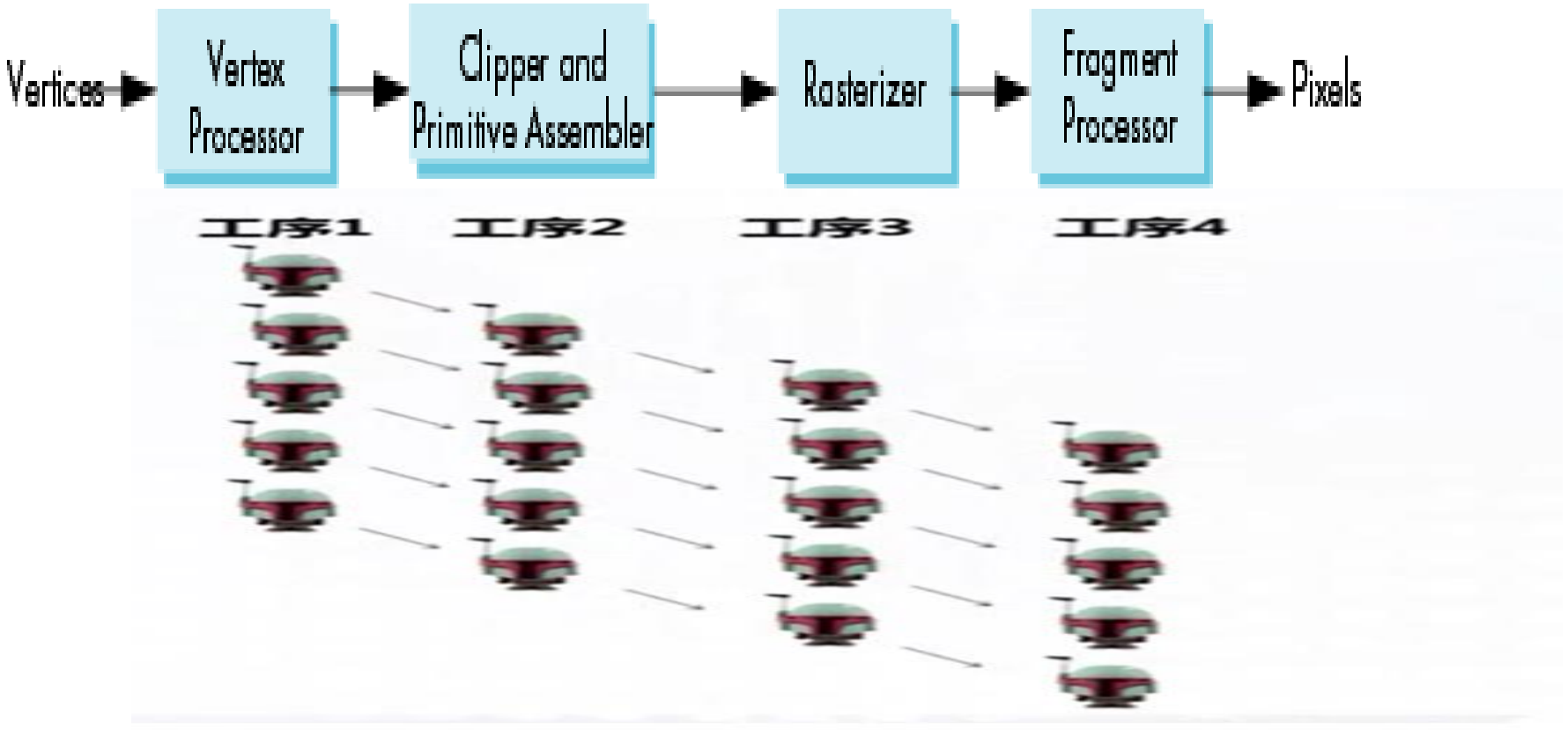
评估流水线的两个性能指标

- 延迟：流水线中模块之间的通过时间称为“延迟”。
- 吞吐量：数据通过系统的速率。

延迟和吞吐量需要权衡。

渲染管线：Pipeline architecture

- Process objects one at a time in the order they are generated by the application
- 按应用程序生成对象的顺序一次处理一个对象(数据不同，操作相同)

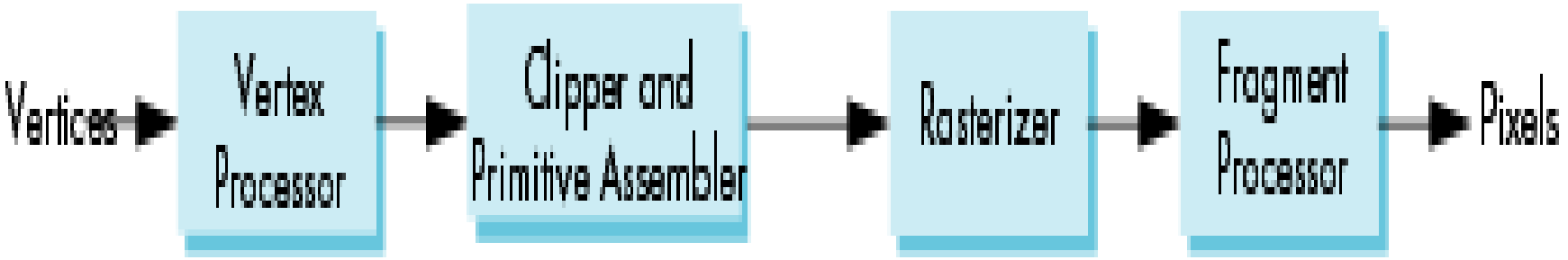


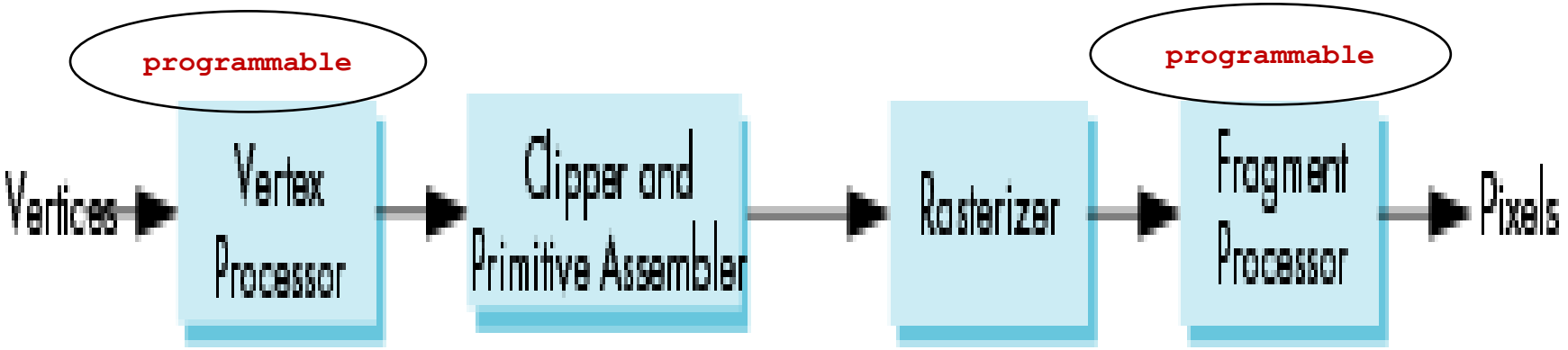
All steps can be implemented in hardware on the graphics card

