非齐次坐标矩阵二维变换

复习矩阵相乘

- 对于 A_{ij}，实际上是把第一个矩阵的第i行向量和第二个矩阵的第j列向量**做点乘**。
- 如果某一个矩阵不存在所需的行向量或列向量（即没有那一行或那一列），那么结果矩阵中该元素不存在。也就是结果矩阵**不包含**该元素**所在行、列**。

线性变换

所谓线性变换，就是可以使用矩阵**乘法**来表示的变换。

缩放矩阵

- 对x、y分别乘以其缩放因子 s_x, s_y.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

切变矩阵

- y值越大，其x值的偏移越大。他是随着y进行线性变化的。也就是 $x' = x + ay$.
- y值是恒定的。x也是在恒定的基础上加了个随y线性变化的偏移量。于是：

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

关于原点旋转矩阵

- 考虑把一个边长为1，左下角顶点是原点的正方形进行逆时针旋转 θ 弧度。因为是线性变换，同样考虑写成以下形式：

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- 现在观察右下角的顶点，其x坐标从1变成了 $cos\theta$，其y坐标从0变成了 $sin\theta$.
    - 因为x的变化是 $x' = Ax + By = Ax$，所以 $cos\theta = A$.（注意，x，y是变化前的，x'是变化后的。）同理可以解出B、C、D（还需要第二个点，因为是正方形，所以都只有一个坐标分量不为0.）。
- 于是有关于原点旋转矩阵。

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

非线性变换

非线性变换无法使用矩阵乘法进行表示，而只能使用矩阵**加减法**。

- 对x、y进行加减法。于是容易得到按向量（t_x, t_y）进行平移的结果：

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

纺射变换

- 也就是把**线性变换和非线性变换结合**起来一起做。
- 而且，因为先乘除，后加减，所以显然是**先线性，后平移**。

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

齐次坐标矩阵二维变换

纺射变换

如果是对一个点 $(x, y, 1)$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} A & C & t_x \\ B & D & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} Ax + Cy + t_x \\ Bx + Dy + t_y \\ 1 \end{bmatrix}$$

如果是对一个向量 $(x, y, 0)$

$$\begin{bmatrix} x' \\ y' \\ 0 \end{bmatrix} = \begin{bmatrix} A & C & t_x \\ B & D & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = \begin{bmatrix} Ax + Cy \\ Bx + Dy \\ 0 \end{bmatrix}$$

- 可见，引入齐次坐标后
  - 能用矩阵乘法表示非线性变换，从而统一。于是纺射变换也写成了线性形式。
  - 能够保证向量在平移前后，长度不变（也就是自动屏蔽其平移操作），而保留其他线性变换操作。
- 为什么点是1，向量是0
  - 因为点和点的差得到向量。
  - 因为能实现上述的平移保护。
- 优化：节约存储空间

    - 因为纺射变换矩阵的最后一行都是001，所以可以不存储。于是这种方式就完美了：在没有增加空间复杂度的情况下把线性变换和非线性变换的表达统一化了。
- 纺射变换表达的非齐次转齐次
    - 线性变换部分：直接拿着原来的矩阵填充左上角的ABCD元素。
    - 非线性变换部分：拿着原来做加法的哪个矩阵填充右边多出来的一列的 t_x, t_y.

复合变换

对一个对象进行多个操作的顺序叠加，称为复合变换。

什么也不做

- 如果一个纺射变换，其线性部分是单位阵，非线性部分是全0，那就是什么也不做。

逆变换

- 如果乘一个变换矩阵的**逆矩阵**，因为它能和原来的某个变换矩阵抵消掉，变成单位阵，也就是相当于上面的“什么也不做”。
    - 特别地，考虑旋转矩阵：

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

    - 如果把每个角度都换成其负值，会变成：
    - $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
    - 于是，有 $R_{-\theta} = R_{\theta}^{T} = R_{\theta}^{-1}$ 成立。这是一个重要性质。也就是非齐次化的旋转矩阵的逆=非齐次化的旋转矩阵转置。
    - 或者说，非齐次的旋转矩阵是一个**正交矩阵**。

