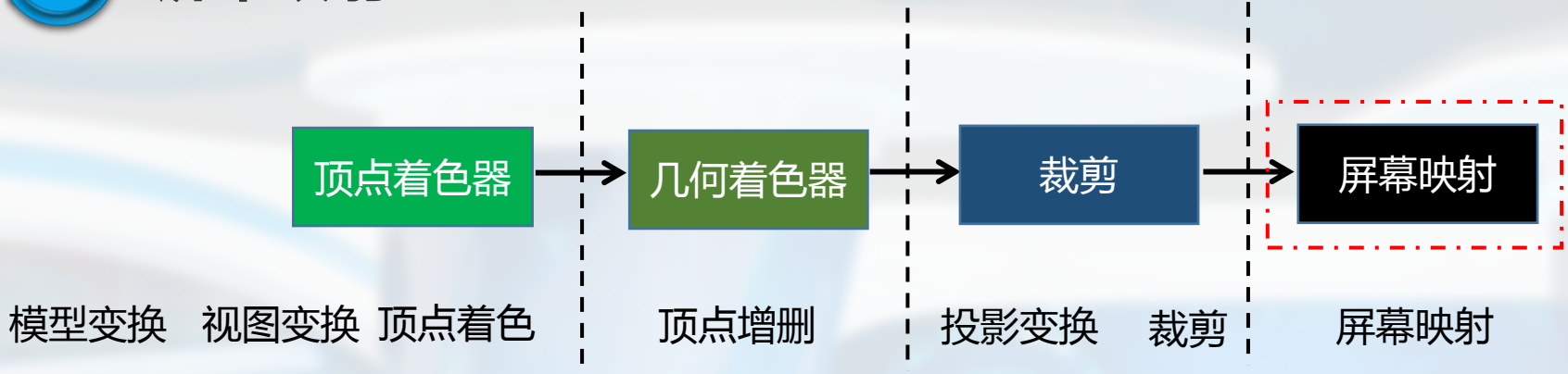
几何阶段的完结：屏幕映射

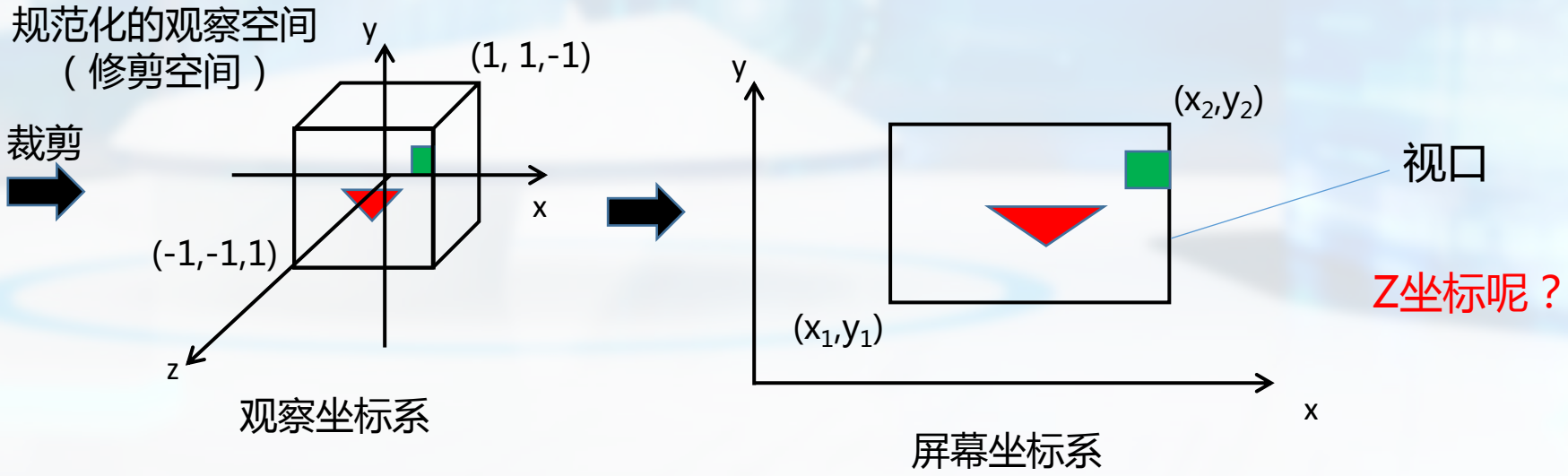
华中科技大学软件学院　万琳

提纲

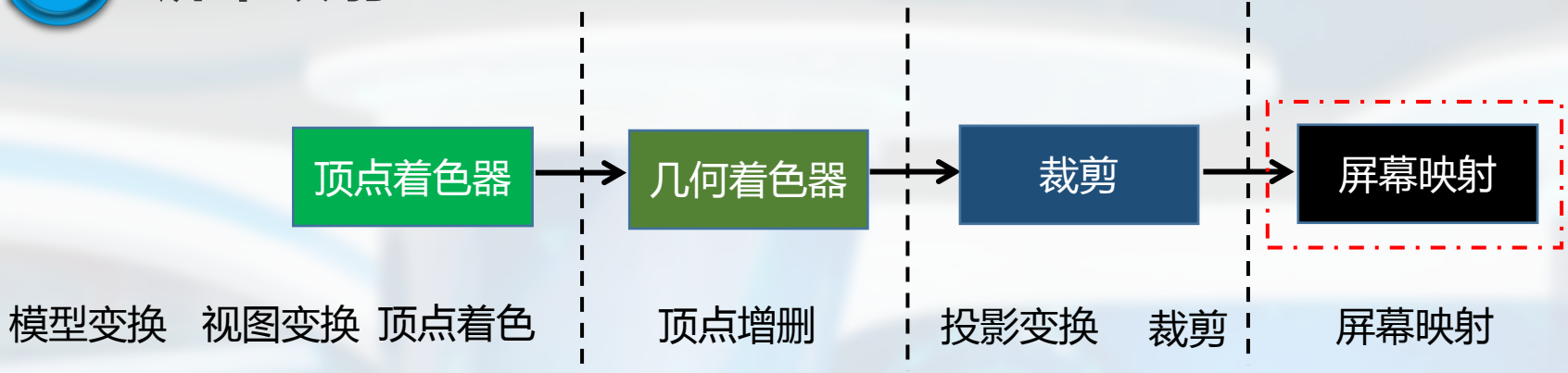
1. 屏幕映射
2. OpenGL的几何阶段实现

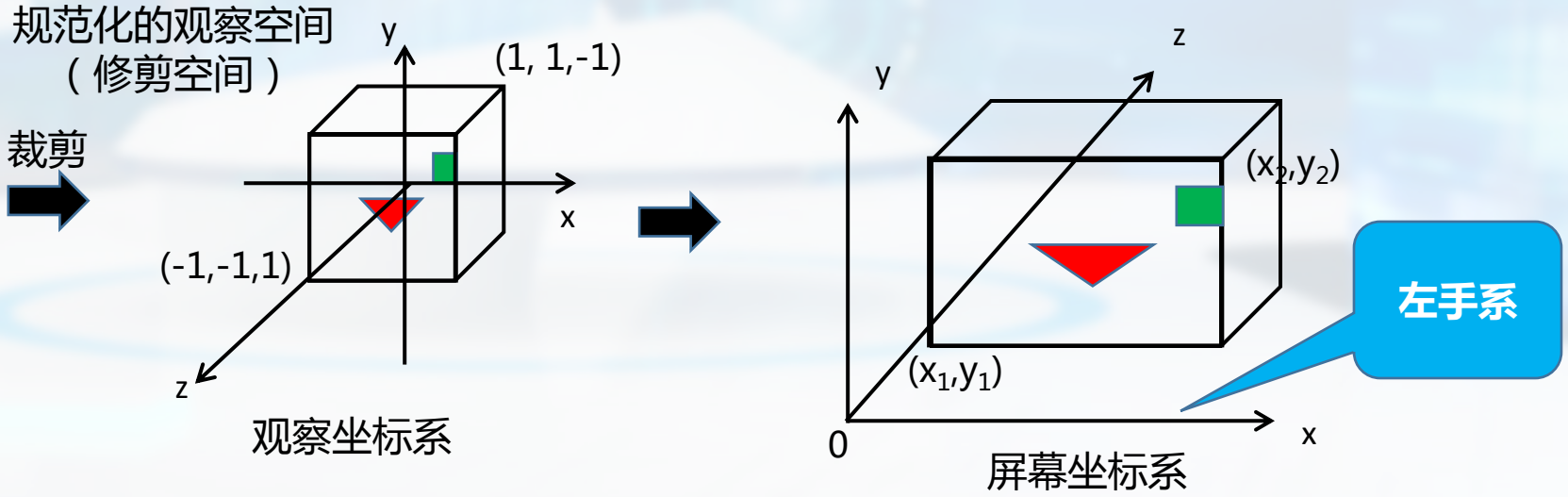
1 屏幕映射


顶点着色器
几何着色器
裁剪
屏幕映射
模型变换
视图变换
顶点着色
顶点增删
投影变换
裁剪
屏幕映射