可编程管线programmable pipeline

固定管线： The fix functions pipeline: Black Box View

可编程管线：The programmable pipeline: extend functions

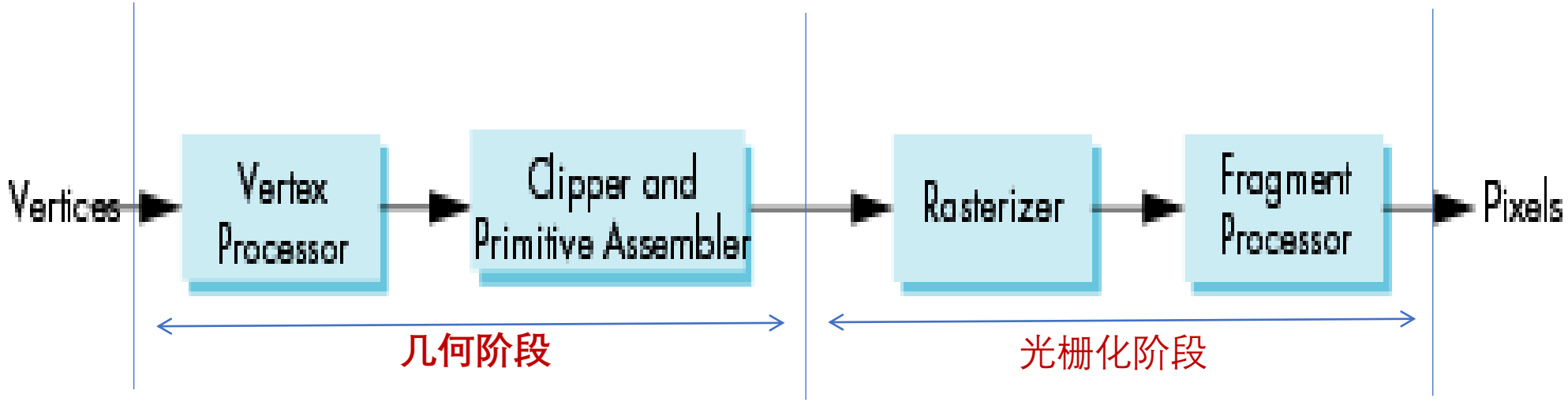




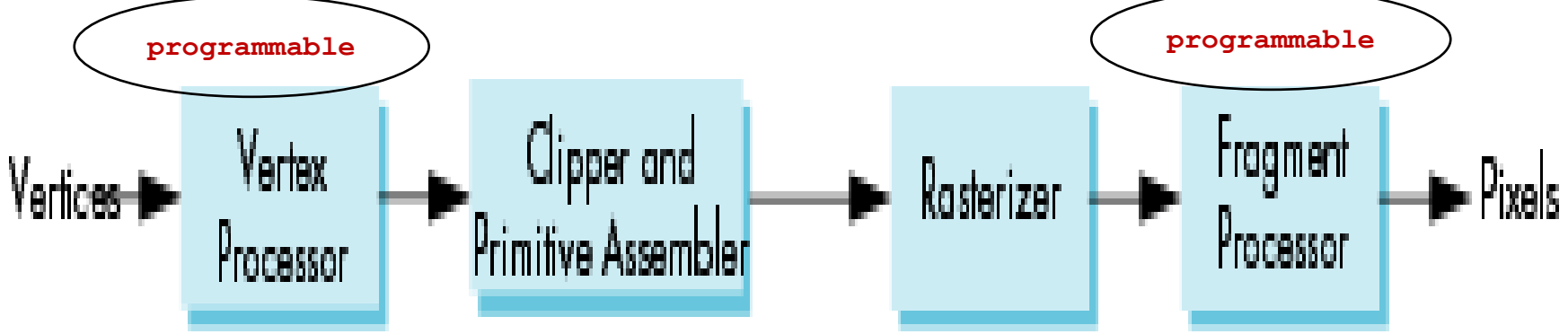
渲染管线模块

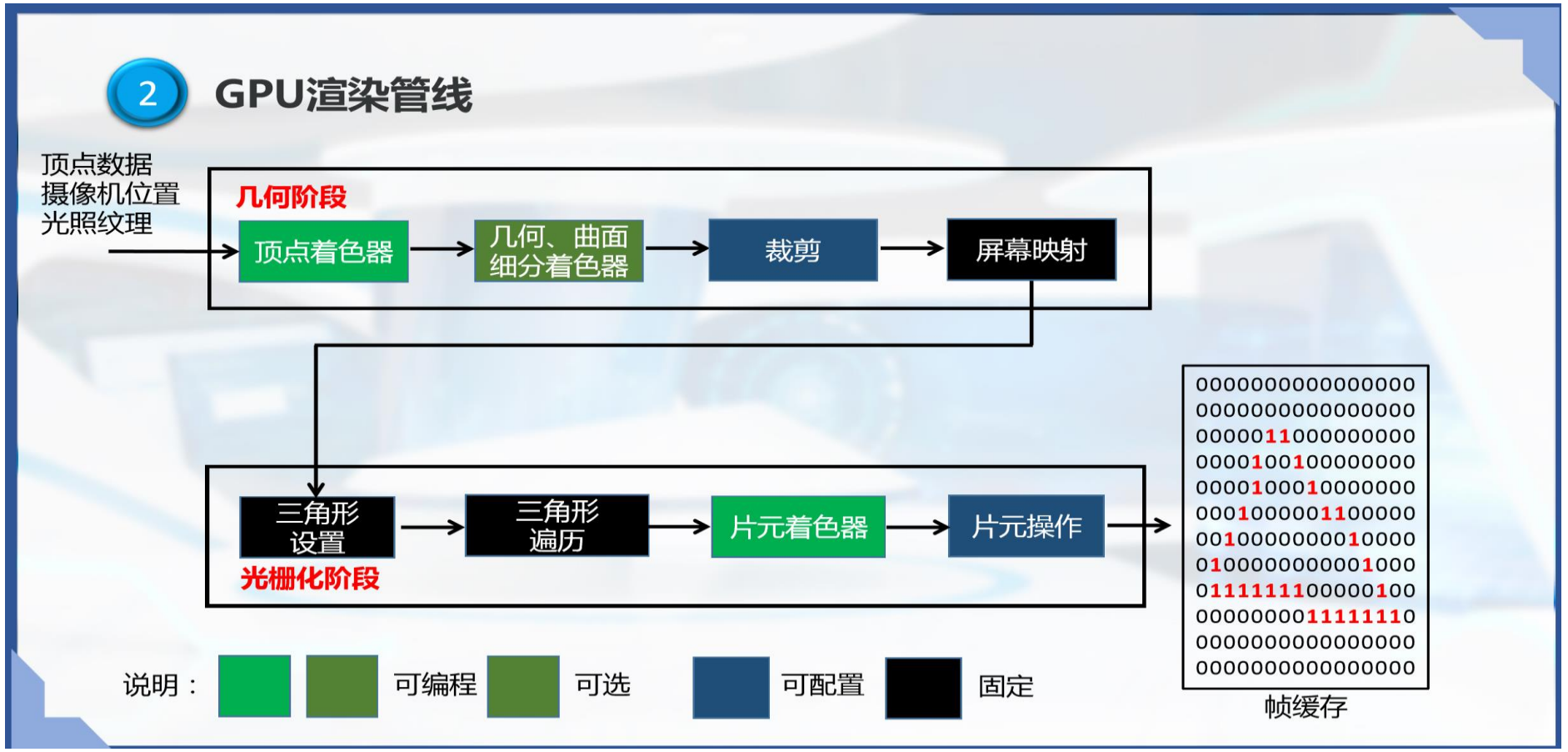
- Process objects one at a time in the order they are generated by the application

  按应用程序生成对象的顺序一次处理一个对象(数据不同，操作相同)

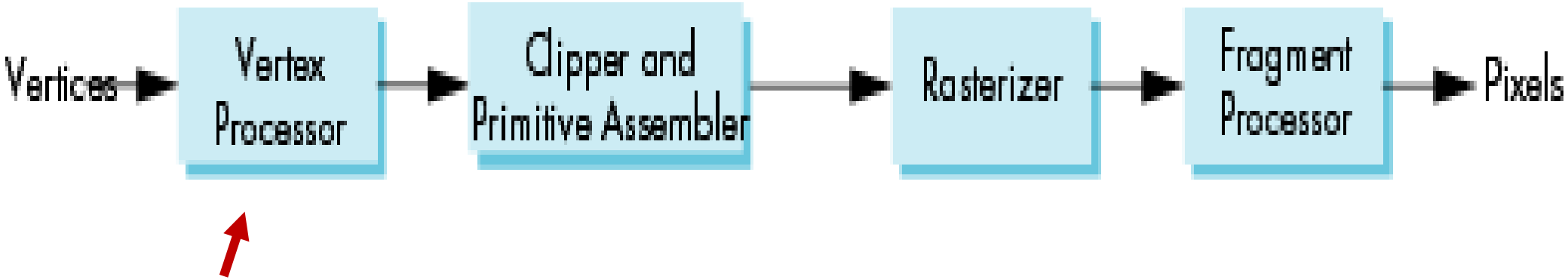


All steps can be implemented in hardware on the graphics card


programmable
programmable
Vertices
Vertex Processor
Clipper and Primitive Assembler
Rasterizer
Fragment Processor
Pixels


2
GPU渲染管线
顶点数据
摄像机位置
光照纹理
几何阶段
顶点着色器
几何、曲面细分着色器
裁剪
屏幕映射
三角形设置
三角形遍历
片元着色器
片元操作
光栅化阶段
0000000000000000
0000000000000000
0000011000000000
0000100100000000
0000100010000000
0001000001100000
0010000000010000
0100000000001000
0111111100000100
0000000011111110
0000000000000000
0000000000000000
帧缓存
说明：
可编程
可选
可配置
固定

