渲染管线中的坐标变换

这里整体梳理一下渲染管线中的坐标变换，总是和相机标定中的针孔模型的一些坐标变换步骤搞混，因此需要进行梳理。

模型坐标系

物体自身的局部坐标系，我理解比如什么CAD数模啊，他们往往存在一个自身的坐标系，是画图或者构建的时候产生的。

世界坐标系

这个就是通常理解的世界坐标系，比如房间里面建立的一个全局坐标系。

模型变换

通过模型变换矩阵将模型坐标系中的点转换到世界坐标系中

可以将模型变换矩阵叫做物体在世界坐标系中的位姿

$$p_{world} = M_{model} * p_{model}$$

视图坐标系

因为之间做相机标定，以为视图坐标系和相机坐标系是一个东西。但是视图坐标系和相机坐标系并不是一个东西。具体定义如下：

相机坐标系：光心为原点，采用右手定则中的右下前定义的坐标系。

视图坐标系：光心为原点，采用右手定则中的右上后定义的坐标系。

因此相机在相机坐标系中看向的是+Z方向。但是在视图坐标系中看向的是-Z方向。

视图变换

把场景中所有物体从**世界坐标系**转换到**视图坐标系**。中间使用到视图变换矩阵M_{view}

$$p_{view} = M_{view} * p_{world}$$

裁剪坐标系

我是第一次知道有这个统一的学术的名字，其实就是那个棱台经过“压扁”后的长方体（大模型说这种描述不标准，标准描述是“**齐次坐标下的标准裁剪体**”，但是我感觉不太好理解，所以延用这种错误的说法，毕竟确实是齐次坐标）。棱台空间的由来：通过近裁剪面和远裁剪面截取视椎体后形成的棱台空间。

是经过投影变换+坐标剪裁之后的点所在的坐标系。

$$p_{clip} = M_{per} * p_{view}$$
$$-w \leq x, y, z \leq w$$

投影变换

$$p_{clip} = M_{per} * p_{view}$$

坐标剪裁

$$-w \leq x, y, z \leq w$$

NDC坐标系

Normalized Device Coordinates，这个就是常说的那个立方体，$NDC \in [-1, 1]^3$

透视除法

$$x_{ndc} = x_{clip} / w_{clip}$$
$$y_{ndc} = y_{clip} / w_{clip}$$
$$z_{ndc} = z_{clip} / w_{clip}$$

屏幕坐标系

我经常和相机标定中的图像坐标系（像素坐标系）搞混，这里详细梳理一下

像素坐标系：原点（图像左上角）+ x轴（向右） + y轴 （向下）

屏幕坐标系：DirectX/Vulkan/现代 OpenGL : 原点（左上角）+ x轴（向右） + y轴 （向下）；传统的OpenGL : 原点（左下角）+ x轴（向右） + y轴 （向上）和API相关

视口变换

将NDC立方体拉伸后回到像素平面上的操作

$$\begin{aligned} p_{screenspace} &= M_{viewport} * p_{ndc} \\ &= \begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} \\ 0 & \frac{h}{2} & 0 & \frac{h}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * p_{ndc} \end{aligned}$$

✅ 总结：视口变换之后做了什么？

阶段	核心任务
图元装配	把顶点组成三角形等图元
光栅化	找出覆盖的像素，并做**透视校正插值**
片段着色	计算每个像素的颜色（光照、纹理等）
输出合并	深度测试、混合、写入最终图像

> 💡 视口变换是“几何阶段”的终点，光栅化是“像素阶段”的起点。

投影变换的公式推导

正交投影

games001

1. 采用右手系 $f < n < 0$
2. 映射立方体的关系 $(l, b, f) : (-1, -1, -1); (r, t, n) : (1, 1, 1)$

$$
\begin{aligned}
-1 &= scale_x * l + offset_x \\
1 &= scale_x * r + offset_x \\
scale_x &= 2/(r-l) \\
offset_x &= 1 - scale_x * l \\
&= 1 - 2l/(r-l) \\
&= (r - l - 2l)/(r-l) \\
&= -(r+l)/(r-l) \\
&= -\frac{r+l}{r-l}
\end{aligned}
$$

同理可以推导出

$$
\begin{aligned}
scale_y &= 2/(t-b) \\
offset_y &= -\frac{t+b}{t-b} \\
scale_z &= 2/(n-f) \\
offset_z &= -\frac{n+f}{n-f}
\end{aligned}
$$

