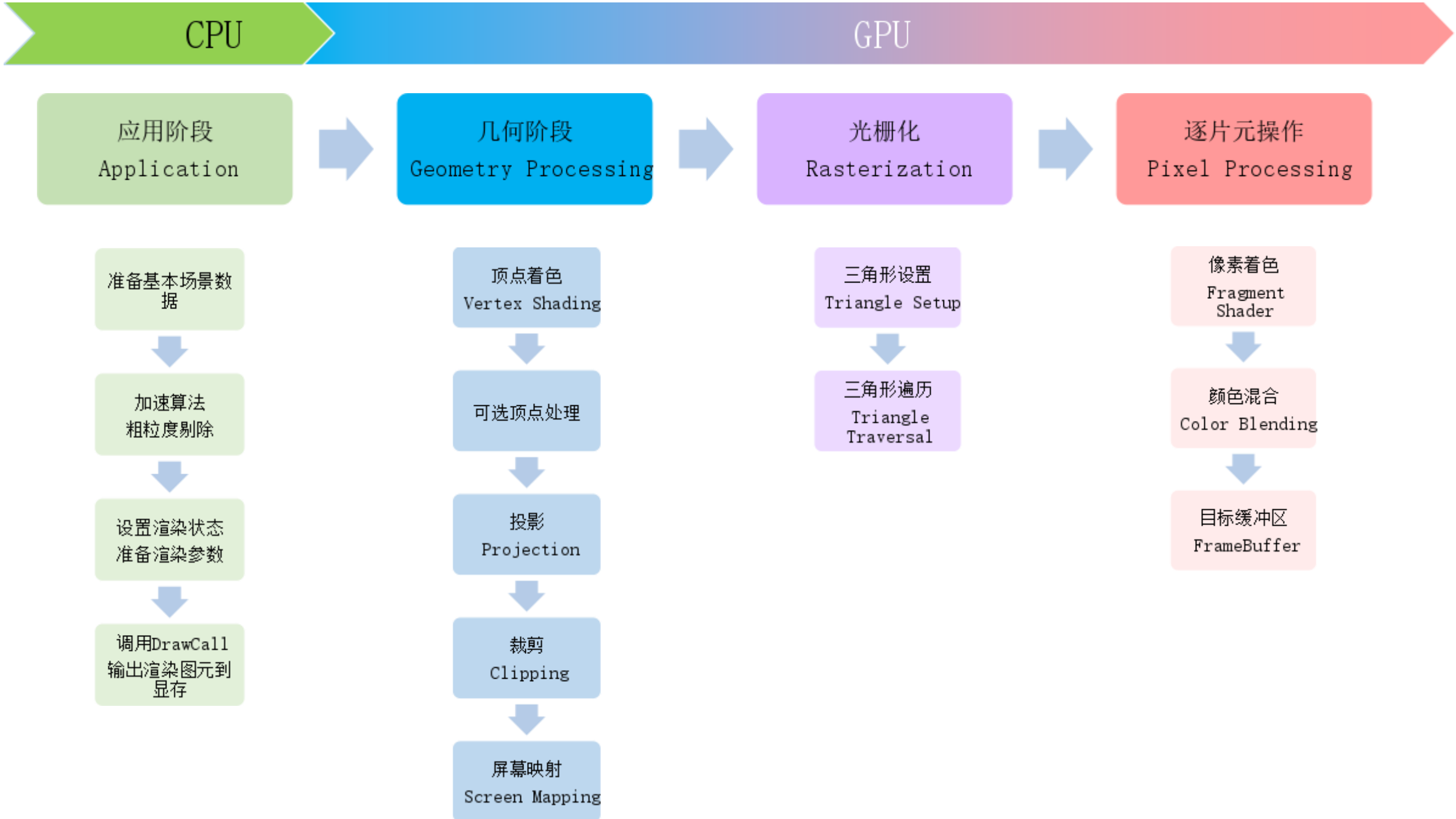
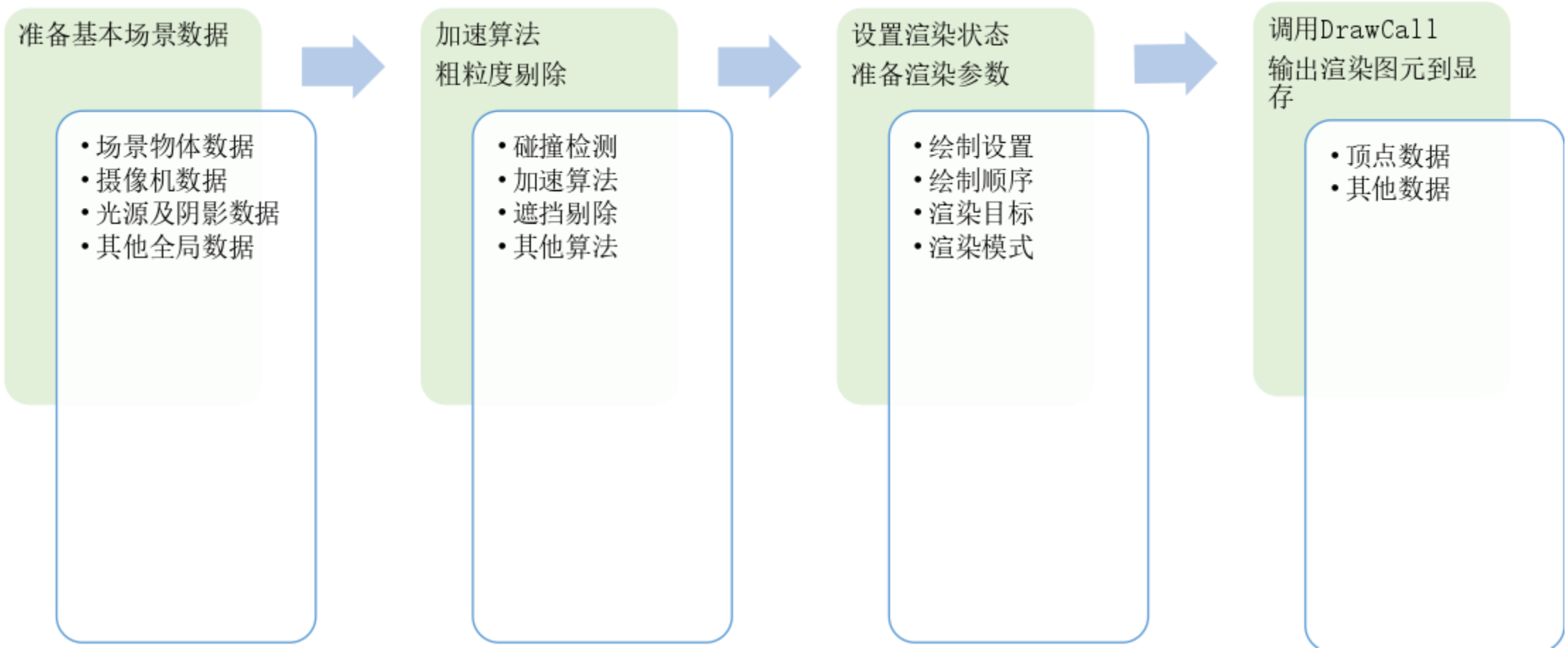
渲染管线简介

0. 整体流程

1. • 应用阶段：粗粒度剔除，进行渲染设置，准备基本数据，输出到几何阶段
2. • 几何阶段：顶点着色器，曲面细分，几何着色器，顶点裁剪，屏幕映射
3. • 光栅化阶段：三角形（点/线）设置，三角形（点/线）遍历，片段着色器
4. • 逐片元操作：裁剪测试，透明度测试，深度测试，模板测试，混合
9. • 后处理


CPU
GPU
应用阶段
Application
几何阶段
Geometry Processing
光栅化
Rasterization
逐片元操作
Pixel Processing
准备基本场景数据
加速算法
粗粒度剔除
设置渲染状态
准备渲染参数
调用DrawCall
输出渲染图元到显存
顶点着色
Vertex Shading
可选顶点处理
投影
Projection
裁剪
Clipping
屏幕映射
Screen Mapping
三角形设置
Triangle Setup
三角形遍历
Triangle Traversal
像素着色
Fragment Shader
颜色混合
Color Blending
目标缓冲区
FrameBuffer

1.应用阶段


准备基本场景数据
• 场景物体数据
• 摄像机数据
• 光源及阴影数据
• 其他全局数据
加速算法
粗粒度剔除
• 碰撞检测
• 加速算法
• 遮挡剔除
• 其他算法
设置渲染状态
准备渲染参数
• 绘制设置
• 绘制顺序
• 渲染目标
• 渲染模式
调用DrawCall
输出渲染图元到显存
• 顶点数据
• 其他数据

