A CG Basic Knowledge Talk

@ IBM & 缘木轩 Club, SYSU

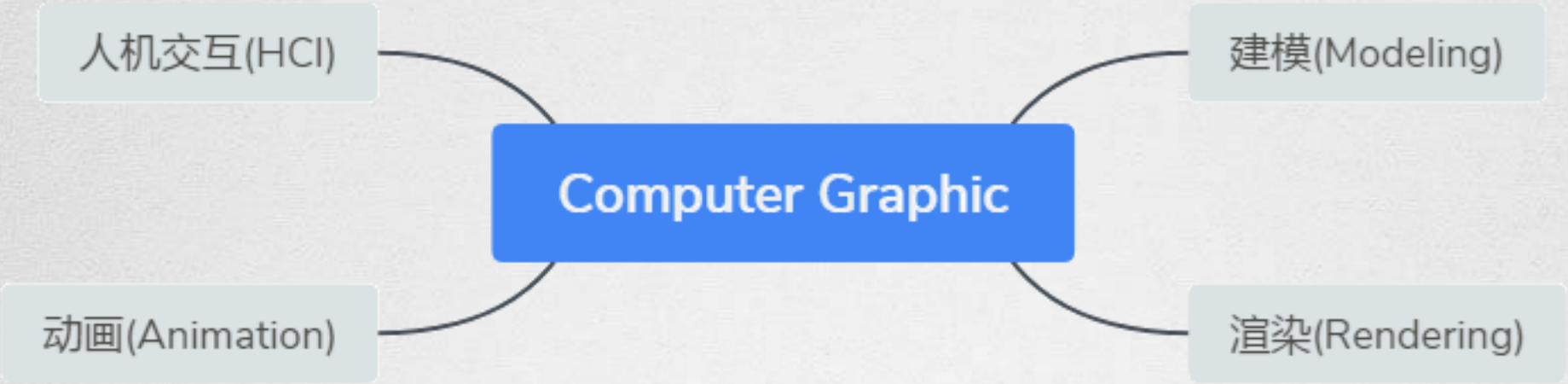
ABOUT ME

Johnny Law

- 15级软件工程（数字媒体）
- 中山大学 IBM 俱乐部技术主席

- luojj26@mail2.sysu.edu.cn
- www.longjj.com
- https://github.com/longjj

计算机图形学是什么?



研究计算机在硬件和软件的帮助下创建计算机图形的科学学科。

- OpenGL? DirectX? …

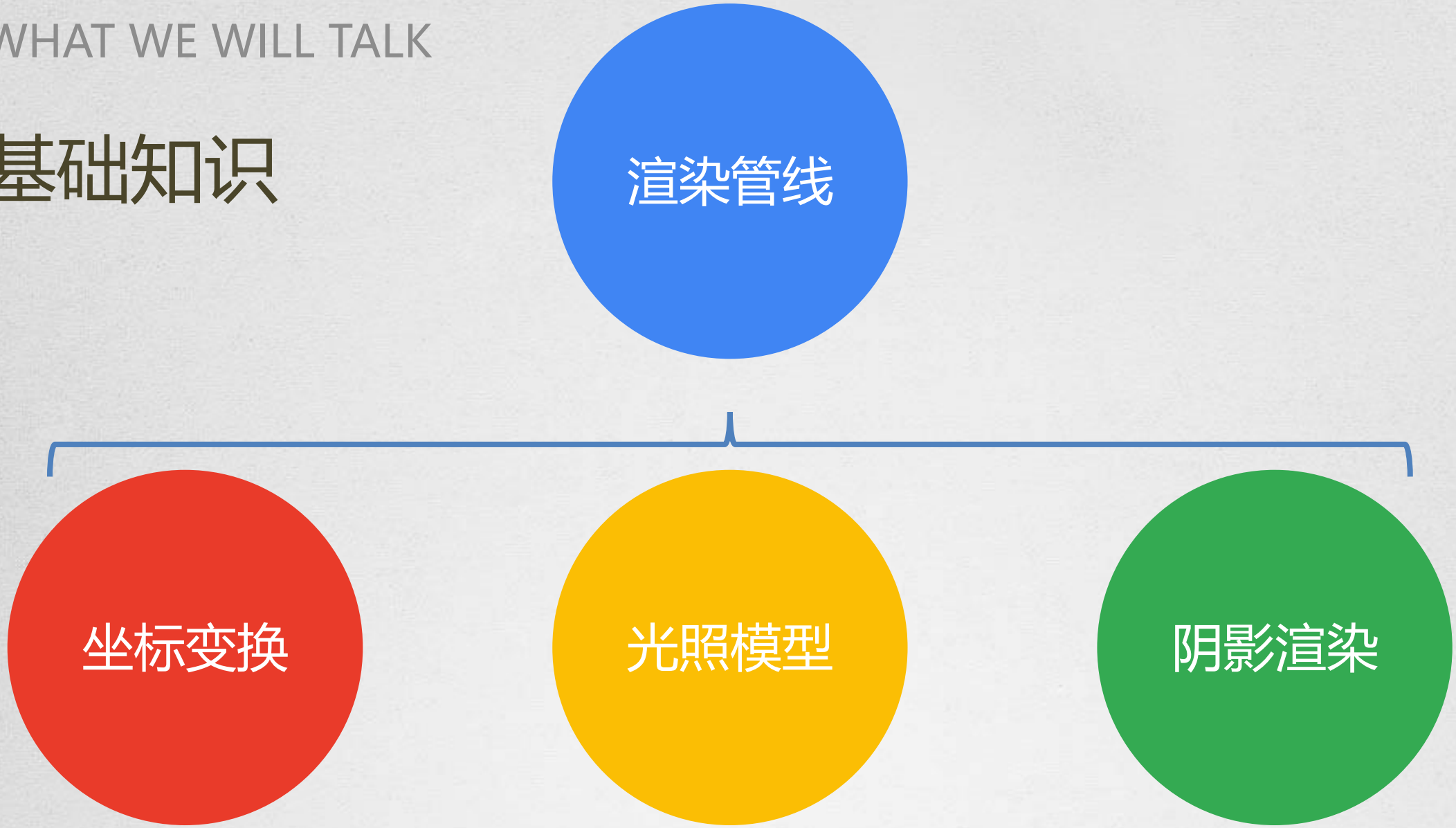
.
.
.