1.Vertex Processing顶点处理



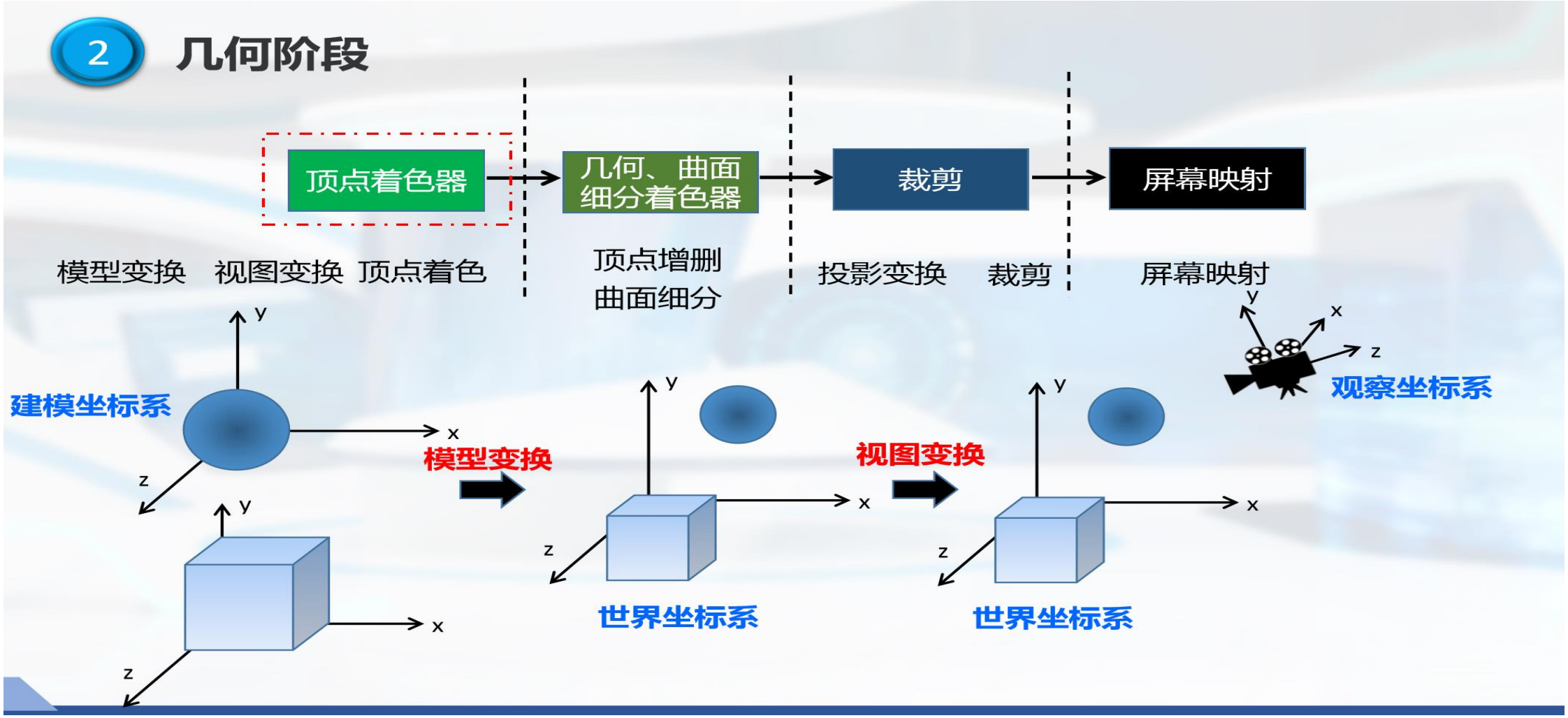
- **Much of the work in the pipeline is in**

converting object representations from one coordinate system to another

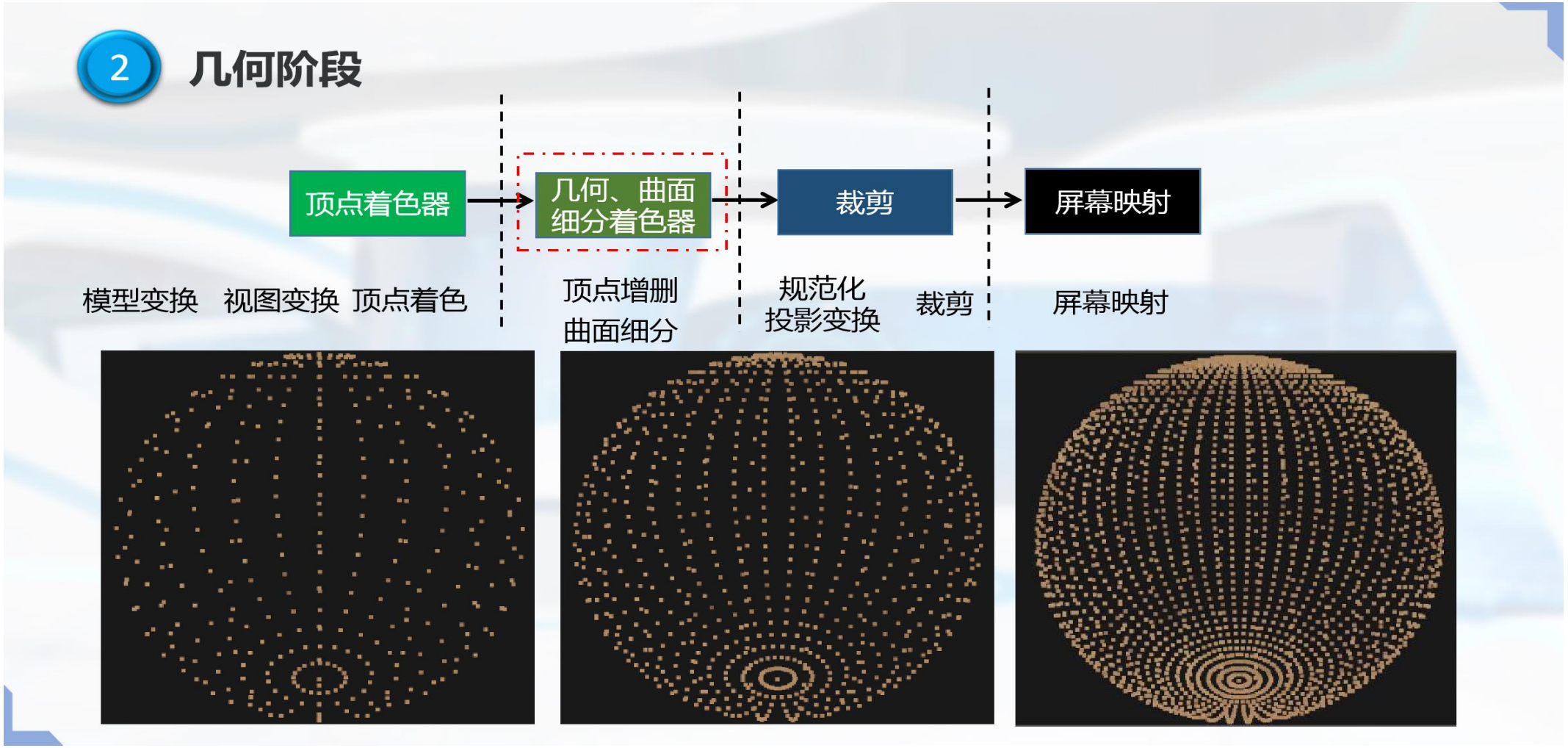
  - **Object coordinates**
  - **Camera (eye) coordinates**
  - **Screen coordinates**

- Every change of coordinates is equivalent to a matrix transformation
- Vertex processor also computes vertex colors

顶点处理：坐标变换


2
几何阶段
顶点着色器
几何、曲面细分着色器
裁剪
屏幕映射
模型变换
视图变换
顶点着色
顶点增删
曲面细分
投影变换
裁剪
屏幕映射
建模坐标系
模型变换
世界坐标系
视图变换
世界坐标系
观察坐标系
x
y
z