1.1 基本场景数据

场景物体数据	光源信息：	摄像机参数：
• 物体变换数据：位置、旋转、缩放等 • 物体网格数据：顶点位置、UV贴图等	• 光源类型：方向光、点光、聚光等 • 位置、方向、角度等其他参数	• 位置、方向、 • 远近裁剪平面 • 正交/透视(FOV) • 视口比例/尺寸等

1.1 基本场景数据

可见光 裁剪	可见场景物体 裁剪
	• 八叉树 • K-D • BVH

1.1 光源和阴影

设置光源

- 方向光：颜色、方向等
- 点光源：颜色、位置、范围等
- 聚光源：颜色、位置、方向、内外圆锥角等

设置阴影

- 是否需要阴影：判断该光源可见范围内是否有可投射阴影的物体
- 阴影参数：对应光源序号、阴影强度、级联参数、深度偏移、近平面偏移等

逐光源绘制阴影贴图：

- 近平面偏移
- 逐级联
  - 计算当前光源+级联对应的观察矩阵、投影矩阵、以及对应到阴影贴图里的视口区域
  - 绘制到阴影贴图

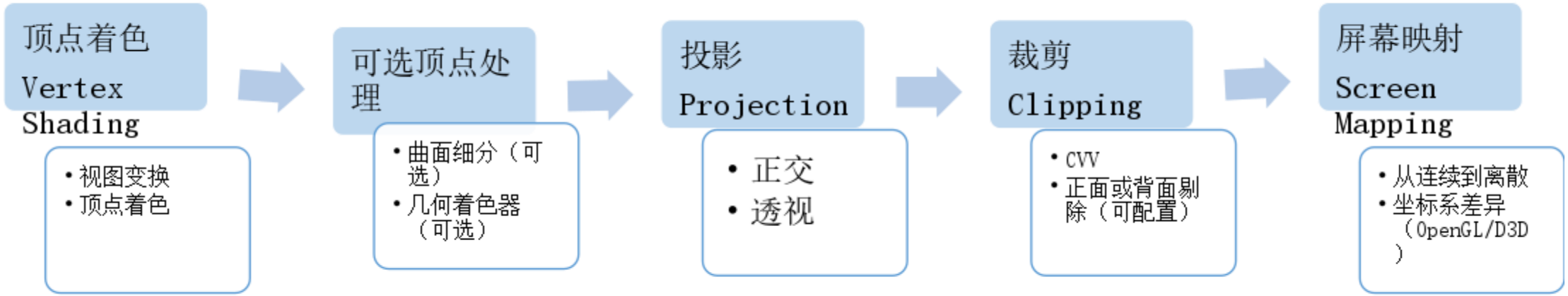
1.3 渲染设置

绘制设置	绘制物体的顺序（可以有多种方式）	渲染目标	渲染模式
• 合批方式	• 相对摄像机的距离 • 材质RenderQueue • UICanvas • 其他方式等	• FrameBuffer • RenderTexture	• 前向渲染 • 延迟渲染

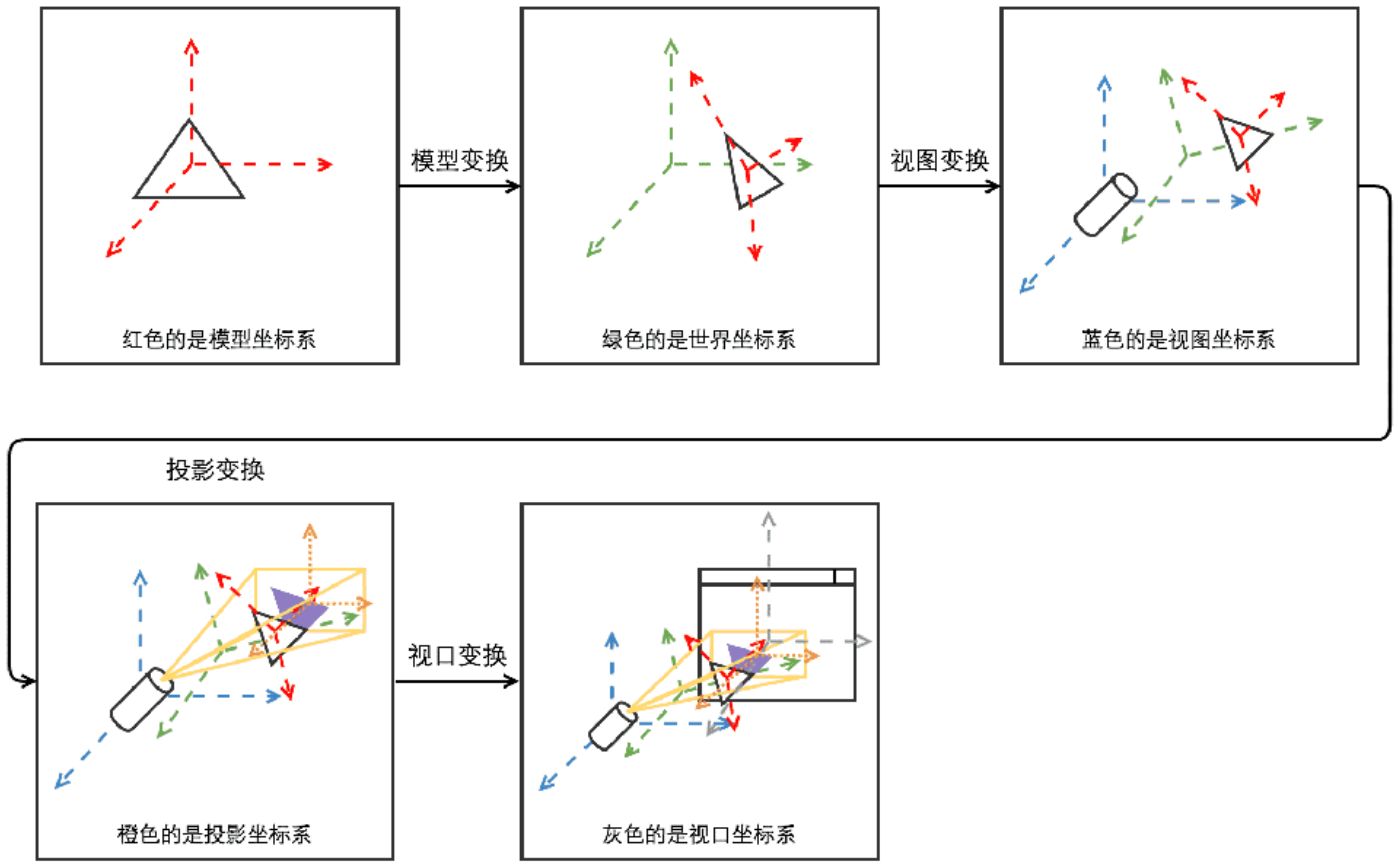
1.4 输出到显存

顶点数据	其他数据
• 位置 • 颜色 • 法线 • 纹理uv坐标 • 其他顶点数据	• MVP变换矩阵 • 纹理贴图 • 其他数据

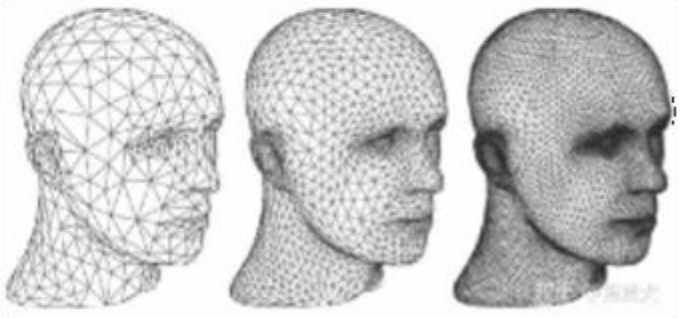
2.几何阶段


顶点着色
Vertex Shading
•视图变换
•顶点着色
可选顶点处理
•曲面细分（可选）
•几何着色器（可选）
投影
Projection
• 正交
• 透视
裁剪
Clipping
•CVV
•正面或背面剔除（可配置）
屏幕映射
Screen Mapping
•从连续到离散
•坐标系差异（OpenGL/D3D）

