

Vector Transformations

Matrices for Vector Transformations

- maths notes Chapter 5 . . .

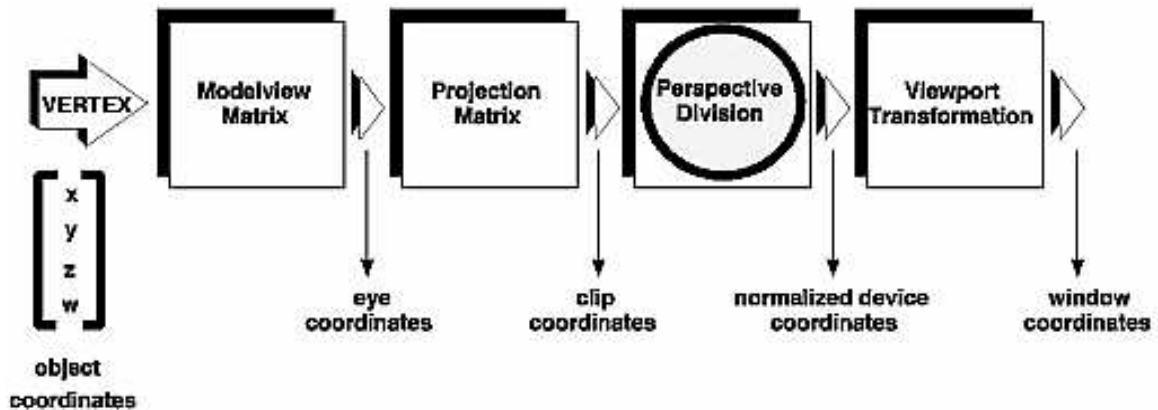


Figure 1: Vertex transformation pipeline.

- In Figure 1, vertex coordinates are operated on as vectors. Depending of the depth of nesting of *objects* — think world, robot, robot-arm, arm-hand, hand-finger, finger-joint, . . . and large number of transformations may be contained in (via composition) the *modelview matrix*
- . . . vectors as components with respect to a basis . . . can use matrix arithmetic to implement linear transformations
- express a vector \mathbf{u} in terms of a weighted sum of basis vectors,

$$\mathbf{u} = u_1 \mathbf{e}_1 + u_2 \mathbf{e}_2, \quad (1)$$

and then apply *linearity* rules,

$$\begin{aligned} \mathbf{v} = T(\mathbf{u}) &= T(u_1 \mathbf{e}_1 + u_2 \mathbf{e}_2), \\ &= u_1 T(\mathbf{e}_1) + u_2 T(\mathbf{e}_2). \end{aligned} \quad (2)$$

- *any* vector can be expressed as a linear combination of the basis vectors

- so we can write

$$T(\mathbf{e}_1) = a_1\mathbf{e}_1 + b_1\mathbf{e}_2, \quad (3)$$

and

$$T(\mathbf{e}_2) = a_2\mathbf{e}_1 + b_2\mathbf{e}_2. \quad (4)$$

- substitute these last two equations into eqn. 3 to get

$$\begin{aligned} T(\mathbf{u}) &= u_1a_1\mathbf{e}_1 + u_1b_1\mathbf{e}_2 + u_2a_2\mathbf{e}_1 + u_2b_2\mathbf{e}_2, \quad (5) \\ &= (u_1a_1 + u_2a_2)\mathbf{e}_1 + (b_1u_1 + b_2u_2)\mathbf{e}_2. \end{aligned}$$

- $\mathbf{v} = v_1\mathbf{e}_1 + v_2\mathbf{e}_2$, then collecting the \mathbf{e}_1 and \mathbf{e}_2 , coefficients, we have,

$$v_1 = a_1u_1 + b_1u_2, \quad (6)$$

$$v_2 = b_1u_1 + b_2u_2. \quad (7)$$

- a matrix equation ...

$$\mathbf{v} = \mathbf{A}\mathbf{u}. \quad (8)$$

\mathbf{A} is a 2 row \times 2 column *matrix*, $\mathbf{A} = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}$, \mathbf{v} is a one column two row matrix, containing a tuple of vector components, i.e. \mathbf{v} is a two-dimensional vector,

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \text{ and } \mathbf{u} \text{ is another a two-dimensional vector,}$$

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}.$$

Points and Vectors

- points, e.g. $P = (P_x, P_y)$
- vectors, e.g. $\mathbf{v} = (v_1 v_2)^t$
- You could be excused for confusing points and vectors. The difference is rather subtle. When we say that P above can be represented by a vector $\mathbf{p} = (p_1, p_2)$, what we mean is that P can be represented by the directed line (vector) $OP = \mathbf{p}$, where O is the *origin* — a *point*, $P = O + \mathbf{p}$.
- **You can add a vector to a point to get a point.**

Some Transformations

- the first column of \mathbf{A} is the vector that results in transforming $\mathbf{e}_1 = (1, 0)^t$. Second column of \mathbf{A} the vector transforms $\mathbf{e}_2 = (0, 1)^t$.