最终可以得到正交投影矩阵

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

透视投影

games001 给出的证明思路是 将透视投影拆解成两步

1. 将视椎（棱台）压缩成长方体 P
2. 正交投影变换 M_{orth}

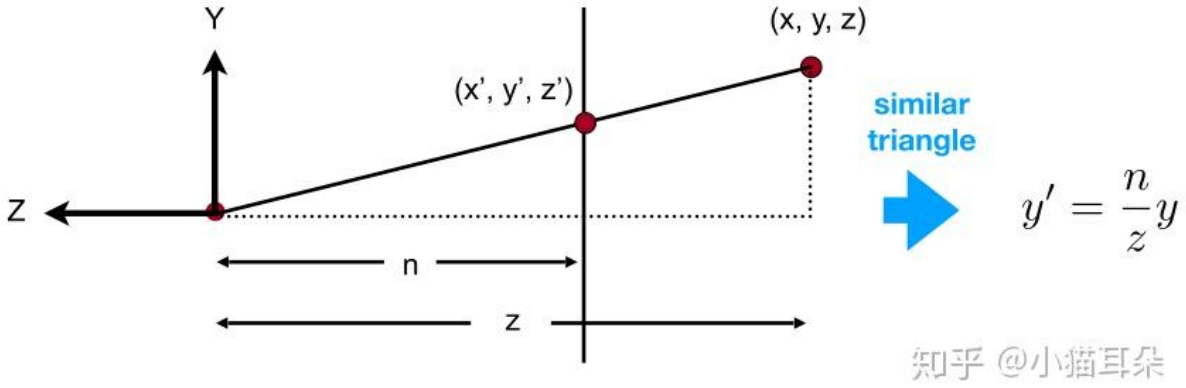
此时存在透视变换矩阵为$M_{per} = M_{orth} * P$

此时我们只需要推导出来P就可以得到 M_{per}了

这里详细拆解步骤一中的条件：

1. 近平面的所有点（例如(l, b, n)）的坐标保持不变
2. 其余点(x, y)的值根据z值进行线性缩放; 远平面上的z值仍然保证为 f

现在来推导一下P矩阵是怎么来的？可能不严谨，但是很好奇是怎么得出来的，大家都说是构造出来的，又很模糊。



结合这张图来建立数学问题如下：

将视椎（棱台）内一点(x, y, z) 通过 转换矩阵 P 转换为坐标 (x', y', z)

写成矩阵形式为

$$\begin{bmatrix} x' \\ y' \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

其中更具相似三角形可知

$$x' = \frac{nx}{z}$$
$$y' = \frac{ny}{z}$$

那么要求解的矩阵转化为

$$\begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

进一步两边同时乘以z转为如下

这里有个bug:为什么右边没变化呢，我是这么理解的，放进？里面去了，也不能不接受吧！！！

为什么能放进？里面去，因为右侧的这个向量我理解就是基分量，你有z的话，应该是线性的，不能出现z^2这种（这属于强行解释了）

1. 现在可以开始构造了，先构造P的前两行
2. 然后构造第四行
3. 根据near+far平面的深度不变，可以构造两个等式$n^2 = Ax + By + Cn + D$,
   $f^2 = Ax + By + Cf + D$。（这里有个问题，为什么选用这两个面的深度构建等式？ 我的理解是视椎（棱台）就是根据这两个超参数确定的，视椎虽然有其他深度的平面，但是没有确定的参数）

   这里确定$ABCD$,其中上述两个等式指的是对平面内的所有点都成立，因此可以知道与x, y线性无关，因此$A = 0, B = 0$;剩下的 CD可以联立方程求解

$$\begin{aligned} \begin{bmatrix} nx \\ ny \\ z^2 \\ z \end{bmatrix} &= \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ ? & ? & ? & ? \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{aligned}$$

P矩阵已知后，就知道了透视投影矩阵了，同时要注意的是，每次对某个点做完透视投影后，还要归一化才行，这一步也对上了之前求解P的时候莫名奇妙的左右两侧乘以了一个z

莫名奇妙的等式左右同时乘以z的终极解释

现在的理解是，z的乘除其实是一步坐标系转换的操作，是归一化坐标系和相机坐标系之间的转换，可以参考相机内外参标定来理解。

最终的透视投影矩阵如下

$$M_{per} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{l+r}{l-r} & 0 \\ 0 & \frac{2n}{t-b} & \frac{b+t}{b-t} & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2nf}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

参考资料

https://zhuanlan.zhihu.com/p/122411512

https://www.bilibili.com/video/BV1ZY41157TR/?spm_id_from=333.337.search-card.all.click&vd_source=bdda56b3baf0c6d4d1d6028820df7bcc

3d高斯坐标变换

这里的变换主要指的是高斯中的均值和协方差矩阵在变换过程中是如何计算的。

视图变换

假设世界坐标系存在如下高斯分布，其中高斯核中心为μ_{world}，协方差矩阵为Σ_{world}

同时设视图变换矩阵拆解成笛卡尔坐标系变换为$M_{view} = [A_{view}, b_{view}]$

上述高斯核经过视图变换后可以得到新的均值和方差如下：

$$\mu_{view} = A_{view} * \mu_{world} + b_{view}$$
$$\Sigma_{view} = A * \Sigma_{world} * A^T$$

