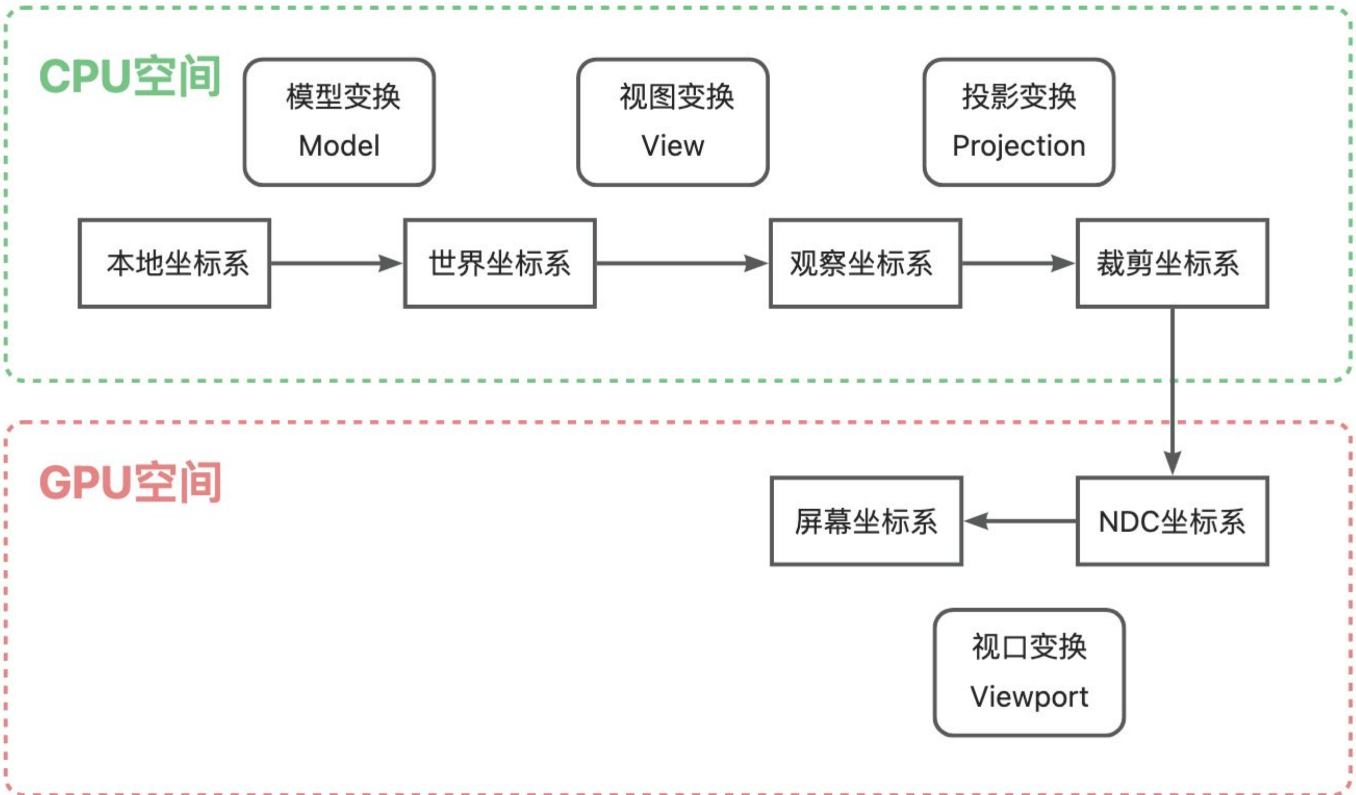
6-5 坐标系统

一、坐标系处理流程



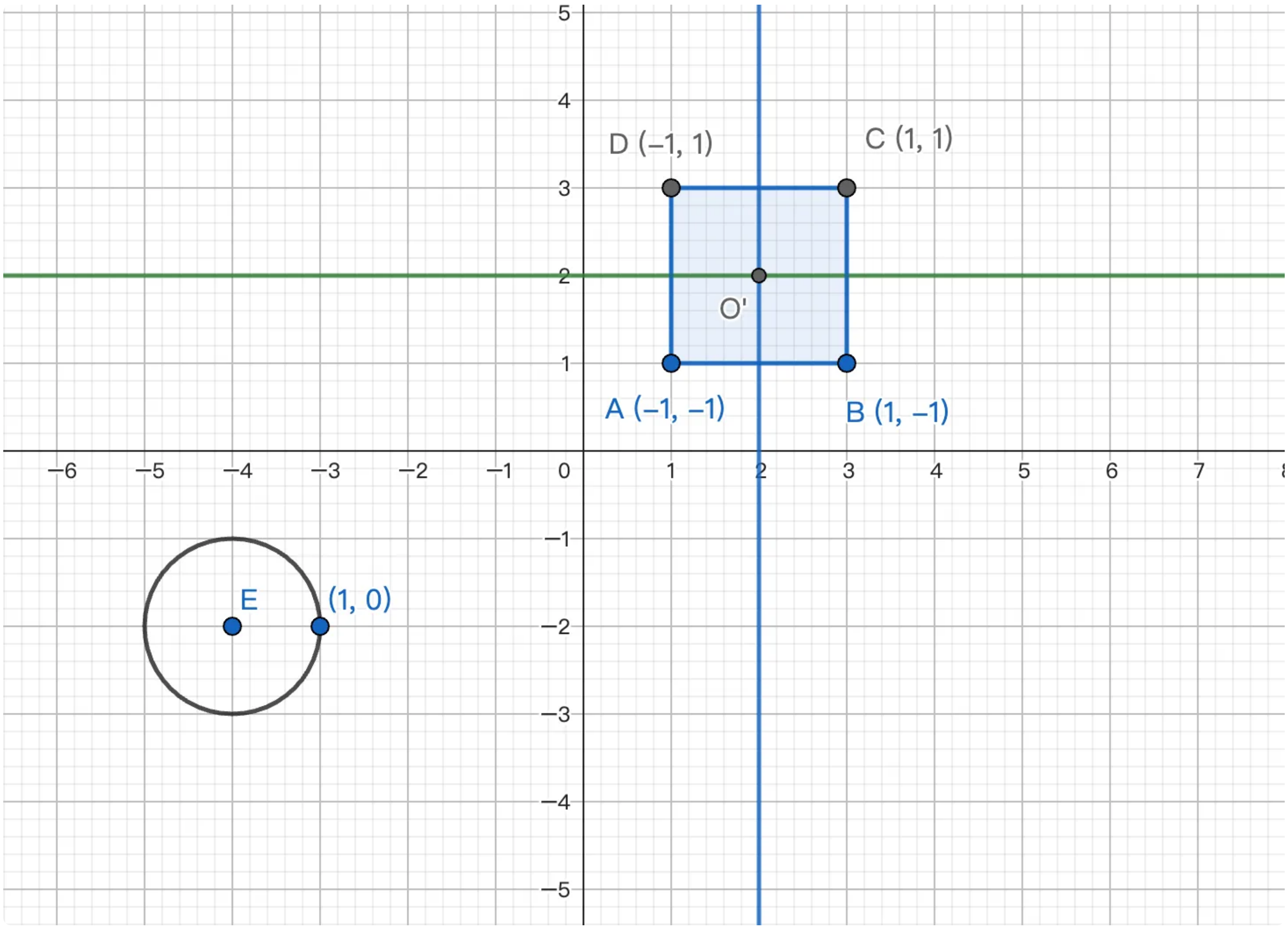
- CPU 中将本地坐标转换成裁剪坐标
  - 顶点在本地坐标系中的坐标经过模型变换，转换到世界坐标系中
  - 然后通过相机观察这个世界，进行视图变换, 将物体从世界坐标系转换到观察坐标系
  - 然后进行投影变换，将物体从观察坐标系转换到裁剪坐标系
- GPU 接收CPU 传递过来的裁剪坐标
  - 接收裁剪坐标，通过透视除法，将裁剪坐标转换成 NDC 坐标
  - GPU 将 NDC 坐标通过视口变换，渲染到屏幕上

二、坐标系分类

(一) 本地坐标系

> 以自身 `几何中心` 为坐标原点构建的坐标系

由于这个概念过于抽象. 我们通过一些实例来理解效果比较好



上图, 蓝色的矩阵有4个顶点, 我们可以认为

- A点的本地坐标是: (−1, −1)
- B点的本地坐标是: (1, −1)
- C点的本地坐标是: (1, 1)
- D点的本地坐标是: (−1, 1)

这样的坐标值就是 `本地坐标系` 下的坐标值

由于这些坐标是相对于自身的 `几何中心O'点` , 因此, 不管这个矩形在哪里, 本地坐标都是不变的, 计算很方便

> **本地空间**
>
> 以 `本地坐标系` 为基础构建的空间就是 `本地空间` (也叫 `模型空间`)