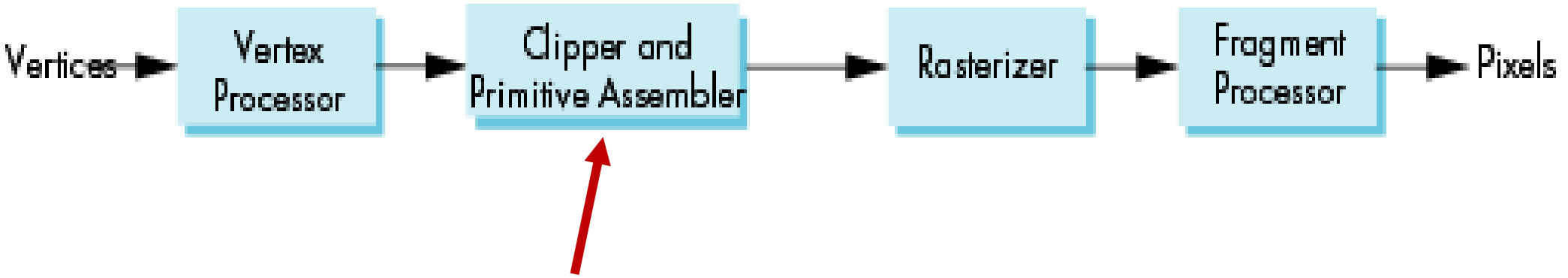
几何着色器-增删顶点


2
几何阶段
顶点着色器
几何、曲面细分着色器
裁剪
屏幕映射
模型变换
视图变换
顶点着色
顶点增删
曲面细分
规范化
投影变换
裁剪
屏幕映射

2. Clipping& Primitive Assembly裁剪和图元组装

- Line segments
- Polygons
- Curves and surfaces

- Line segments
- Polygons
- Curves and surfaces

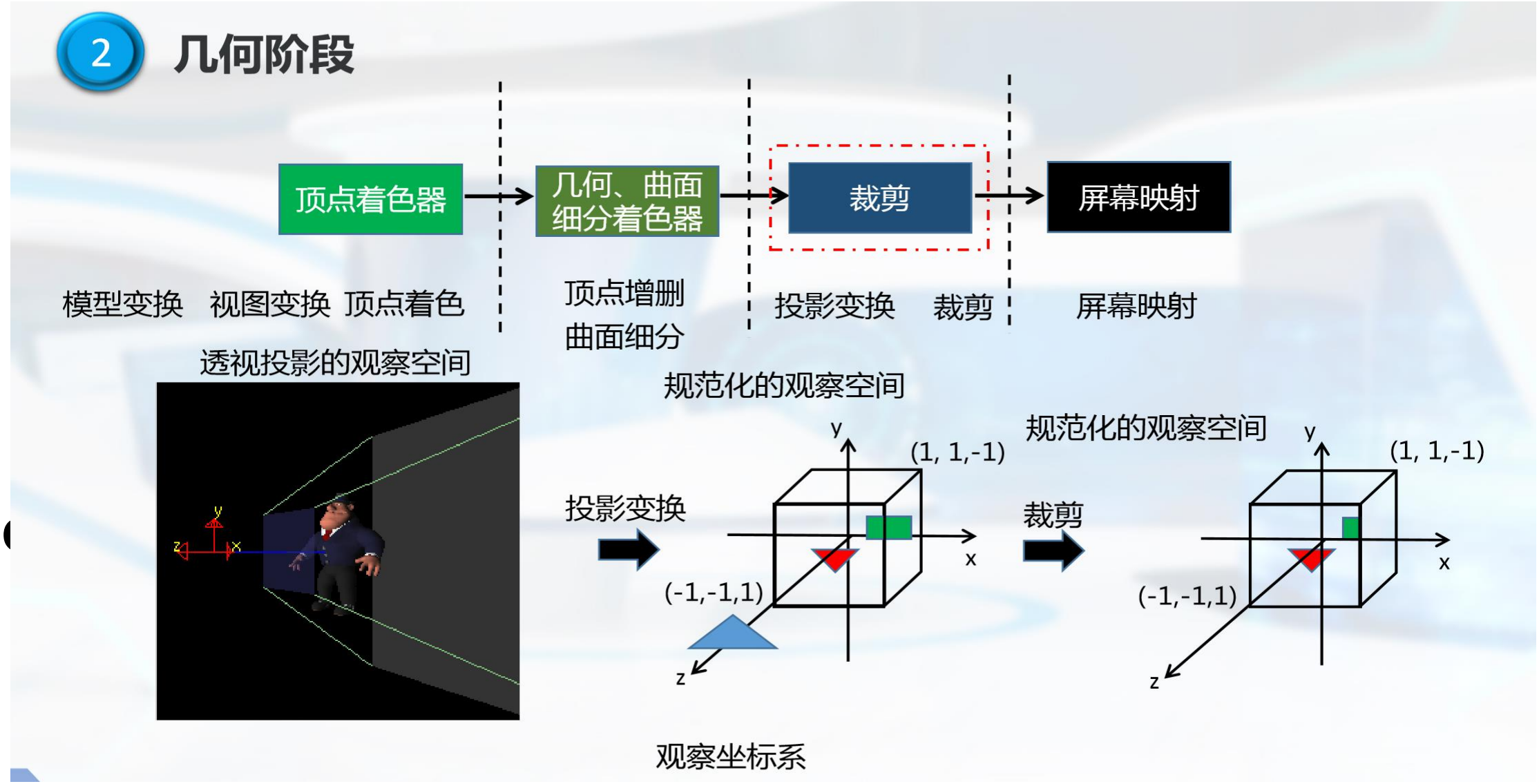


Primitive Assembly:

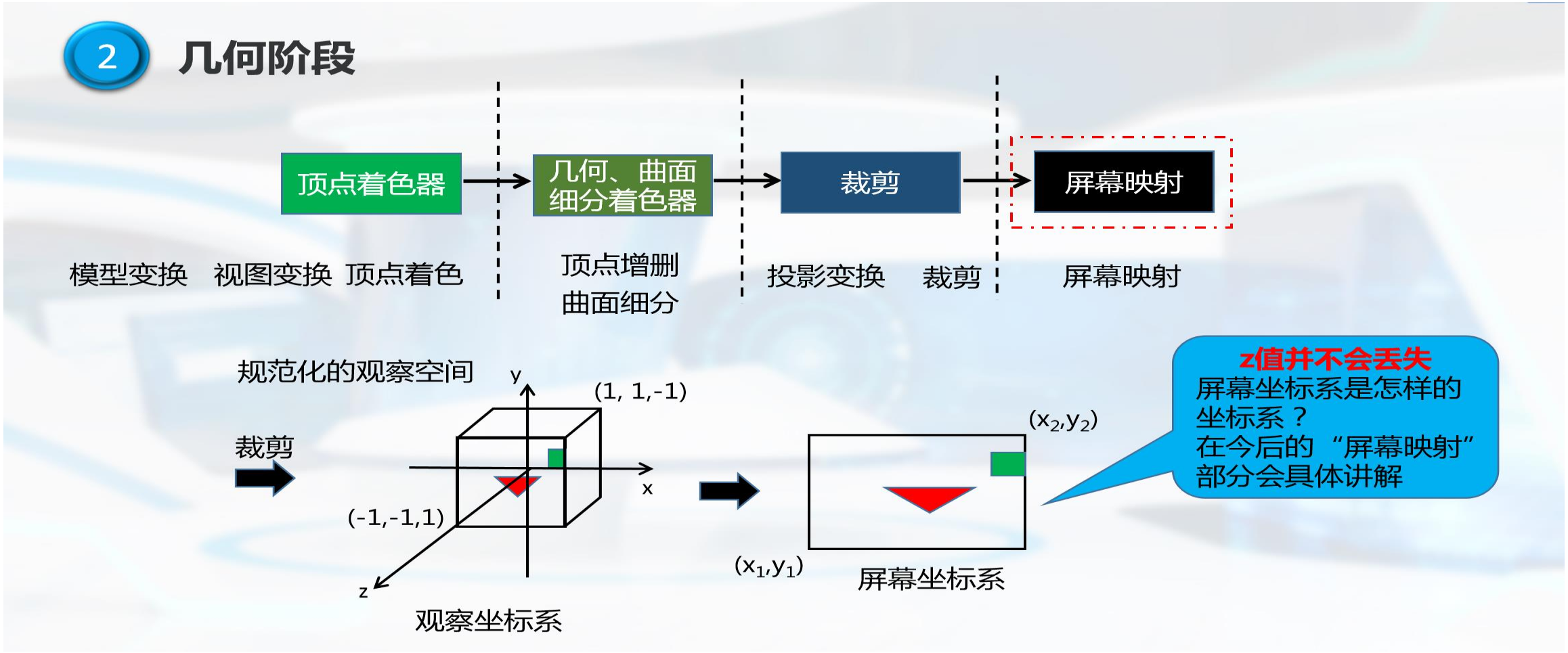
顶点需要被组装成有顺序的形式以表达为几何对象“图元”，才能被裁剪

Vertices must be collected into geometric objects before clipping

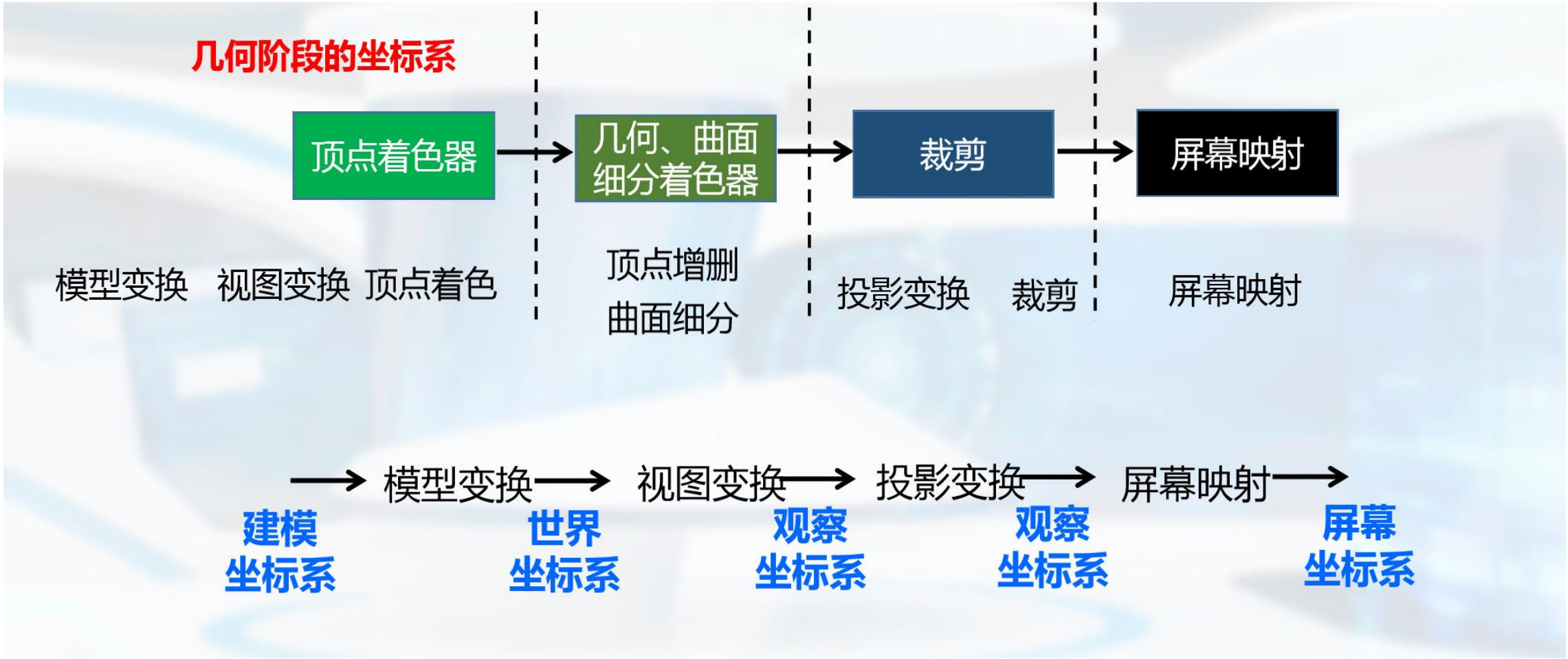
2. Clipping& Primitive Assembly裁剪和图元组装



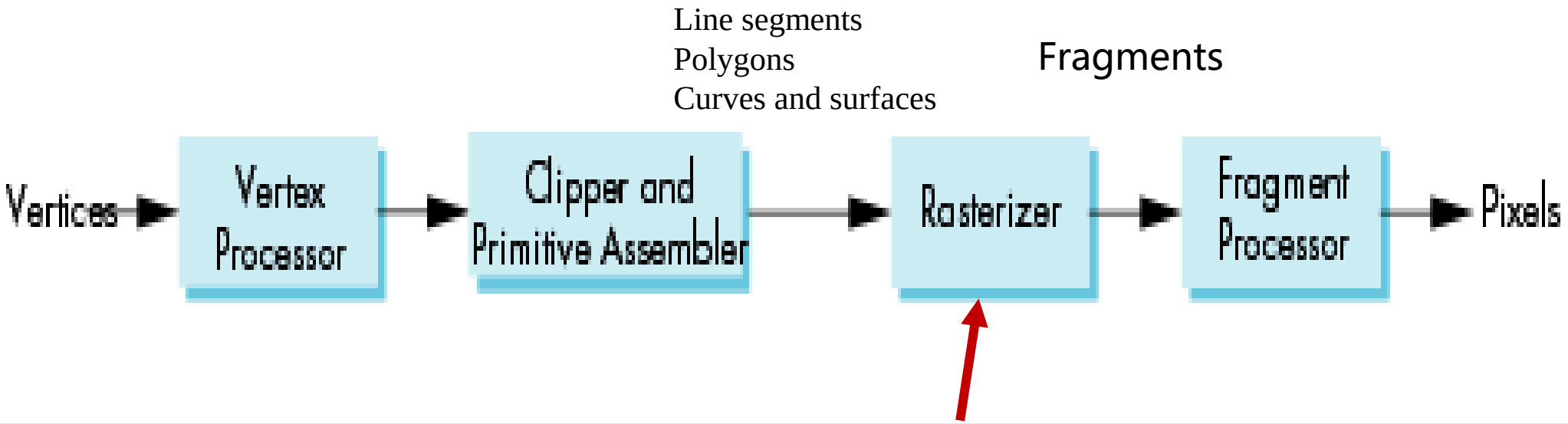
2. Clipping& Primitive Assembly裁剪和图元组装



总结：几何阶段（坐标变化）


几何阶段的坐标系
顶点着色器
几何、曲面细分着色器
裁剪
屏幕映射
模型变换
视图变换
顶点着色
顶点增删
曲面细分
投影变换
裁剪
屏幕映射
模型变换
视图变换
投影变换
屏幕映射
建模坐标系
世界坐标系
观察坐标系
观察坐标系
屏幕坐标系