投影变换

观察一下透视投影变换矩阵，很明显发现最后一行导致了整体的变换过程不是线性的，我理解是需要$1/z$的操作（透视除法），因此不是线性的。

那么均值的计算方法可以采用**齐次坐标矩阵变换+透视除法**的方式获取得到。

协方差矩阵由于非线性的原因，采用雅可比矩阵进行线性化近似后，才能求取

协方差矩阵计算公式如下：

$$\begin{aligned} \Sigma_{clip} &= J_{per} * \Sigma_{view} * J_{per}^T \\ &= J_{per} * A * \Sigma_{world} * A^T * J_{per}^T \\ &= A_{orth} * J_{press} * A_{view} * \Sigma_{world} * A_{view}^T * J_{press}^T * A_{orth}^T \end{aligned}$$

视口变换

均值直接按照视口变换矩阵计算即可。

协方差矩阵计算如下：

这里有个很重要的特点就是：$[ignoreZ]A_{viewport} = [ignoreZ]A_{orth}^{-1}$

为什么呢，因为正交投影是把长方体压缩成正方体;但是视口变换是重新拉伸成长方形。因此忽略掉z,可以认为是逆过程。

为什么忽略掉z呢？因为视口变换的最终结果是2D的图像，没有z的事了。

因此你会看到为什么很多的推导说协方差矩阵不经过视口变换啦，只到了压缩就位置了。是这么来的。

$$
\begin{aligned}
\Sigma_{viewport}^{ignore\_z} &= A_{viewport}^{ignore\_z} * \Sigma_{clip} * A_{viewport}^{ignore\_z^T} \\
&= J_{press}^{ignore\_z} * A_{view} * \Sigma_{world} * A_{view}^{T} * J_{press}^{ignore\_z^T}
\end{aligned}
$$

雅可比矩阵

下面推导一下J_{press}, 根据矩阵对向量求导得到雅可比矩阵

$$
\begin{aligned}
p_{press+1/z} &= \frac{1}{z}\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\
&= \frac{1}{z}\begin{bmatrix} nx \\ ny \\ nz+fz-nf \\ z \end{bmatrix} \\
&= \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ n+f-\frac{nf}{z} \\ 1 \end{bmatrix}
\end{aligned}
$$

其中可以知道

$$
\begin{aligned}
f_1(x) &= \frac{nx}{z} \\
f_2(y) &= \frac{ny}{z} \\
f_3(z) &= n+f-\frac{nf}{z}
\end{aligned}
$$

根据雅可比求解规则可知

$$
\begin{aligned}
J_{press+1/z} &= \begin{bmatrix} \frac{df_1}{dx} & \frac{df_1}{dy} & \frac{df1}{dz} \\ \frac{df_2}{dx} & \frac{df_2}{dy} & \frac{df2}{dz} \\ \frac{df_3}{dx} & \frac{df_3}{dy} & \frac{df3}{dz} \end{bmatrix} \\
&= \begin{bmatrix} \frac{n}{z} & 0 & -\frac{nx}{z^2} \\ 0 & \frac{n}{z} & -\frac{ny}{z^2} \\ 0 & 0 & -\frac{nf}{z^2} \end{bmatrix}
\end{aligned}
$$

球谐函数

$$f(t) \approx \sum_{l} \sum_{m=-l}^{l} c_l^m y_l^m(\theta, \phi)$$

$$y_l^m = \begin{cases} \sqrt{(2)} K_l^m cos(m\phi) P_l^m cos\theta, & \mathrm{m} > 0 \\ \sqrt{(2)} K_l^m sin(-m\phi) P_l^{-m} cos\theta, & \mathrm{m} < 0 \\ K_l^0 P_l^0 cos\theta, & \mathrm{m} = 0 \end{cases}$$