规范化的观察空间
（修剪空间）
裁剪
y
(1, 1,-1)
x
(-1,-1,1)
z
观察坐标系
y
(x2,y2)
(x1,y1)
x
屏幕坐标系
视口
Z坐标呢？

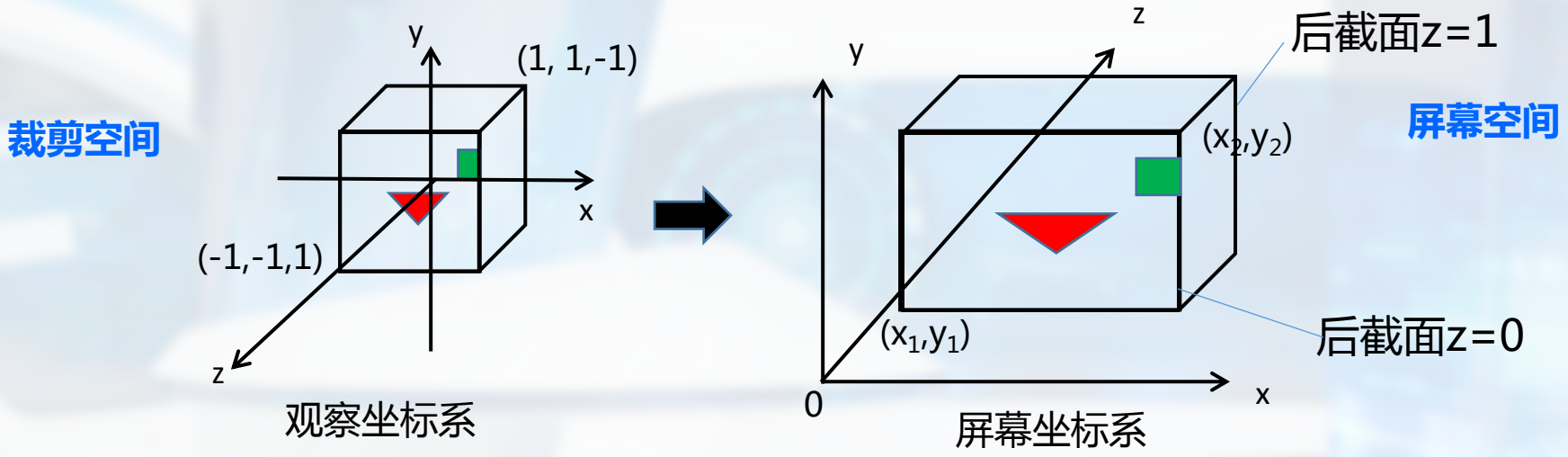
1 屏幕映射


顶点着色器
几何着色器
裁剪
屏幕映射
模型变换
视图变换
顶点着色
顶点增删
投影变换
裁剪
屏幕映射


规范化的观察空间
（修剪空间）
裁剪
y
(1, 1,-1)
x
(-1,-1,1)
z
观察坐标系
z
y
(x_2,y_2)
(x_1,y_1)
0
x
左手系
屏幕坐标系