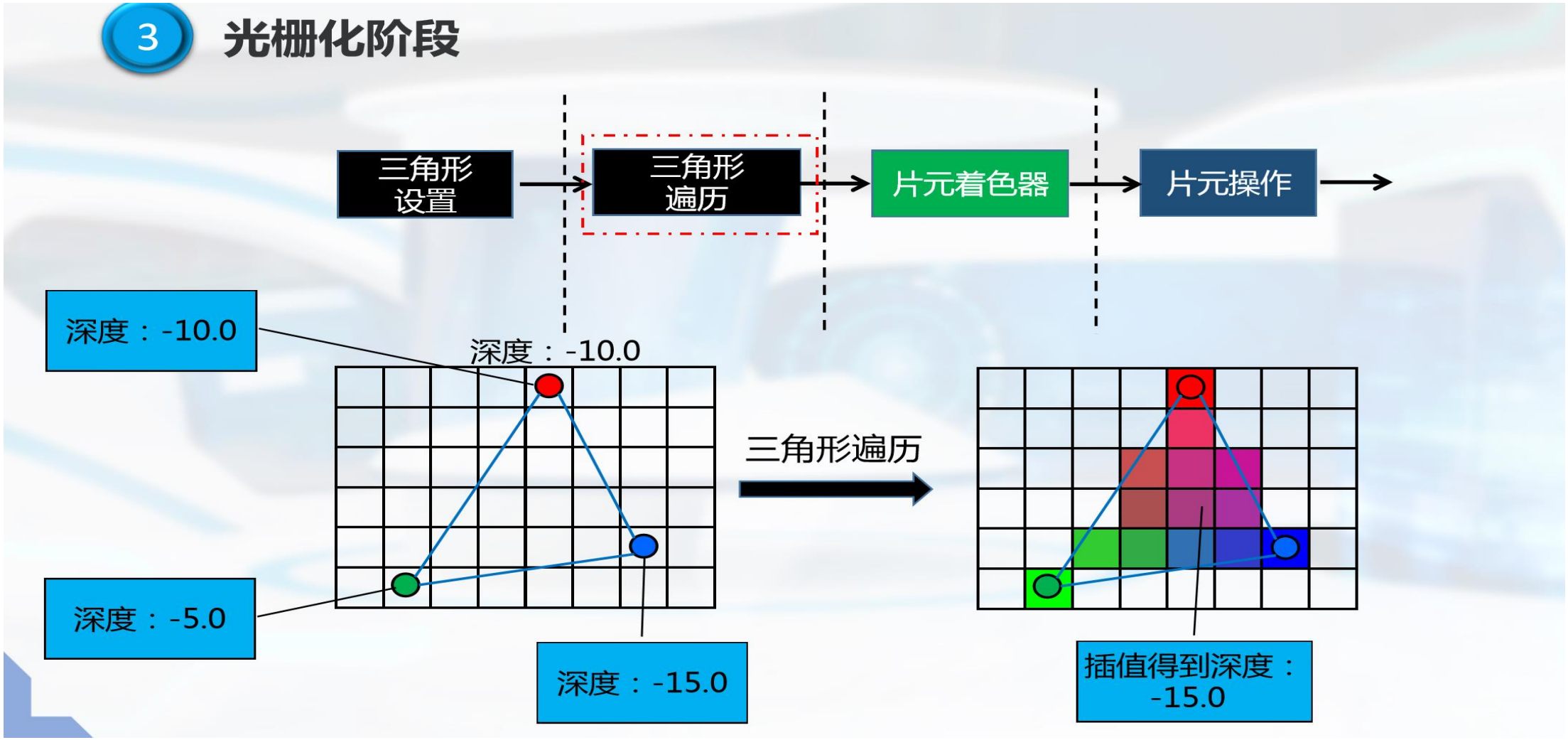
3.Rasterization光栅化



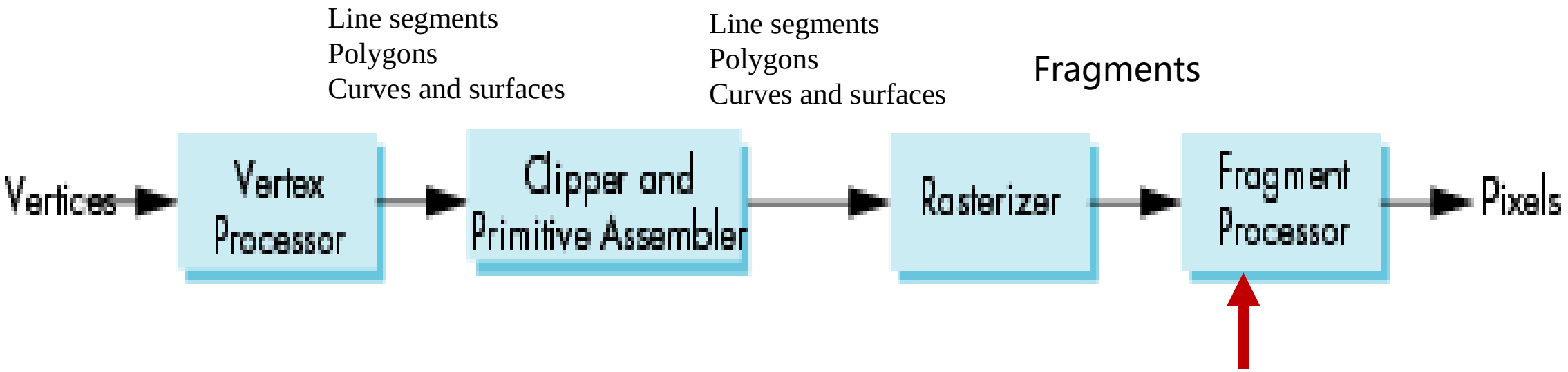
Rasterizer produces a set of fragments for each object

- Fragments are "potential pixels"
  - Have a location in frame bufffer
  - Color and depth attributes

