Computer Graphics

CGT 520

Computer Graphics Technology Dept.
Purdue University

February 3, 2015

Outline

Point and vector representation

Specifying **vector** components $\mathbf{a} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$ implies a choice of basis vectors $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}$ such that $\mathbf{a} = \alpha_1 \mathbf{v_1} + \alpha_2 \mathbf{v_2} + \alpha_3 \mathbf{v_3}$.

Specifying **point** coordinates $\mathbf{p} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$ implies a choice of basis vectors $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}$ and a reference point, $\mathbf{p_0}$, such that $\mathbf{p} = \mathbf{p_0} + \beta_1 \mathbf{v_1} + \beta_2 \mathbf{v_2} + \beta_3 \mathbf{v_3}$.

Frame

To specify a **frame:** in 3D

a reference point, $\mathbf{P_0}$, and basis vectors $\{\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}\}$

Notation : $F = (P_0, v_1, v_2, v_3)$

The Cartesian frame

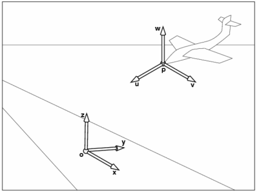
The 3 basis vectors, e_1, e_2, e_3, are chosen to be unit length and mutually perpendicular.

- $e_1 \cdot e_1 = e_2 \cdot e_2 = e_3 \cdot e_3 = 1$
- $e_1 \cdot e_2 = e_1 \cdot e_3 = e_2 \cdot e_3 = 0$

Frames in OpenGL

Vertices can have different coordinates in all of the different frames:

- Model (sometimes called Object or Local frame)
- World frame
- Camera (or Eye) frame
- Window (or Screen) frame



The transformations from one frame to another are specified by using 4×4 matrices.

Coordinate system

Three non-coplanar vectors v_1, v_2, v_3 define a **coordinate system**.
We can then **uniquely** represent any vector, w, as a linear combination of the basis vectors

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$$

Using matrix notation, we can rewrite this as

$$w = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Frame

Three non-coplanar basis vectors v_1, v_2, v_3 and a reference point P_0 define a **frame**.
Any point, P, can be represented in this frame as some vector displacement added to the reference point (origin).

$$\begin{aligned} P &= P_0 + \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 \\ P &= P_0 + \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \end{aligned}$$

Homogeneous coordinates

First, let's define $0 \cdot P = \mathbf{0}$ and $1 \cdot P = P$.

Then we can write a new representation for the point $P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3$ in terms of the 4 elements of the frame

$$P = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ 1 \end{bmatrix}^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

We will call $(\beta_1, \beta_2, \beta_3, 1)$ the homogeneous coordinates of the point P. Usually these coordinates will be represented as a column vector.

Homogeneous coordinates

Let's write a new representation for the vector $w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$.

In this case the vector representation does not depend on the reference point. So, the coefficient multiplying P_0 is 0.

$$w = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 0 \end{bmatrix}^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

We will call $(\alpha_1, \alpha_2, \alpha_3, 0)$ the homogeneous coordinates of the vector w.

Change of coordinate system

Given 2 frames, $F_1 = (P_0, v_1, v_2, v_3)$
and $F_2 = (Q_0, u_1, u_2, u_3)$, write F_2 in terms of F_1.

Why? Given coordinates of a point in F_1, we can compute coordinates in F_2.

How? First write the basis vectors of F_2 in terms of the basis vectors of F_1.

$$\begin{aligned} u_1 &= \gamma_{11}v_1 + \gamma_{12}v_2 + \gamma_{13}v_3 \\ u_2 &= \gamma_{21}v_1 + \gamma_{22}v_2 + \gamma_{23}v_3 \\ u_3 &= \gamma_{31}v_1 + \gamma_{32}v_2 + \gamma_{33}v_3 \end{aligned}$$

Then write Q_0 in terms of frame F_1

$$Q_0 = P_0 + \gamma_{41}v_1 + \gamma_{42}v_2 + \gamma_{43}v_3$$

Change of coordinate system

We can write all 4 equations in matrix form

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

We will be using **M** to denote the 4×4 matrix which transforms frame (P_0, v_1, v_2, v_3) into (Q_0, u_1, u_2, u_3)

$$\mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

This is known as an **affine transformation** matrix.