1 屏幕映射

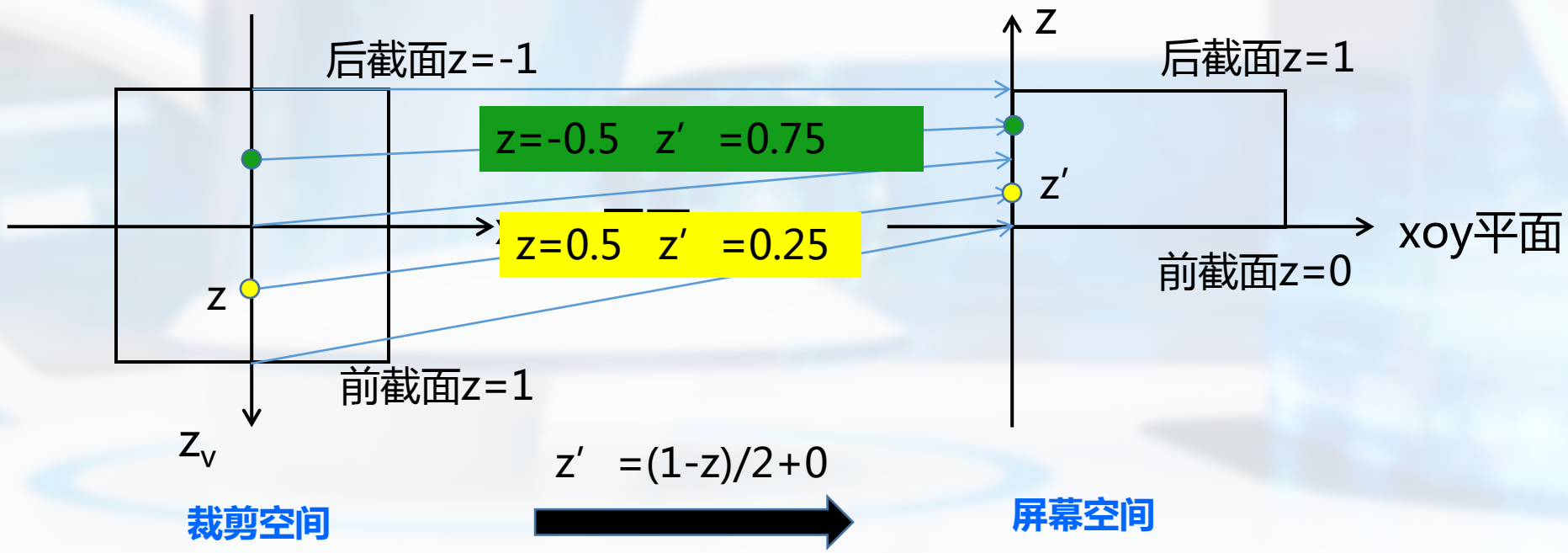
屏幕映射： 裁剪空间向屏幕空间的映射



1 屏幕映射

过程分析：

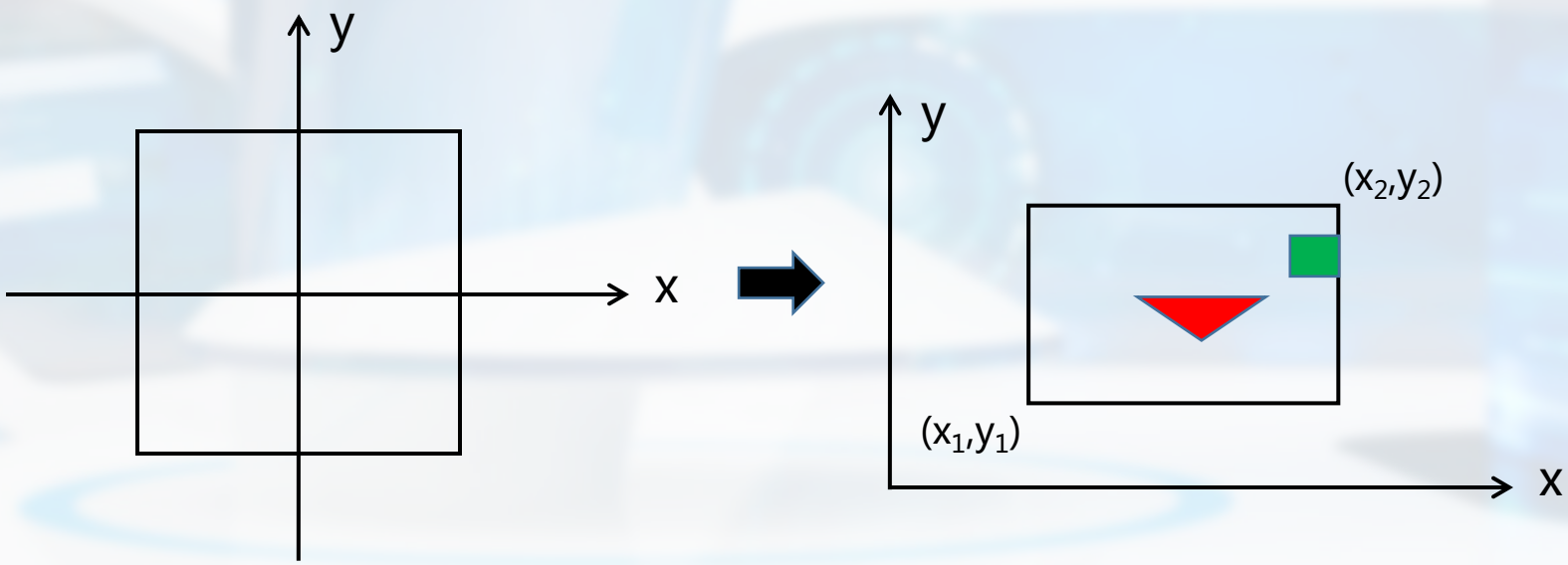
➢z方向的变化



1 屏幕映射

过程分析：

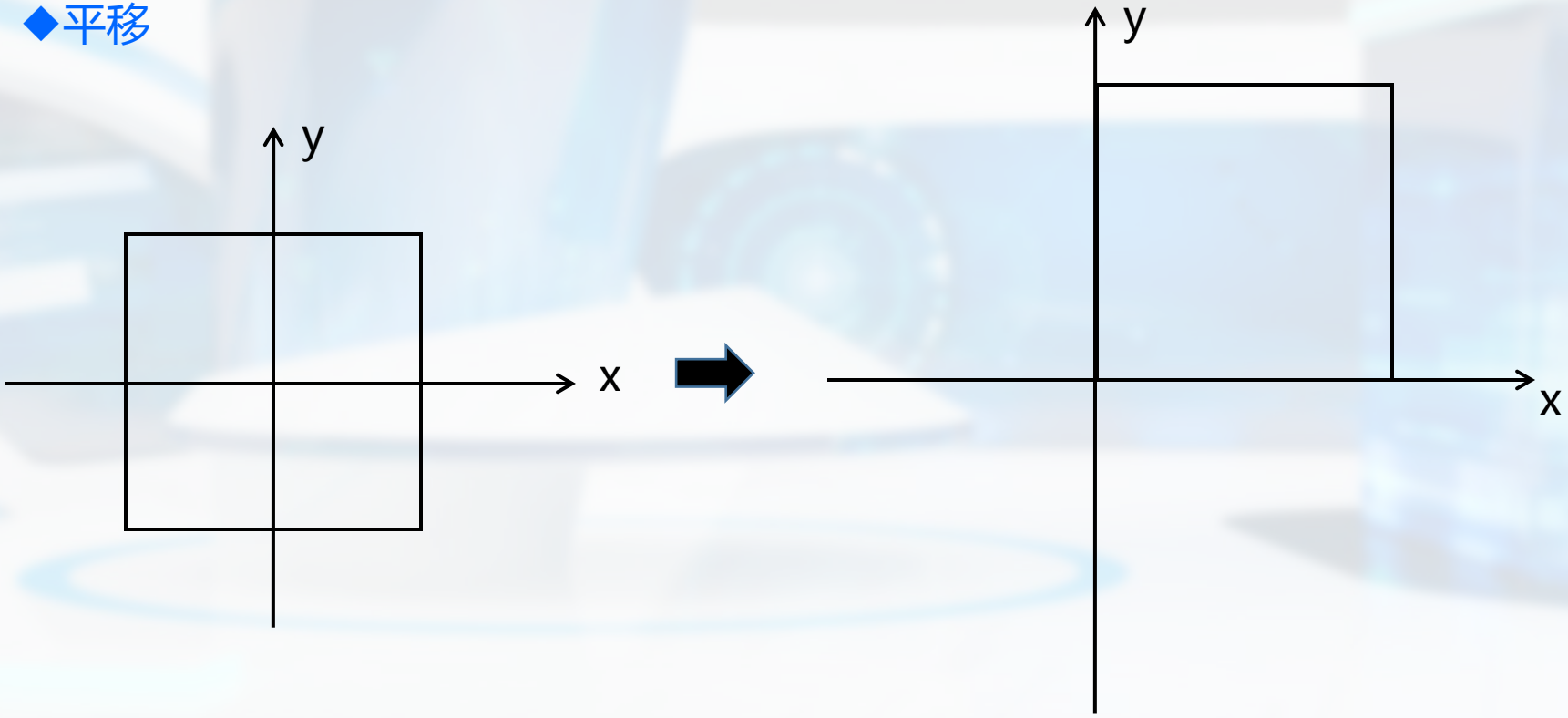
➢XOY平面上的变化



1 屏幕映射

过程分析：

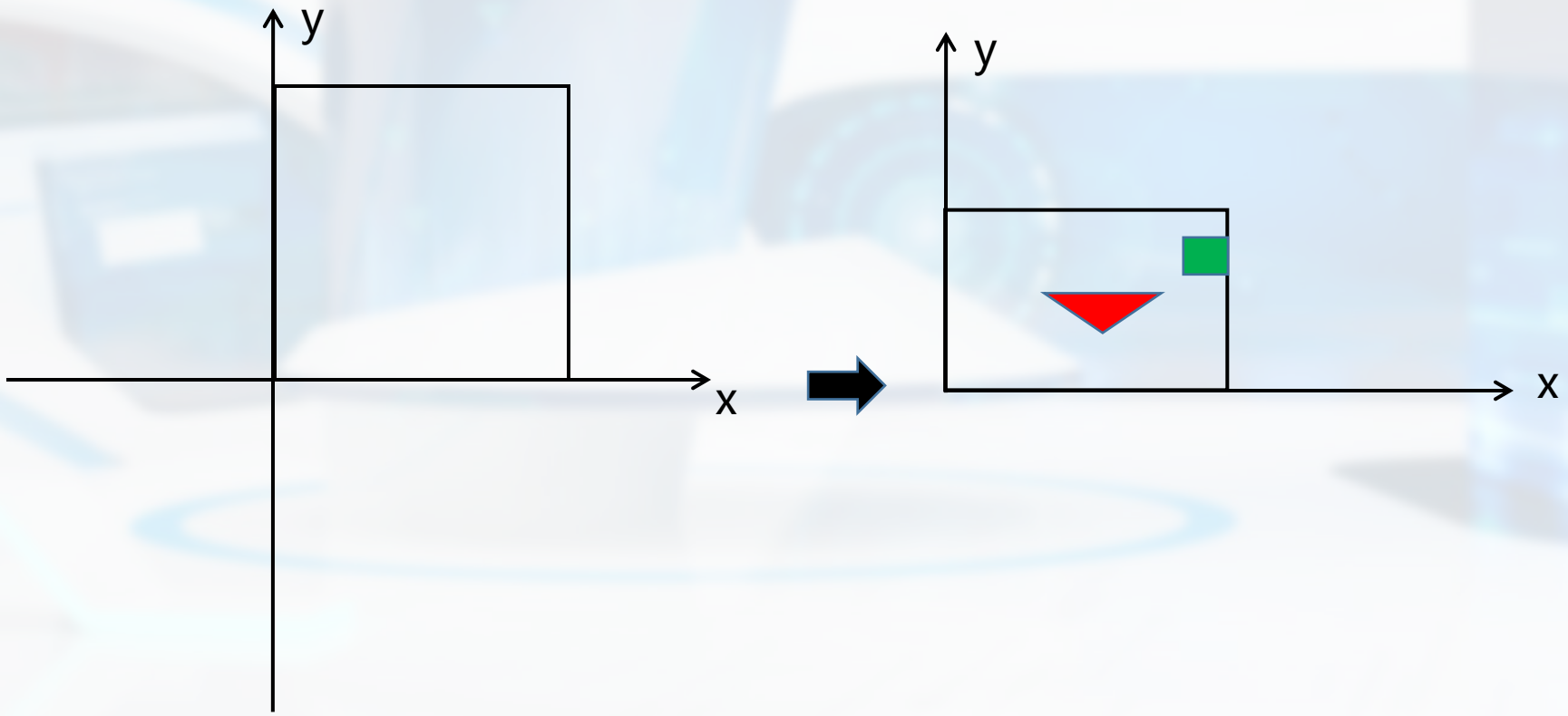
◆平移