光栅化：将图形转换为图像（插值）



4.Fragment Processing

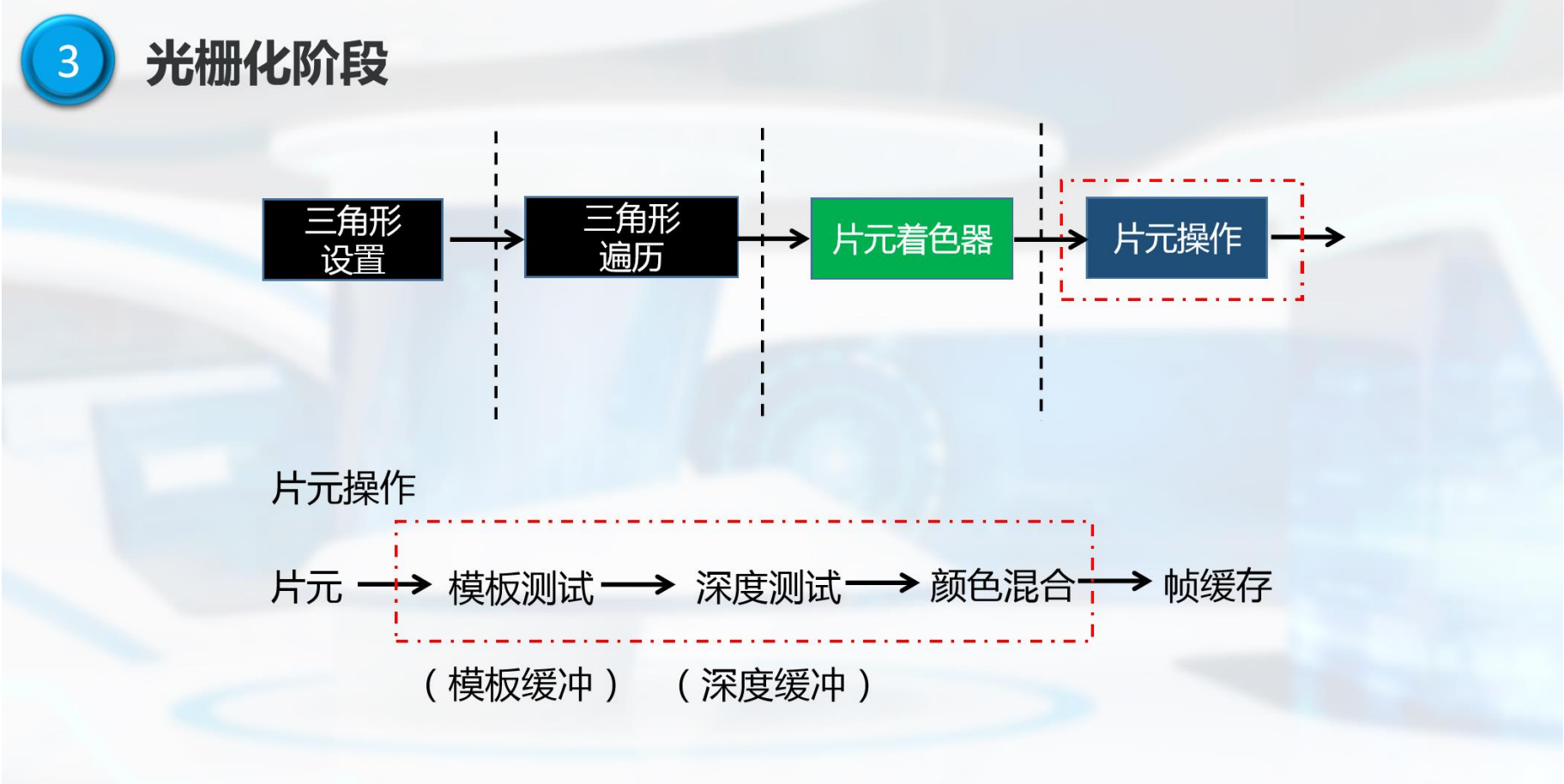


Fragments are processed

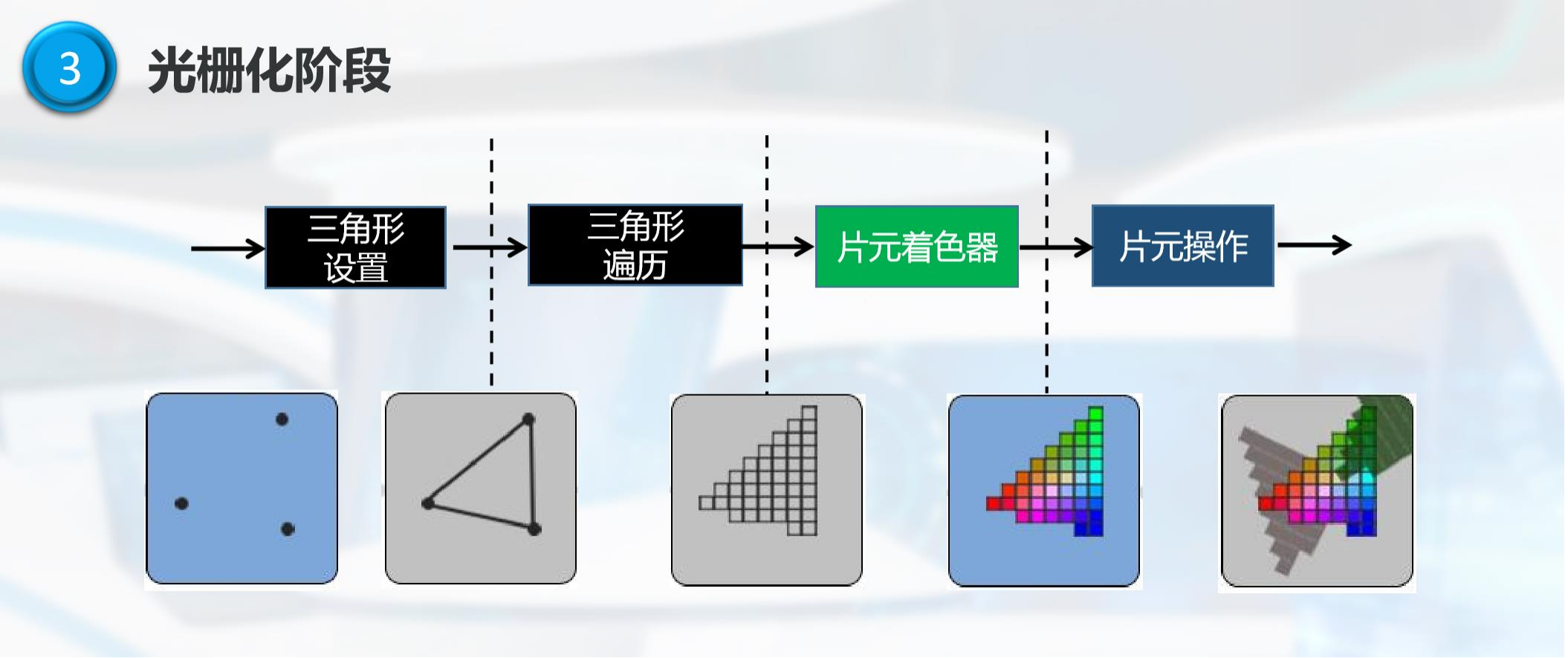
to determine the color of the corresponding pixel in the frame buffer

- Colors can be determined by texture mapping or interpolation of vertex colors
- Fragments may be blocked by other fragments closer to the camera Hidden-surface removal

片元融合得到像素色


3
光栅化阶段
三角形设置
三角形遍历
片元着色器
片元操作
片元操作
片元
模板测试
深度测试
颜色混合
帧缓存
（模板缓冲）
（深度缓冲）

总结：光栅化阶段（从顶点到像素）



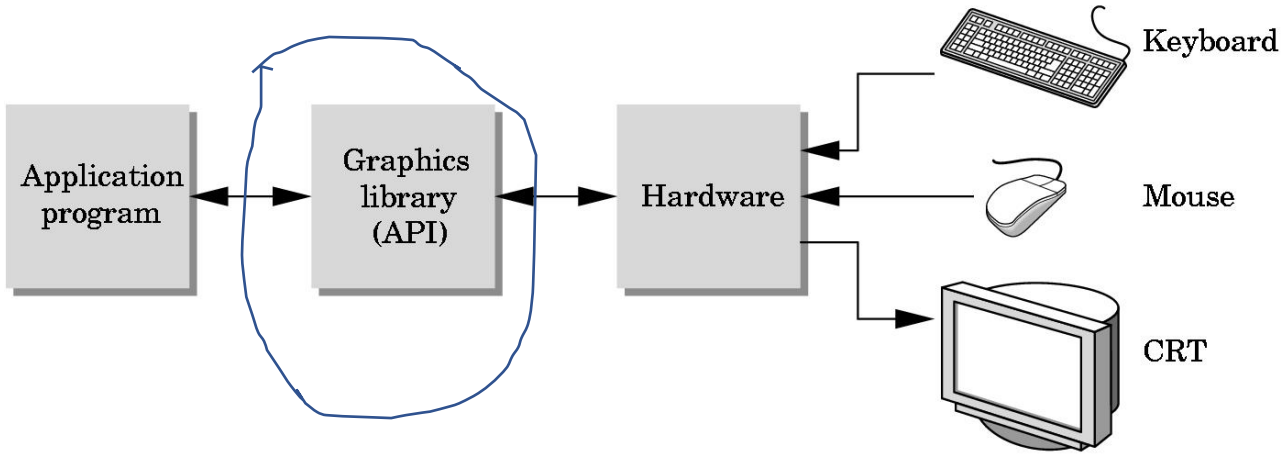
Outlines

- **Imaging Principle and models(图形处理所基于的模型)**
  - Elements of Image Formation-生成的要素
  - Synthetic camera Model –虚拟相机模型
  - Global lighting & Local lighting Model-全局/局部光照模型
- **Imaging Implement （图形处理的实现框架）**
  - Model-Render mode 建模渲染模式
  - Architecture (render pipeline) 可编程渲染管线
- **Imaging Implement （图形系统的使用-编程）**
  - CG API

The Programmer's Interface(API)

- **Programmer** sees the **graphics system** through **a software interface**:

the Application Programmer Interface (API)



API Contents：GL函数

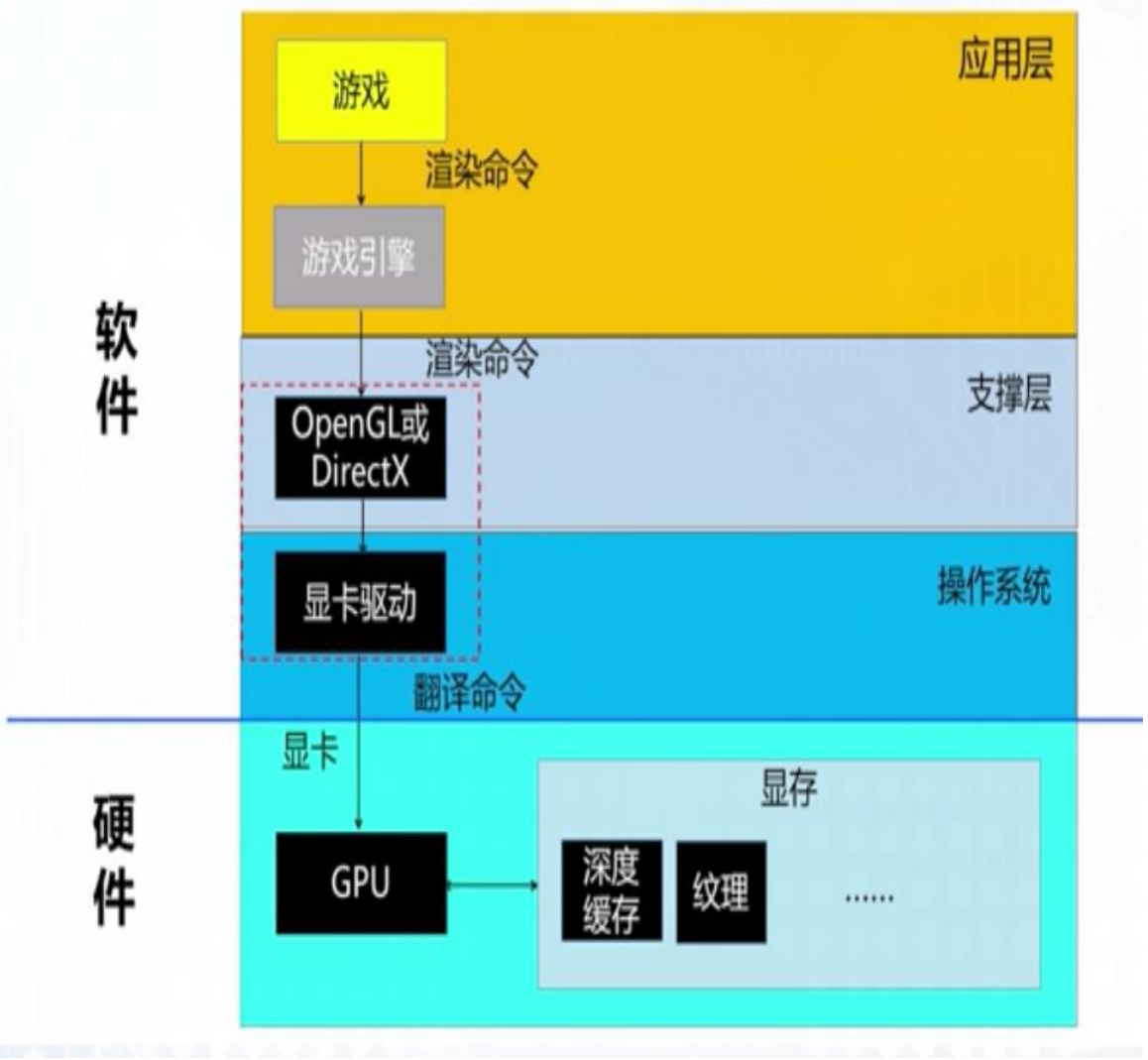
- **Functions that specify what we need to form an image**
  - Objects & Materials
  - Viewer
  - Light Sources
- **Other information**
  - Input from devices such as mouse and keyboard
  - Capabilities of system