API调用

- OpenGL
- DirectX (Windows)
- Vulkan
- …

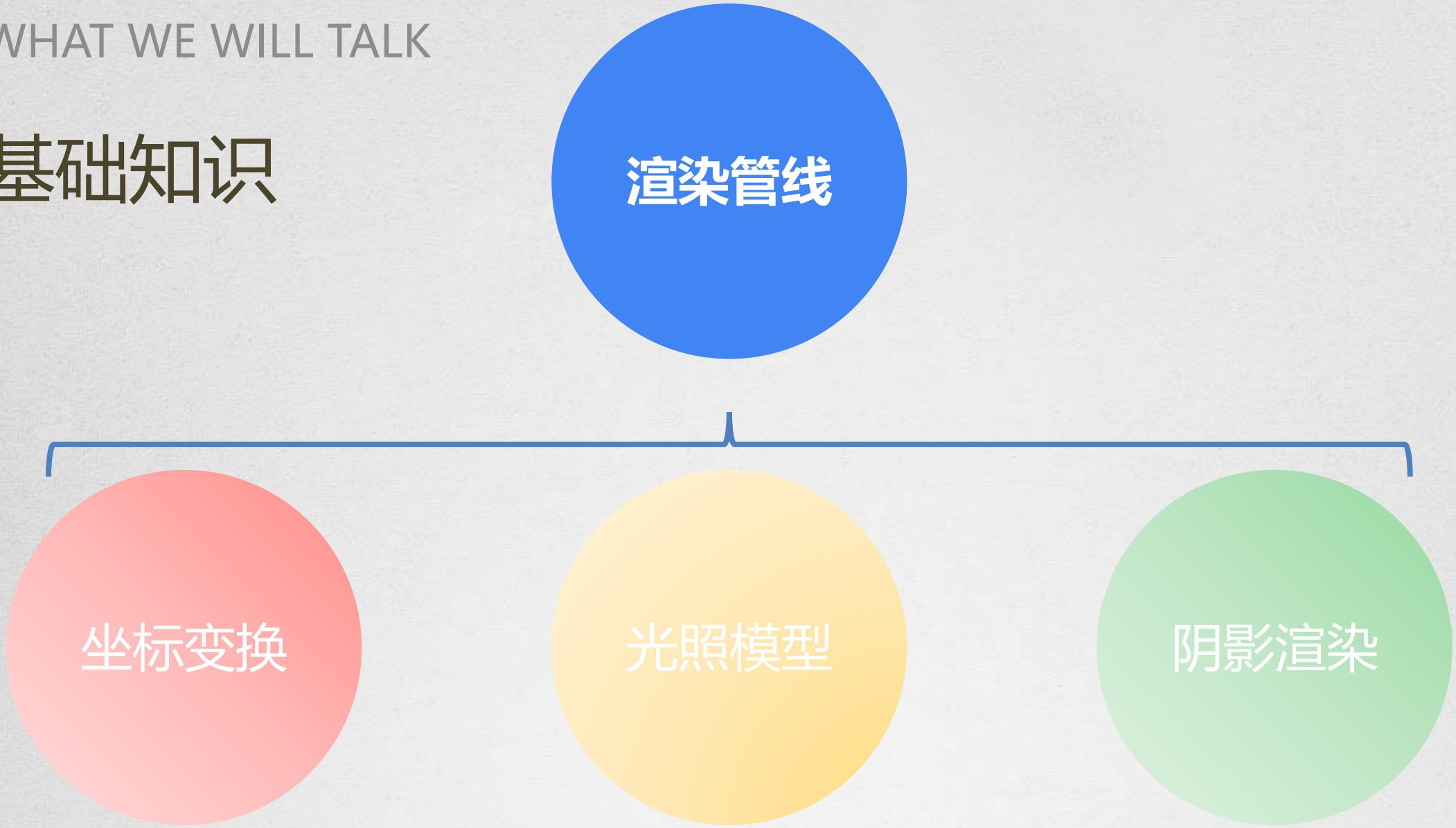
WHAT WE WILL TALK

基础知识



WHAT WE WILL TALK

基础知识



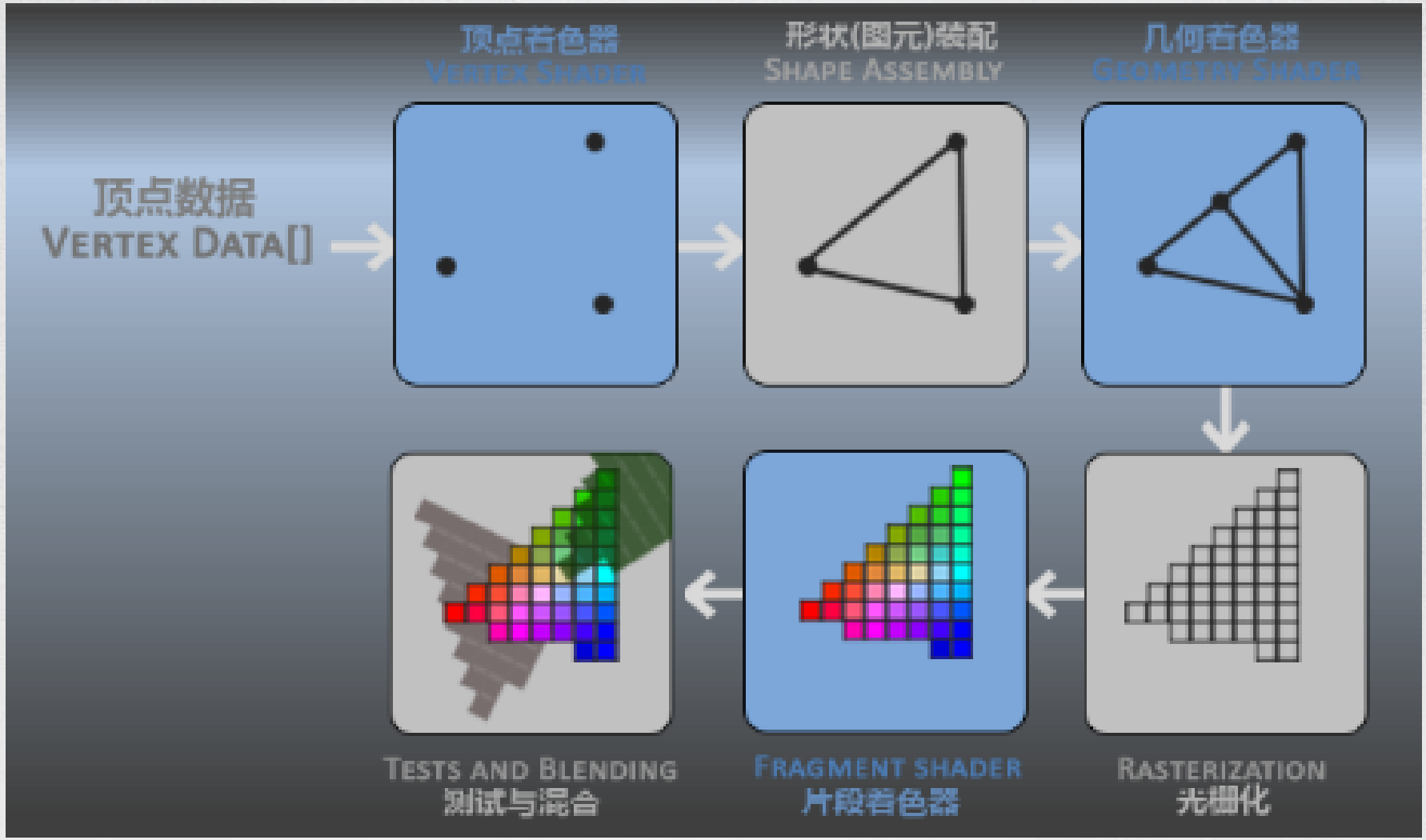
渲染管线主要做的两件事情

- 1. 3D坐标 → 2D坐标
- 2. 2D坐标 → 有颜色的像素

2D 坐标 ≠ 2D 像素

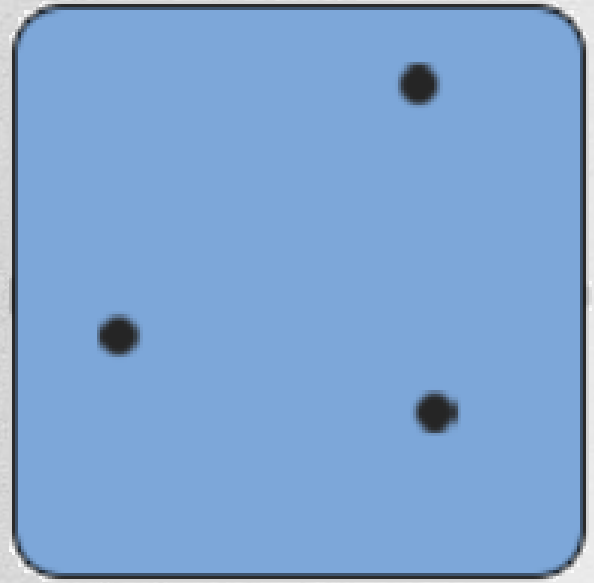
Rendering Pipeline

渲染流程图

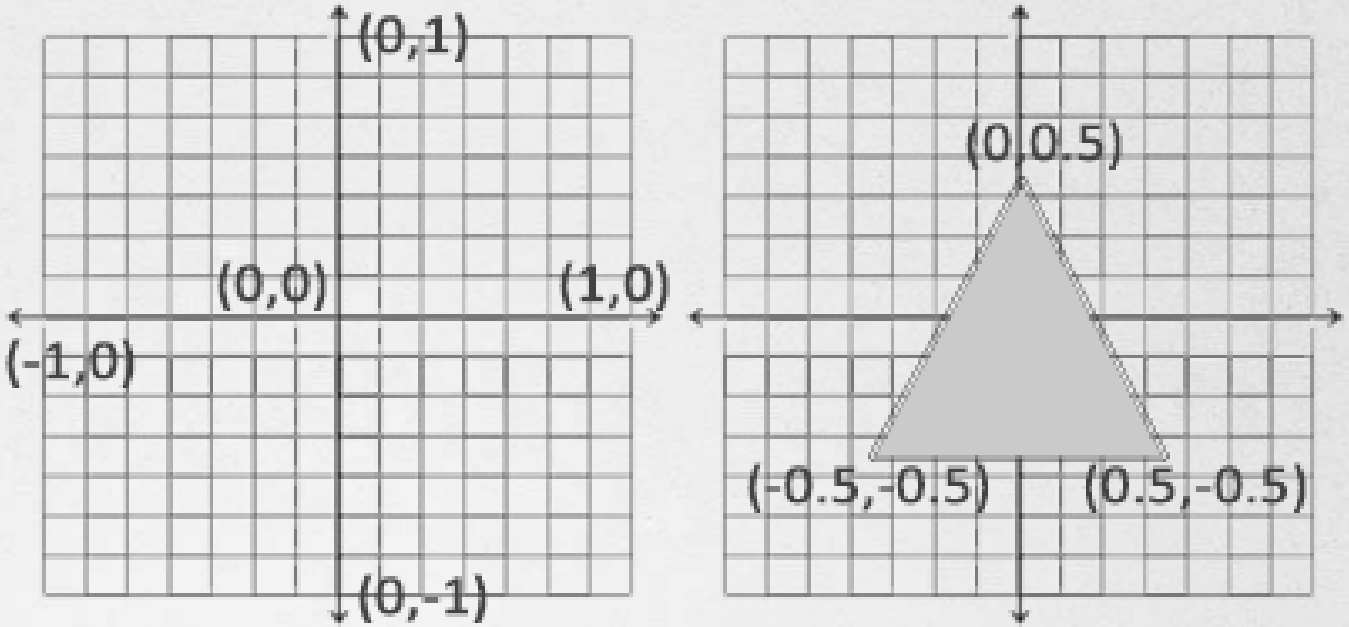


Rendering Pipeline

顶点着色器
VERTEX SHADER

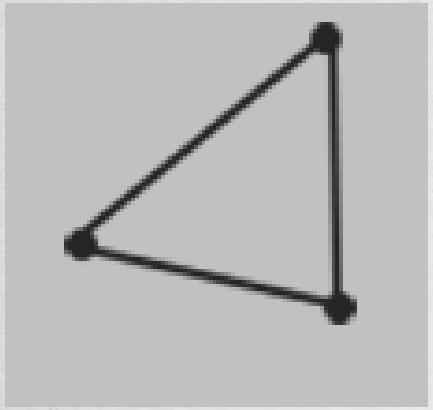


标准化设备坐标
(Normalized Device Coordinates, NDC)



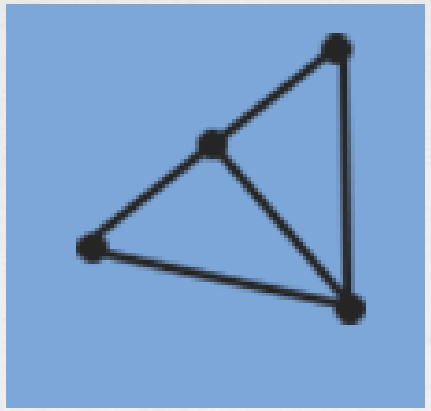
Rendering Pipeline

形状(图元)装配阶段



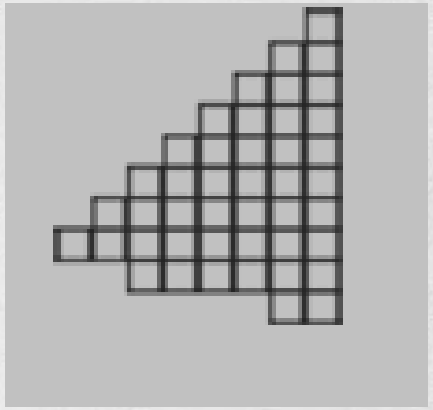
图元装配阶段将顶点着色器输出的所有顶点作为输入，将所有的点装配成指定图元的形状；