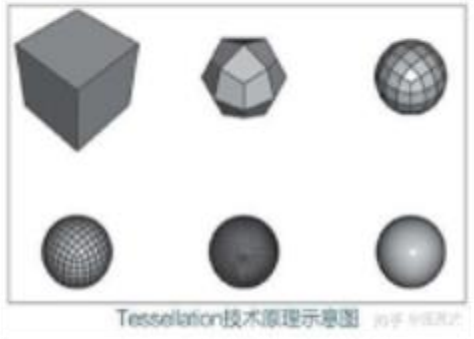
2.1 顶点着色器-视图变换



2.2 曲面细分



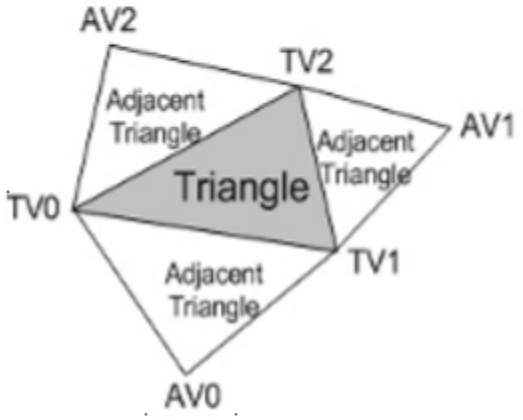
A large primitive change into a bunch of smaller primitives



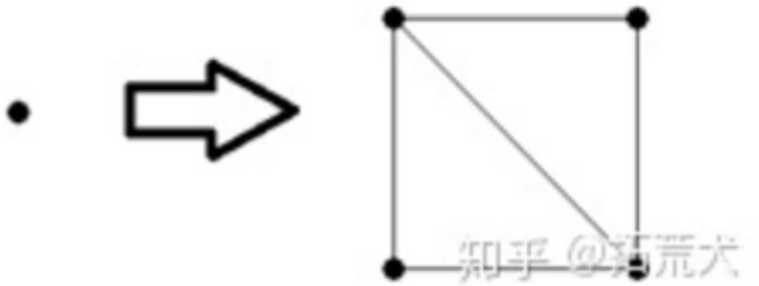
将一个正方体通过不断添加Vertex的方式，来形成一个光滑的球体

Tessellation Pipeline

2.2 几何着色器（基于图元的操作）

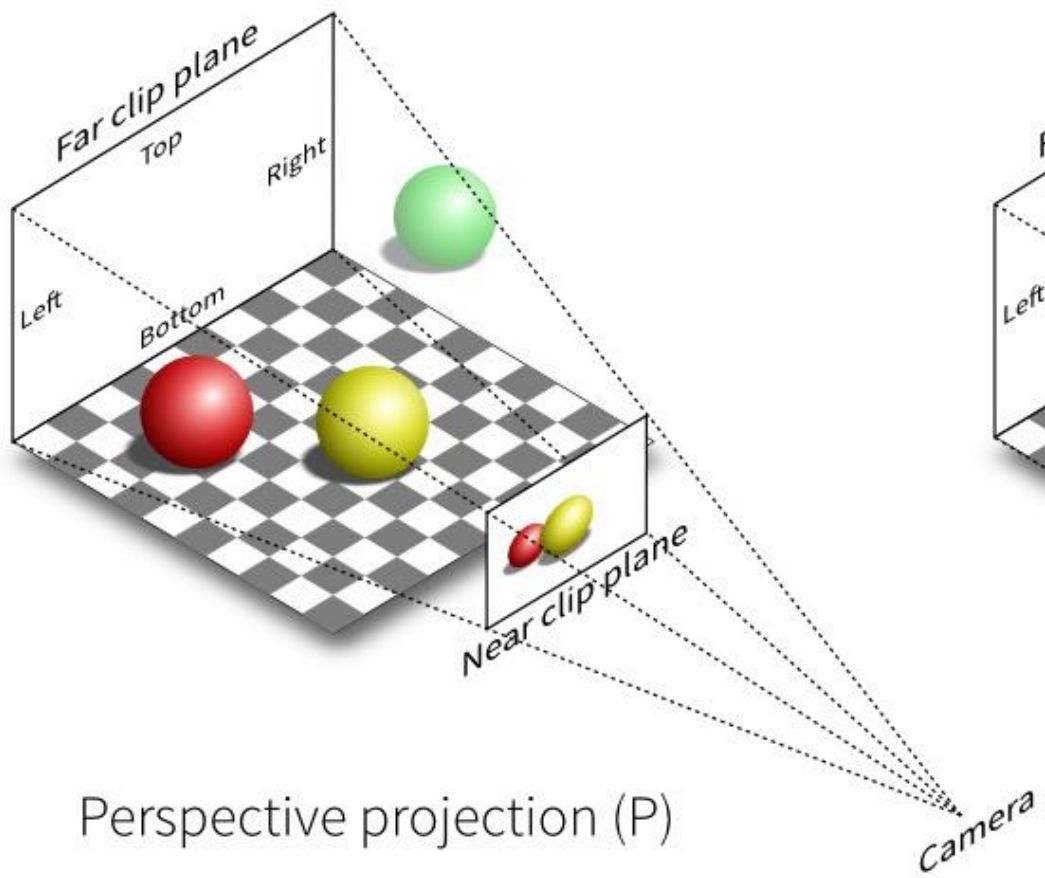


在图元外添加额外的Vertex，将原始图元转换成新图元，以构建一个不一样的模型

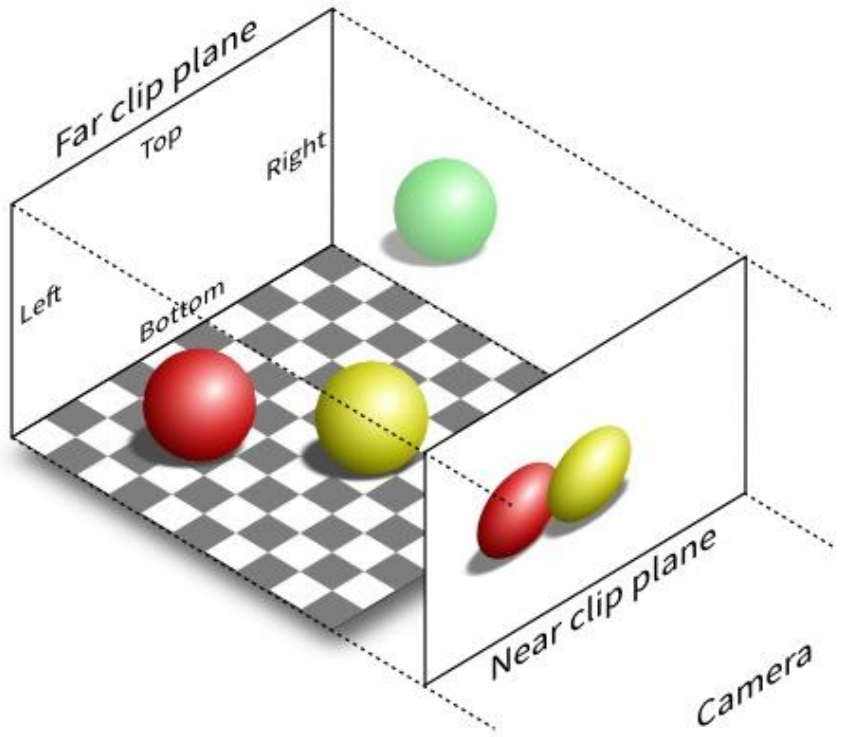


只用一个Vertex来表示每一个颗粒，只需要在Geometry Shader阶段将每一个Vertex拓展成两个三角形

2.3 投影

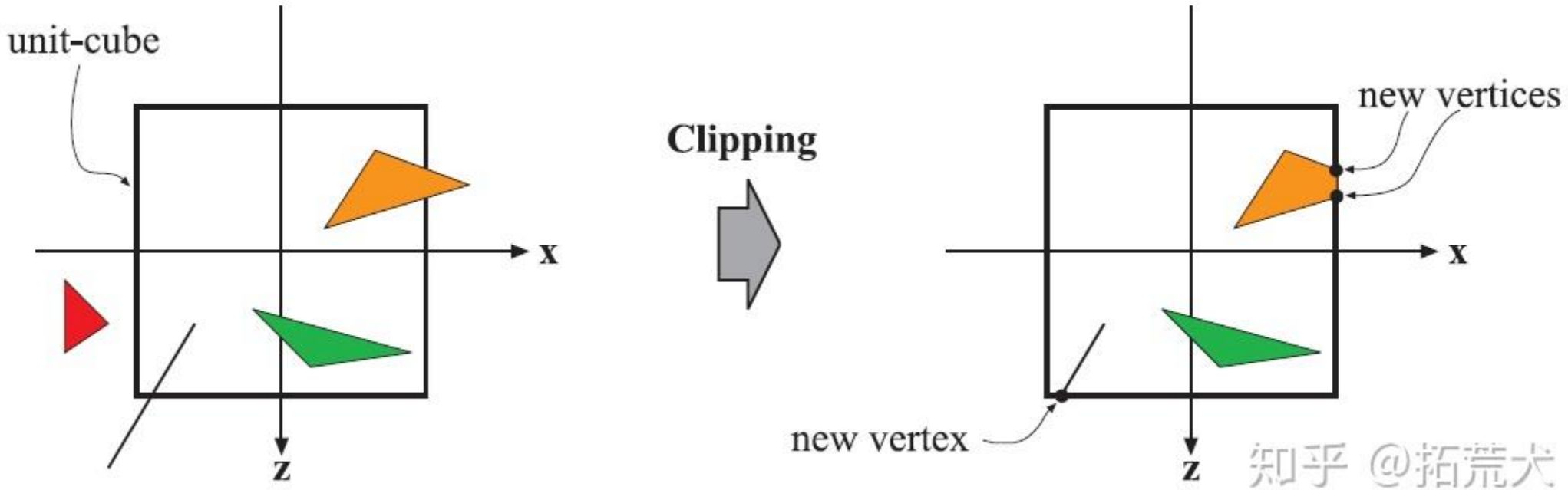


Perspective projection (P)

