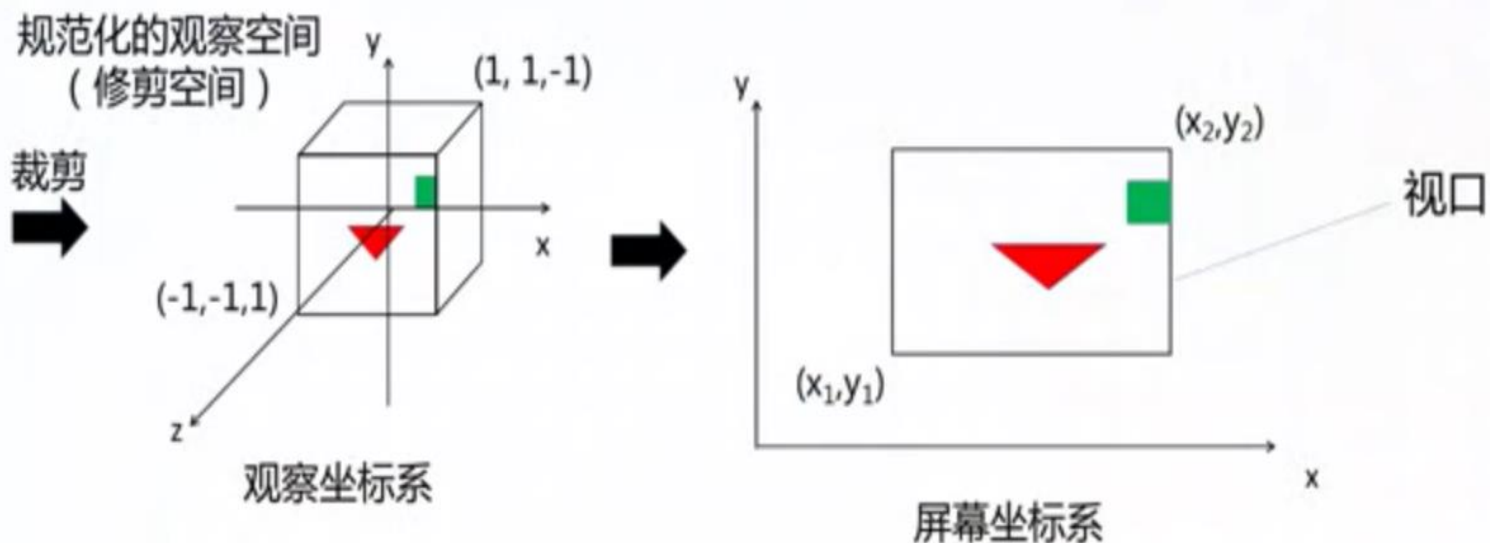
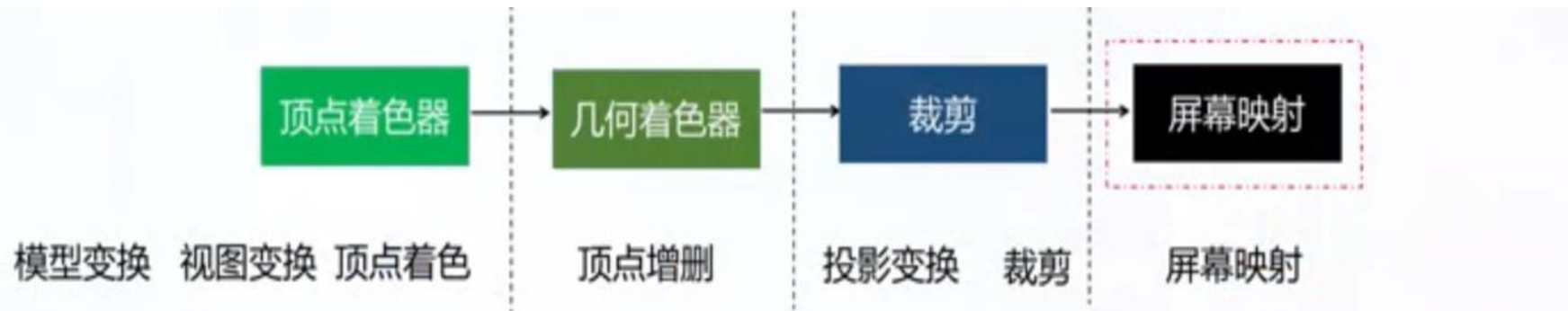
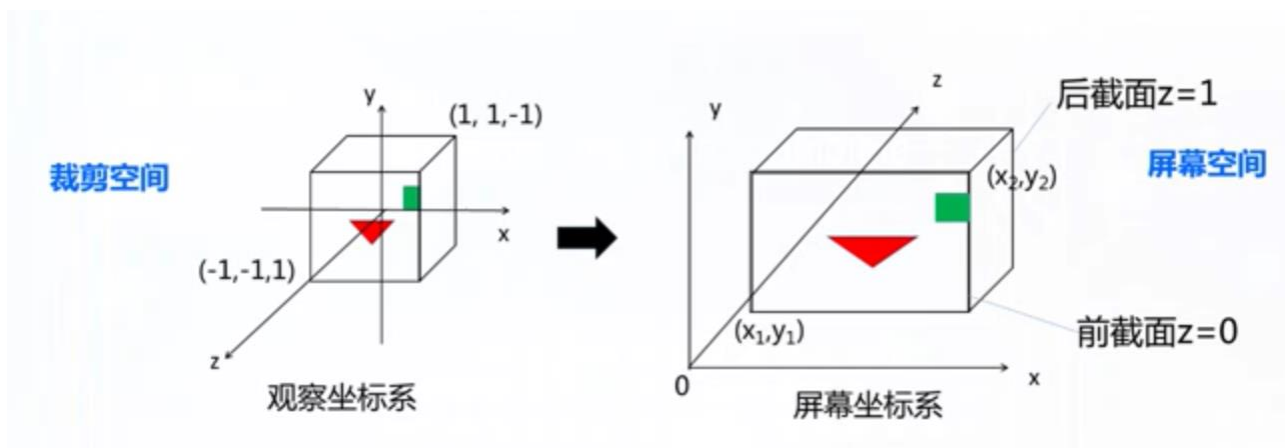


