3D Technical Introduction

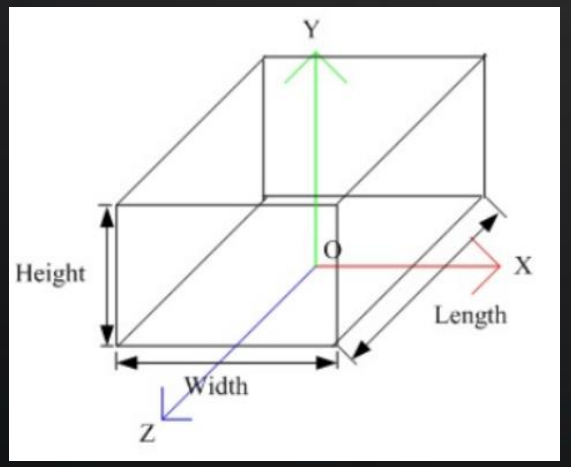
（四）坐标系与Camera投影原理

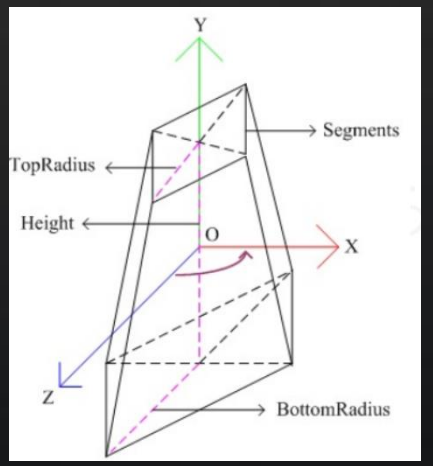
王丰 （Feng WANG）

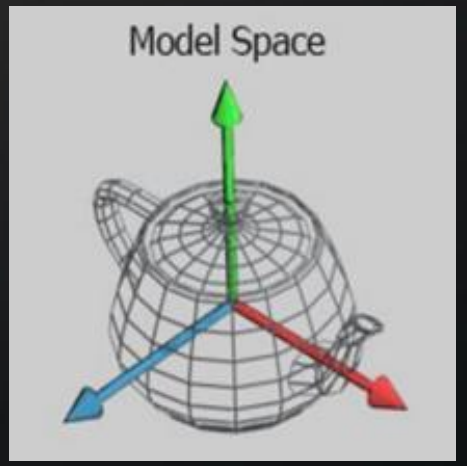
模型坐标系

1. 基本概念

在为物体实体构建网格（Mesh）模型时，用于各顶点（Vertex）的参考坐标系，称之为模型坐标系。
在模型坐标系中，各个顶点的值，不会随着物体在空间的不同位置、旋转、缩放而发生变化。