几何着色器



几何着色器把图元式的一系列顶点的集合形作为输入，它可以通过产生新顶点构造出新的（或是其它的）图元来生成其他形状。

光栅化阶段



光栅化阶段把图元映射为最终屏幕上相应的像素，生成供片段着色器使用的片段(Fragment)。在片段着色器运行之前会执行裁切(Clipping)。

暂停一下

图元(Primitive): 提示 OpenGL 对顶点数据进行渲染的渲染类型。

Eg. GL_POINTS、 GL_TRIANGLES、 GL_LINE_STRIP。

片段(Fragment): OpenGL 中的一个片段是 OpenGL 渲染一个像素所需的所有数据。

Rendering Pipeline

片段着色器

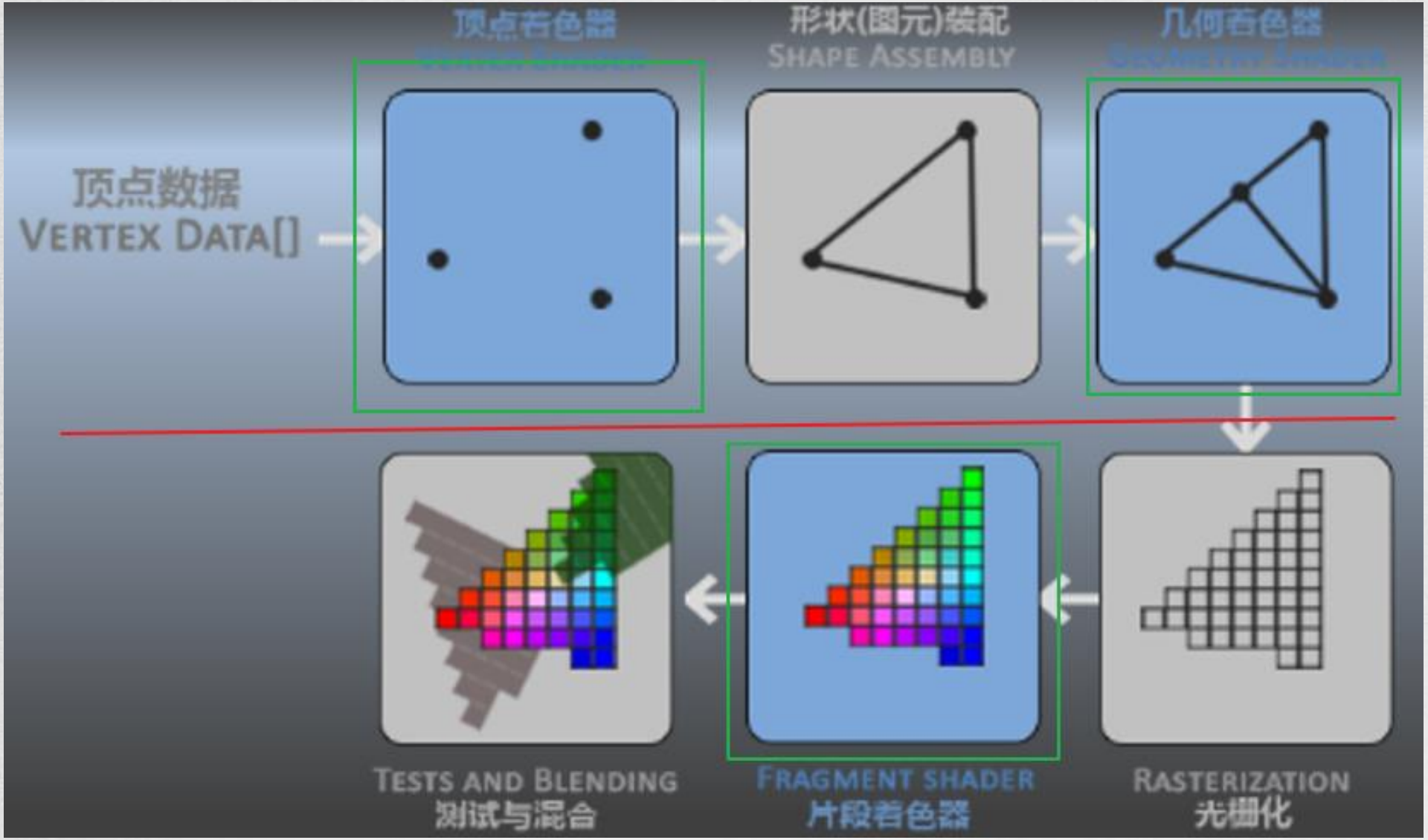
计算一个像素的最终颜色，这也是所有OpenGL高级效果产生的地方。

- Lighting
- Shading

测试与混合阶段

- 检测片段的对应的深度值，用它们来判断这个像素是其它物体的前面还是后面，决定是否应该丢弃。
- 检查alpha值（透明度）并对物体进行混合(Blend)。

Rendering Pipeline



GLSL

in / out

Communication between shaders

Layout

Vertex data input

uniform

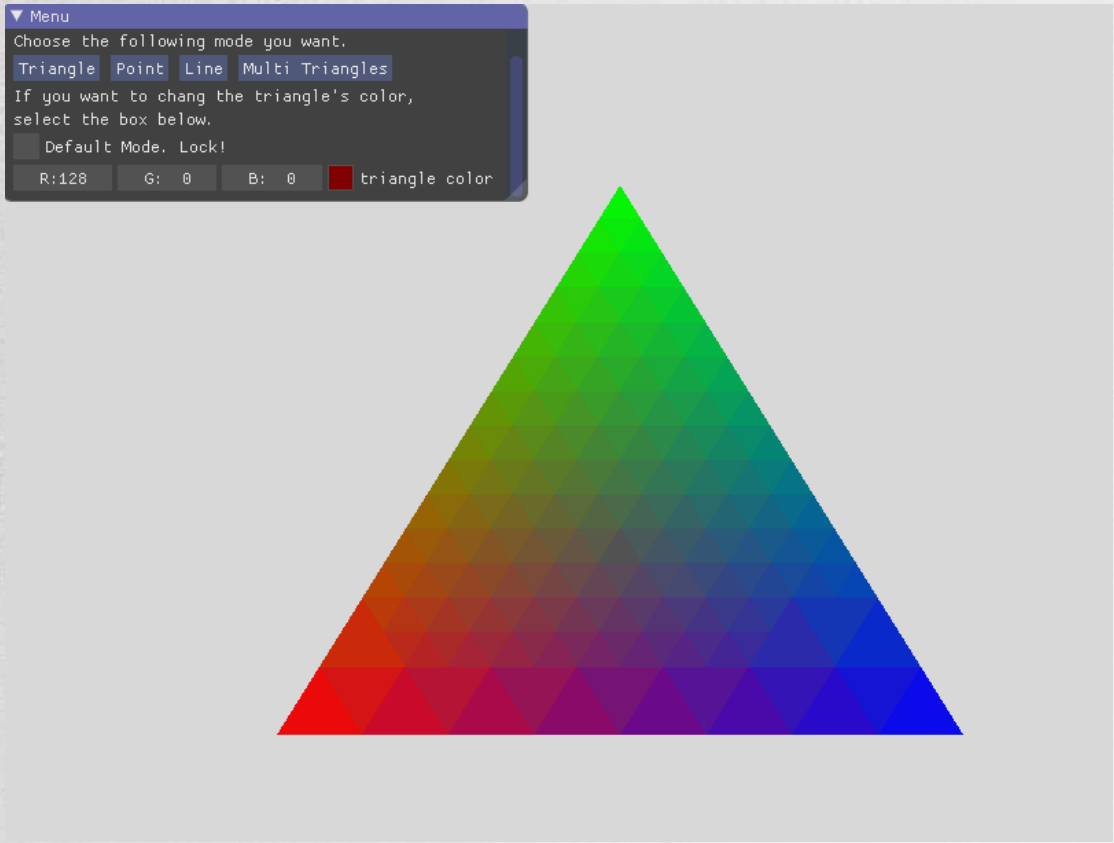
Variables controlled by Main()