(二) 世界坐标系

但是, 在一个世界(空间)中, 不可能只有一个物体, 要描述整个世界(包含多个物体)时, 本地坐标反而不好描述了

因此, 规定了一个统一的坐标系(世界坐标系), 所有物体都在同一个 `世界坐标系` 下描述

> 可以参考 `地心说与日心说` 的故事
>
> - 地心说相当于本地坐标系
> - 日心说相当于世界坐标系

▼ 地心说与日心说

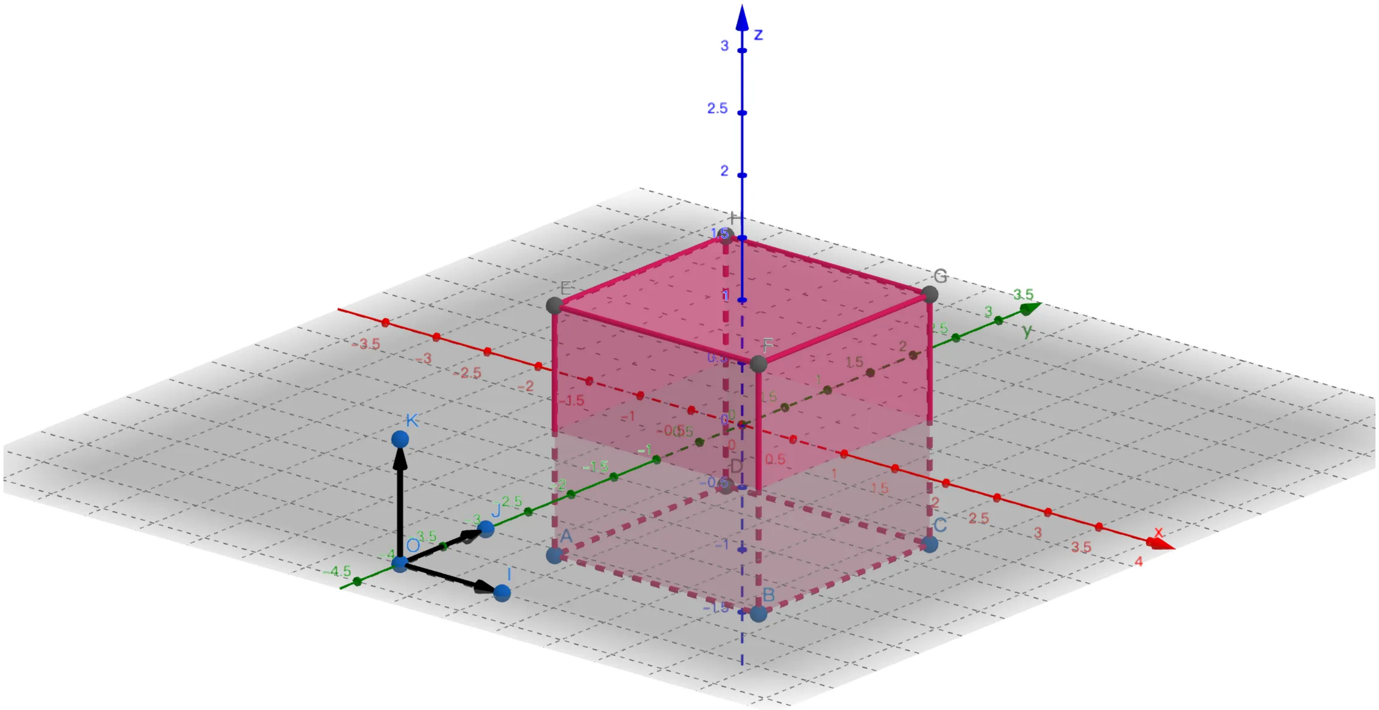
百度百科–地心说和日心说

在上述图中.

- A点的世界坐标是: (1, 1)
- B点的世界坐标是: (3, 1)
- C点的世界坐标是: (3, 3)
- D点的世界坐标是: (1, 3)

> **世界空间**
>
> 以 `世界坐标系` 为基础构建的空间就是 `世界空间`

(三) 观察坐标系



这里 `O'` 就是相机所在的 `世界坐标系` 下的位置.