1 屏幕映射

过程分析：

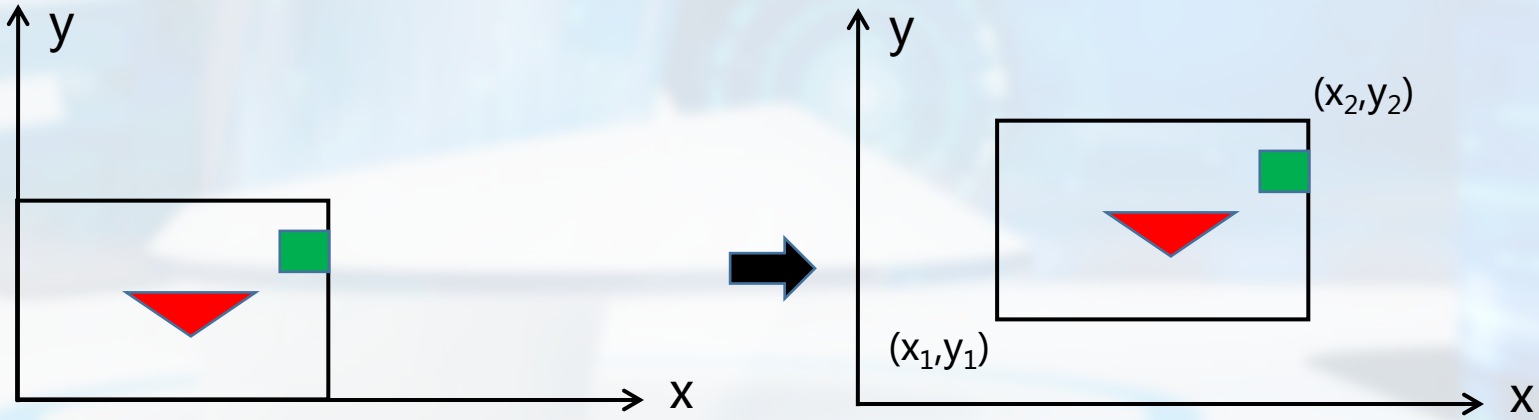
◆比例



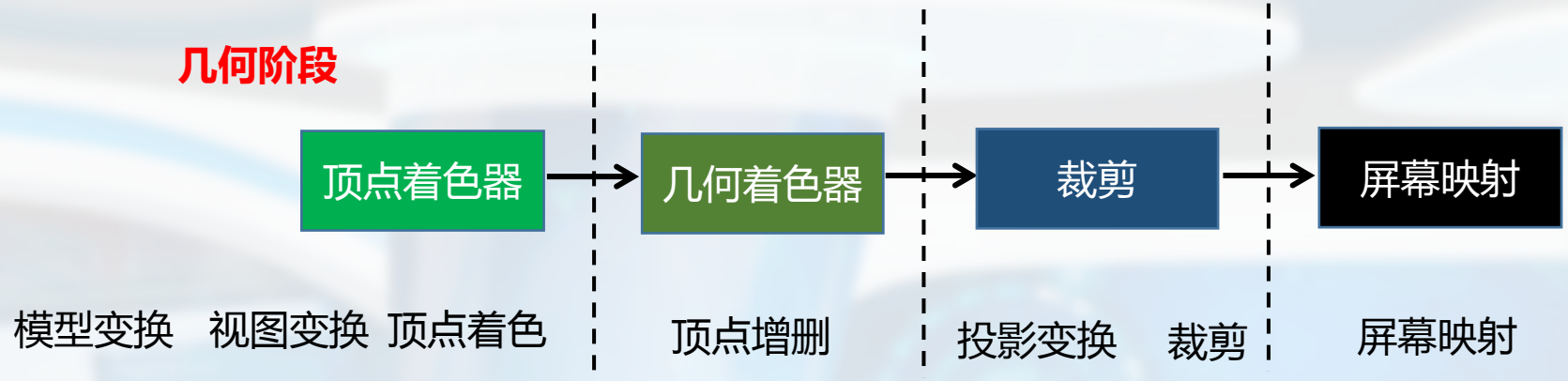
1 屏幕映射

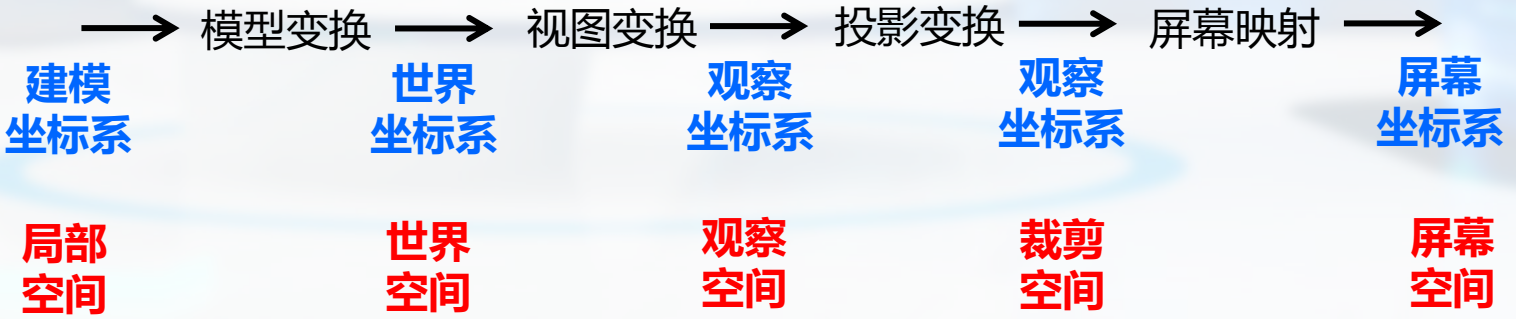
过程分析：

◆反向平移



2 OpenGL的几何阶段实现





2 OpenGL的几何阶段实现

OpenGL中的矩阵

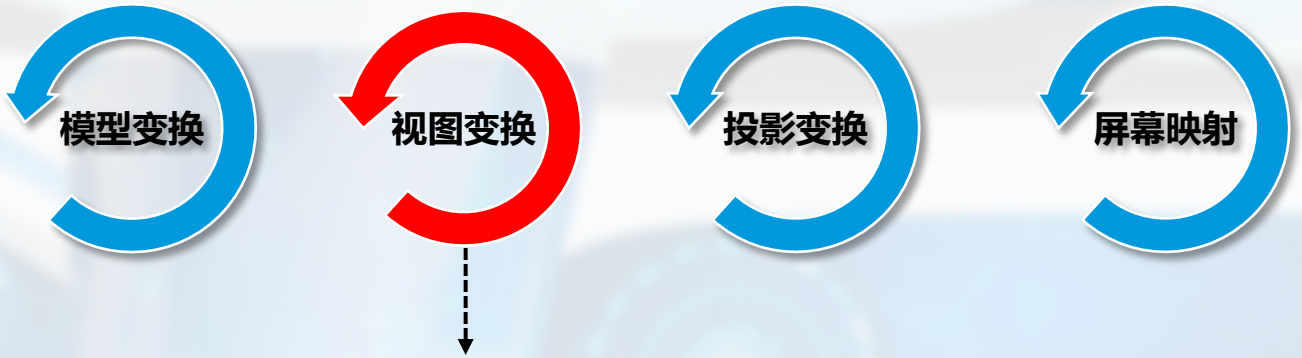
理论部分：6.2 三维模型，动起来

实践部分：OpenGL中用模型矩阵model来实现

```
glm::mat4 model(1);//定义model矩阵
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));//平移
model = glm::rotate(model,(float)glfwGetTime(),glm::vec3(0.0f, 1.0f, 0.0f));//旋转
model = glm::scale(model, glm::vec3(0.5, 0.5, 0.5)); //缩放
```