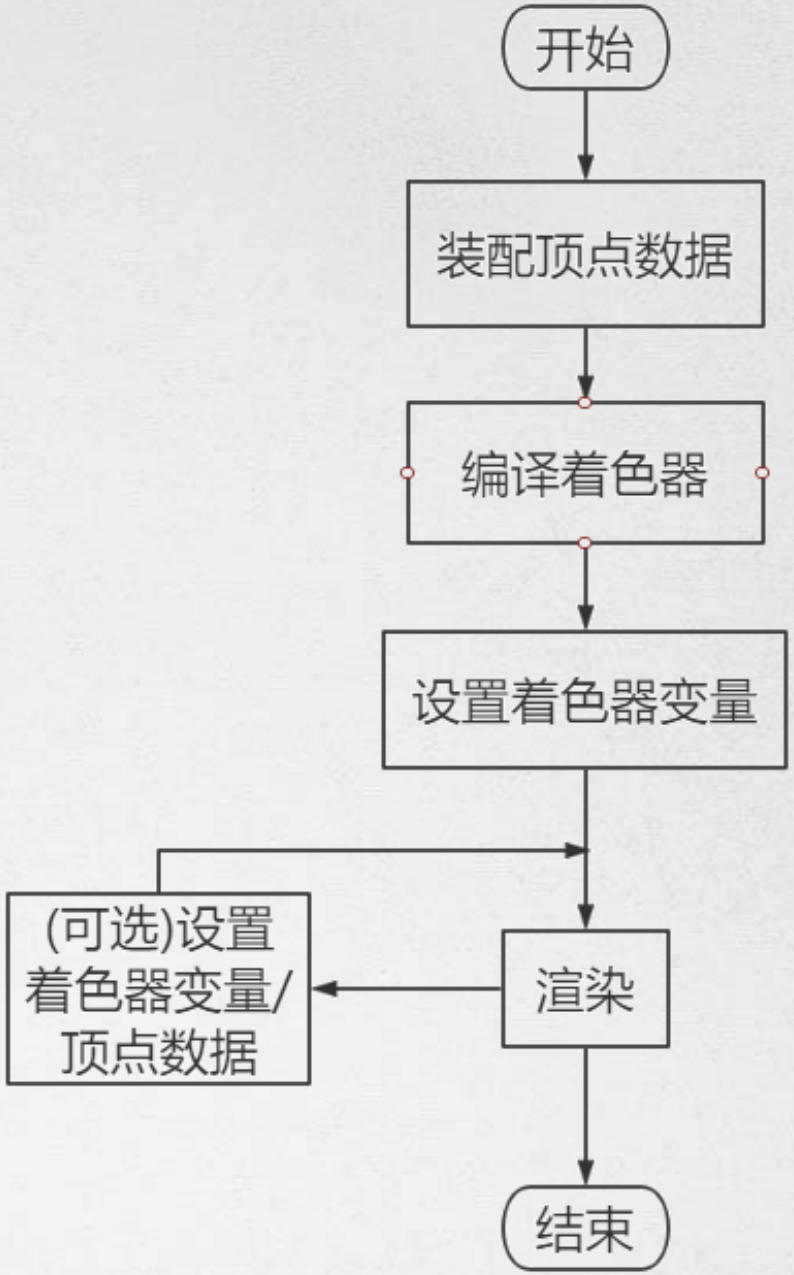
gl_Position / FragColor

Some predefined variables

Rendering Pipeline

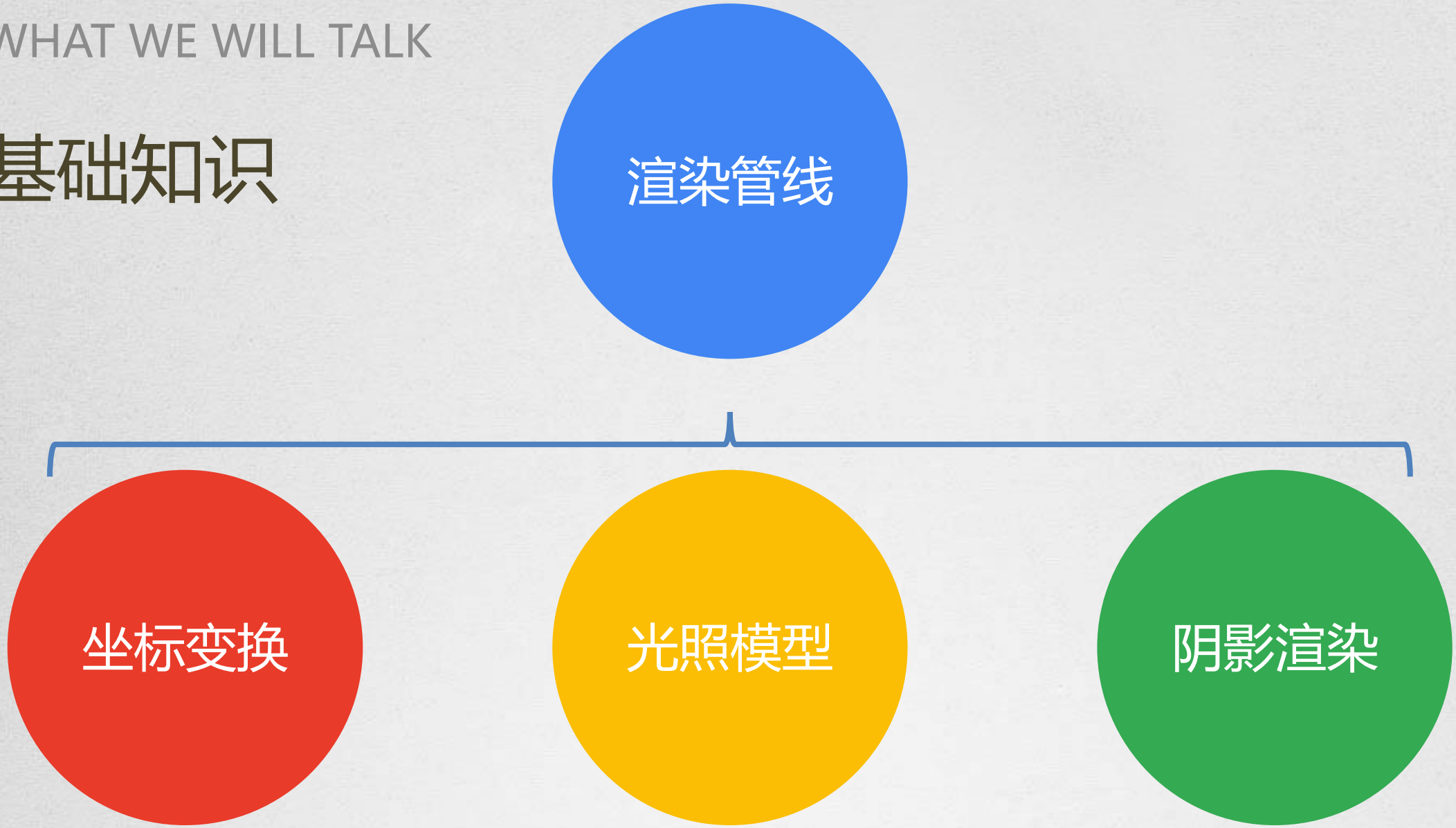
基本程序流程





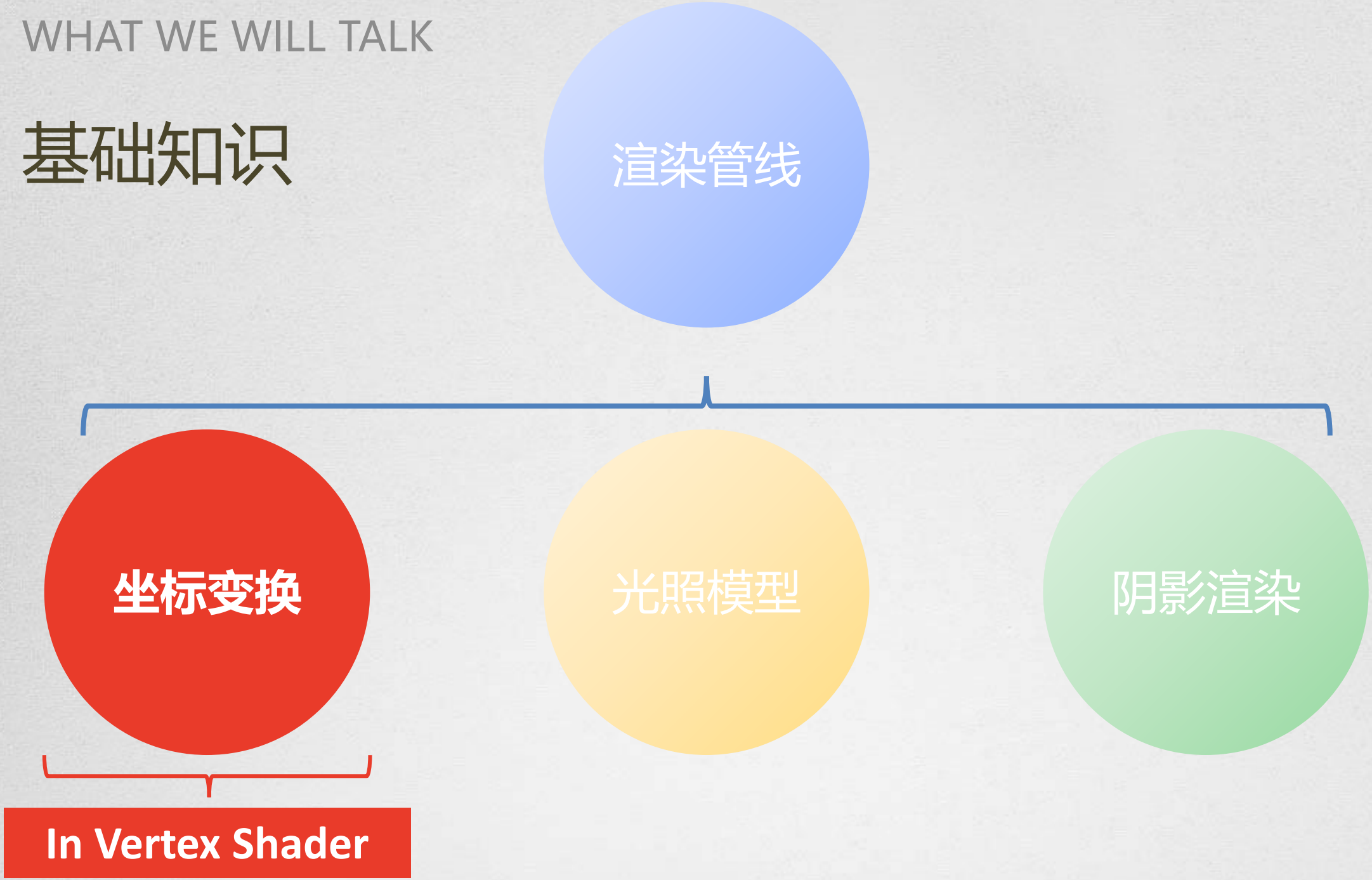
WHAT WE WILL TALK

基础知识



WHAT WE WILL TALK

基础知识

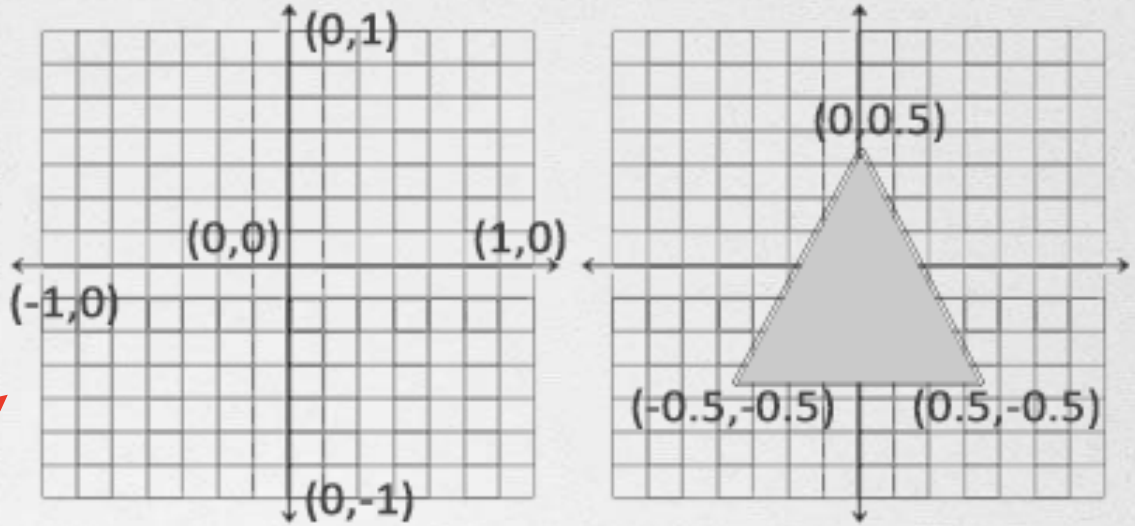


缘由

1. 不够直观

2. 动态渲染

标准化设备坐标
(Normalized Device Coordinates, NDC)



Homogeneous Coordinates

2D Point:

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D Vector:

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

Why Homogeneous Coordinates

Transformation in 2D:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$M = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} = \left[\begin{array}{cc|c} \multicolumn{2}{c|}{A} & \mathbf{t} \\ \hline 0 & 0 & 1 \end{array}\right]$$

2D → 3D Rotation Example (around X axis)

2D

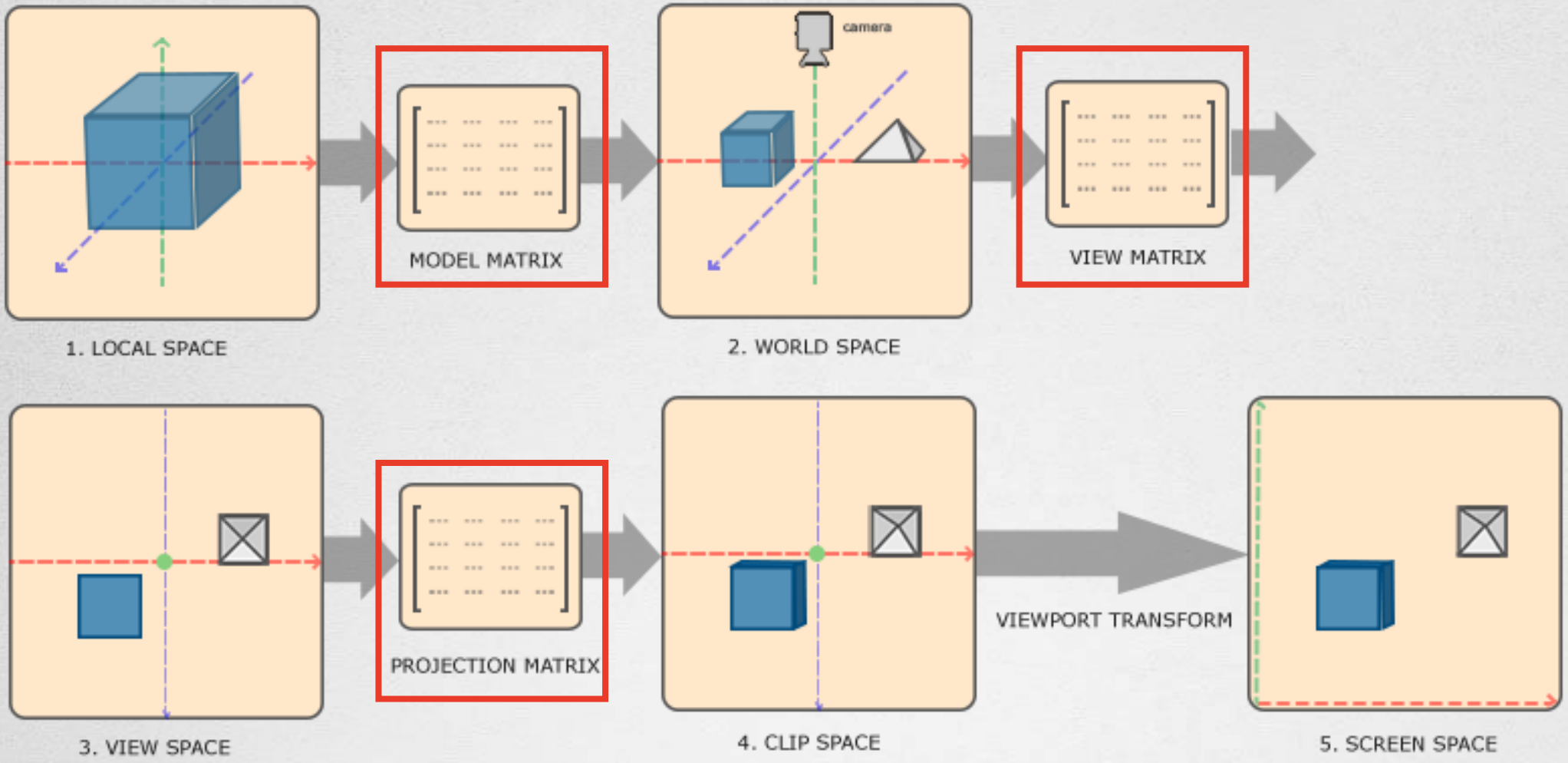
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