- 图元类型和属性设置
- 绘制函数
- 变换函数
- 光源设置
- 相机设置
- 查询函数
- 其它交互控制函数

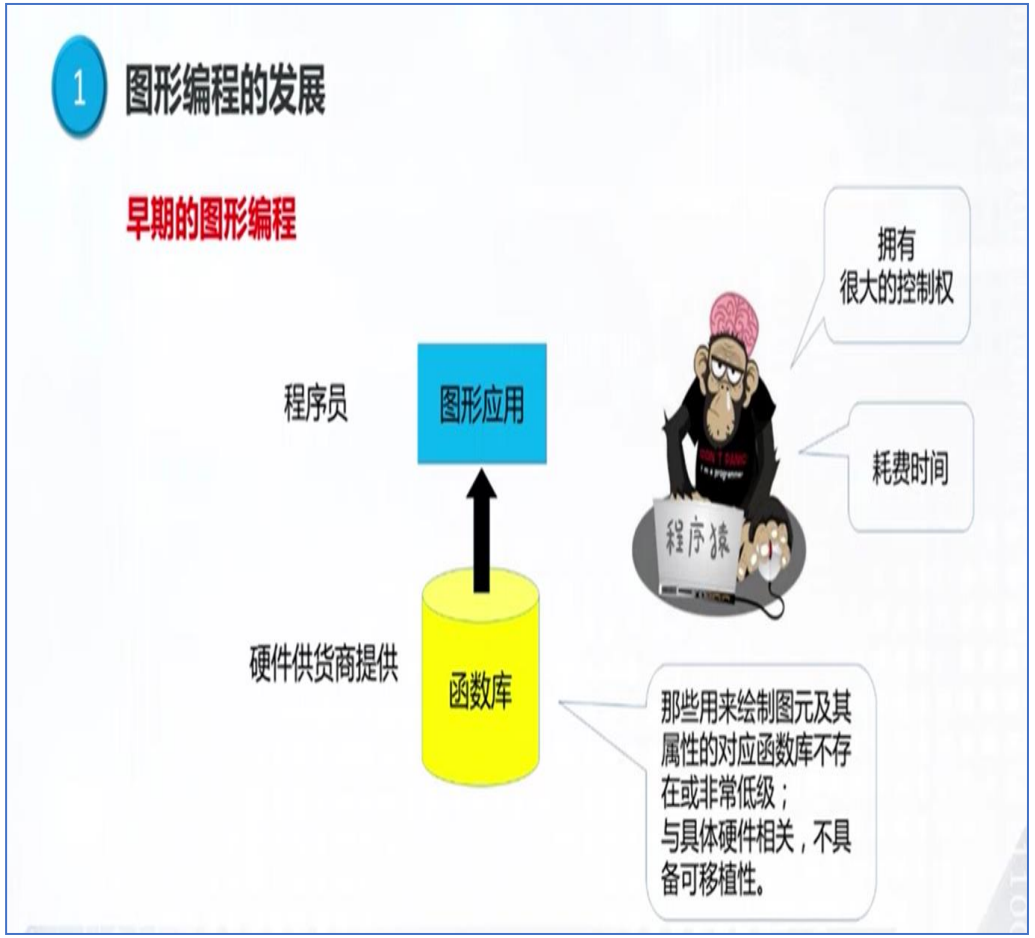
图形系统软件层次

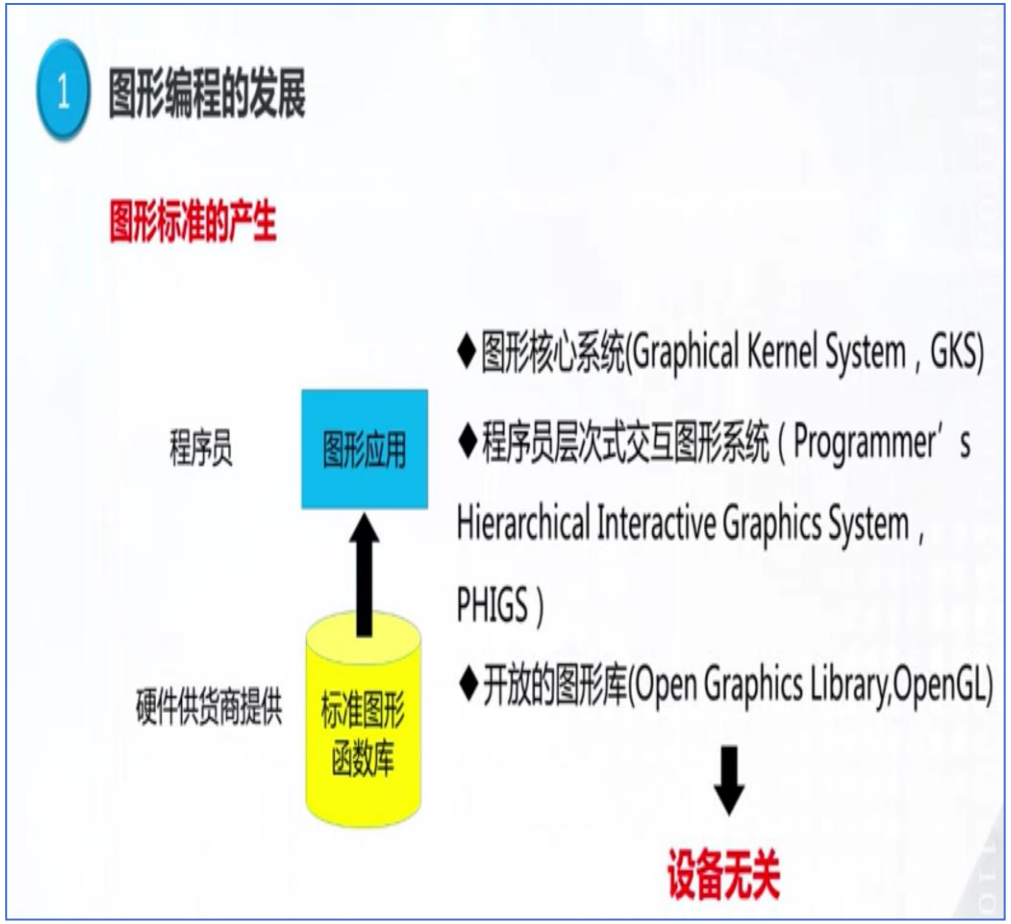
从一个游戏看图形软件的构成：

其他的问题：有没有用到游戏引擎？是基于OpenGL还是DirectX？



图形函数库GI(Graphics Library)标准化





从固定到可编程

固定管线

图形API提供了一个对硬件进行操作的标准接口；从内部实现上来说，API对程序员提出的各种绘制图元或属性的请求都采用固定的方式来处理。

这种内部实现方式通常称为**固定功能渲染流水线**。

固定到可编程

钩函数hooks的出现：突破固定功能流水线的限制，使用可编程着色器修改流水线中某些特定步骤的行为。

Review

1）图形系统及硬件组成（第1节课）

显示处理器processor：将顶点表示图形转换为帧缓存中的像素图

2）成像原理及图形系统的软件实现

成像原理：虚拟相机成像模型（计算位置）

全局光照/局部光照模型（计算颜色）

实现框架：渲染管线~局部光照管线框架（四步）

3）应用编程接口API: 提供标准图形库，包含主要函数功能有

定义描述物体顶点属性，绘制图元类型，

设置光源，相机，投影方式，裁剪窗口

系统设置，交互设备控制等

Render sketch map