复合变换的顺序性

- 因为矩阵乘法不满足交换律，所以变换矩阵乘在左边的顺序决定了变换结果。
    - **注意，一定要是左乘，否则什么也不是。**
    - 如何正确地确定顺序？整体思想。例如A乘B乘被变换向量，那么先化简一步，B和待变换向量乘起来了以后，变成了新向量，这个时候再和A乘即可。
    - **因此，是从右往左依次施加变换结果。**

复合变换矩阵

- 既然已经推倒出来了上面的顺序复合变换的结论，那么如果把被变换向量拿走，只剩下左边按从右到左顺序相乘的一系列矩阵，取该相乘结果作为一个单独的变换矩阵，那么这也是一个齐次坐标纺射变换矩阵，只不过它不属于任何一种标准类型，但是它可以表示一种复杂的变换。
- 好处：例如当前我们只有绕原点旋转的变换矩阵，那么如何得到绕任一点旋转的这样的矩阵呢？**可以利用先平移到原点，再旋转，再平移回去的复合构成的矩阵。**

齐次坐标矩阵三维变换

三维纺射变换

绕x，y或z轴旋转

$$R_x(\alpha)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & cos\alpha & -sin\alpha & 0\\ 0 & sin\alpha & cos\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$R_y(\alpha)=\begin{bmatrix}cos\alpha & 0 & sin\alpha & 0\\ 0 & 1 & 0 & 0\\ -sin\alpha & 0 & cos\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$R_z(\alpha)=\begin{bmatrix}cos\alpha & -sin\alpha & 0 & 0\\ sin\alpha & cos\alpha & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

- 绕哪个轴转，哪个就变成单位阵的那一列、行。
- x、z都是照搬了二维下旋转α角的情况，y是照搬了二维下旋转 $-\alpha$的情况。然后按照原来的**相对位置**填入。

绕任意轴旋转

- 利用变换的复合。

引入：MVP变换

- 首先进行模型（Model）变换
  - 用于把**场景搭建**完成。
- 然后进行视图（View）变换
  - 用于**移动相机，找个好角度**。
- 最后是投影（Projection）变换
  - 用于**按下快门**。

ModelView变换（模型视图变换）

相机相关参数

- 位置（e向量）
- lookat（g向量，gaze）
- up（t向量，top）
- 描述了相机**放在哪里，看向哪里，是正着看还是歪着看（直观理解：竖屏拍照和横屏拍照）**。

相机标准位置

- 位置在原点
- lookat向-z
- up向+y

为什么Model和View都来了

- 假如相机在世界坐标中的**任意一个位置漫游**，然后我们需要把他变换到相机**标准位置**，但是又要保证它看到的东西不会发生变化。
- 于是，就可以在变换相机的同时**对所有的模型施加完全相同的变换**。
- 也就是，Model和View进行了相同的变换。

如何进行ModelView变换

- 首先进行平移，即按照e的负向量平移。
  - $$T_{view} = \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- 然后进行旋转。这里需要把相机的up、lookat分别旋转到+y、-z分解一下任务，就是：
  - 世界坐标x轴要对齐lookat和up的叉乘
  - 世界坐标y轴要对齐up
  - 世界坐标z轴要对齐lookat的负值
- 直接旋转不容易，但是可以考虑把一个标准的相机旋转回当前相机的位置，于是可以让这个相机施加下列变换：
  - $$R_{view}^{-1} = \begin{bmatrix} x_{y\times t} & x_t & x_{-g} & 0 \\ y_{y\times t} & y_t & y_{-g} & 0 \\ z_{y\times t} & z_t & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
  - 假设要使用这个变换，对齐次向量(1,0,0,0)进行变换，其结果是 $(x_{y\times t}, y_{y\times t}, z_{y\times t}, 0)$，也就是能够成功把X轴方向旋转到相机的lookat和up的叉乘所在方向上。对标准位置Y轴(0,1,0,0)、Z轴(0,0,1,0)的变换同理。可见，该矩阵可以实现这样的变换。
- 然后，对上述矩阵求逆，对当前的相机施加，就可以把他转到标准位置去。因为把标准位置转到当前相机位置的逆操作就是把当前相机转到标准位置去。
  - 注意，因为这是一个旋转矩阵，而刚才证明了旋转矩阵是一个正交矩阵，所以可以直接求转置，得到其逆矩阵：
    -