3D

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformation

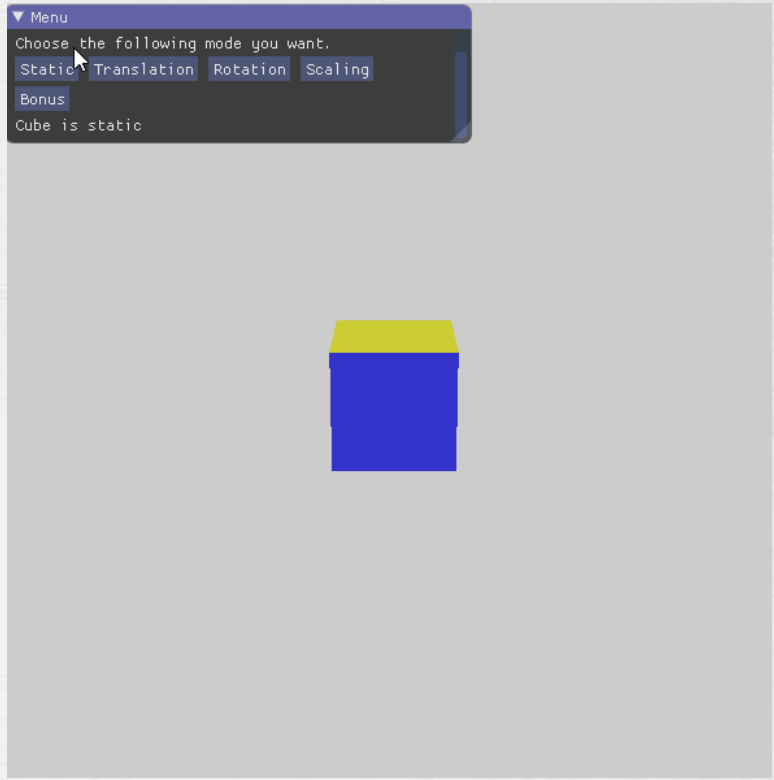
3 Important Matrices



Implementation

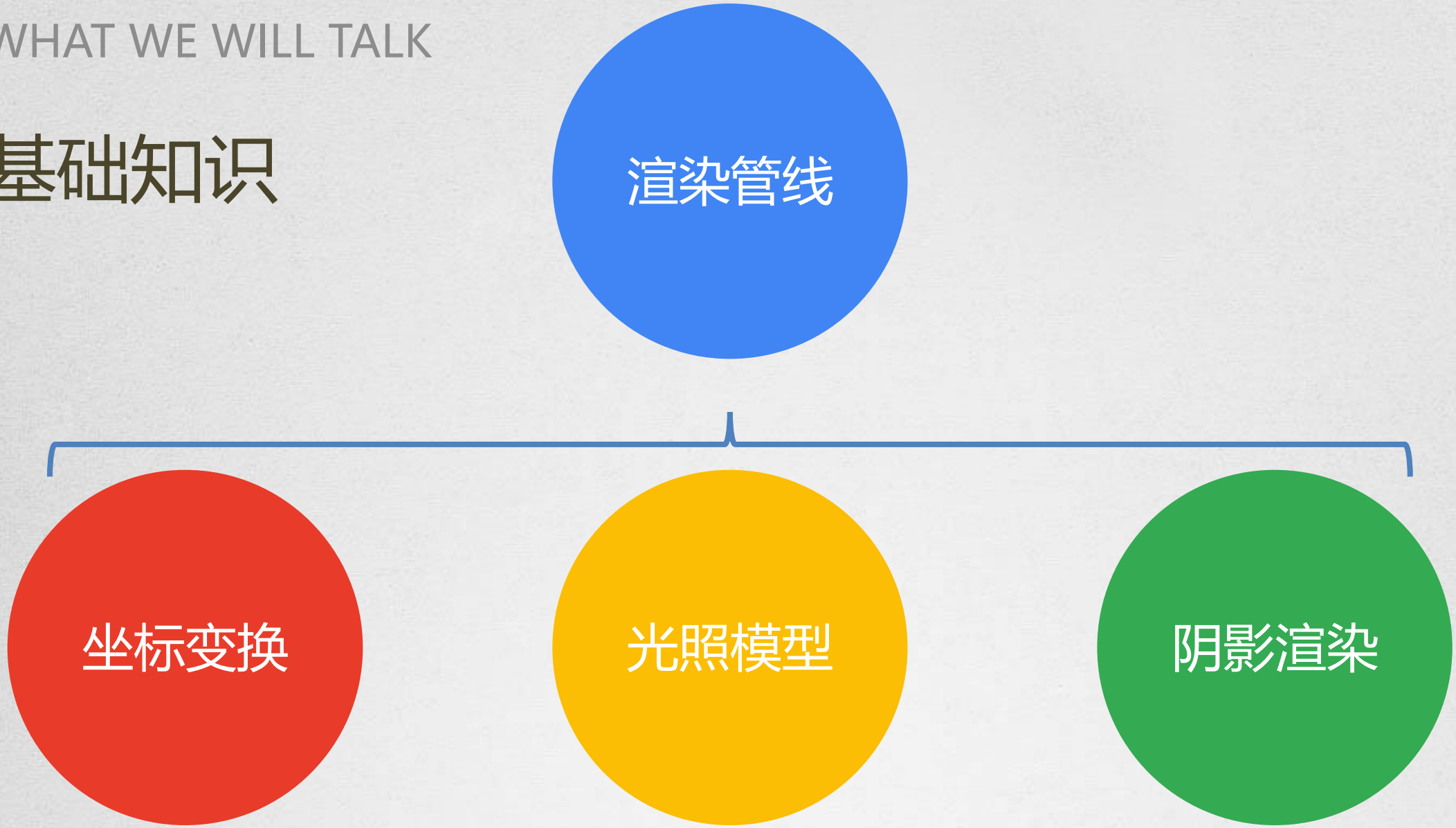
$$V_{clip} = M_{projection} \cdot M_{view} \cdot M_{model} \cdot V_{local}$$

```
#version 330 core

layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aColor;

out vec4 vColor;
uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

void main() {
    gl_Position = projection * view * model * vec4(aPos, 1.0);
    vColor = vec4(aColor, 1.0);
}
```



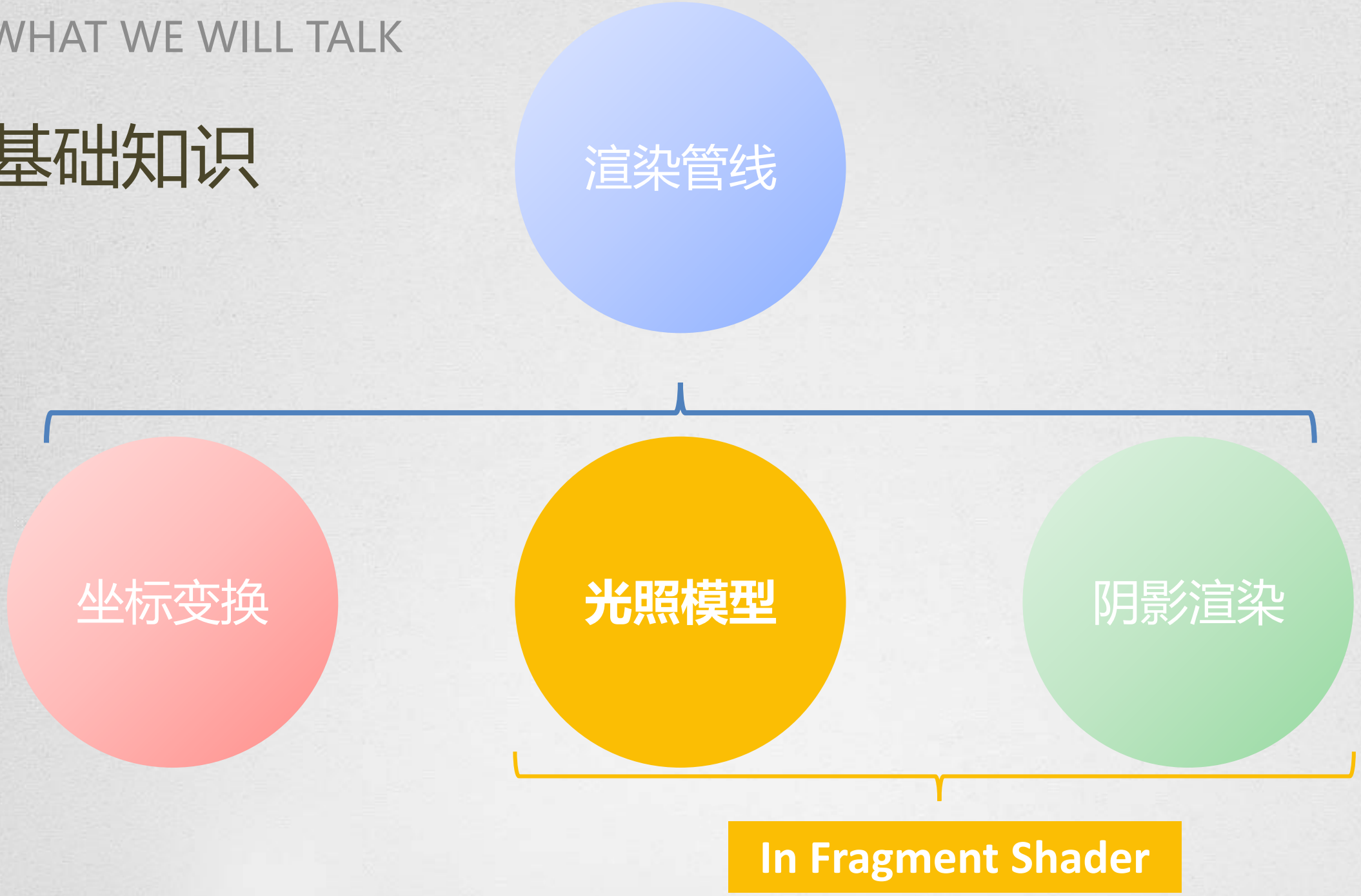
WHAT WE WILL TALK

基础知识



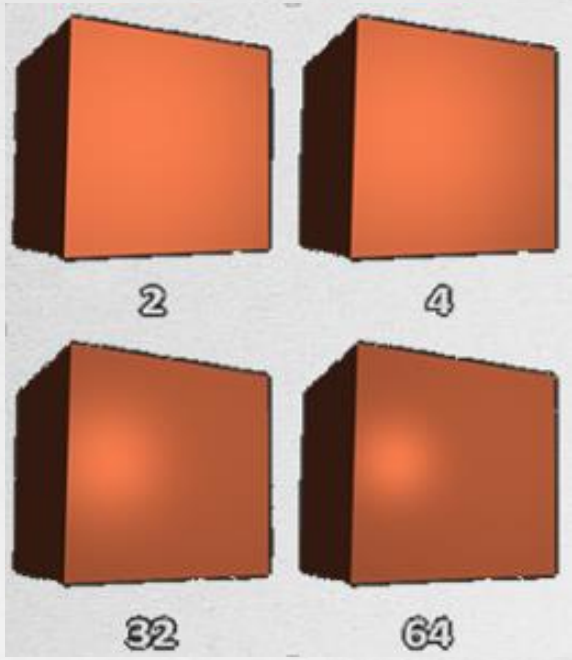
WHAT WE WILL TALK

基础知识



光照模型分类

全局光照明模型
(复杂)

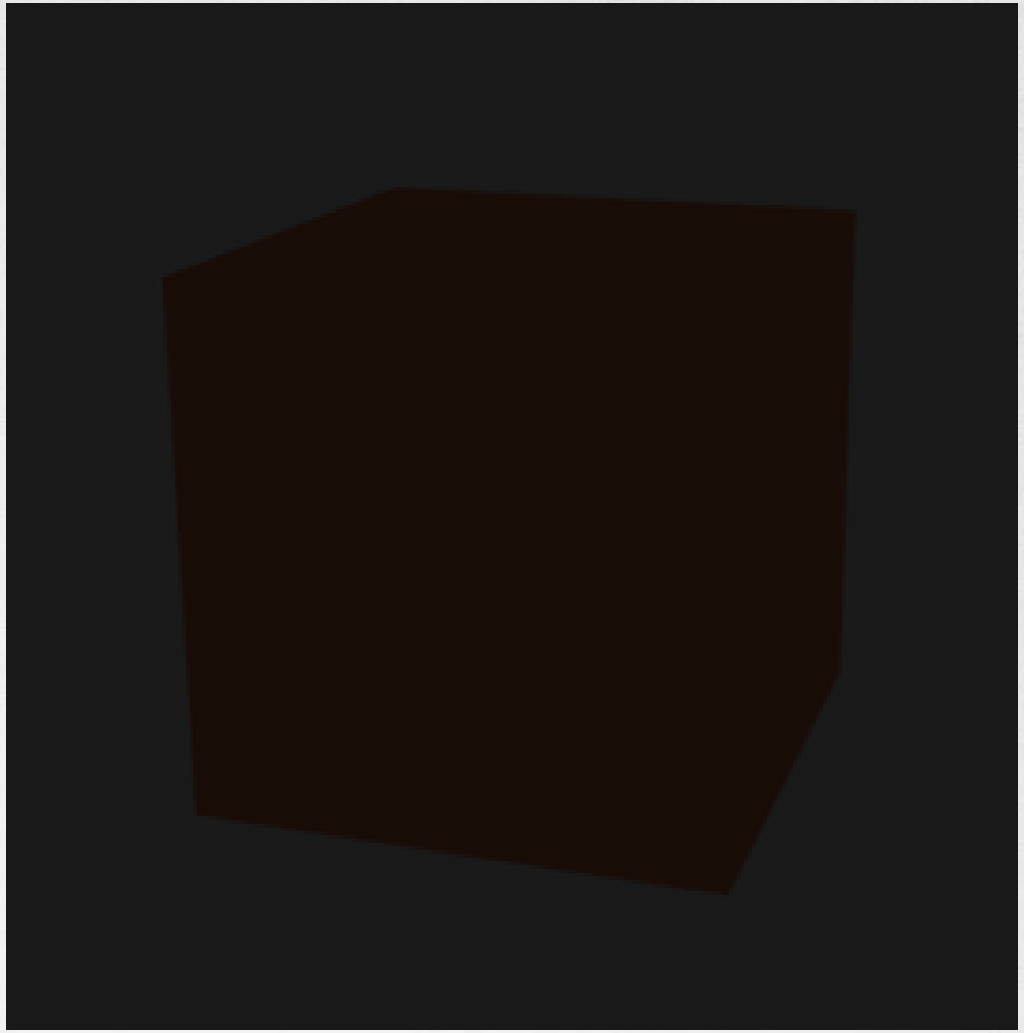


局部光照明模型
(简单)