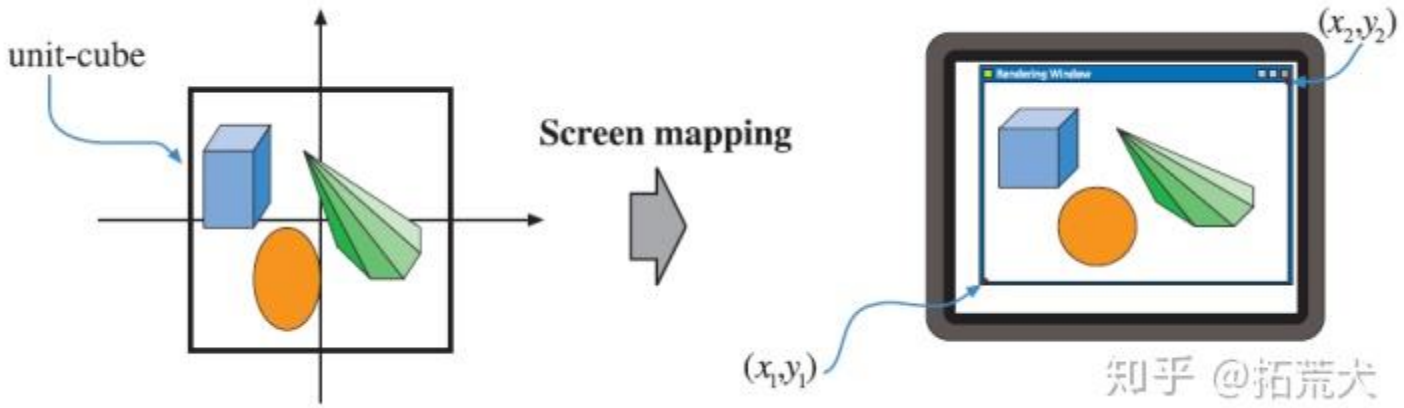


Orthographic projection (O)