一般来说，模型的中心点要放置在模型坐标系的原点

2. 常用建模工具

3DMax，Blend等，这些工具可以导出obj文件，一种标准的mesh数据。
不过不同工具出于不同的目的，一般有自己的存储格式。

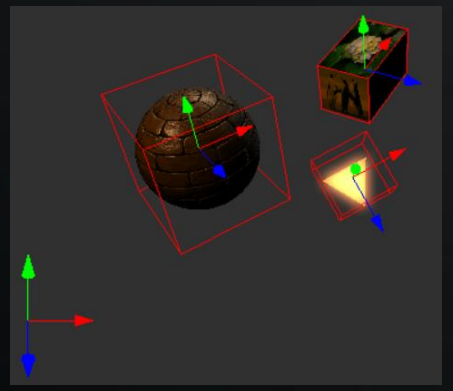
惯性坐标系

1. 基本概念

把一个网格模型放到3D场景里时，这个模型必然有固定的位置、旋转角度、缩放等信息，这个模型跟其他模型也会构成相对位置关系，这些就衍生出了惯性坐标系的概念。

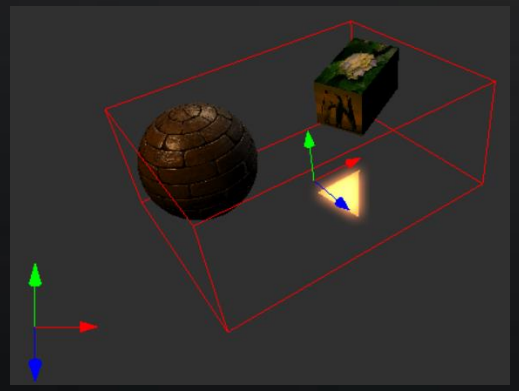
1）Local坐标系

直接与网格模型关联



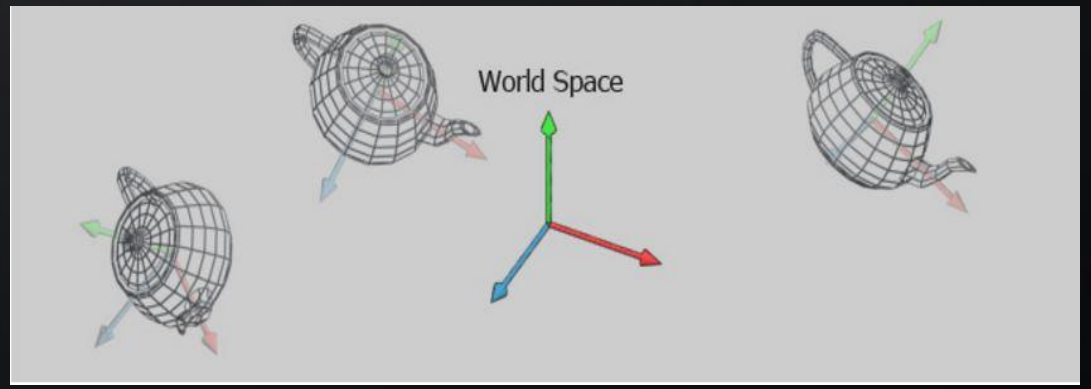
2）父坐标系

一个抽象概念，用于构建坐标系的嵌套关系



3）世界坐标系

一个抽象概念，一般保持不变。可以定义一个root父坐标系，与世界坐标系重合，但是可以改变其参数



Scale：本地坐标系坐标轴的刻度精细度，网格模型沿本地坐标轴缩放。
Rotate：本地坐标系坐标轴的朝向，以父坐标系坐标轴的朝向为参照。
Position：本地坐标系原点在父坐标系中的位置。