$$R_{view} = \begin{bmatrix} x_{y\times t} & y_{y\times t} & y_{y\times t} & 0 \\ x_t & y_t & z_t & 0 \\ x_{-g} & y_{-g} & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

    - 于是，我们就得到了ModelView的旋转矩阵。
- **对相机和各模型分别先施加ModelView平移矩阵，再施加ModelView的旋转矩阵，就完成了ModelView变换。**

Projection变换

和ModelView变换的衔接

- ModelView把相机标准化了，并且把场景里面的东西跟着标准化了。

Projection正交投影（Orthographic Projection）和透视投影（Perspective Projection）的直观区别

- 主要在于原本**平行**的线是否还保持**平行**。
- 后者能实现近大远小，前者不能。

正交投影

> 任务：想要在空间的任**一块**划定一个正交投影体（长方体），**求它的投影结果。**
>
> 于是再让空间中的**每个物体都再施加正交投影矩阵变换**，让空间中的物体和正交投影体一起变换到归一化的情况。即可在中心（0，0，0），边长2的标准正方体内直接输出结果。

- **把空间中所有物体的z都设为0**，就得到了正交投影，非常简单。（需要消隐）
- 标准正交投影体是一个立方体，边长为2，中心在原点。
- 现有一个非标准正交投影体，是一个范围为 `x:(left,right),y:(bottom,top),z:(far,near)` 构成的长方体，且其中心在世界坐标任意位置。现在要做的事情就是把他转换成标准正交投影体。**把非标准正交投影体转换为正交投影体的过程称为正交投影，同时施加到空间任何物体上。**
- 则先平移，后缩放即可简单地完成要求：

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

- 注意
    - 因为是往标准位置变换，平移因子都取了**负数**。
    - 缩放因子的含义是：先除以该边的长度，化为1，再乘2，于是可以得到边长为的目标了。
    - 因为是先平移，后缩放，所以不能用单个纺射变换表示。

- 平移因子是取了中点，所以是“+”；缩放因子是取了长度，所以是“-”。平移因子的“2”是为了取中点，缩放因子的“2”是为了缩放到边长为2.

透视投影

> 任务：和正交投影是一样的。只不过这里**使用了视锥，而非长方体。**
>
> 最终在近平面取景。把近平面和远平面**之间**（包括远平面上）的所有物体都投影到近平面上。

- 可以拆分为两个步骤：先把视锥体**挤压**成长方体，再执行**正交投影**。也就是先挤压，再调用正交投影。但是要做出如下规定，保证挤压的结果唯一：
  - 近平面上的任何一个点在挤压前后都不能变化位置（即近平面没有任何变化）
  - 远平面上的任何一个点的z值在挤压前后不能发生变化，而x、y可以（即远平面没有脱离其原来所在平面）
- 第一步骤的矩阵我们称为 $M_{prosp\to ortho}$.也就是专心研究如何按照上述的约束对视锥体进行挤压，从而能把透视投影转换为正交投影。
  - 现在考虑相似三角形：设近平面的z值是n，远平面的是f。设视锥体内任一点的z值为z。则对每个点的坐标 `(x,y,z)` 有下列关系成立（**z是未知的，因为我们不清楚在挤压后，z将会发生什么变化**）：
  - $$\begin{aligned} x' &= \frac{n}{z}x \\ y' &= \frac{n}{z}y \\ z' &= ? \end{aligned}$$
  - 现在想要得到下面的变换（其中，(x,y,z)在近平面和远平面之间，视锥体内的任意位置；(x',y',z')**在挤压后的近平面和远平面之间**，且z可能发生了变化）：
  - $$M_{prosp\to ortho}\begin{bmatrix} x \\ y \\ ? \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ ? \\ 1 \end{bmatrix}$$
  - 其中，根据上述相似三角形变换，然后根据齐次坐标的性质（即各分量同乘一个数字，仍能表示同一个点或向量），有
  - $$\begin{bmatrix} x' \\ y' \\ ? \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n}{z}x \\ \frac{n}{z}y \\ ? \\ 1 \end{bmatrix} = \begin{bmatrix} nx \\ ny \\ ?\times z \\ z \end{bmatrix}$$
  - 即现在是
  - $$M_{prosp\to ortho}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nx \\ ny \\ ? \\ z \end{bmatrix}$$
  - 下面开始构造 $M_{prosp\to ortho}$.因为在施加该矩阵后，x、y根据上面的规律，变换成了nx、ny，所以可以填出下面的矩阵：