2 OpenGL的几何阶段实现

OpenGL中的矩阵



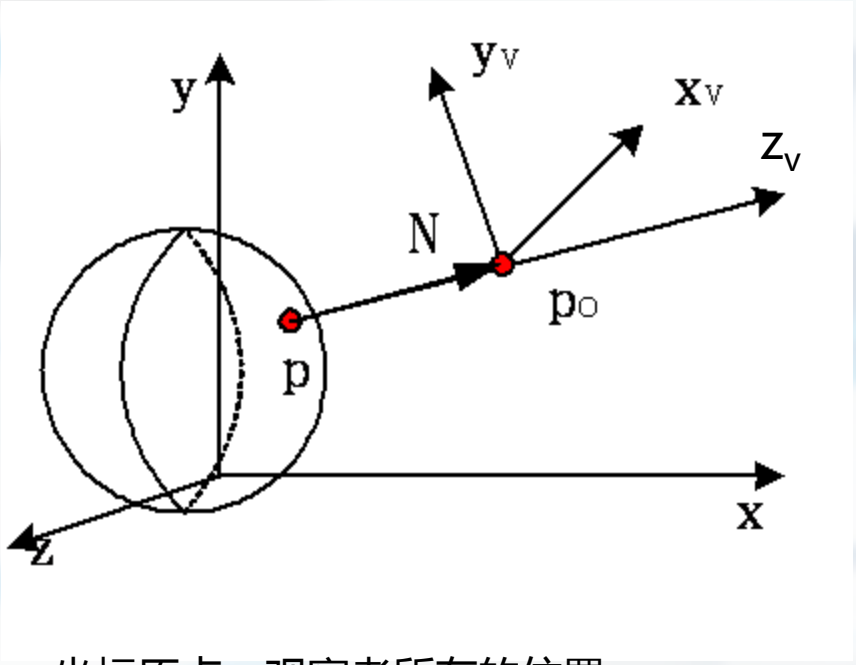
理论部分：6.3 观察者也能动

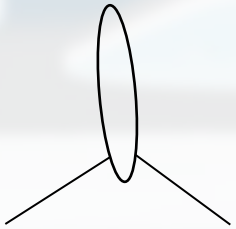
实践部分：OpenGL中用视图矩阵view来实现

```
glm::mat4 view(1);
view=glm::lookAt(camera_position ,camera_position+camera_front, camera_up);
//glm::LookAt函数的参数分别为一个摄像机位置、目标位置和摄像机上向量。
```

2 OpenGL的几何阶段实现

指定观察坐标系：LookAt函数





```
glm::mat4 view(1);
view=glm::lookAt(camera_position ,camera_position+camera_front, camera_up);
//glm::LookAt函数的参数分别为一个摄像机位置、目标位置和摄像机上向量。
```

坐标原点：观察者所在的位置

z_v ：视点和观察物体上焦点的连线

y_v ：向上的方向

x_v ：按照右手定则确定的方向

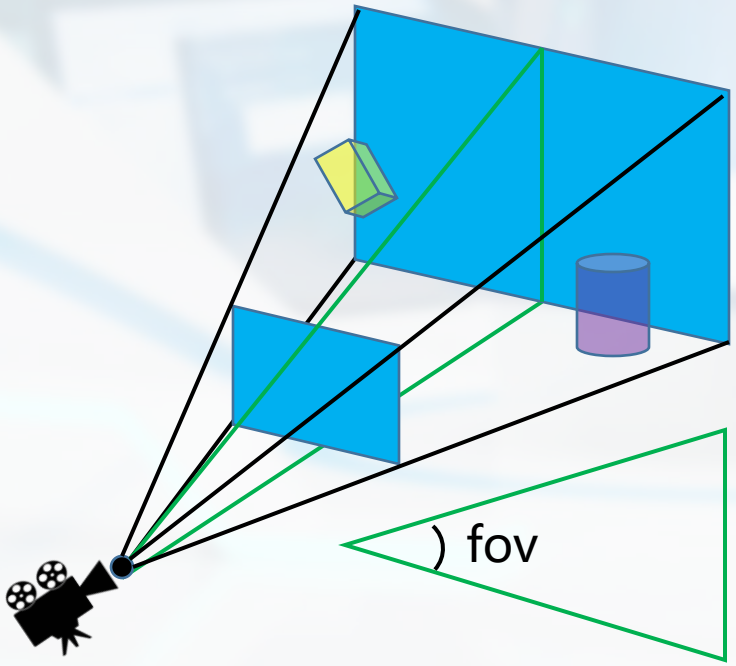
P_0：camera_position

P： camera_position+camera_front

y_v ： camera_up

2 OpenGL的几何阶段实现

OpenGL中的矩阵



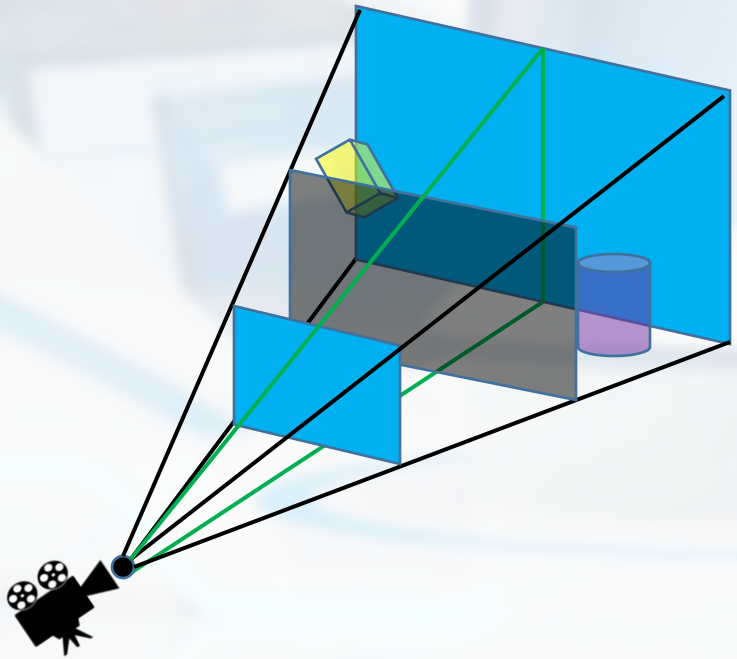
理论部分：7.2 规范化的投影变换

实践部分：OpenGL中用视图矩阵projection来实现

```
glm::mat4 proj =glm::perspective(45.0f, (float)width/(float)height,
0.1f, 100.0f);
```