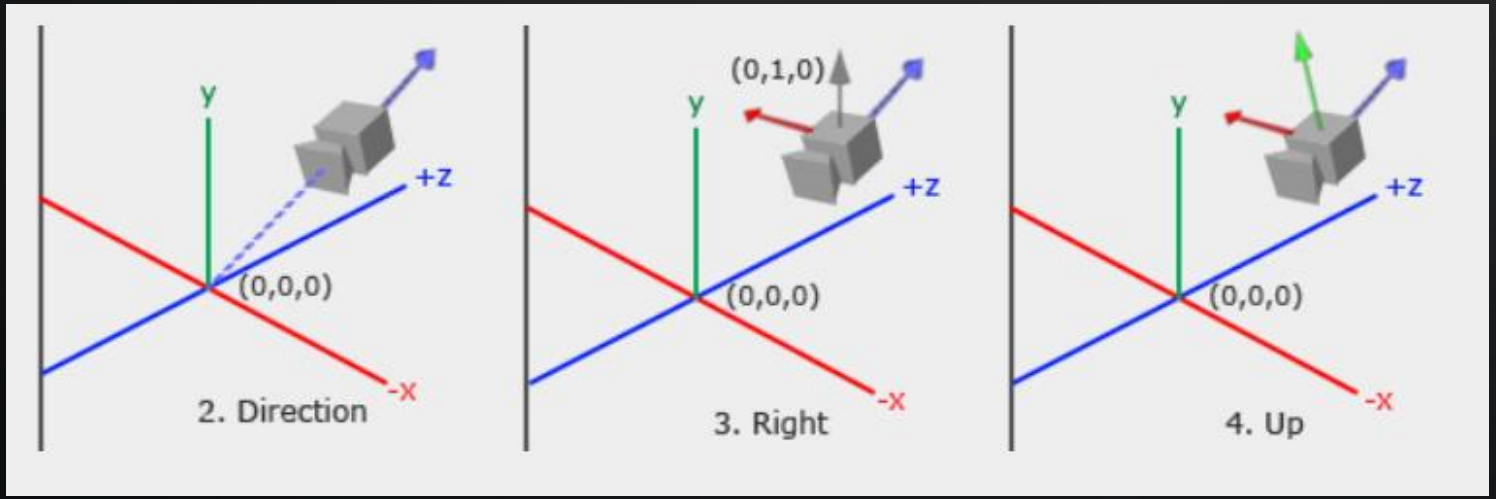
2. 累积原则

mtxModel = mtxScale * (mtxRY * mtxRX * mtxRZ) * mtxTranslate * mtxParent1 * mtxParentN;

Camera

1. View矩阵

在3D虚拟世界中，摄像机就像人的眼睛，用来观看整个3D场景空间，并将观看到的场景，经过视角变换和投影变换后，在2DViewport中呈现。



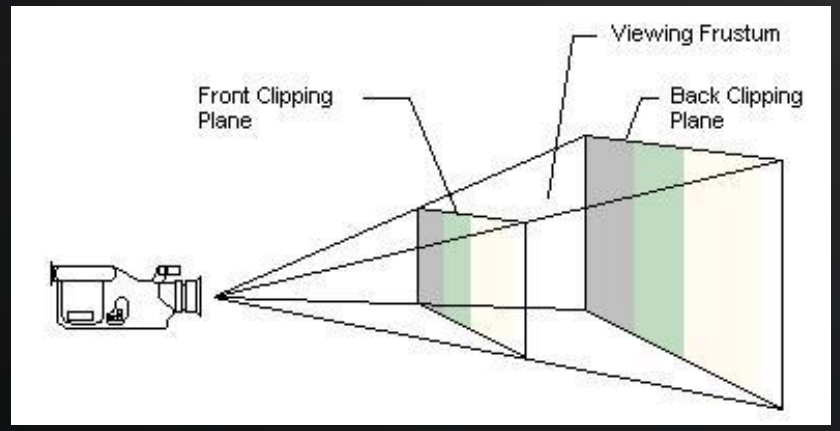
$$\begin{bmatrix} right_x & up_x & forward_x & position_x \\ right_y & up_y & forward_y & position_y \\ right_z & up_z & forward_z & position_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

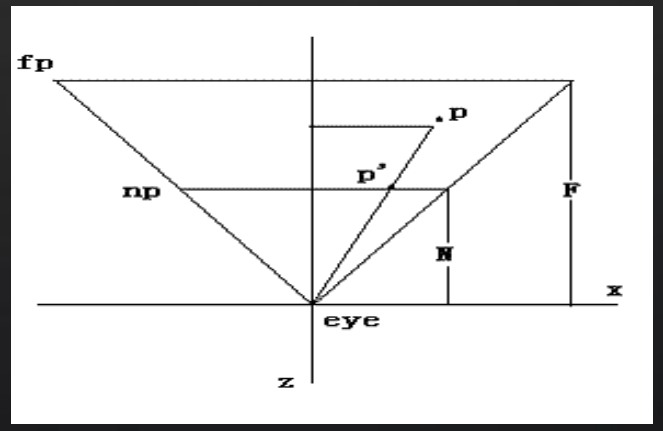
Lookat(Matrix4& m, const Vector3& eye, const Vector3& lookDir（target）, const Vector3& upDir)

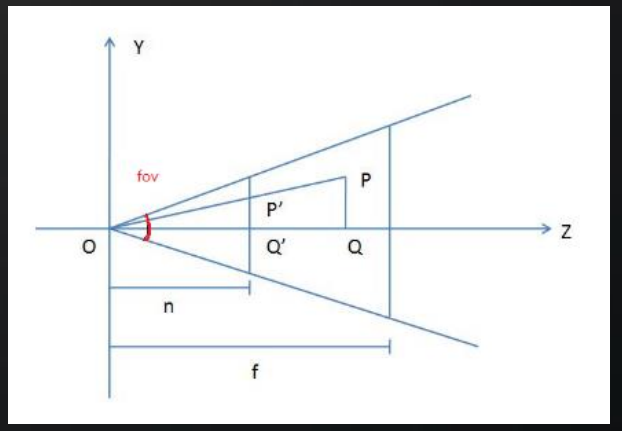
相机在世界中的model matrix的inverse即是view matrix, 或者说相机与物体之间是相对运动。
物体不动，移动相机 << == >> 移动物体，相机不动

