Representation

Objectives

- Introduce concepts such as dimension and basis
- Introduce coordinate systems for representing vectors spaces and frames for representing affine spaces
- Discuss change of frames and bases
- Introduce homogeneous coordinates

Linear Independence

- A set of vectors $v_1, v_2, \ldots, v_n$ is *linearly independent* if

  $\alpha_1 v_1 + \alpha_2 v_2 + .. \alpha_n v_n = 0$ iff $\alpha_1 = \alpha_2 = \ldots = 0$
- If a set of vectors is linearly independent, we cannot represent one in terms of the others
- If a set of vectors is linearly dependent, as least one can be written in terms of the others

Dimension

- In a vector space, the maximum number of linearly independent vectors is fixed and is called the *dimension* of the space
- In an *n*-dimensional space, any set of n linearly independent vectors form a *basis* for the space
- Given a basis $v_1, v_2, \ldots, v_n$, any vector v can be written as

$$v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$$

where the $\{\alpha_i\}$ are unique

Representation

- Until now we have been able to work with geometric entities without using any frame of reference, such as a coordinate system
- Need a frame of reference to relate points and objects to our physical world.
  - For example, where is a point? Can't answer without a reference system
  - World coordinates
  - Camera coordinates

Coordinate Systems

- Consider a basis $v_1, v_2, \ldots, v_n$
- A vector is written $v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$
- The list of scalars $\{\alpha_1, \alpha_2, \ldots \alpha_n\}$ is the *representation* of v with respect to the given basis
- We can write the representation as a row or column array of scalars

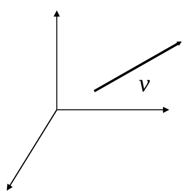
$$\mathbf{a} = [\alpha_1 \ \ \alpha_2 \ \ \ldots \ \ \alpha_n]^T = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \alpha_n \end{bmatrix}$$

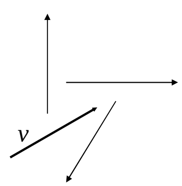
Example

- $v = 2v_1 + 3v_2 - 4v_3$
- $\mathbf{a} = [2 \ 3 \ -4]^T$
- Note that this representation is with respect to a particular basis
- For example, in OpenGL we start by representing vectors using the object basis but later the system needs a representation in terms of the camera or eye basis

Coordinate Systems

- Which is correct?

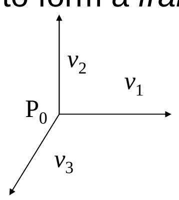




- Both are because vectors have no fixed location

Frames

- A coordinate system is insufficient to represent points
- If we work in an affine space we can add a single point, the *origin*, to the basis vectors to form a *frame*



Representation in a Frame

- Frame determined by (P_0, v_1, v_2, v_3)
- Within this frame, every vector can be written as

  $v=\alpha_1 v_1+ \alpha_2 v_2 +\ldots+\alpha_n v_n$
- Every point can be written as

  $P = P_0 + \beta_1 v_1+ \beta_2 v_2 +\ldots+\beta_n v_n$

Confusing Points and Vectors

Consider the point and the vector

$P = P_0 + \beta_1 v_1+ \beta_2 v_2 +\ldots+\beta_n v_n$

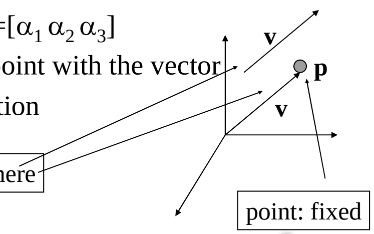
$v=\alpha_1 v_1+ \alpha_2 v_2 +\ldots+\alpha_n v_n$

They appear to have the similar representations

$\mathbf{p}=[\beta_1\ \beta_2\ \beta_3]$ $\mathbf{v}=[\alpha_1\ \alpha_2\ \alpha_3]$

which confuses the point with the vector

A vector has no position



A Single Representation

If we define $0\bullet P = \mathbf{0}$ and $1\bullet P = P$ then we can write

$v=\alpha_1 v_1+ \alpha_2 v_2 +\alpha_3 v_3 = [\alpha_1\ \alpha_2\ \alpha_3\ 0]\ [v_1\ v_2\ v_3\ P_0]^T$

$P = P_0 + \beta_1 v_1+ \beta_2 v_2 +\beta_3 v_3 = [\beta_1\ \beta_2\ \beta_3\ 1]\ [v_1\ v_2\ v_3\ P_0]^T$

Thus we obtain the four-dimensional *homogeneous coordinate* representation

$\mathbf{v} = [\alpha_1\ \alpha_2\ \alpha_3\ 0]^T$

$\mathbf{p} = [\beta_1\ \beta_2\ \beta_3\ 1]^T$

Homogeneous Coordinates

The homogeneous coordinates form for a three dimensional point [x y z] is given as