- 以 `O'` 为原点
- 以视线方向为 `z轴` (方向是从视点指向目标点)
- 再依据上方向构建 `x轴`
- 最后计算 `y轴`

形成的 `左手坐标系` , 相当于以 `相机为中心` 观察整个世界

> 1. 以相机为中心就是所谓的 `第一人称视角`
> 2. 以世界为中心就是所谓的 `第三人称视角`

> **可视空间**
>
> 经过 `正交参数` 或者 `透视参数` 配合相机位置能够确定世界空间中哪些部分是可以被看到的, 这就是 `可视空间`
>
> 超出 `可视空间` 外的部分会被 '裁剪' 掉

(四) 裁剪坐标系

经过投影变换后, 得到的坐标的xyz值, 范围在[-1, 1]之间, w值不一定是1, 然后传递给 `gl_Position`

这些坐标值就是在 `裁剪坐标系` 中的值

不论在 `世界坐标` 中的值是1000, 还是10000, 经过 `视图变换` 和 `投影变换` 后, 坐标值都会落在 `裁剪坐标系` 的[-1, 1]区间

> 这里 `裁剪坐标系` 中的[-1, 1]相当于 `世界坐标系` 中的[-∞, +∞]

> **裁剪空间**
>
> 一个[-1,1]的立方体, 投影变换就是将 `可视空间` 映射到 `裁剪空间`

(五) NDC坐标系

NDC坐标系(Normalize Device Coordinates): 归一化设备坐标.

我们在计算 `透视投影` 时一个点的齐次坐标w分量不一定为1

将一个顶点, 比如(2,2,2,2)传给 `gl_Position` 后, GPU会执行 `透视除法` , 将xyz的值同时除以w, 就得到NDC坐标值了

```glsl
gl_Position = vec4(2,2,2,2)
gl_Position = vec4(1,1,1,1) // 表示同一个点
```

任何一个顶点(w≠0)都可以做如下转化, 但是表示的含义不变

```glsl
(x, y, z, w)
(x/w, y/w, z/w, 1)
```

> 1. NDC坐标系是GPU处理的
> 2. NDC坐标系跟裁剪坐标系的值都在[-1, 1]的区间, 但是w分量不同

(六) 屏幕坐标系

屏幕是有像素组成的. 屏幕坐标系是以像素值为单位的

GPU在渲染到屏幕时, 会根据 `gl.viewport` 这个API, 将[-1, 1]之间的值转换成视口大小.

三、坐标变换

(一) 模型变换

通过将图形上每个点的 `本地坐标` 应用一个变换矩阵, 就可以求得 `世界坐标` .