2.4 裁剪



2.5 屏幕映射

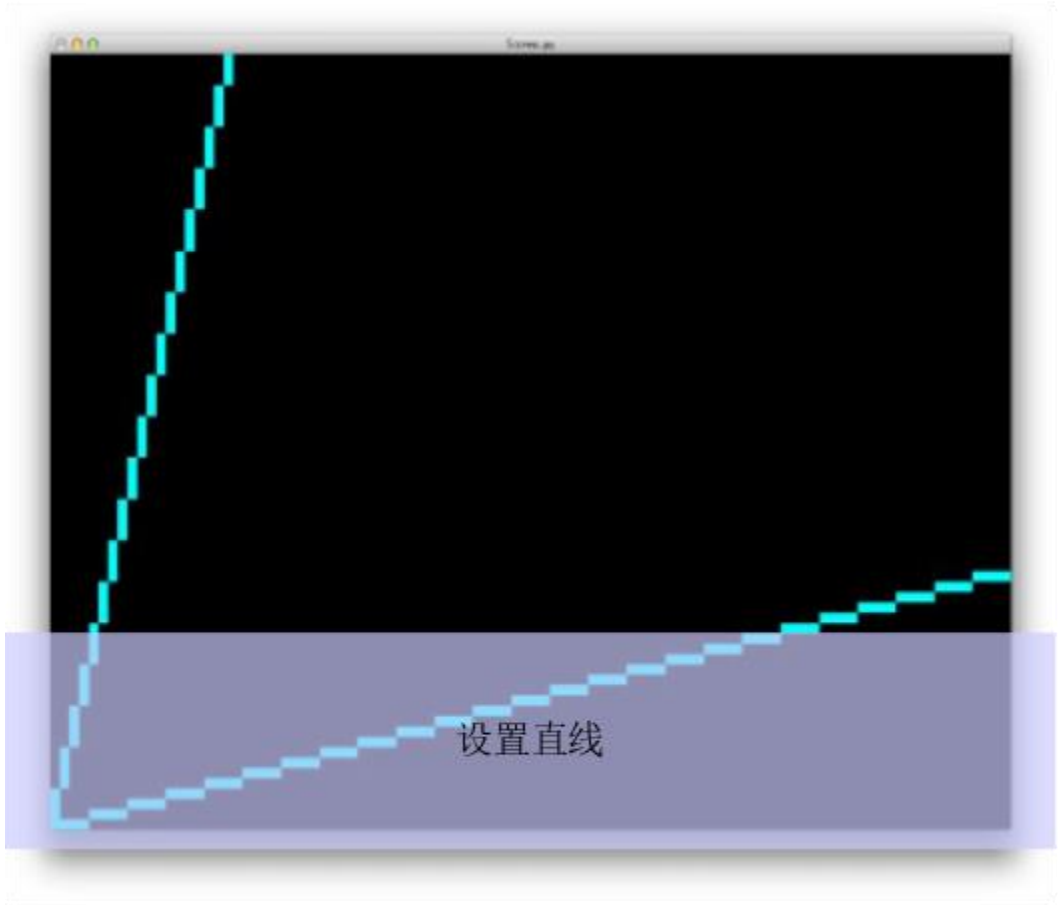


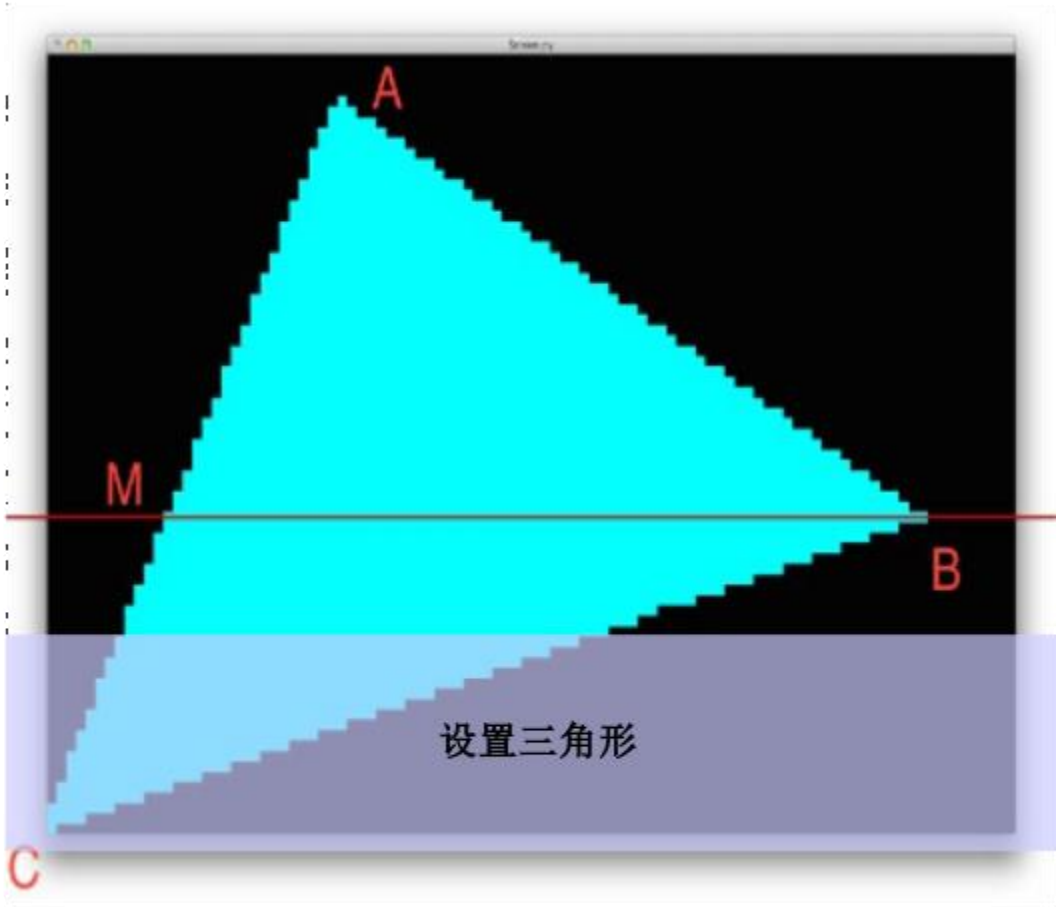
3. 光栅化阶段

三角形设置
Triangle Setup

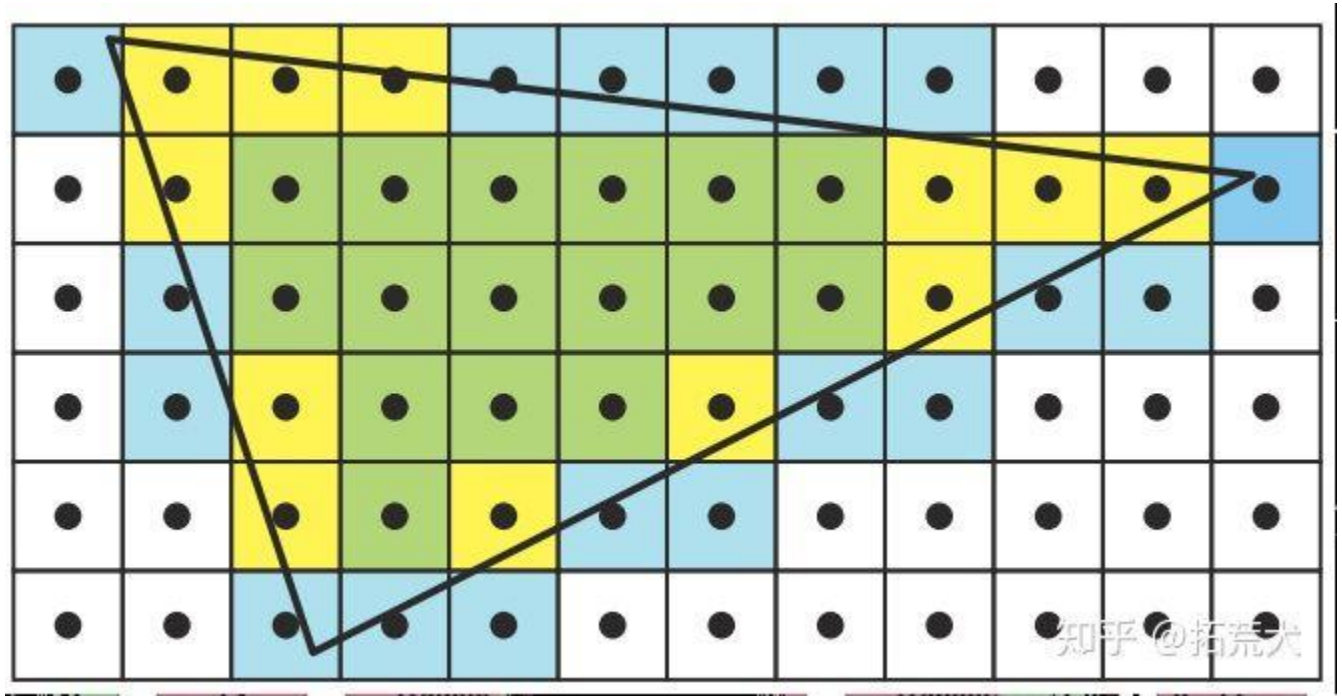
三角形遍历
Triangle Traversal

3.1 三角形设置



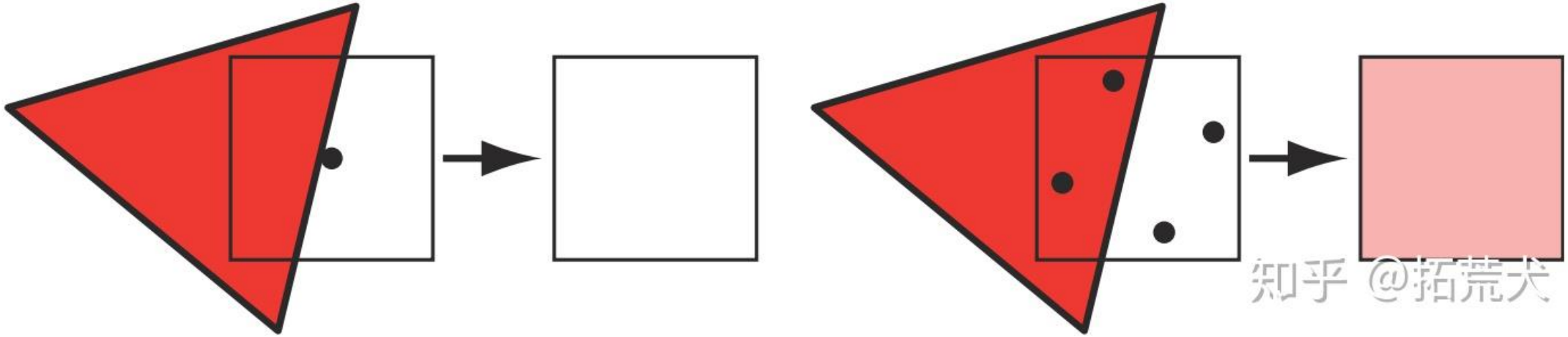


3.2 三角形遍历



3.2 抗锯齿(MSAA)

SSAA	MSAA	FXAA / TXAA
• 渲染到一个分辨率放大n倍的buffer • 对放大n倍的buffer下采样	• 在光栅化阶段 • 计算多个覆盖样本	• 后处理技术，不在这个渲染阶段