- $$M_{prosp\to ortho}=\begin{bmatrix}n&0&0&0\\0&n&0&0\\?&?&?&?\\0&0&1&0\end{bmatrix}$$
- 可以验证，现在可以得到
- $$M_{prosp\to ortho}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}=\begin{bmatrix}nx\\ny\\?\\z\end{bmatrix}$$
- 但是现在还需要待定 $M_{prosp\to ortho}$ 中的“? ”。也就是继续填充矩阵的第三行，最终得出我们的从透视投影到正交投影转化的变换矩阵来。
- 根据完全近平面不变，可以得出矩阵第三行和待变换点相乘和其结果的关系：(前面两个0是因为这里的结果没有出现x、y的值，所以肯定是0)
- $$[0\ 0\ A\ B]\begin{bmatrix}x\\y\\n\\1\end{bmatrix}=n^2$$
- 也就是，原来在近平面上的点的z是n，变换后得到 n^2.这个平方是因为我们之前利用齐次坐标的性质，同时乘以n所得。
- 同理，根据远平面不会脱离其所在平面（即z值恒定），可以得出矩阵第三行和待变换点相乘和其结果的关系：
- $$[0\ 0\ A\ B]\begin{bmatrix}x\\y\\f\\1\end{bmatrix}=f^2$$
- 然后综合上述两个关系，得到：
- $$\begin{cases}An+B=n^2\\Af+B=f^2\end{cases}$$
- 解得
- $$\begin{cases}A=n+f\\B=-nf\end{cases}$$
- 于是，得到最终的从透视投影向正交投影转换的矩阵：
- $$M_{prosp\to ortho}=\begin{bmatrix}n&0&0&0\\0&n&0&0\\0&0&n+f&-nf\\0&0&1&0\end{bmatrix}$$

- 最终，继续进行正交投影，即可得到透视投影矩阵，于是就把空间中任意位置的任意大小的视锥体正则化为了中心在原点，边长为2的正交投影体了：

$$M_{persp} = M_{ortho} M_{prosp \to ortho}$$

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

  - ~~(好耶。完结撒花)~~
- 此外，如何定义一个视锥体?
  - 如果把视锥体看做一个摄像机，那么实际上是在近平面上成像，因为最后一步转换成了正交投影。
  - 给出FOVy（垂直可视角度），宽高比和n（near）即可。
  - 因为根据FOVy和宽高比可以算出FOVx；根据FOVy和n可以算出高度，然后根据宽高比又能算出宽度。

Viewport变换

- 视口变换完成最后一步工作：把正则化了的正交投影体的投影结果变换为屏幕空间。
- 具体操作是：把变长为2的正方形缩放成屏幕的宽高对应的矩形，然后平移，使之以左下角为(0,0).
- 矩阵就是一个纺射变换。因为是先缩放，后平移，所以可用单个纺射变换表示。

$$M_{viewport} = \begin{bmatrix} \frac{width}{2} & 0 & 0 & \frac{width}{2} \\ 0 & \frac{height}{2} & 0 & \frac{height}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$