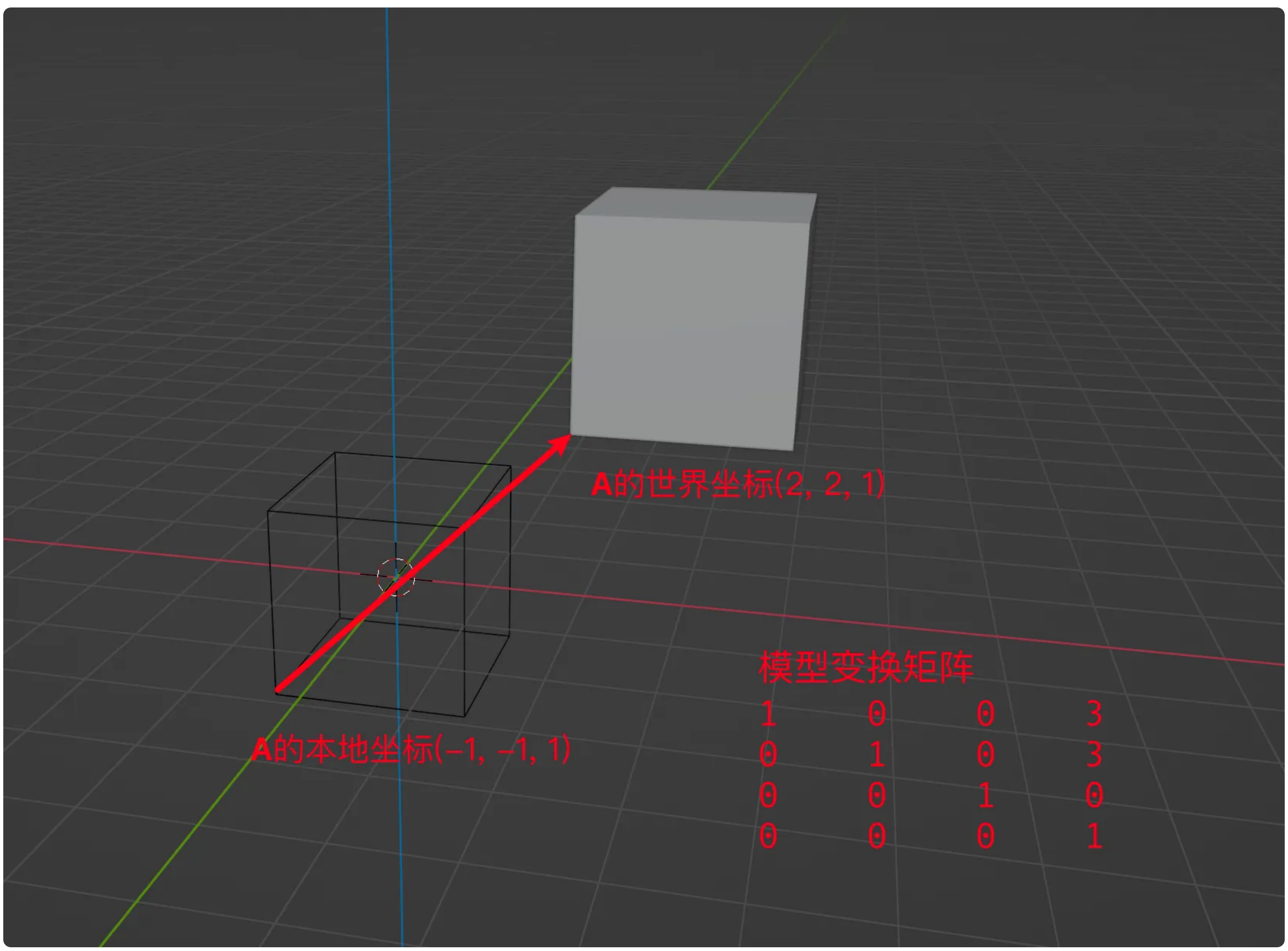
这个变换矩阵就是 `模型矩阵` (平移, 旋转, 缩放的 `复合矩阵`)

> 举例

$$\begin{pmatrix}1 & 0 & 0 & 3\\0 & 1 & 0 & 3\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}-1\\-1\\1\\1\end{pmatrix}=\begin{pmatrix}2\\2\\1\\1\end{pmatrix}$$

我们对 `A点的本地坐标` 应用 `平移矩阵` 就会得到 `A点的世界坐标`

同理, 对其他点应用 `平移矩阵` 都可以将本地坐标转换成世界坐标



(二) 视图变换

假设

- 视点: (3, 3, 5)
- 目标点(3, 3, 0)
- 上方向(0, 1, 0)