2 OpenGL的几何阶段实现

指定裁剪空间：以透视投影为例

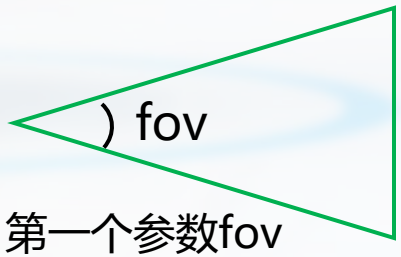


glm::mat4 proj =glm::perspective(45.0f, 1.3f, 0.1f, 100.0f);

在观察空间中指定裁剪空间：

➢第一个参数定义了视野的角度fov

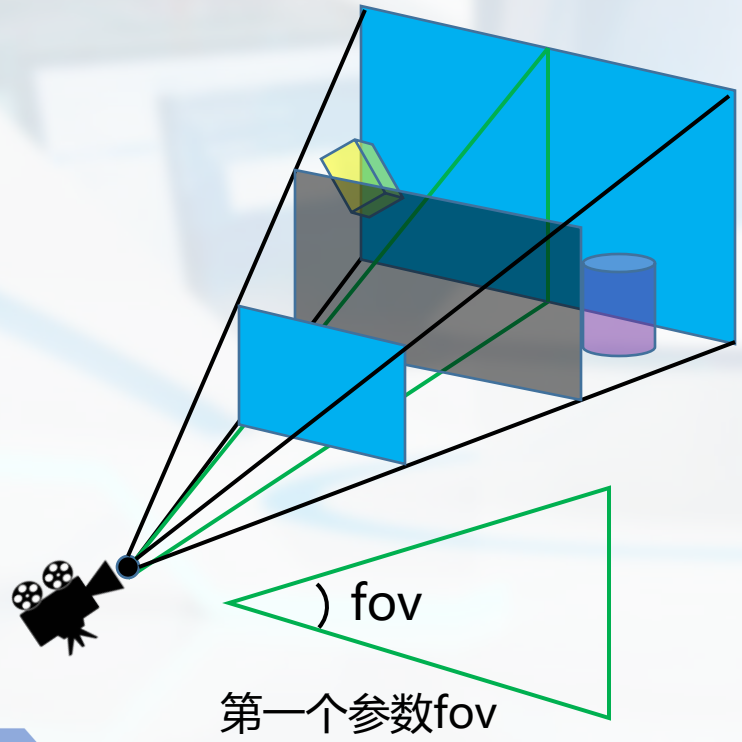
对于一个真实的观察效果，它的值经常设置为45.0，但想要看到更多结果你可以设置一个更大的值。



第一个参数fov

2 OpenGL的几何阶段实现

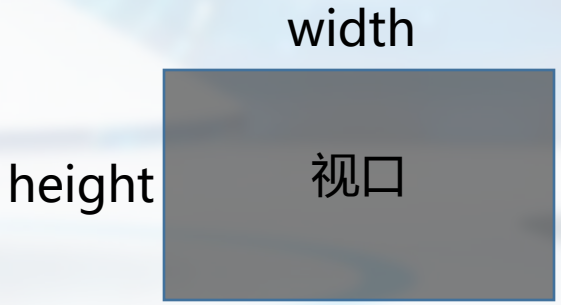
指定裁剪空间：以透视投影为例



glm::mat4 proj =glm::perspective(45.0f, 1.3f, 0.1f, 100.0f);

在观察空间中指定裁剪空间：

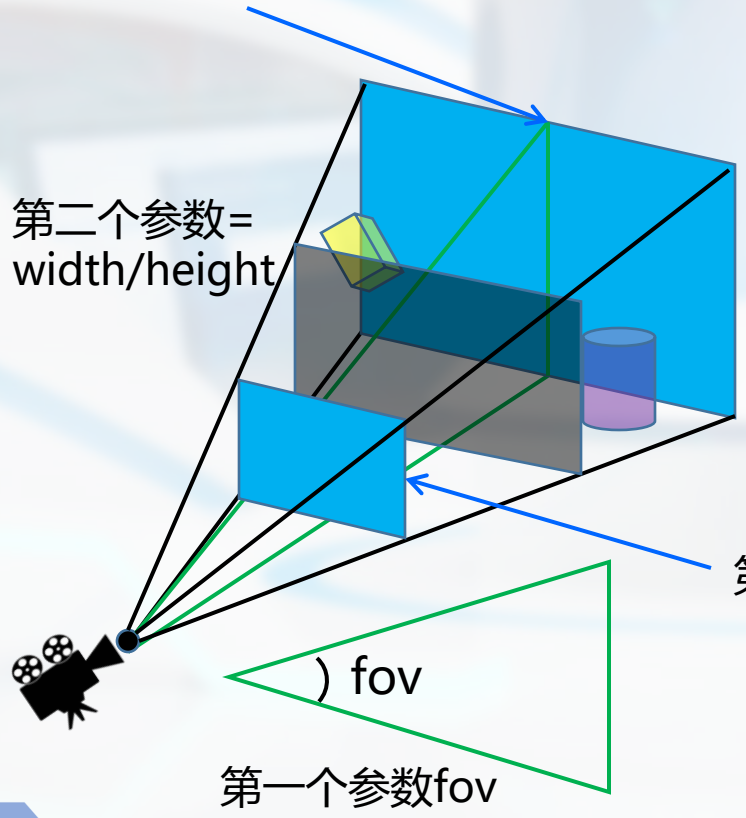
- 第一个参数定义了视野的角度fov
- 第二个参数设置了宽高比，由视口的高除以宽



第二个参数=width/height

2 OpenGL的几何阶段实现

指定裁剪空间：以透视投影为例



glm::mat4 proj =glm::perspective(45.0f, 1.3f, 0.1f, 100.0f);