4. 逐片元操作

像素着色

Fragment Shader

颜色混合

Color Blending

- Alpha Test
- Depth Buffer Test
- Stencil Test
- Blending

目标缓冲区

- FrameBuffer
- RenderTexture

Opengl per-fragment spec

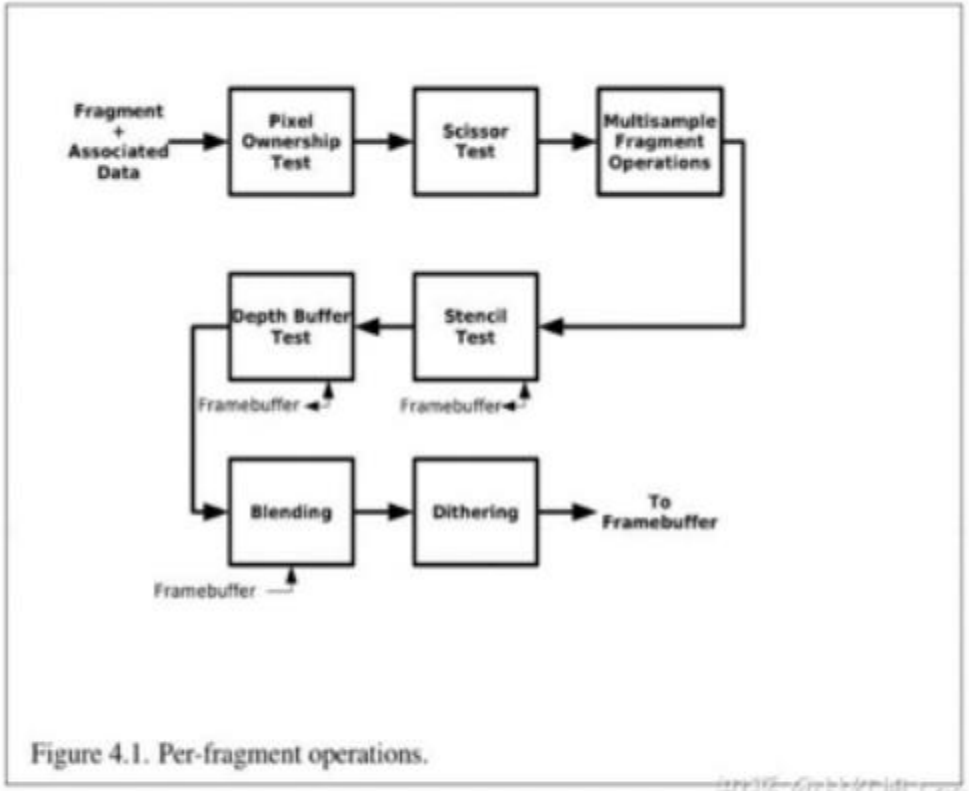
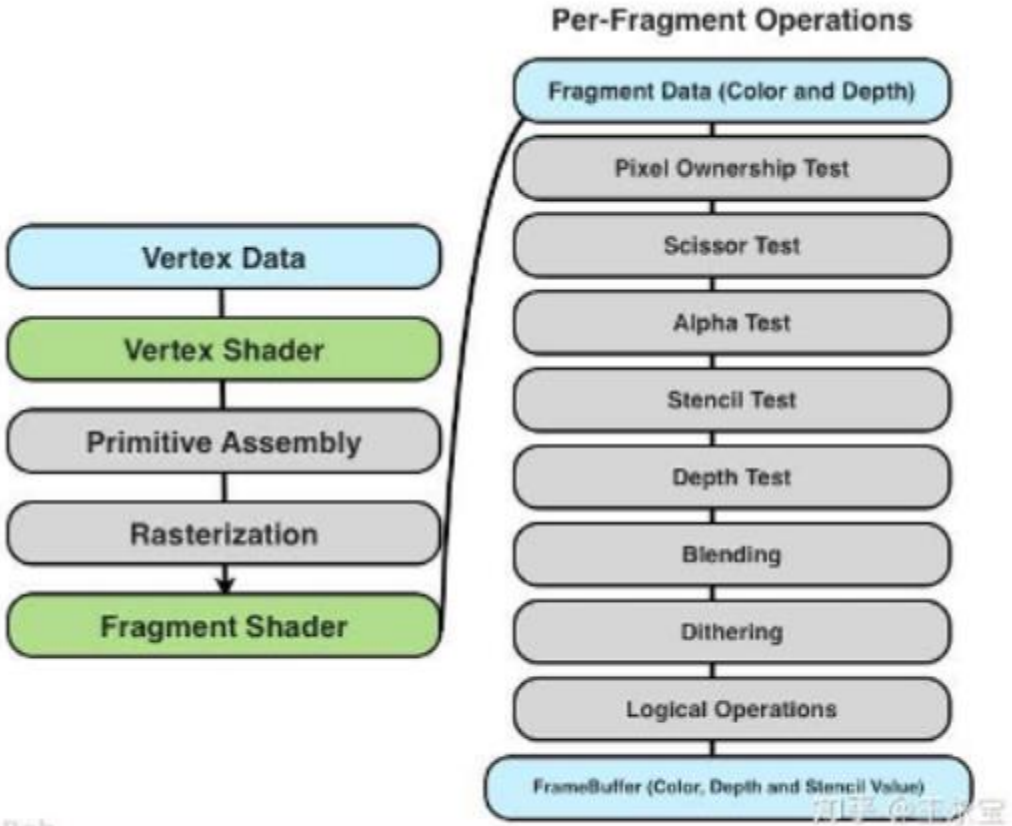


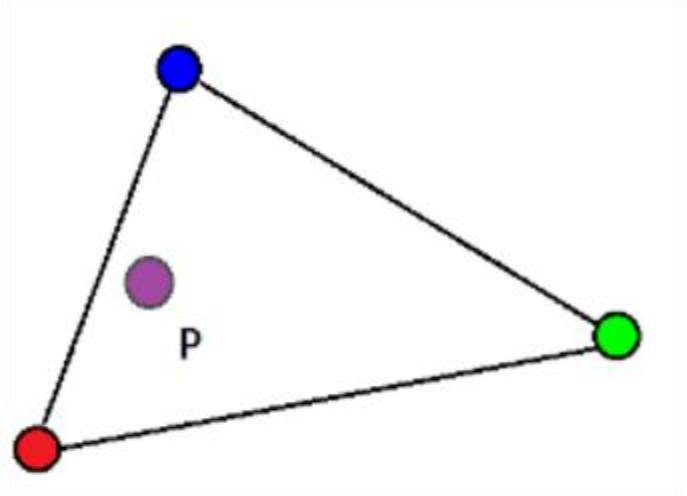
Figure 4.1. Per-fragment operations.