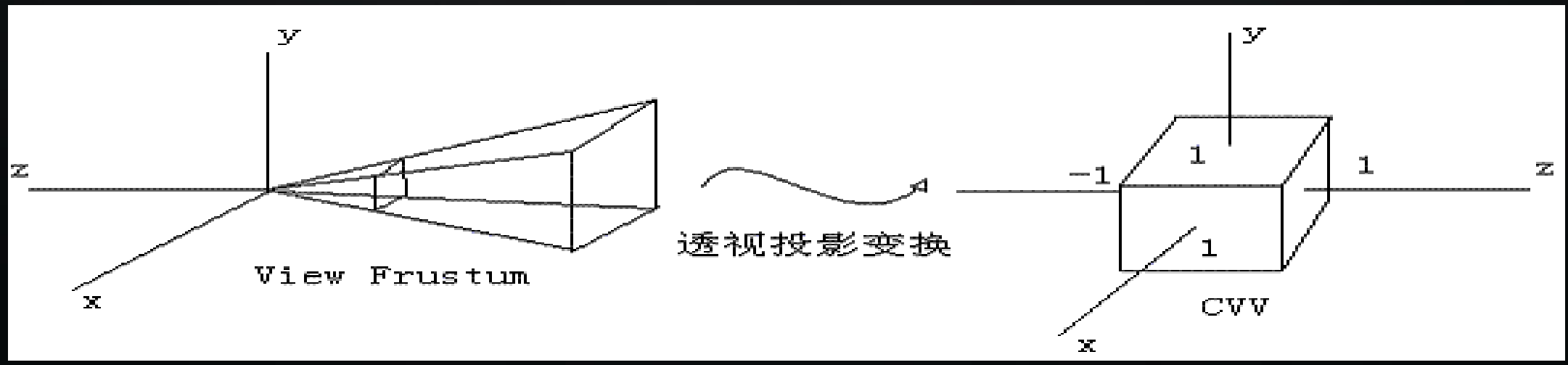
2. 视景体

在3D场景中，摄像机只能够在一定范围内观测到实体模型，这个可观测范围称之为视景体。



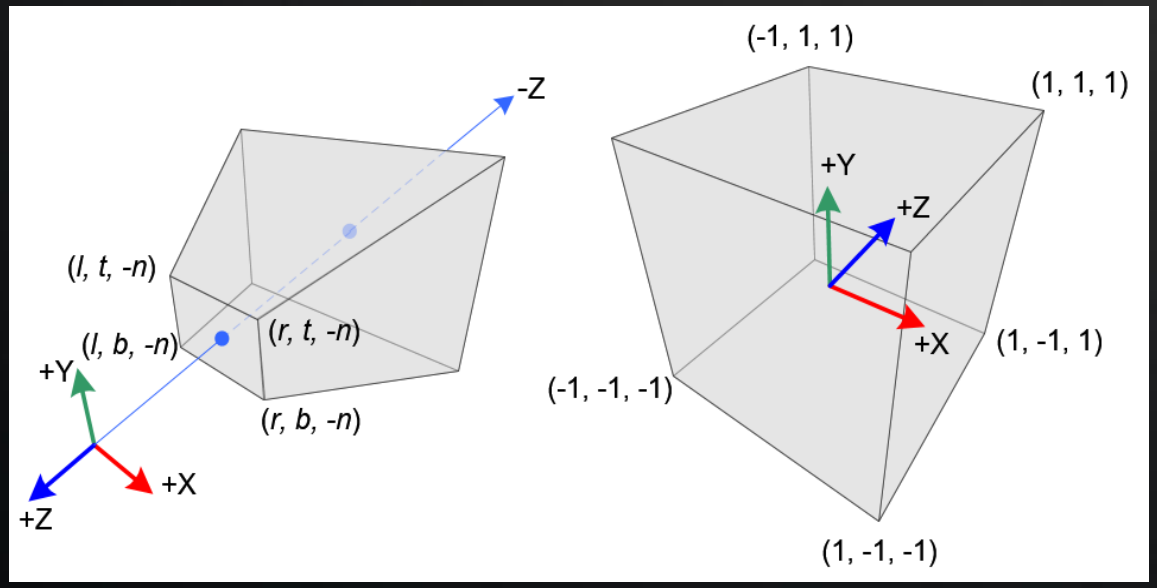






3. Projection-透视投影矩阵

透视投影时，呈现出近大远小的效果。

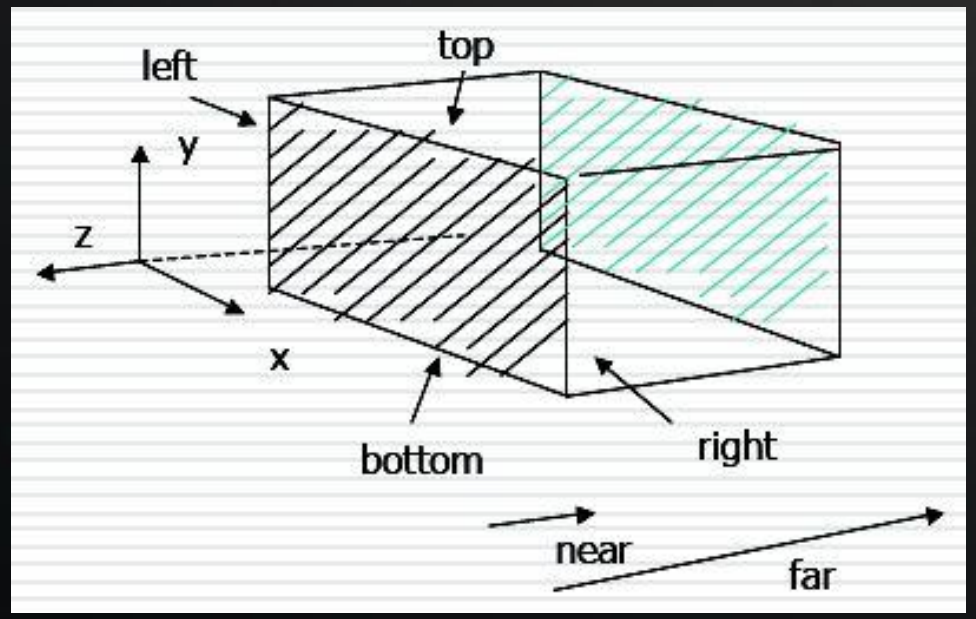


$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

void setPerspective(float fov, float aspect, float zNear, float zFar);

4. Projection-正交投影矩阵

正交投影时，实体模型无论远近，其大小不发生变化。



$$\mathbf{P}_o = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{1}{f-n} & -\frac{n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