Light = 环境光 + 漫反射光 + 镜面光

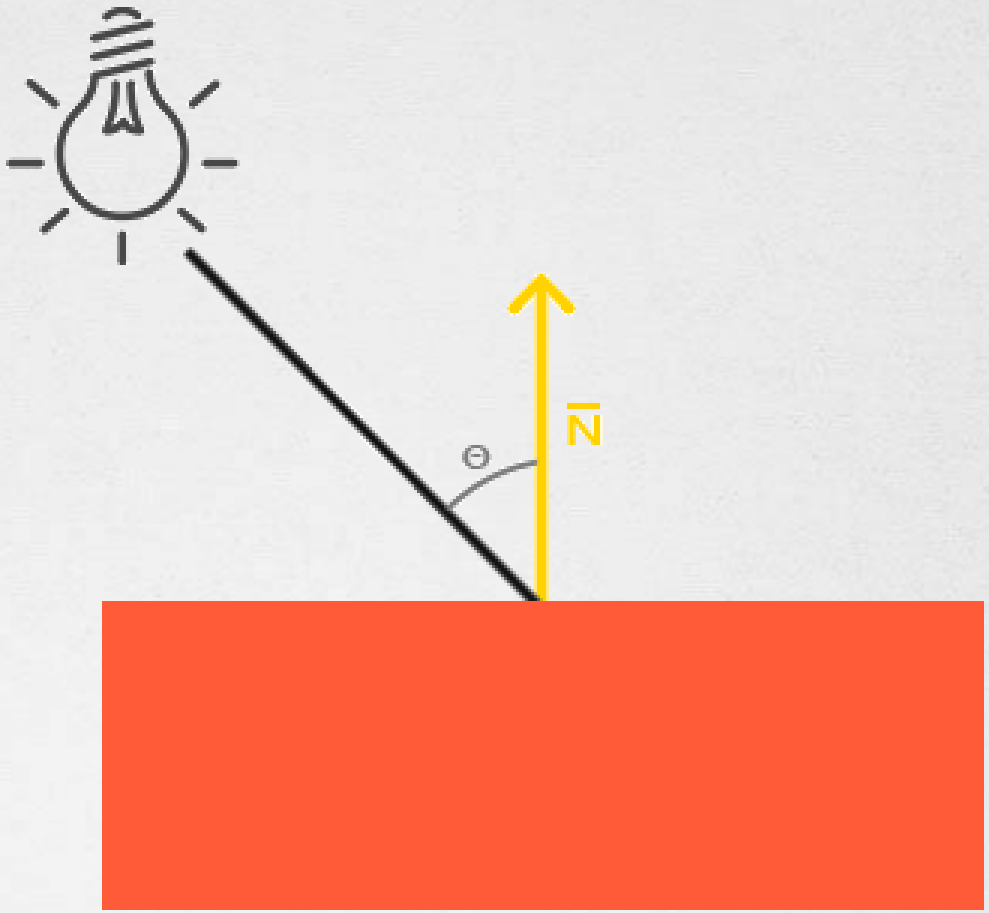
环境光照

一般为常量

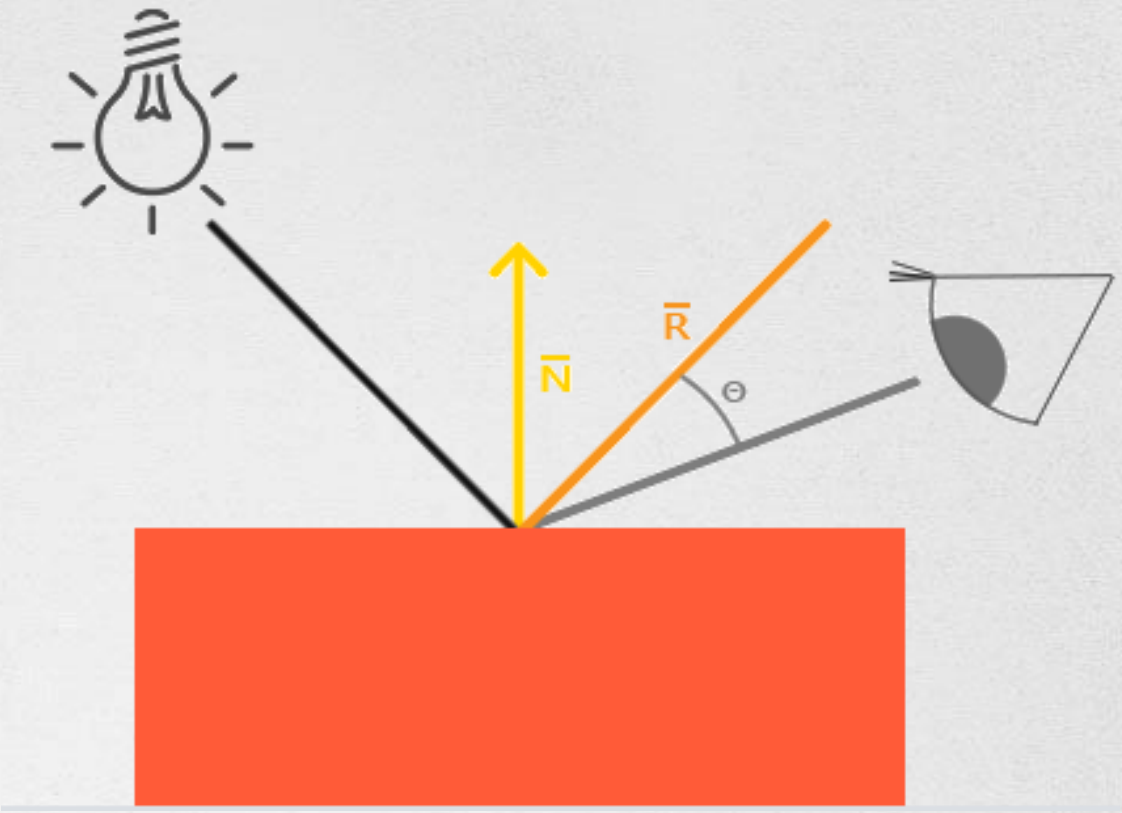


漫反射光照

- 法向量
- 定向的光线
- 法线矩阵(Normal Matrix)



镜面光照



specularStrength 由材质决定

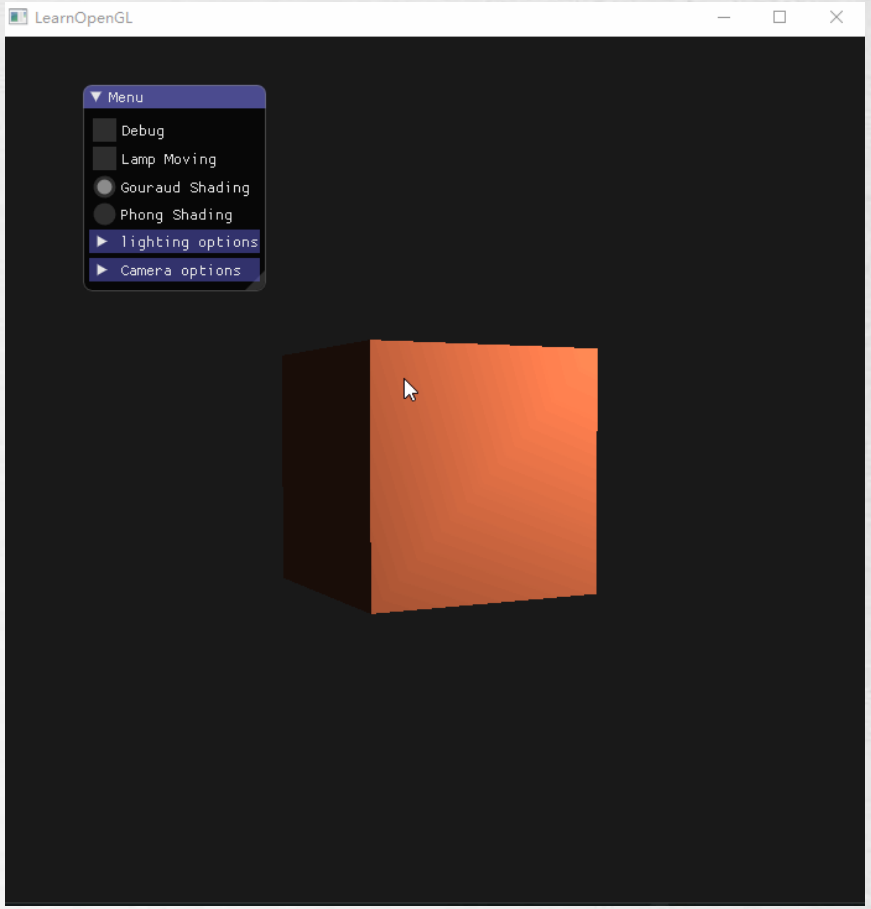
```
float specularStrength = 0.5;
float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
vec3 specular = specularStrength * spec * lightColor;
```