深度测试/模版测试/透明度测试先后顺序是什么样的？ - 王永宝的回答 - 知乎

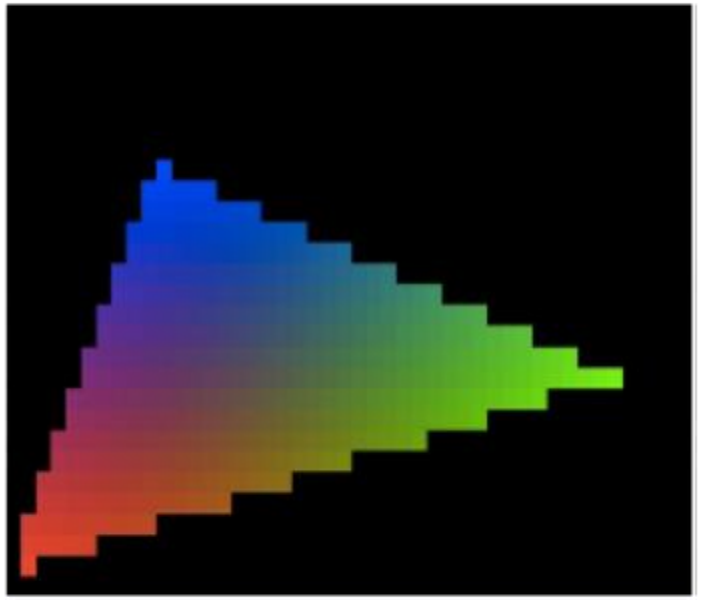


深度测试/模版测试/透明度测试先后顺序是什么样的？ - 林红旭Leo的回答 - 知乎

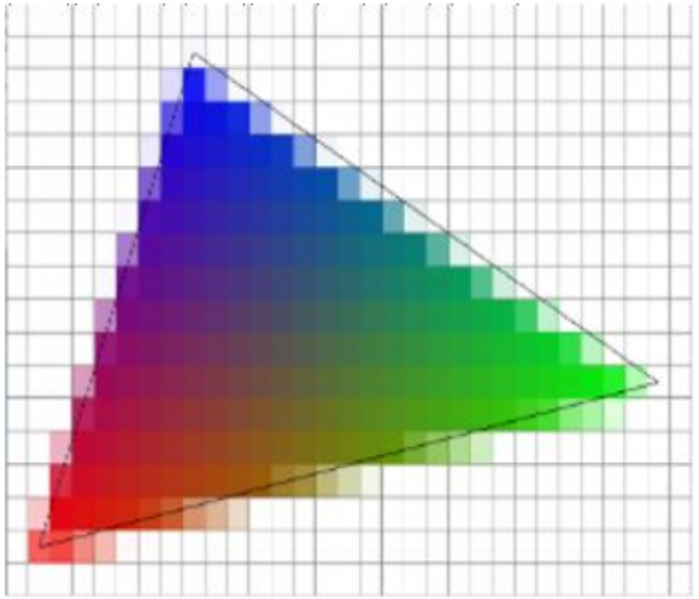
4.1 像素着色



P点颜色=3个顶点颜色做插值

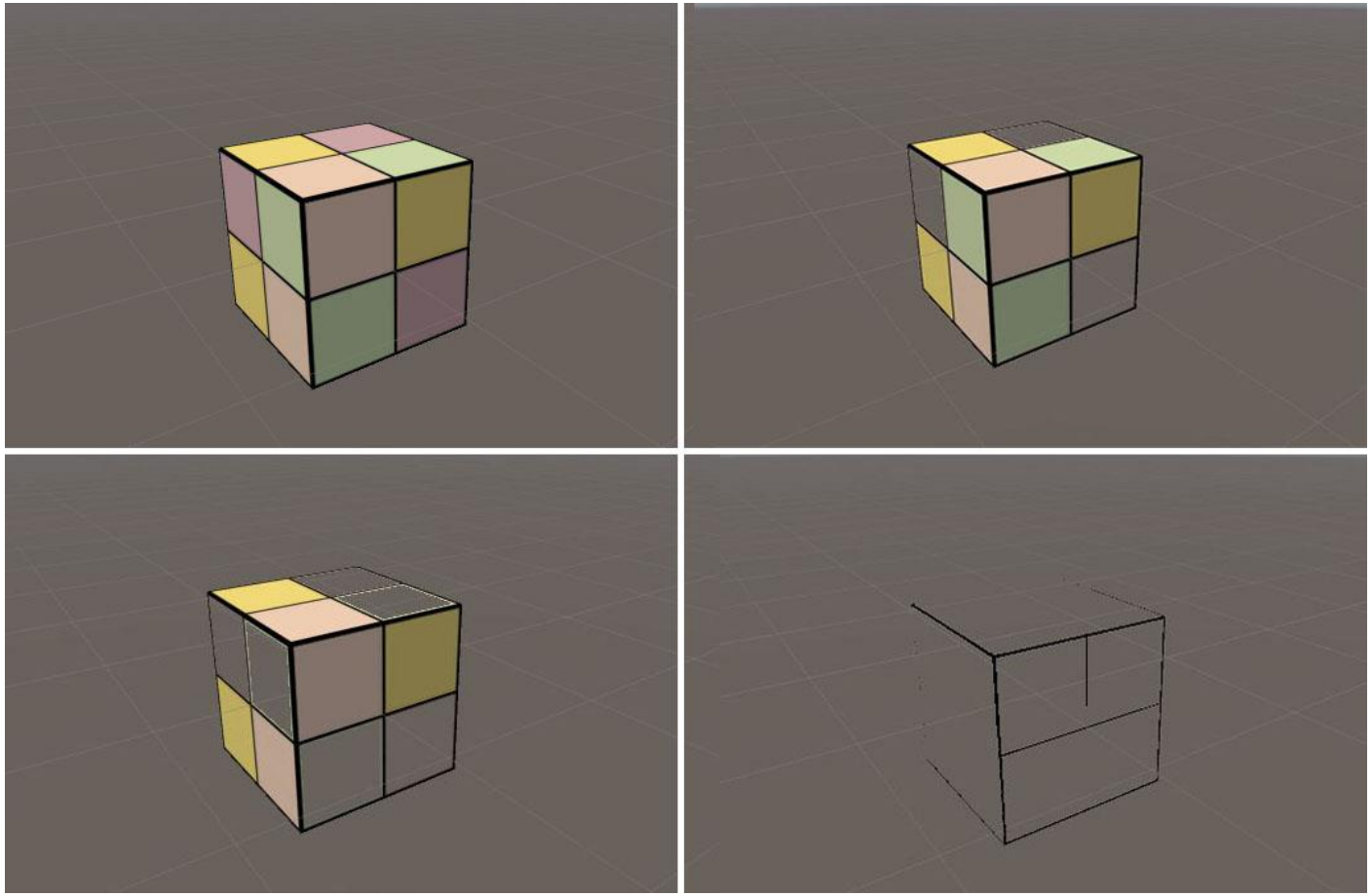


着色结果

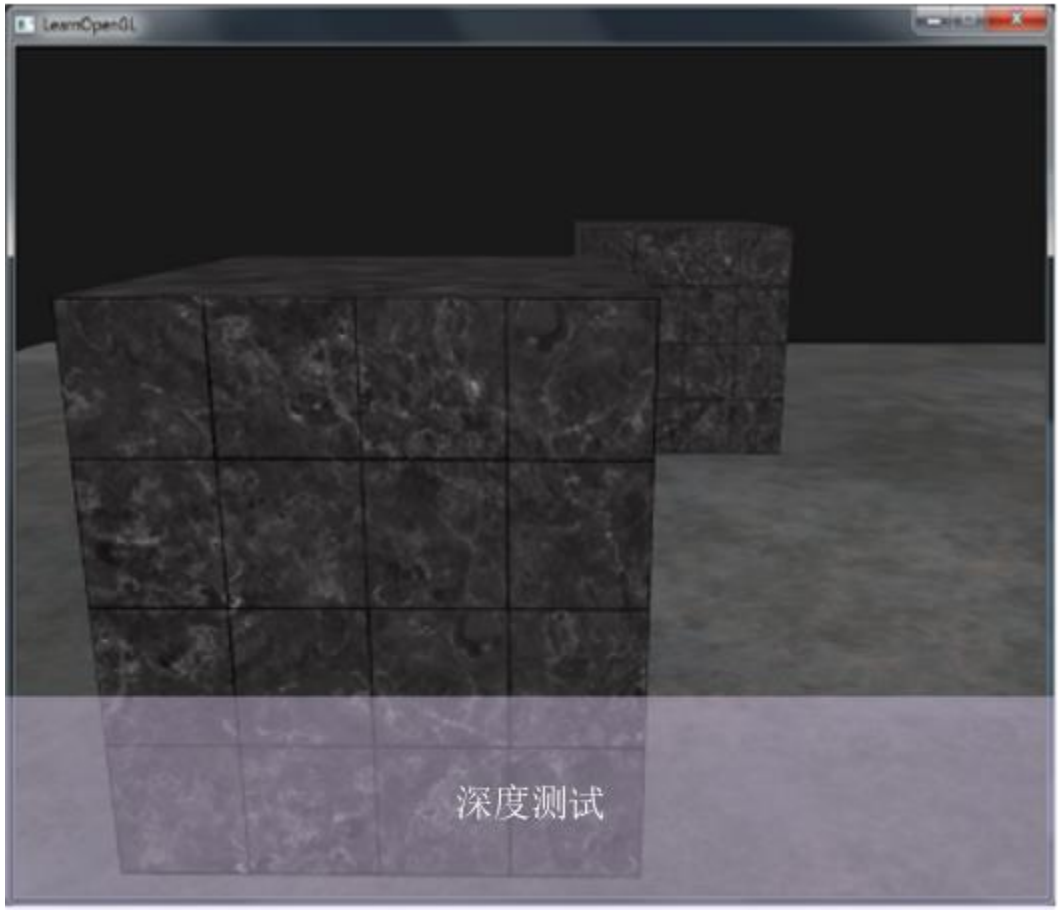


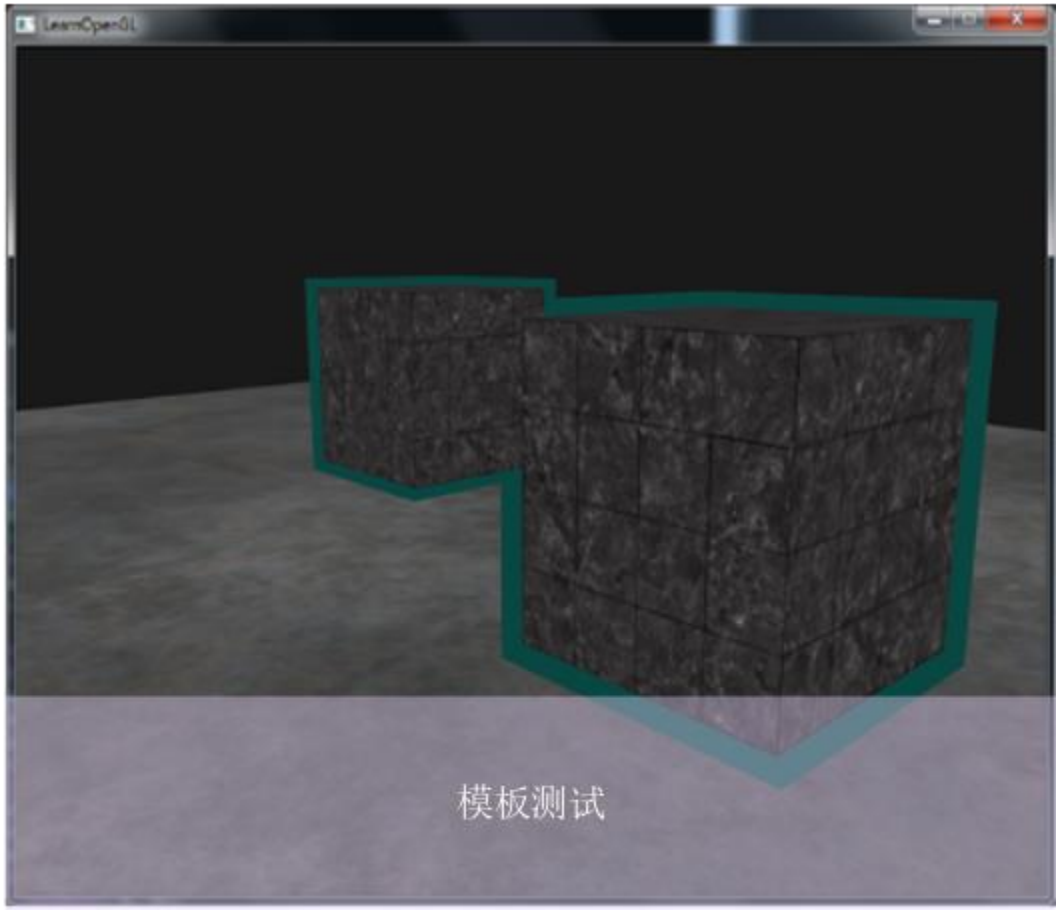
使用抗锯齿后的着色结果

4.2 颜色混合 – 透明度测试

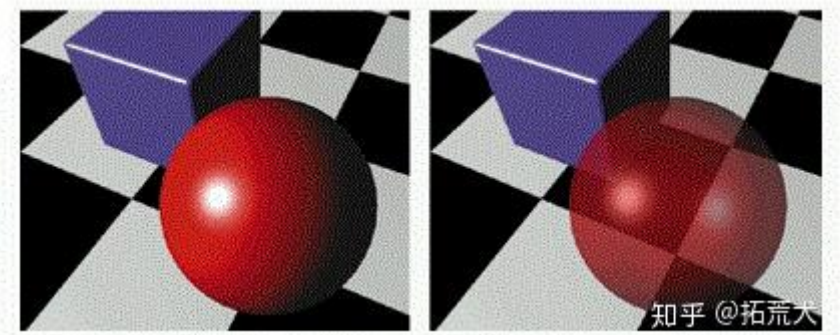


4.2 颜色混合 – 深度测试/模板测试

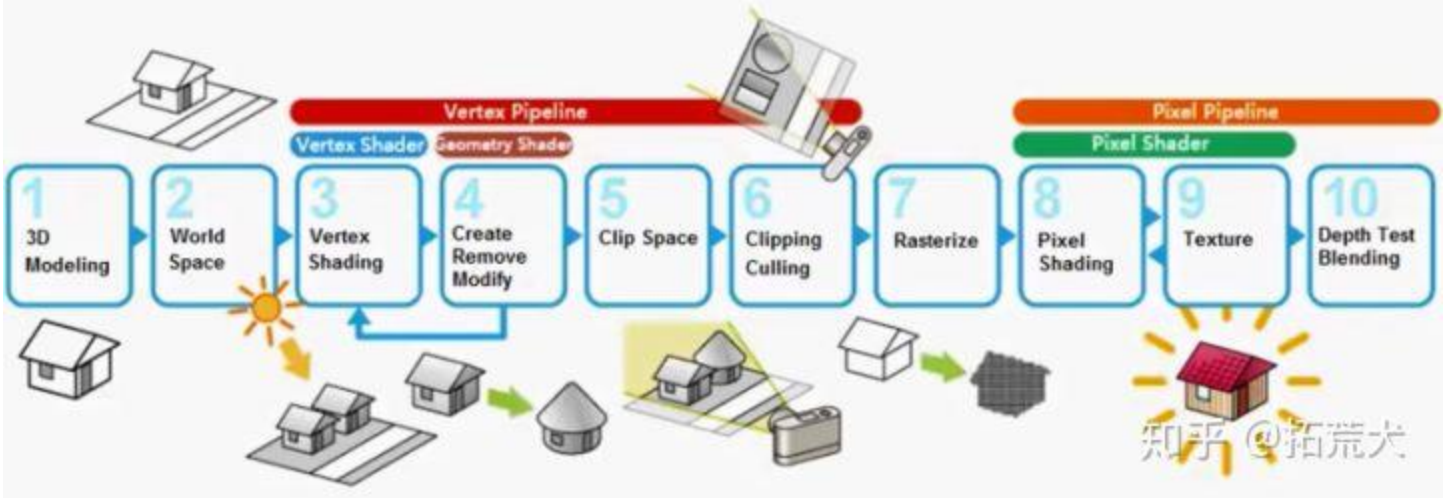




4.2 混合



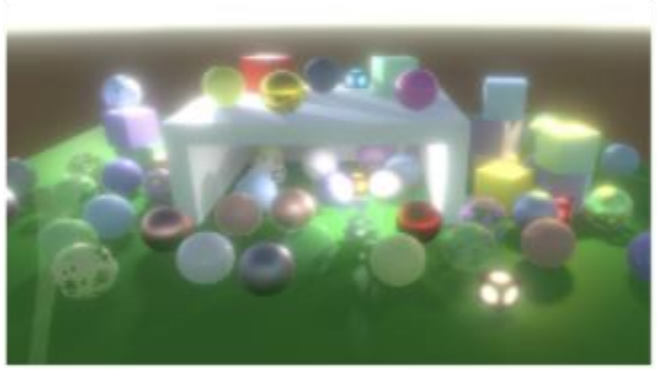
回顾 1-4 整体流程的一个例子



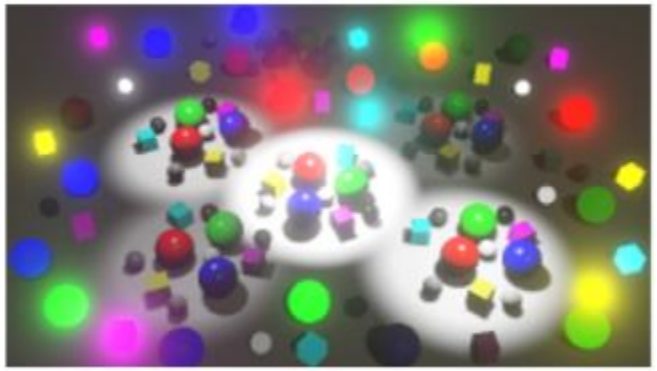
5.后处理

前面的渲染流程完成后，再对最后输出的缓冲区/渲染贴图进行处理，可以看成是应用于一个矩形面片贴图的图像处理。

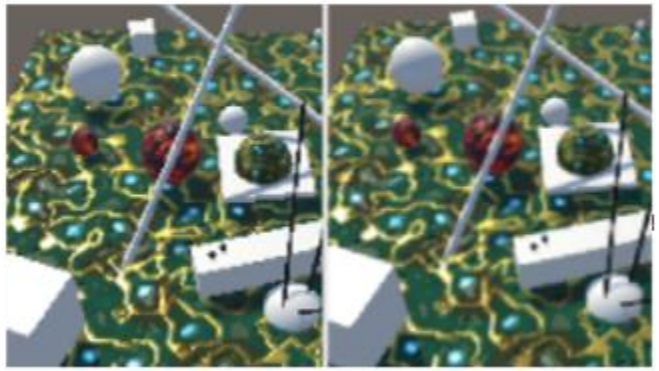
Bloom



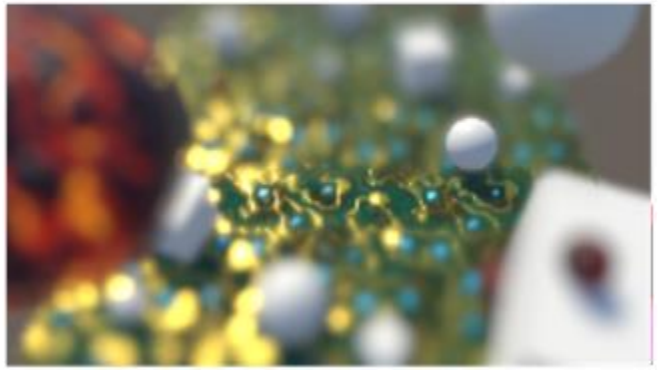
HDR



FXAA（原图/处理后）



Depth of View



边缘检测