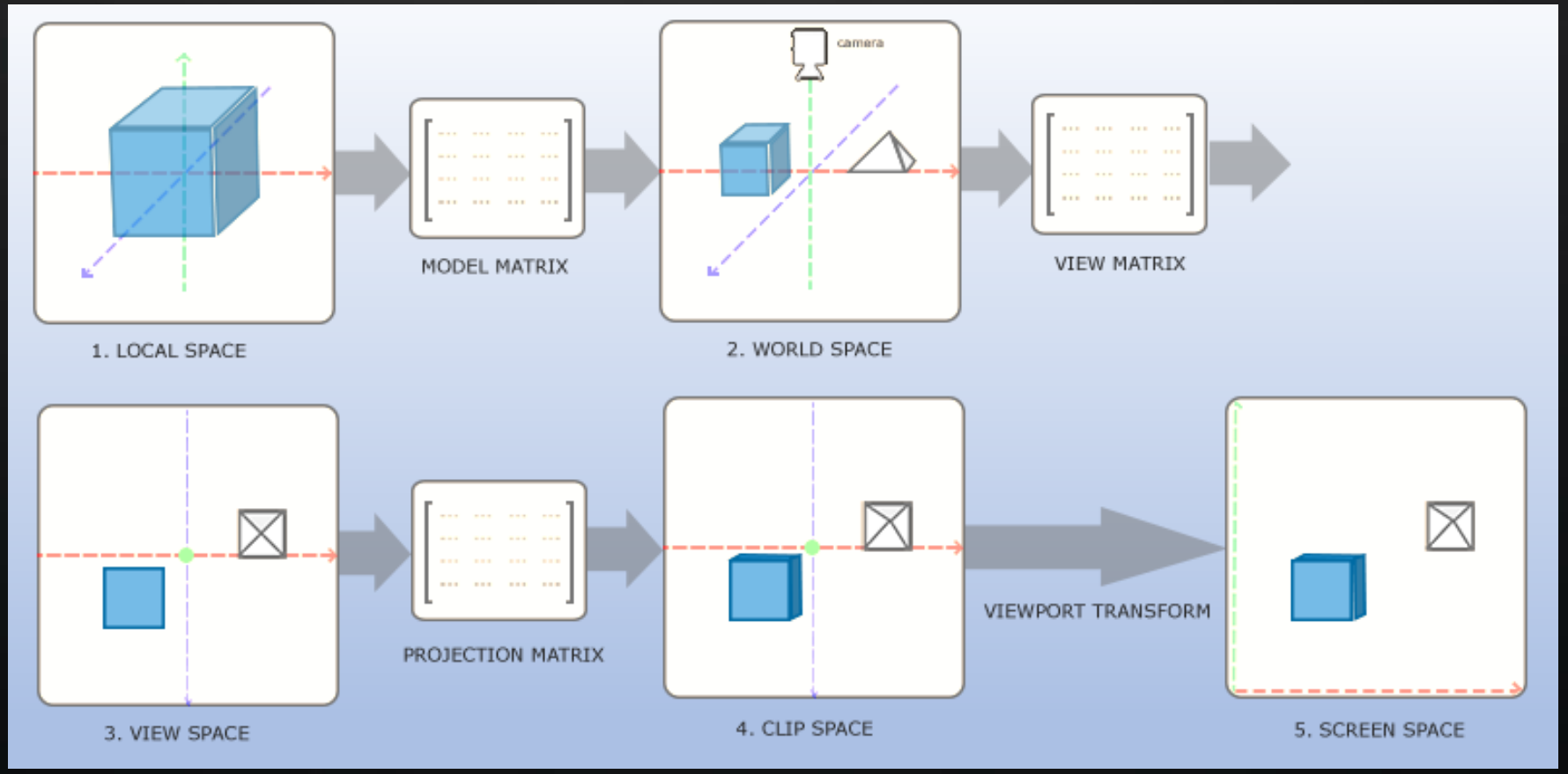
void setOrtho(float left, float top, float right, float bottom, float zNear, float zFar);

5. Viewport变换

ndc空间（笛卡尔坐标系）(-1，+1）转换到屏幕坐标系

```
screen_x = (nx * 0.5 + 0.5) * screen_width;
screen_y = (-ny * 0.5 + 0.5) * screen_height;
```