# 观察流程



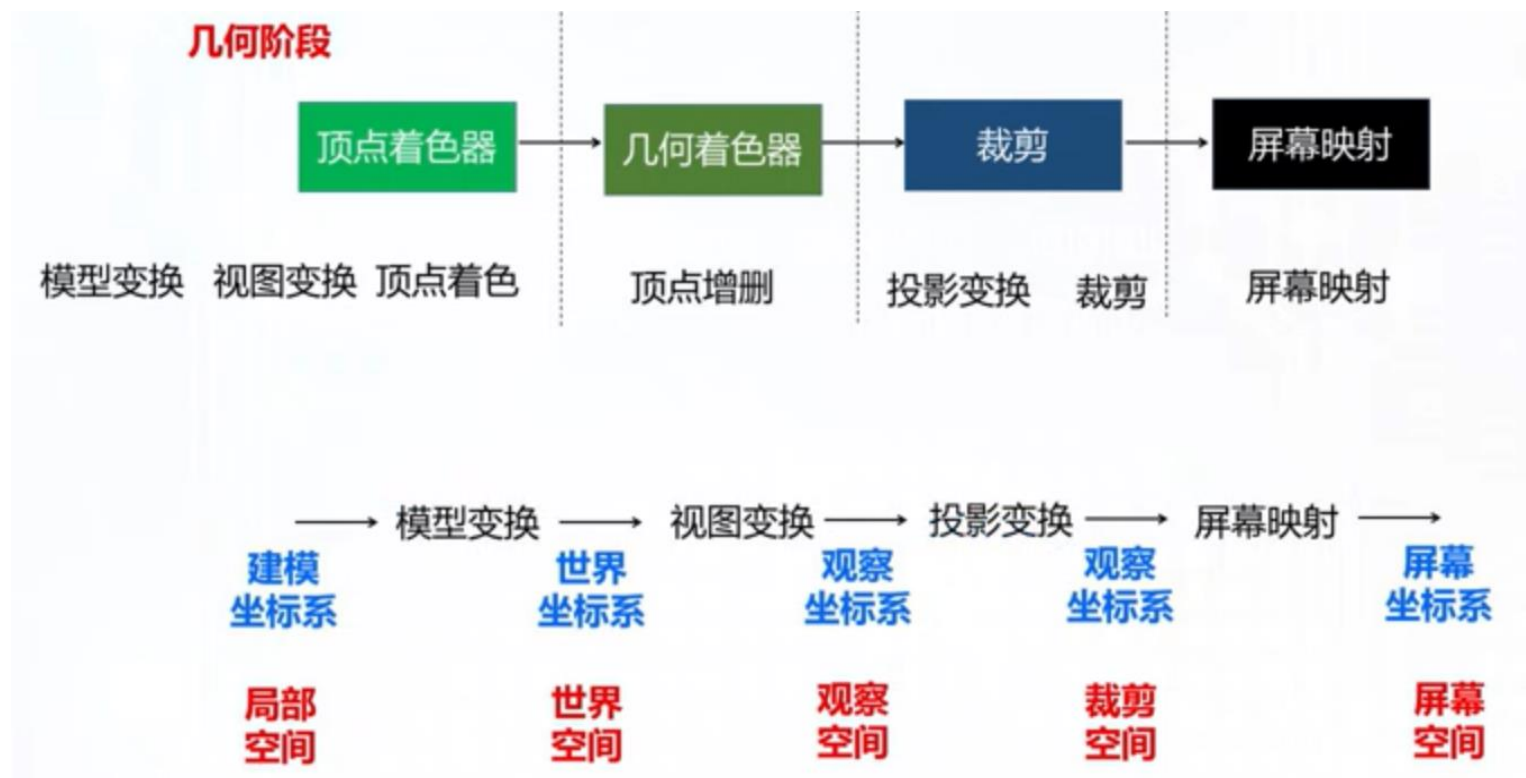
# 屏幕空间

- 注意还是3D的，并且是左手系

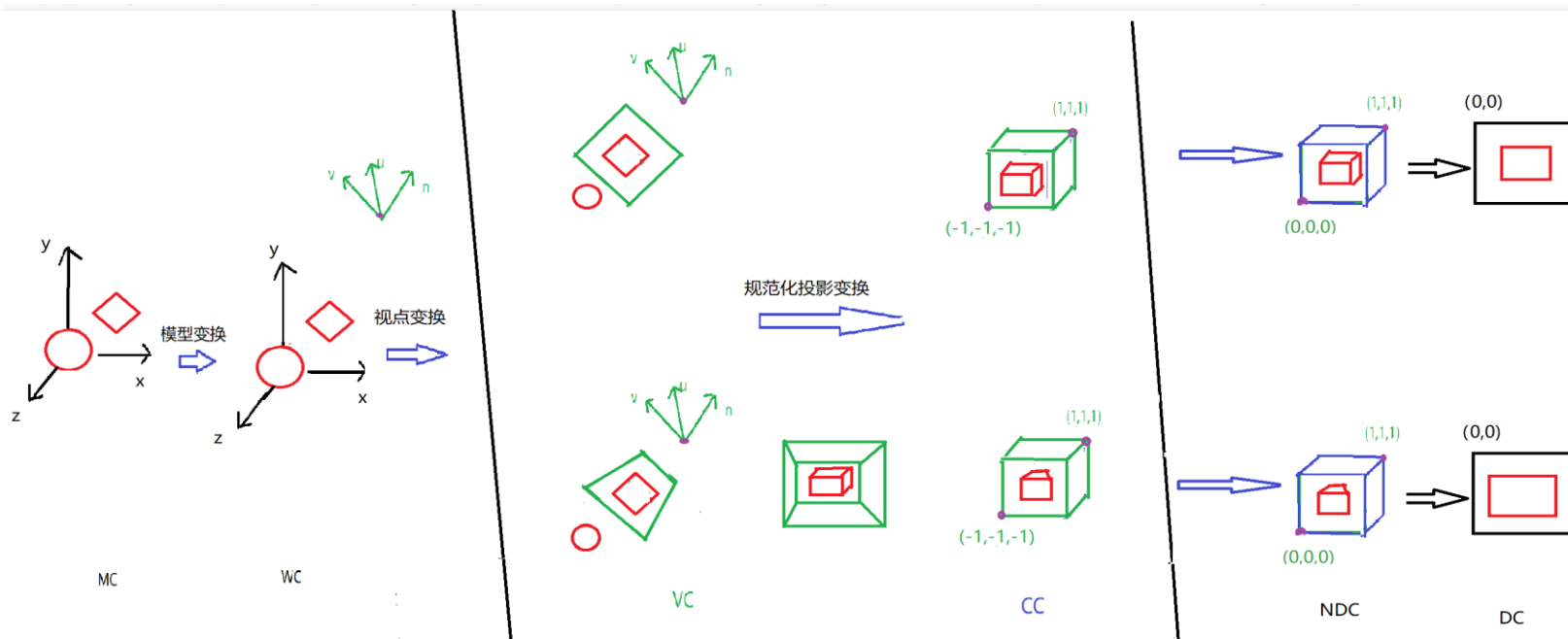


# 观察流程中的变换

- 四种类型变换
- 屏幕映射系统自动完成（窗区视区变换）



# 需要编程实现MVP变换



- **M模型变换Model Trans.**
  - 物体局部坐标**MC**转换到全局坐标**WC**下
- **V观察变换View Trans.**
  - 在**WC**下设定相机和观察方向，将物体坐标转换到**VC**坐标
- **P: 投影变换Projection Trans.**
  - 在**VC**下取窗口，将视见体物体坐标转换到规范化视见体中**CC**坐标

# OpenGL顶点着色器实现MVP

- 采用列向量表示顶点，矩阵乘法顺序PVM

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
```

# Webgl观察流水几何处理实例

- 其它实例参见QQ群文件”lec12WEBGL观察实例”，（也可参angel实例ref2）