径向模糊

引用

- RTR4 第二章 图形渲染管线 - 0110君的文章 - 知乎 https://zhuanlan.zhihu.com/p/208987296
- GPU Rendering Pipeline——GPU渲染流水线简介 - 拓荒犬的文章 - 知乎https://zhuanlan.zhihu.com/p/61949898
- 卡通渲染（上）：致从没看懂过着色器代码的你 - 羡辙的文章 - 知乎 https://zhuanlan.zhihu.com/p/25595069
- 实时渲染中的坐标系变换（5）：投影变换-3 - IgorKakarote的文章 - 知乎 https://zhuanlan.zhihu.com/p/115395322
- 实时渲染中的坐标系变换（3）：投影变换-1 - IgorKakarote的文章 - 知乎 https://zhuanlan.zhihu.com/p/113662566
- GLSL-几何着色器详解跟实例 https://www.cnblogs.com/mazhenyu/p/3831986.html
- https://docs.microsoft.com/zh-cn/windows-hardware/drivers/display/geometry-shader-stage
- 渲染器 2 —— 三角形的光栅化 - 萧井陌的文章 - 知乎 https://zhuanlan.zhihu.com/p/20148016
- 深度测试/模版测试/透明度测试先后顺序是什么样的？ - 林红旭 Leo的回答 - 知乎 https://www.zhihu.com/question/384124671/answer/1121443495
- https://learnopengl-cn.github.io/