在观察空间中指定裁剪空间：

- 第一个参数定义了视野的角度fov
- 第二个参数设置了宽高比，由视口的高除以宽
- 第三个参数设置了前截面的位置
- 第四个参数设置了后截面的位置

2 OpenGL的几何阶段实现

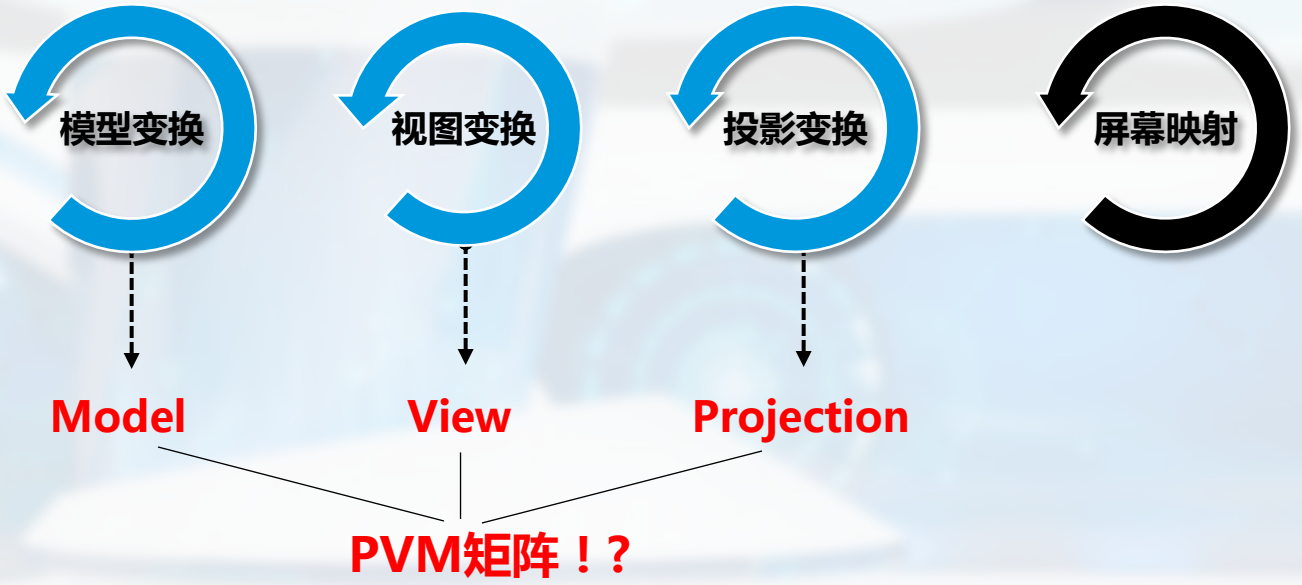
OpenGL中的矩阵

理论部分：本节内容

实践部分：GPU渲染管线中固定部分

2 几何阶段实例

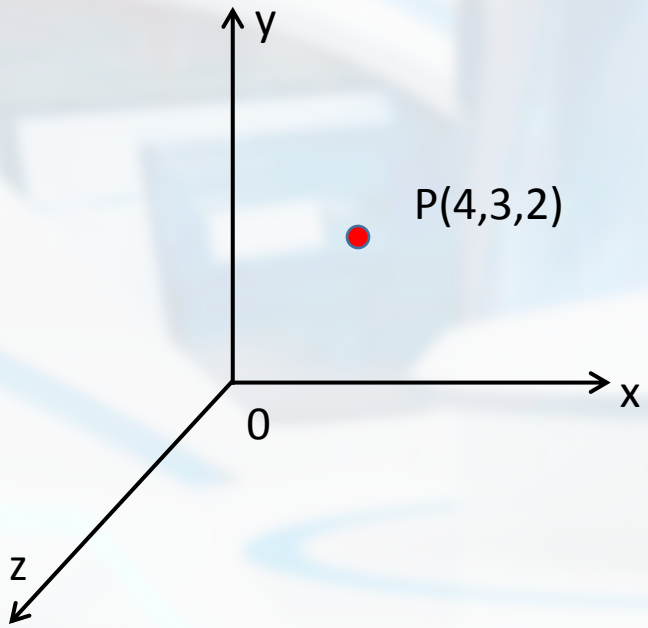
OpenGL中的矩阵




```
gl_Position = projection * view * model * vec4(aPos, 1.0);
//注意变换的顺序不能出错，在计算时，是从右往左进行计算
```

2 几何阶段实例

之前的变换基于行矩阵：



三维坐标系下点p(x,y,z)变换后为p′（x′，y′，z′）：

则所有的变换可以用矩阵T_{3D}来表示！

行矩阵

$$p'=\begin{bmatrix}x' & y' & z' & 1\end{bmatrix}=p\cdot T_{3D}=\begin{bmatrix}x & y & z & 1\end{bmatrix}\cdot\left[\begin{array}{ccc|c} a & b & c & p \\ d & e & f & q \\ h & i & j & r \\ \hline l & m & n & s \end{array}\right]$$

T_1 T_3 T_2 T_4

2 几何阶段实例

现在在OpenGL中改为列矩阵：

三维坐标系下点p(x,y,z)变换后为p′（x′，y′，z′）：

则所有的变换可以用矩阵T_{3D}来表示！

行矩阵

$$p'=\begin{bmatrix}x' & y' & z' & 1\end{bmatrix}=p\cdot T_{3D}=\begin{bmatrix}x & y & z & 1\end{bmatrix}\cdot\begin{bmatrix}a & b & c & p\\ d & e & f & q\\ h & i & j & r\\ l & m & n & s\end{bmatrix}$$

$$p'=\begin{bmatrix}x'\\ y'\\ z'\\ 1\end{bmatrix}=T_{3D}\cdot p=\begin{bmatrix}a & d & h & l\\ b & e & i & m\\ c & f & j & n\\ p & q & r & s\end{bmatrix}\cdot\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

列矩阵

2 OpenGL的几何阶段实现

OpenGL中的PVM观察变换

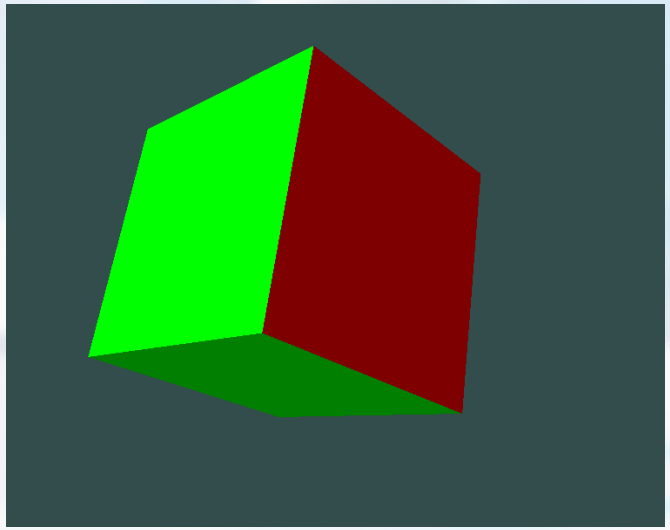
```
gl_Position = projection * view * model * vec4(aPos, 1.0);
```

P V M 观 察 变 换

2 OpenGL的几何阶段实现

实验　旋转的立方体

立方体会根据时间，自动旋转。




谢谢
软件学院　万琳