- 1.CubeTrackBall
- 2.SunPlanet
- 3.Honolulu&Hat
- 4.FigureRobot
- CC-LEFTHAND-孜孜以求
- Common
- ref1-wave
- ref2-wirehat
- 1.CubeTrackBall立方体建模和虚拟跟踪球.ppt
- 2.SunPlanet过程建模.ppt
- 3.Honolulu&Hat结构化网格模型.ppt
- 4.Figure层次建模.ppt
- readme.txt
- 编程注意事项.ppt
- 观察流水线总结.txt
- 图3D观察流程FromMCtoDC.png

# webGL裁剪坐标:左手坐标系

## ■ 参见“webGL编程指南”附录D 总结

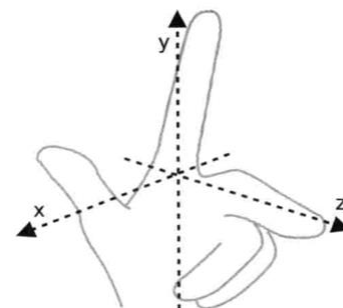
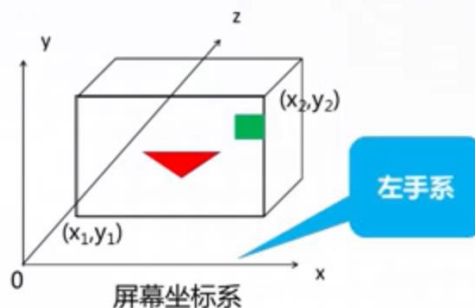
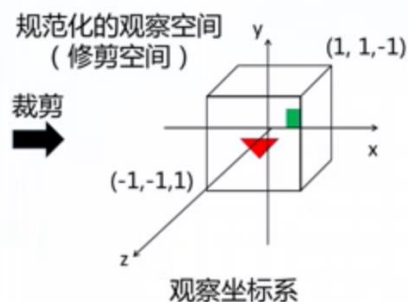


图 D.4 左手坐标系

总之，我们知道 WebGL 并不强制使用右手或左手坐标系。我们了解了，大多数 WebGL 库和 WebGL 程序都采用了传统的右手坐标系，本书也是这样做的。但是 WebGL 的默认行为（比如，在裁剪空间中使用左手坐标系）却与此冲突。为了解决这个冲突，我们可以通过翻转  $z$  坐标值进行补偿，这样就能够继续使用传统的右手坐标系了。但是，如前所述，这只是一个传统，只是大多数人遵守而已。如果你不了解 WebGL 的默认行为，以及处理它的过程，搞不好什么时候这个问题就会出来难为你了。



# 解决方法

- 在MC,WC,VC下都统一采用右手坐标系进行计算，在顶点着色器程序最后进行Z反向处理即可。

```
mat4 M_InverseZ = mat4(  
    1.0, 0.0, 0.0, 0.0,  
    0.0, 1.0, 0.0, 0.0,  
    0.0, 0.0, -1.0, 0.0,  
    0.0, 0.0, 0.0, 1.0  
);  
gl_Position = M_InverseZ*vPosition;
```

！如果用系统提供的API函数（如投影函数）已经进行了处理，就不用进行反Z操作

！注意：shader中的矩阵是“列先序”存储的；向量是“列向量”