两种局部光照模型

Light = 环境光 + 漫反射光 + 镜面光

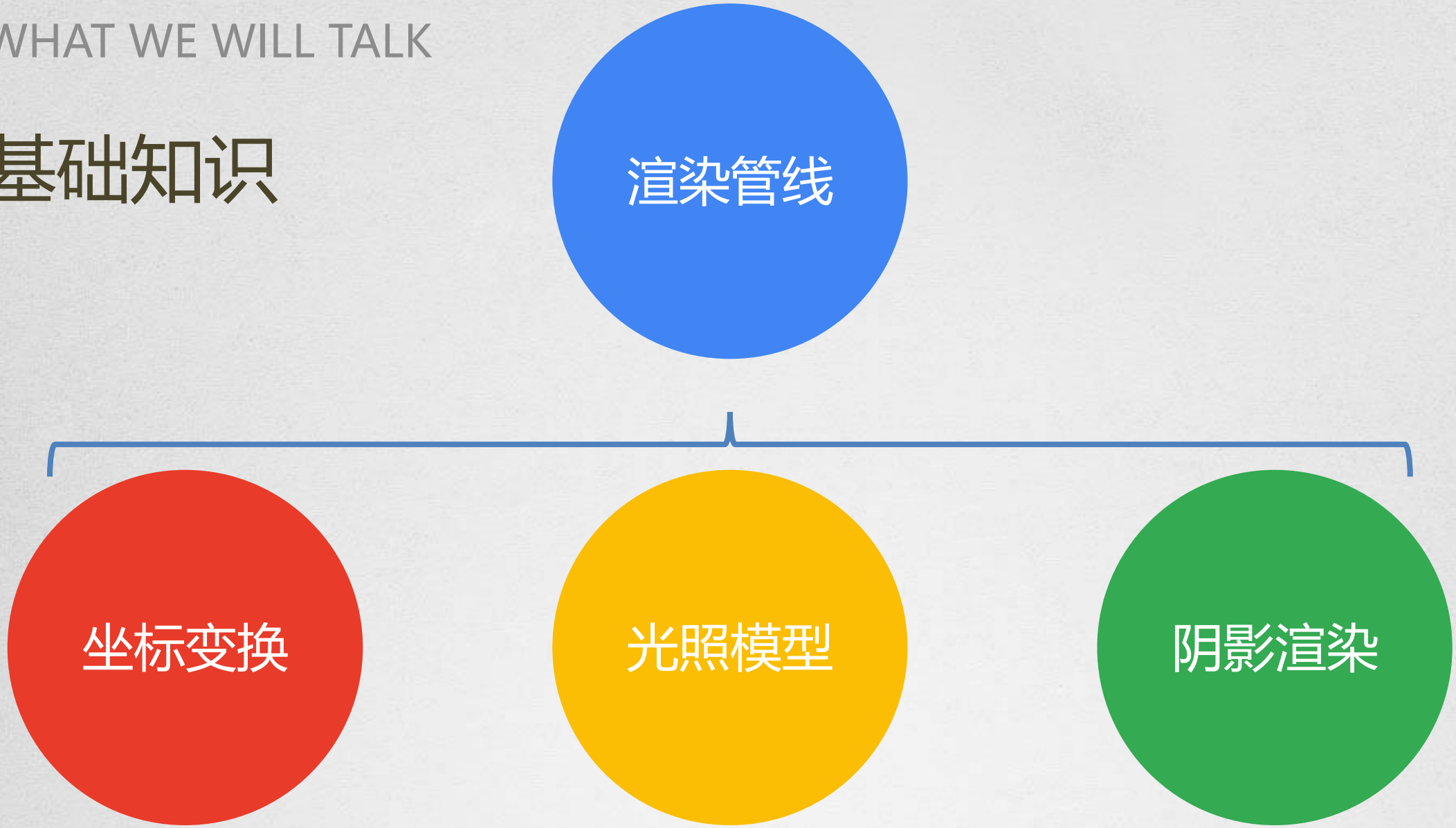
Gouraud: 将上式放在顶点着色器中计算

Phong : 将上式放在片段着色器中计算



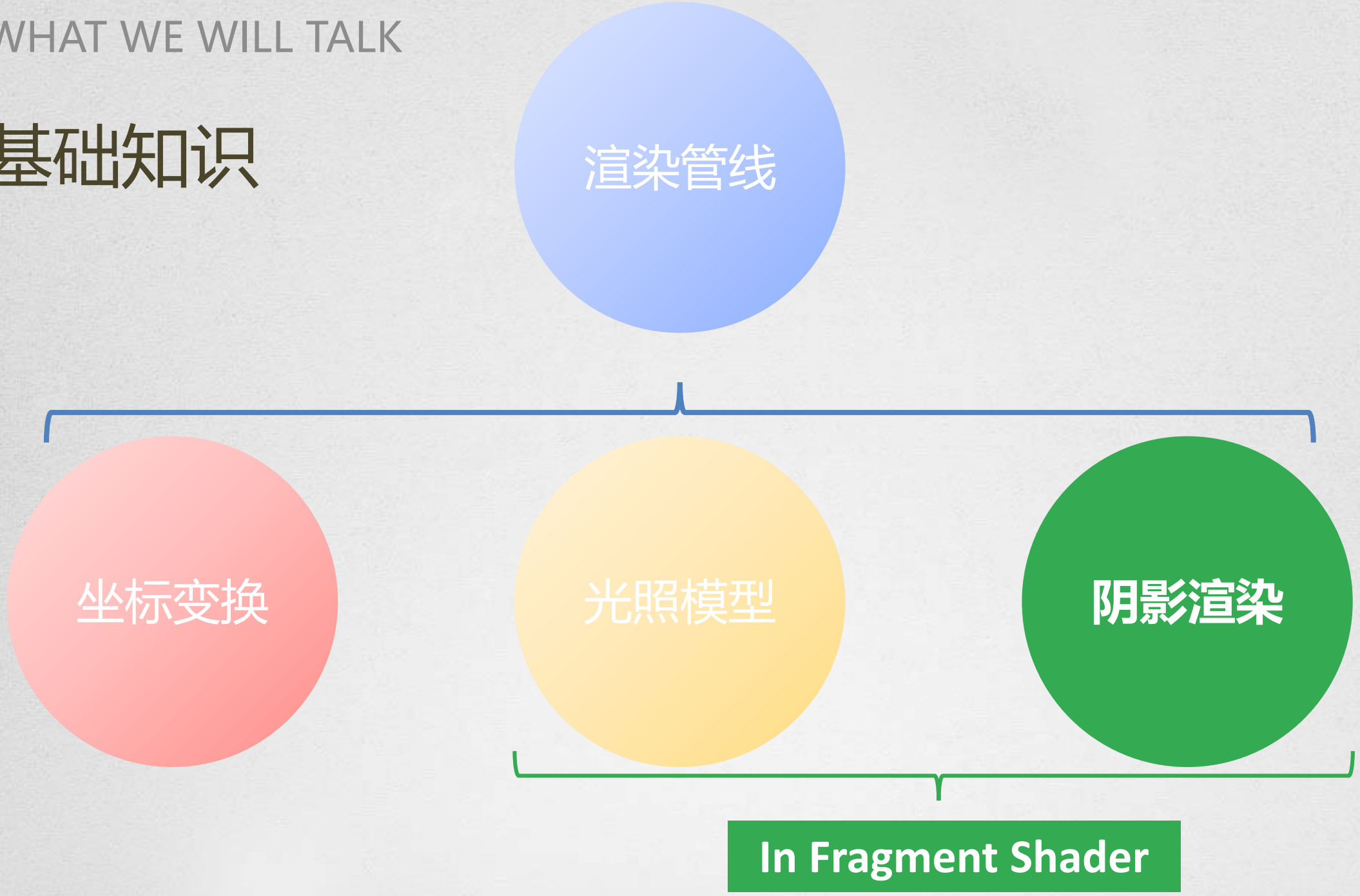
WHAT WE WILL TALK

基础知识

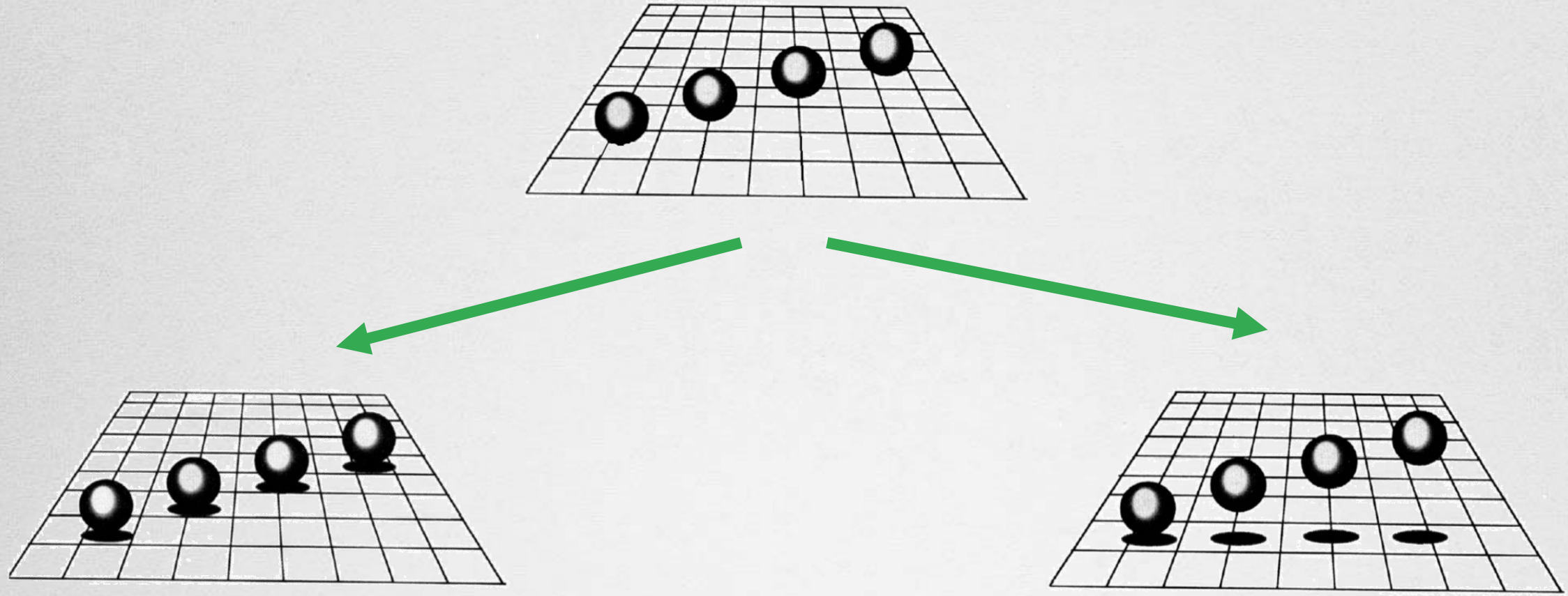


WHAT WE WILL TALK

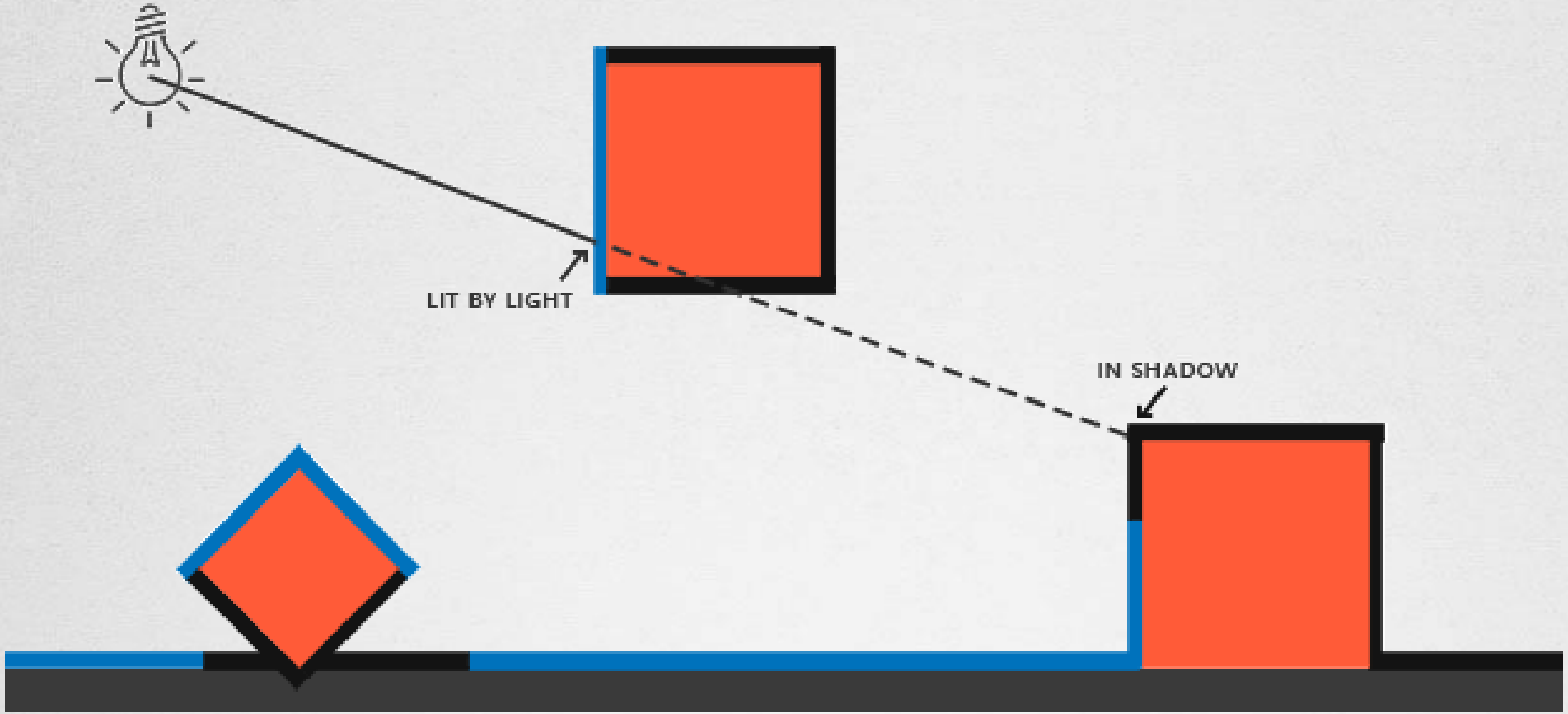
基础知识



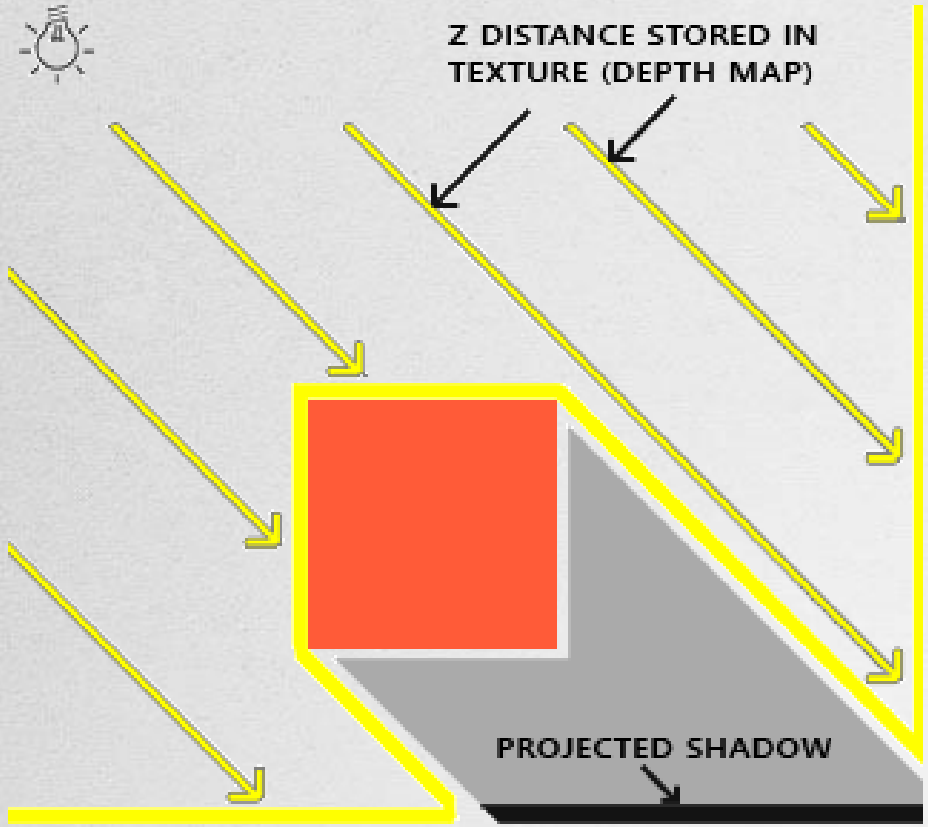
Why shadow is important?