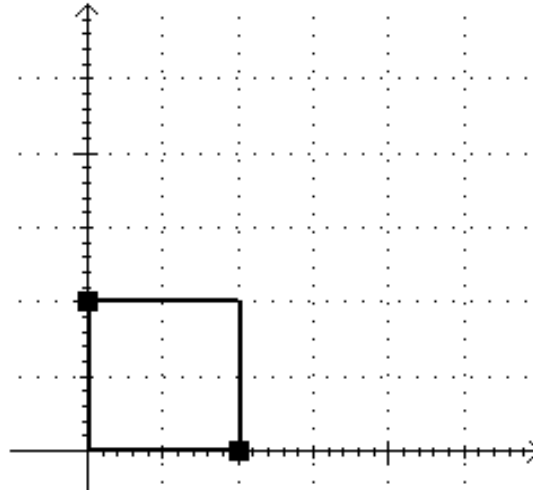


Figure 2: 2D Rectangle. $(1, 0)$ and $(0, 1)$ are marked as solid points.

- **Scaling**

We transform using $\begin{bmatrix} 2 & 0 \\ 0 & 1.5 \end{bmatrix}$. Sure enough, we get $(1, 0) \rightarrow (2, 0)$ and $(0, 1) \rightarrow (0, 1.5)$, see Figure 3(a), just as we expected. Next transform using $\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$. We get $(1, 0) \rightarrow (1, 0)$ and $(0, 1) \rightarrow (0, 2)$, see Figure 3(b), again as expected.

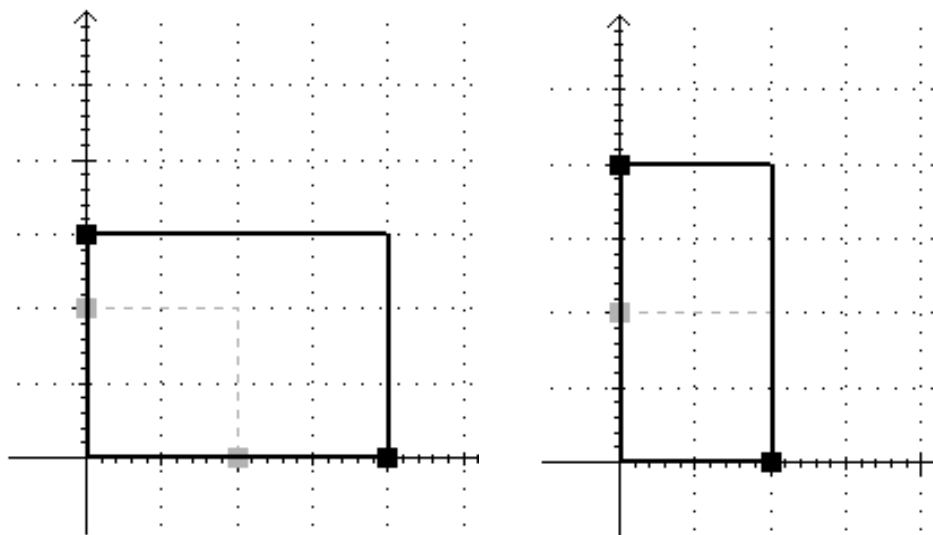


Figure 3: Scaling. (a) scale x by 2, y by 1.5; (b) scale x by 1, y by 2.

• Rotation

We transform using $\begin{bmatrix} \cos 30^\circ & -\sin 30^\circ \\ \sin 30^\circ & \cos 30^\circ \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 \\ 0.5 & 0.866 \end{bmatrix}$.

Sure enough, we get $(1, 0) \rightarrow (0.866, 0.5)$ and $(0, 1) \rightarrow (-0.5, 0.866)$, see Figure 4(a), just as we expected.

Next transform using $\begin{bmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{bmatrix} = \begin{bmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{bmatrix}$.

We get $(1, 0) \rightarrow (0.707, 0.707)$ and $(0, 1) \rightarrow (-0.707, 0.707)$ see Figure 4(b), just as expected.

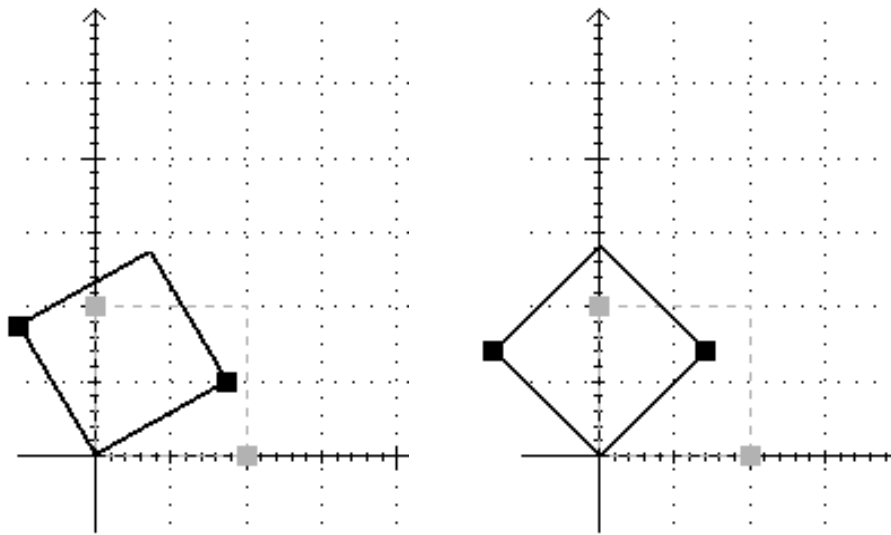


Figure 4: Rotation. (a) 30 degrees; (b) 45 degrees.

- But we have a big problem, we cannot do *translation* using matrices ...

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \begin{bmatrix} u_x \\ u_y \end{bmatrix}. \quad (9)$$

- the *homogeneous coordinate* representation fixes this.