累积变换

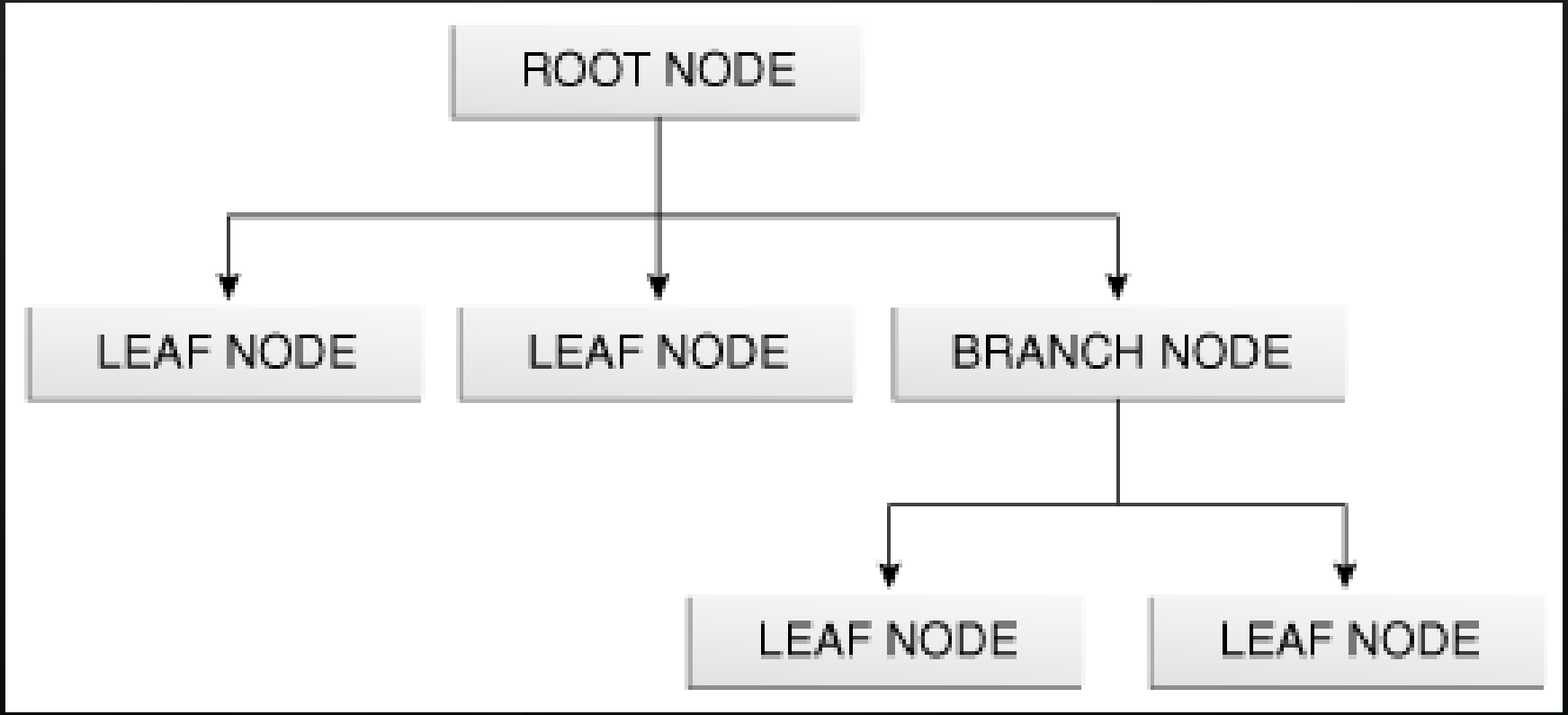


MVP = ModelMatrix * ViewMatrix * ProjectionMatrix（3D应用程序中，行矩阵），结果输入到Shader里

MVP = ProjectionMatrix * ViewMatrix * ModelMatrix (直接Shader中处理，列矩阵) gl_Position = MVP * pos