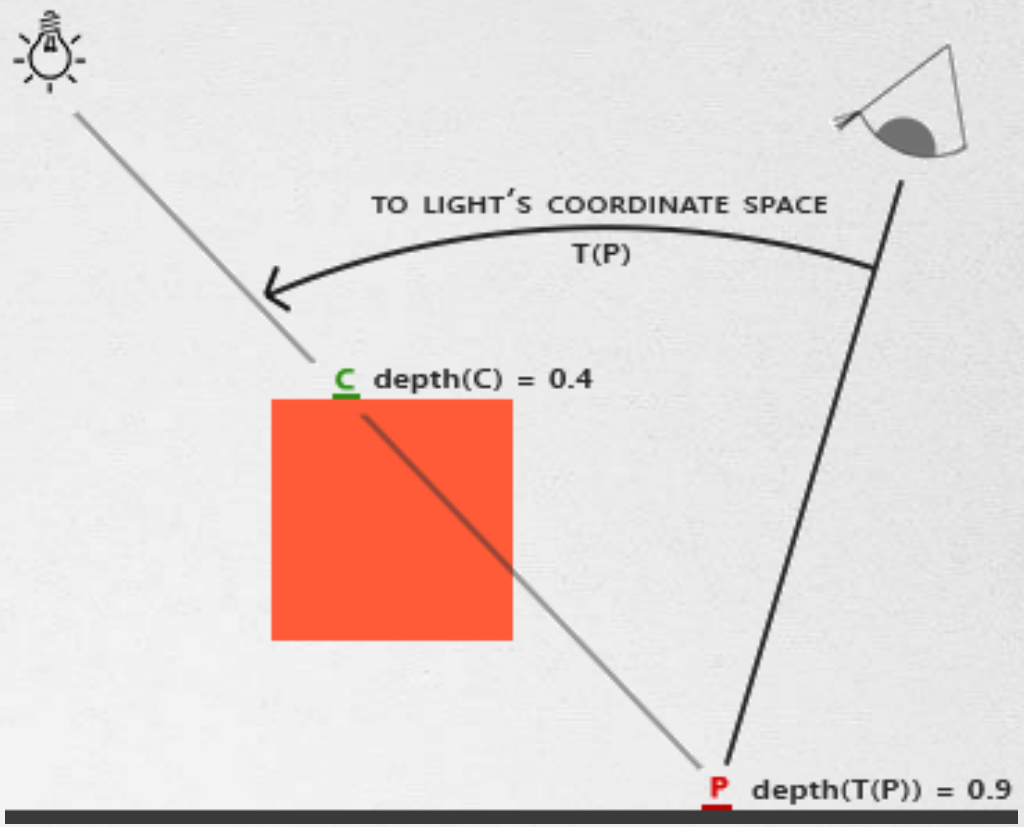
Basic shadow rendering



Shadow Mapping

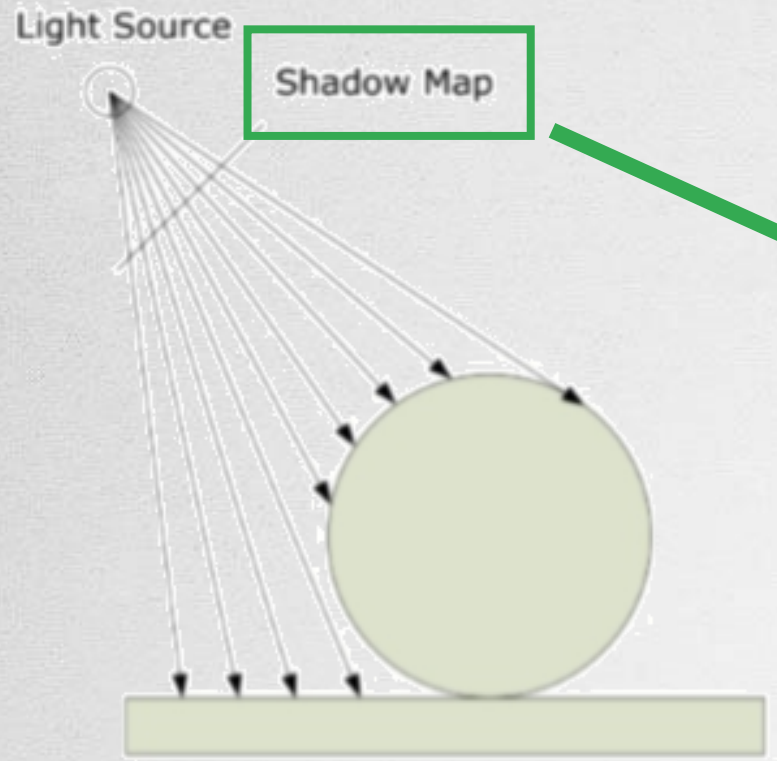


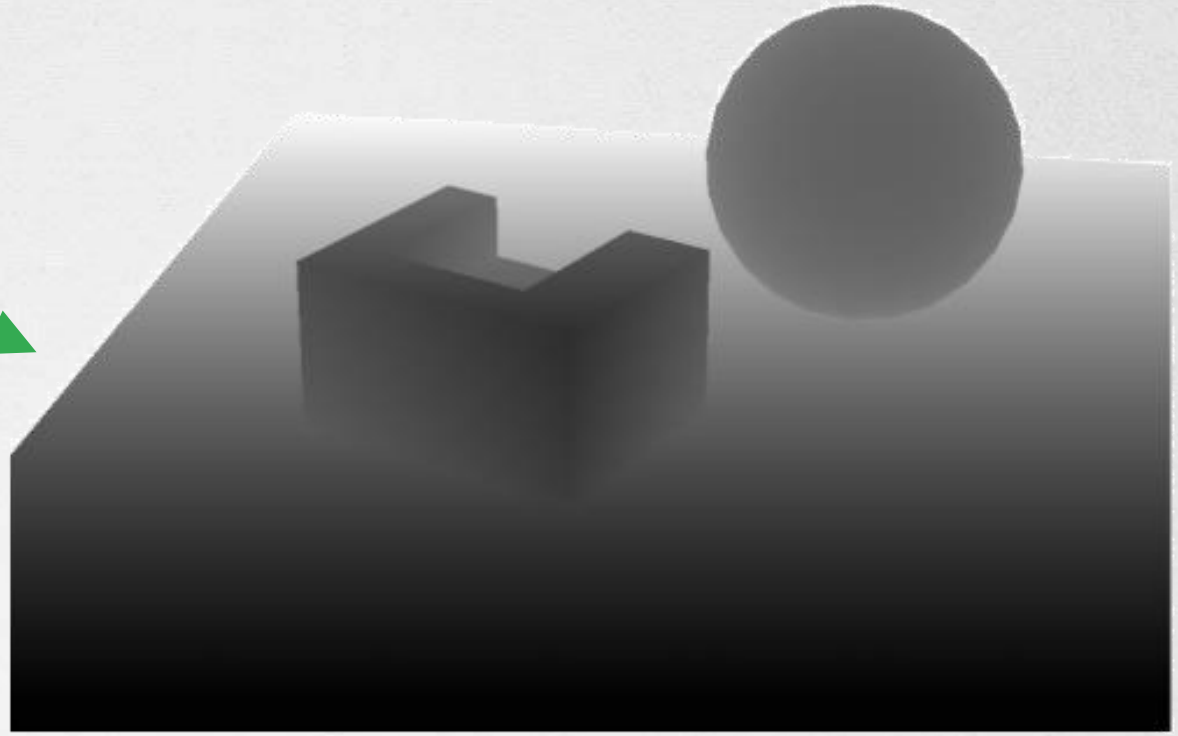
Step 1



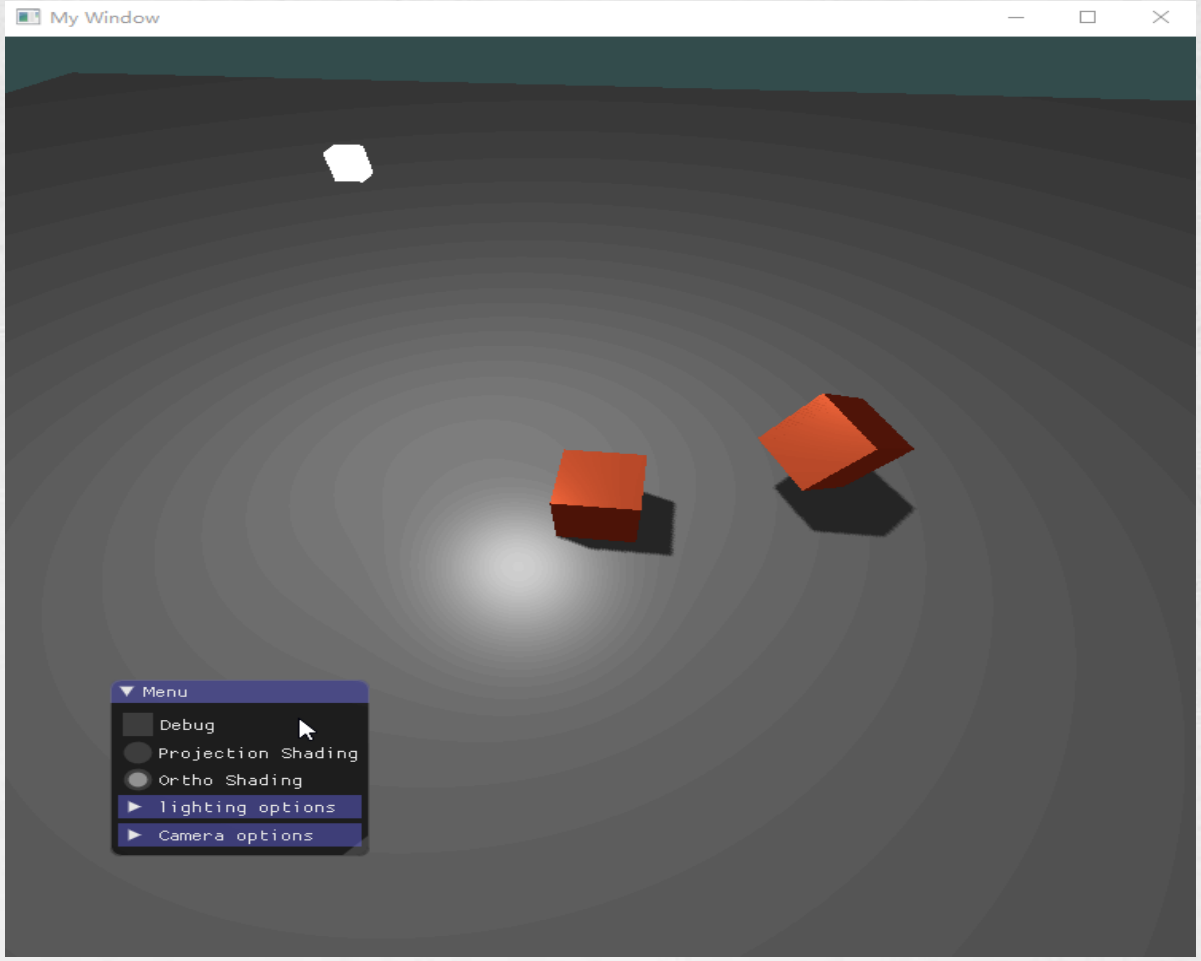
Step 2

Step 1 details





Step 2 Demo


My Window
Menu
Debug
Projection Shading
Ortho Shading
lighting options
Camera options

What’s not included …

本次分享没有涉及到的内容

- 纹理映射
- 曲线绘制
- 阴影优化方法
- More …

Main Reference

- Learn-OpenGL-CN: https://learnopengl-cn.github.io/
- Courseware of Computer Graphics 2018 by A.prof. Chengying Gao, Sysu.
- Demo from my own homework.

Q&A

Thank You