Change of coordinate system

Let **a** be the homogeneous representation of a point in (v_1, v_2, v_3, P_0) and **b** be the homogeneous representation of the same point in (u_1, u_2, u_3, Q_0). (Note: the following holds if **a** and **b** are vectors also)

$$\mathbf{b}^T \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = \mathbf{b}^T \mathbf{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

Since the representation of the point is unique (**a** only has one representation in frame (P_0, v_1, v_2, v_3)), we have

$$\mathbf{b}^T \mathbf{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} = \mathbf{a}^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

Change of coordinate system

Since $\mathbf{b}^T\mathbf{M} = \mathbf{a}^T$ and since $(\mathbf{b}^T\mathbf{M})^T = \mathbf{M}^T\mathbf{b}$ we can write

$$\mathbf{a} = \mathbf{M}^T\mathbf{b}$$

If matrix $\mathbf{M}$, transforms frame $F_1 = (P_0, v_1, v_2, v_3)$ into $F_2 = (Q_0, u_1, u_2, u_3)$, then $\mathbf{M}^T$ transforms homogeneous representations in F_2 into representations in F_1.

We can transform representations in the other direction ($F_1 \rightarrow F_2$) by inverting the matrix:

$$\mathbf{b} = (\mathbf{M}^T)^{-1}\mathbf{a}$$

Transformations

- Transformation : a function that takes a point (or vector) and maps it to another point (or vector)
  - $T(v) = w$
- Linear Transformation : Preserves linear combinations
  - $T(\alpha_1 v_1 + \alpha_2 v_2) = \alpha_1 T(v_1) + \alpha_2 T(v_2)$
- Affine Transformation : Preserves affine combinations
  - $T(\alpha v_1 + (1-\alpha)v_2) = \alpha T(v_1) + (1-\alpha)T(v_2)$

Properties:

- Every linear transformation is equivalent to a change in frames : we can perform linear transformations by multiplying homogeneous representations by a matrix.
- Affine transformations preserve lines : this allows us to transform lines (and planes) by transforming the points on the lines (or planes).

Affine transformations

When working with homogeneous representations, the affine transformation should **not** map points to vectors, or vice-versa, so the matrix will leave the last component unchanged:

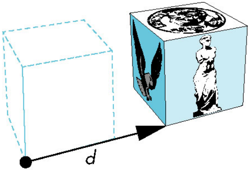
$$\mathbf{A} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation $\mathbf{v} = \mathbf{A}\mathbf{u}$ will map points to points and vectors to vectors.

Translation

We could use the point-vector addition operation to perform translation of points:

$$P' = P + d = \begin{bmatrix} P_x + d_x \\ P_y + d_y \\ P_z + d_z \end{bmatrix}$$



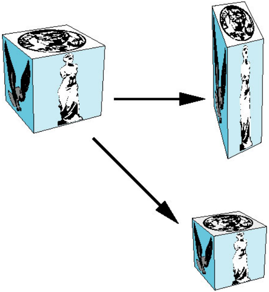
(b)

Translation

We can also write this as an affine transformation of the homogeneous representation:

$$\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} P_x + d_x \\ P_y + d_y \\ P_z + d_z \\ 1 \end{bmatrix}$$

Scaling



Properties

- The origin is unchanged by a scaling operation.
- Independent scaling can be applied along the 3 coordinate axes.

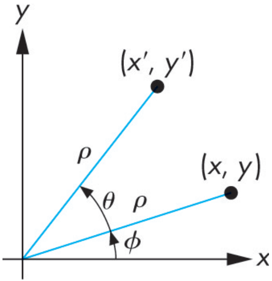
Scaling

$$\begin{bmatrix} \beta_x & 0 & 0 & 0 \\ 0 & \beta_y & 0 & 0 \\ 0 & 0 & \beta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} \beta_x P_x \\ \beta_y P_y \\ \beta_z P_z \\ 1 \end{bmatrix}$$

Properties

- The origin is unchanged by a scaling operation.
- Independent scaling can be applied along the 3 coordinate axes.
- Can also be applied to vectors.

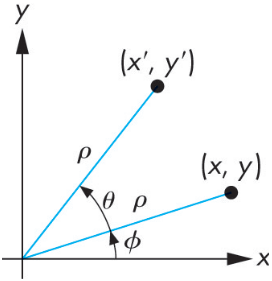
2D Rotation : Geometric derivation



Rewrite in polar coordinates:

$$
\begin{aligned}
x &= \rho \cos \phi \\
y &= \rho \sin \phi \\
x' &= \rho \cos(\theta + \phi) \\
y' &= \rho \sin(\theta + \phi)
\end{aligned}
$$

2D Rotation : Geometric derivation



Rewrite in polar coordinates:

$$
\begin{aligned}
x &= \rho \cos \phi \\
y &= \rho \sin \phi \\
x' &= \rho \cos(\theta + \phi) \\
y' &= \rho \sin(\theta + \phi)
\end{aligned}
$$