A点经过视图变换后的坐标值

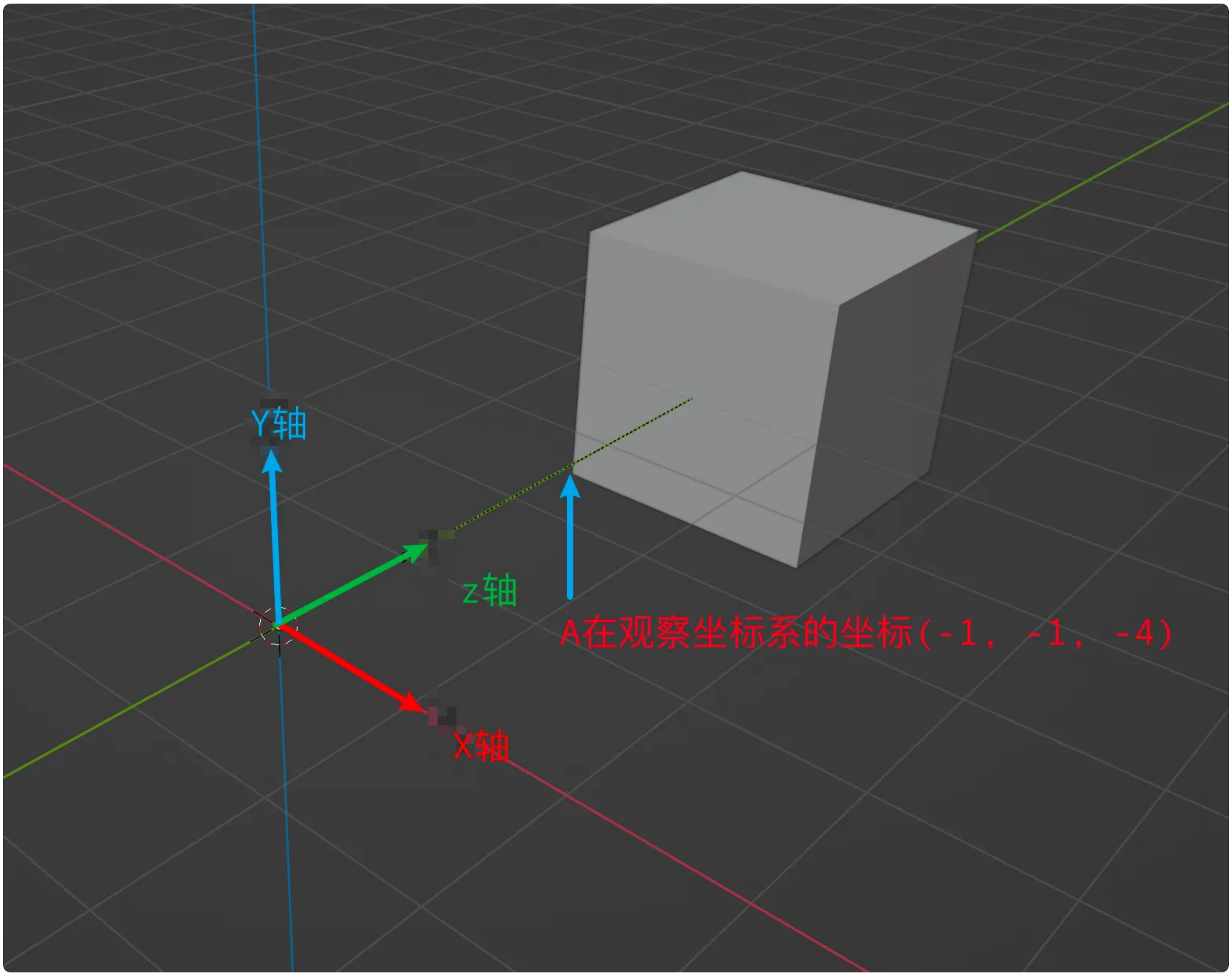
$$\begin{pmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ -4 \\ 1 \end{pmatrix}$$

对A点的 `世界坐标值` 应用 `视图矩阵` 得到的还是 `世界坐标系下的坐标值`

这里确实有点费解.

因为 `视图矩阵` 同时作用于相机和物体, 等效于对物体应用变换

> **注意**
>
> 在观察坐标系下, A点的z值应该为正, 因为**观察坐标系是左手坐标系**
>
> A点在 `观察坐标系` 下的坐标值应该是(–1, –1, 4, 1), 而不是(–1, –1, –4, 1)
>
> 这就是在做投影变换时, 对z反转的原因.



(三) 投影变换

假设做正交投影, 参数如下:

- left: –5
- right: 5
- top: 5
- bottom: –5
- near: 1
- far: 10

这里的数值都是 `观察坐标系` 下的数据.

正常情况下的 `正交投影矩阵` 应该如下:

$$\begin{pmatrix} \frac{2}{10} & 0 & 0 & 0 \\ 0 & \frac{2}{10} & 0 & 0 \\ 0 & 0 & \frac{2}{9} & -\frac{11}{9} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

但是这个投影矩阵需要作用于观察坐标系的A点(–1, –1, 4, 1).为了计算方便, 就做了一个负值调整

$$\begin{pmatrix} \frac{2}{10} & 0 & 0 & 0 \\ 0 & \frac{2}{10} & 0 & 0 \\ 0 & 0 & -\frac{2}{9} & -\frac{11}{9} \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ -1 \\ -4 \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{2}{10} \\ -\frac{2}{10} \\ -\frac{3}{9} \\ 1 \end{pmatrix}$$

(四) 整体公式

$$P_M \cdot V_M \cdot M_M \cdot A = A'$$

$$\begin{pmatrix} \frac{2}{10} & 0 & 0 & 0 \\ 0 & \frac{2}{10} & 0 & 0 \\ 0 & 0 & -\frac{2}{9} & -\frac{11}{9} \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{2}{10} \\ -\frac{2}{10} \\ -\frac{3}{9} \\ 1 \end{pmatrix}$$