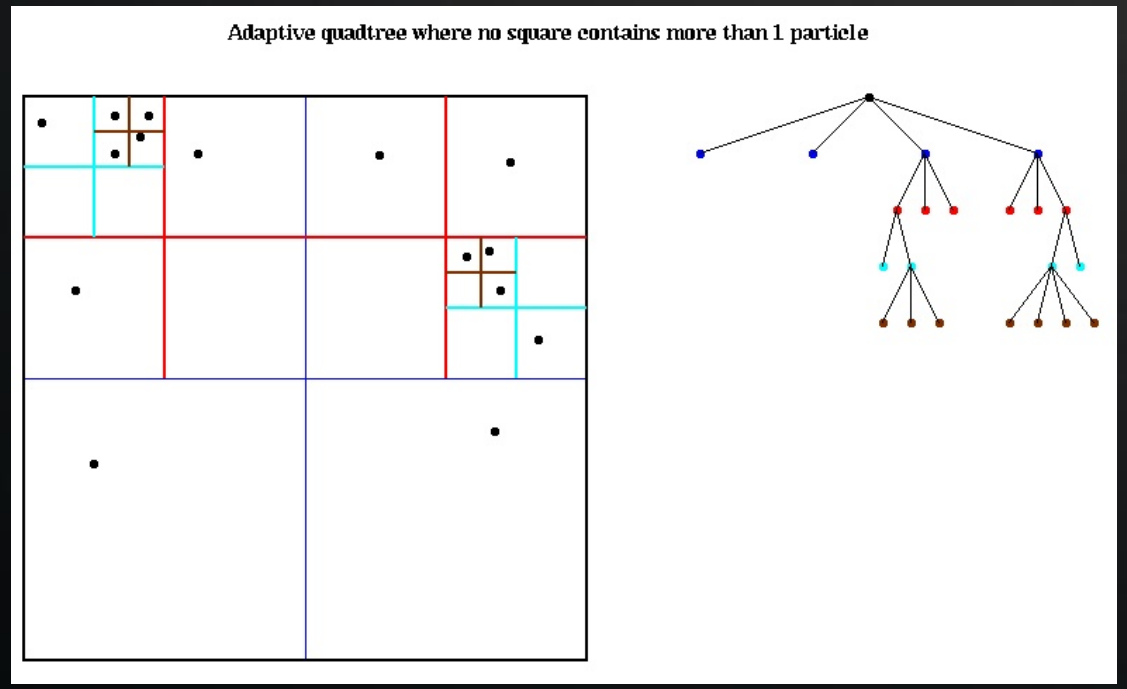
场景管理

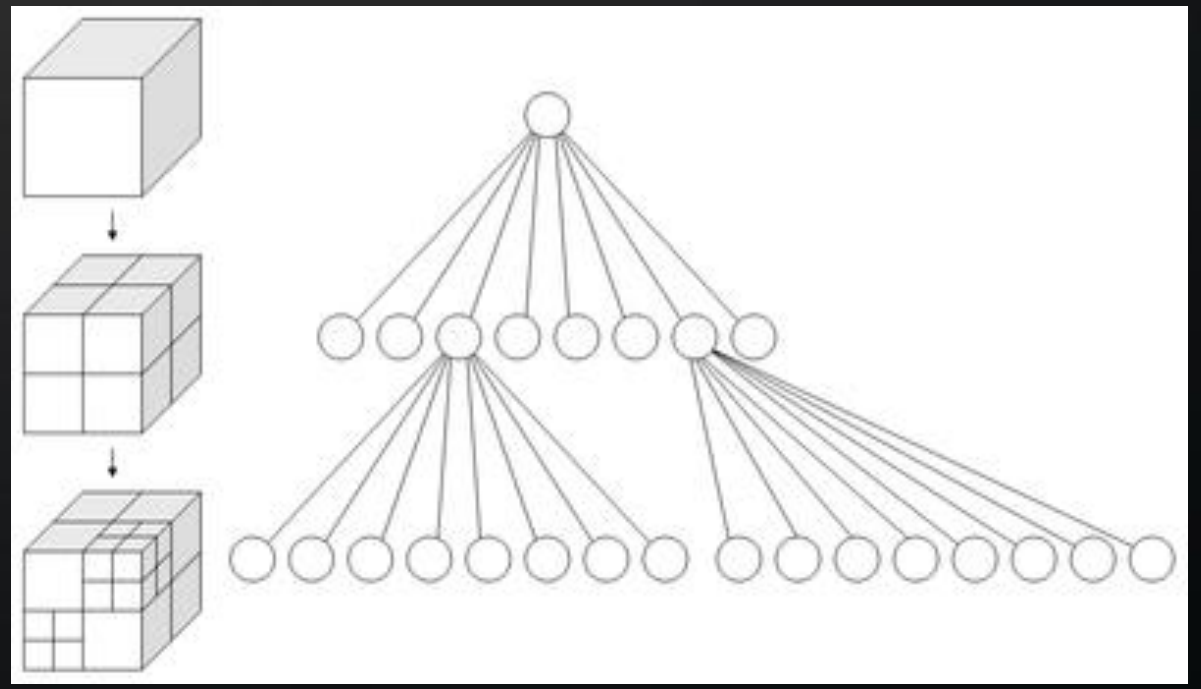
1. 场景树


ROOT NODE
LEAF NODE
LEAF NODE
BRANCH NODE
LEAF NODE
LEAF NODE

场景管理

2. 四叉树&八叉树





目的：快速剔除查找

THANK YOU