$\mathbf{p} = [x'\ y'\ z'\ w]^T = [wx\ wy\ wz\ w]^T$

We return to a three dimensional point (for $w\neq 0$) by

$x \leftarrow x'/w$

$y \leftarrow y'/w$

$z \leftarrow z'/w$

If $w=0$, the representation is that of a vector

Note that homogeneous coordinates replace points in three dimensions by lines through the origin in four dimensions

For w=1, the representation of a point is [x y z 1]

Homogeneous Coordinates and Computer Graphics

- Homogeneous coordinates are key to all computer graphics systems
  - All standard transformations (rotation, translation, scaling) can be implemented with matrix multiplications using 4 x 4 matrices
  - Hardware pipeline works with 4 dimensional representations
  - For orthographic viewing, we can maintain w=0 for vectors and w=1 for points
  - For perspective we need a *perspective division*

Change of Coordinate Systems

- Consider two representations of a the same vector with respect to two different bases. The representations are

$$\mathbf{a} = [\alpha_1\ \alpha_2\ \ \alpha_3]^T$$
$$\mathbf{b} = [\beta_1\ \ \beta_2\ \ \beta_3]^T$$

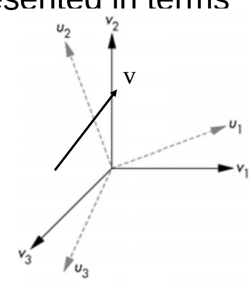
where

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 = [\alpha_1\ \alpha_2\ \alpha_3]\ [v_1\ v_2\ v_3]^T$$
$$= \beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3 = [\beta_1\ \beta_2\ \beta_3]\ [u_1\ u_2\ u_3]^T$$

Representing second basis in terms of first

Each of the basis vectors, u_1,u_2, u_3, are vectors that can be represented in terms of the first basis

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



The coefficients define a 3 x 3 matrix

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

Matrix Form

Then $\mathbf{u} = \mathbf{M}\,\mathbf{v}$

and the two representations of $\mathbf{w}$ can be related by

$$\mathbf{w} = \mathbf{b}^T\,\mathbf{u} = \mathbf{b}^T\,\mathbf{M}\mathbf{v} = \mathbf{a}^T\,\mathbf{v}$$

Hence $\mathbf{b}^T\,\mathbf{M} = \mathbf{a}^T$ or

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

See text for numerical examples

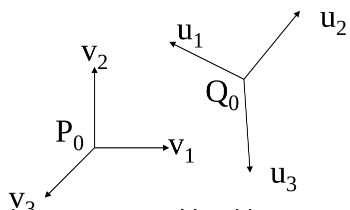
Change of Frames

- We can apply a similar process in homogeneous coordinates to the representations of both points and vectors

Consider two frames:

(P_0, v_1, v_2, v_3)

(Q_0, u_1, u_2, u_3)



- Any point or vector can be represented in either frame
- We can represent Q_0, u_1, u_2, u_3 in terms of P_0, v_1, v_2, v_3

Representing One Frame in Terms of the Other

Extending what we did with change of bases

$u_1 = \gamma_{11}v_1+\gamma_{12}v_2+\gamma_{13}v_3$
$u_2 = \gamma_{21}v_1+\gamma_{22}v_2+\gamma_{23}v_3$
$u_3 = \gamma_{31}v_1+\gamma_{32}v_2+\gamma_{33}v_3$
$Q_0 = \gamma_{41}v_1+\gamma_{42}v_2+\gamma_{43}v_3 +\gamma_{44}P_0$

defining a 4 x 4 matrix

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

Working with Representations

Within the two frames any point or vector has a representation of the same form

$\mathbf{a}=[\alpha_1\ \alpha_2\ \ \alpha_3\ \alpha_4\,]$ in the first frame
$\mathbf{b}=[\beta_1\ \beta_2\ \ \beta_3\ \beta_4\,]$ in the second frame

where $\alpha_4 = \beta_4 = 1$ for points and $\alpha_4 = \beta_4 = 0$ for vectors and

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

The matrix $\mathbf{M}$ is 4 x 4 and specifies an affine transformation in homogeneous coordinates

Affine Transformations

- Every linear transformation is equivalent to a change in frames
- Every affine transformation preserves lines
- However, an affine transformation has only 12 *degrees of freedom* because 4 of the elements in the matrix are fixed and are a subset of all possible 4 x 4 linear transformations

The World and Camera Frames

- When we work with representations, we work with n-tuples or arrays of scalars
- Changes in frame are then defined by 4 x 4 matrices
- In OpenGL, the base frame that we start with is the world frame
- Eventually we represent entities in the camera frame by changing the world representation using the model-view matrix
- Initially these frames are the same (**M**=**I**)

Other Coordinates

- Clip coordinates
  - projected to clip coordinates
  - cube centered around origin used for clipping
- Normalized device coordinates
  - produced by division by w, called perspective division
- Window coordinates
  - using viewport produce 3 dimensional representation in pixel units

Moving the Camera

If objects are on both sides of z=0, we must move camera frame

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$