$$P_n(x) = \frac{1}{2^n . n!} \frac{d^n}{dx^n} [(x^2 - 1)^n]$$

$$P_l^m = (-1)^m (1 - x^2)^{\frac{m}{2}} \frac{d^m}{dx^m} P_l(x)$$

$$K_l^m = \sqrt{\frac{(2l + 1)(l - |m|)!}{4\pi(l + |m|)!}}$$

其中l表示阶数，通常取0,1,2,3; θ表示俯仰角；ϕ表示方位角；c_l^m是系数，同时是三维的向量，对应着RGB三个通道。y_l^m是基函数。

其中系数一般有16项，其中是1+3+5+7=16，使用矩阵表达的话，会是一个16*3维的矩阵。

Nerf 体渲染

累积不透明度

$$T(t) = exp(-\int_{t_n}^{t} \sigma(r(s)) ds)$$

上述公式推导如下：

定义变量：$T(t)$:光线传播到t点没有停止的概率。（累积透射率）

同理可知，$T(t + dt)$:又向前传播了dt距离仍没有停止的概率。

$\sigma(t)$：表示体素密度/不透明度。$\sigma(t) \in [0, 1]$

$$T(t+dt) = T(t).(1-dt.\sigma(t))$$
$$\frac{T(t+dt)-T(t)}{dt} = -T(t).\sigma(t)$$
$$\frac{dT(t)}{dt} = -T(t).\sigma(t)$$
$$\dot{T}(t) = -T(t).\sigma(t)$$
$$\frac{\dot{T}(t)}{T(t)} = -\sigma(t)$$
$$\int_{t_n}^{t} \frac{\dot{T}(t)}{T(t)}dt = \int_{t_n}^{t} -\sigma(t)dt$$
$$ln(T(t))|_{t_n}^{t} = \int_{t_n}^{t} -\sigma(t)dt$$
$$ln(T(t))|_{t_n=0}^{t} = \int_{t_n}^{t} -\sigma(t)dt$$
$$ln(T(t)) - ln(T(0)) = \int_{0}^{t} -\sigma(t)dt$$
$$ln(T(t)) - ln(1) = \int_{0}^{t} -\sigma(t)dt$$
$$ln(T(t)) = \int_{0}^{t} -\sigma(t)dt$$
$$T(t) = exp(\int_{0}^{t} -\sigma(t)dt)$$

其中：

$dt.\sigma(t)$：可以理解为遮挡率

$1-dt.\sigma(t)$：可以理解为透射率

累积透射率：当前累积透射率 × 新增累积透射率

渲染颜色

按照累积不透明度对射线上所有采样点的颜色进行加权求和，得到最终的颜色。

$$C(\vec{r}) = \int_{0}^{t} T(t)\delta(\vec{r}(t))\vec{c}(\vec{r}(t),\vec{d})dt$$
$$\vec{r} = \vec{o} + t.\vec{d}$$

渲染离散化

- 将光线$[0,s]$划分为N个等距区间$[T_n \to T_{n+1}]$，其中$n = 0,1,2,\ldots,N-1$
- 间隔的长度记为δ_n
- 假定区间内的体素密度σ_n和颜色C_n是固定的

  这里认为最终的颜色是每段区间贡献的光强叠加之和：$C = \sum_{n=0}^{N-1} I(T_n \to T_{n+1})$

  因此推导单个区间内的离散化：

$$
\begin{aligned}
I(T_n \to T_{n+1}) &= \int_{t_n}^{t_{n+1}} T(t)\sigma_n C_n dt \\
&= \sigma_n C_n \int_{t_n}^{t_{n+1}} exp(-\int_0^t \sigma(s)ds)dt \\
&= \sigma_n C_n \int_{t_n}^{t_{n+1}} exp(-(\int_0^{t_n} \sigma(s)ds + \int_{t_n}^t \sigma(s)ds))dt \\
&= \sigma_n C_n \int_{t_n}^{t_{n+1}} exp(-\int_0^{t_n} \sigma(s)ds) exp(-\int_{t_n}^t \sigma(s)ds)dt \\
&= \sigma_n C_n exp(-\int_0^{t_n} \sigma(s)ds) \int_{t_n}^{t_{n+1}} exp(-\int_{t_n}^t \sigma(s)ds)dt \\
&= \sigma_n C_n exp(-\sum_{i=0}^{n-1} \sigma_i \delta_i) \int_{t_n}^{t_{n+1}} exp(-\int_{t_n}^t \sigma(s)ds)dt \\
&= \sigma_n C_n T(0 \to t_n) \int_{t_n}^{t_{n+1}} exp(-\int_{t_n}^t \sigma(s)ds)dt \\
&= \sigma_n C_n T(0 \to t_n) \int_{t_n}^{t_{n+1}} exp(-\int_{t_n}^t \sigma_n ds)dt \\
&= \sigma_n C_n T(0 \to t_n) \int_{t_n}^{t_{n+1}} exp(-\sigma_n (t - t_n))dt \\
&= \sigma_n C_n T(0 \to t_n) [-\frac{1}{\sigma_n} exp(-\sigma_n (t - t_n))|_{t_n}^{t_{n+1}}] \\
&= C_n T(0 \to t_n)[1 - exp(-\sigma_n \delta_n)] \\
&= C_n exp(-\sum_{i=0}^{n-1} \sigma_i \delta_i)[1 - exp(-\sigma_n \delta_n)]
\end{aligned}
$$

因此最终可以归纳出来

$$
C = \sum_{i=1}^{N} T_i (1 - exp(-\sigma_i \delta_i)) C_i
$$

$$
T_i = exp(-\sum_{j=0}^{i-1} \sigma_j \delta_j)
$$