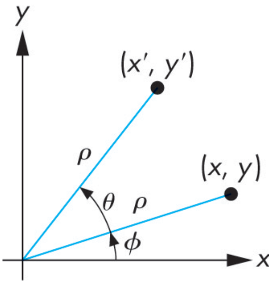
Using the trig identities

$$
\begin{aligned}
\cos(\theta + \phi) &= \cos \phi \cos \theta - \sin \phi \sin \theta \\
\sin(\theta + \phi) &= \cos \phi \sin \theta + \sin \phi \cos \theta
\end{aligned}
$$

Rewrite x', y'

$$
\begin{aligned}
x' &= \rho \cos \phi \cos \theta - \rho \sin \phi \sin \theta \\
y' &= \rho \cos \phi \sin \theta + \rho \sin \phi \cos \theta
\end{aligned}
$$

2D Rotation : Geometric derivation



Rewrite x', y' in terms of x, y

Rewrite in polar coordinates:

$$
\begin{aligned}
x &= \rho \cos \phi \\
y &= \rho \sin \phi \\
x' &= \rho \cos \phi \cos \theta - \rho \sin \phi \sin \theta \\
y' &= \rho \cos \phi \sin \theta + \rho \sin \phi \cos \theta
\end{aligned}
$$

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

2D Rotation : Geometric derivation

We can rewrite this system of equations

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

in matrix form as

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

This is equivalent to 3D rotation about the z-axis

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotation in homogeneous coordinates

To rotate homogeneous points (w = 1) or vectors (w = 0) we use a 4×4 matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

This can also be derived by converting frames ...

Rotation in homogeneous coordinates

Rotation about the x-axis:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Rotation about the y-axis:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Matrix concatenation : sequences of transformations

Matrix multiplication is associative.

We can combine affine transformations by multiplying matrices. If $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are transformation matrices, we can apply the **sequence** to a point, $\mathbf{p}$ by multiplying:

$$\mathbf{q} = \mathbf{CBAp}$$

this is equivalent to applying the matrix-vector multiplication 3 times:

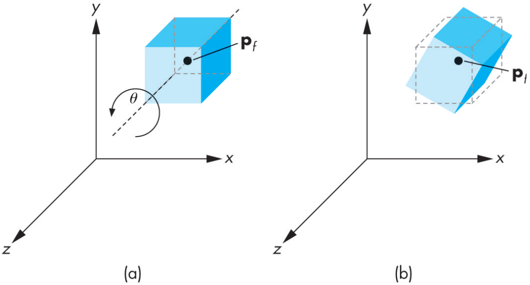
$$\mathbf{q} = \mathbf{C}(\mathbf{B}(\mathbf{Ap}))$$

or computing the matrix-matrix product $\mathbf{M} = \mathbf{CBA}$ and then transforming the point

$$\mathbf{q} = \mathbf{Mp}$$

Example : rotation about a fixed point

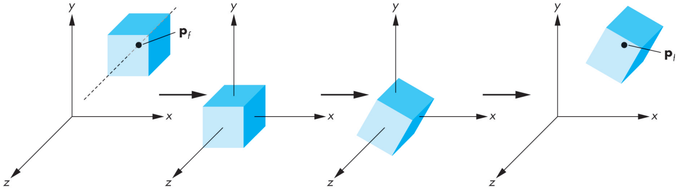
Suppose we want to rotate a cube about its own center.



- Recall that the rotation transformation rotates points about the coordinate frame origin.
- The origin is a **fixed point** of the transform : it remains unchanged after transformation.

Example : rotation about a fixed point

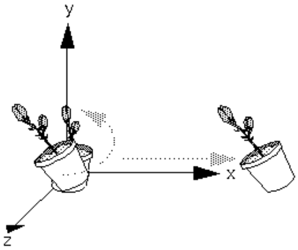
We can rotate the cube about its own center by combining 3 transformations.

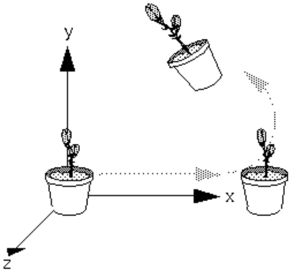


- Translate the center of the cube to the origin : $\mathbf{T}(-\mathbf{p_f})$
- Apply the desired rotation : $\mathbf{R}(\theta)$
- Translate the center of the cube back to its original location : $\mathbf{T}(\mathbf{p_f})$
- This can be done with a single matrix $\mathbf{M} = \mathbf{T}(\mathbf{p_f})\mathbf{R}(\theta)\mathbf{T}(-\mathbf{p_f})$

Two ways to combine rotation and translation

Matrix multiplication does not commute.





- Rotate then translate
- Translate then rotate

Just keep in mind that rotation occurs **about the origin** of the frame.