Week 05: Coordinate Systems and Frames

CS-537: Interactive Computer Graphics

Dr. Chloe LeGendre
Department of Computer Science

Some materials from the companion slides of Angel and Shreiner, “Interactive Computer Graphics, A Top-Down Approach with WebGL.”

Objectives

- Define dimensionality and basis
- Introduce coordinate systems for representing vector spaces and frames for representing affine spaces
- Introduce homogenous coordinates
- Introduce change of representations for both vectors and points

Linear Independence

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

$$\alpha_1 v_1+\alpha_2 v_2+..\ \alpha_n v_n=0 \text{ if and only if } \alpha_1=\alpha_2=\ldots=0$$

- 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**, at 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
    - Although we briefly described “screen space” or window coordinates for the WebGL canvas
    - … and “clip space” coordinates – how we specified points in Assignment 1
- Especially now when moving to 3D, we need a frame of reference to relate points and objects to our physical world.
    - For example, where is a point?
        - We can’t answer this question without a reference system
    - Two possibilities (and we’ll see more later):
        - World coordinates (we define an arbitrary origin point)
        - Camera coordinates (define the origin at the camera’s center of projection)

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

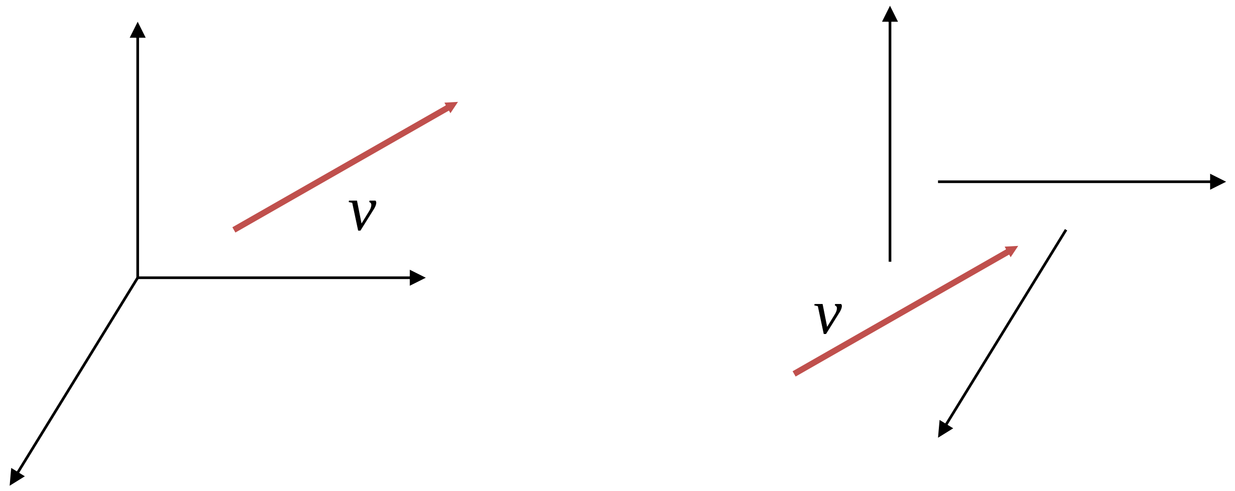
$$\mathbf{a}=[\alpha_1 \ \ \alpha_2 \ \ \ldots \ \alpha_n]^T = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ \alpha_n \end{bmatrix}$$

Example Representation

- $v=2v_1+3v_2-4v_3$
- $\mathbf{a}=[2\ 3\ –4]^T$
- Note that this representation is with respect to a particular basis
- For example, in WebGL we will 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 (II)

- Which is correct?



- Both! Because vectors have no fixed location in space.

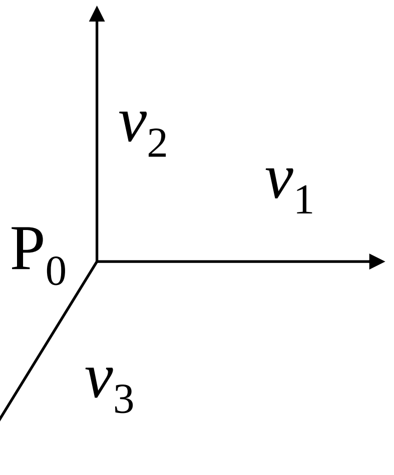
Frames

- 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*
- Frame in 3D defined by 3 linearly independent vectors and a 3D origin point (see diagram at the right)*:*
  - 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

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



Points vs. Vectors

- Consider the point and the vector

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

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

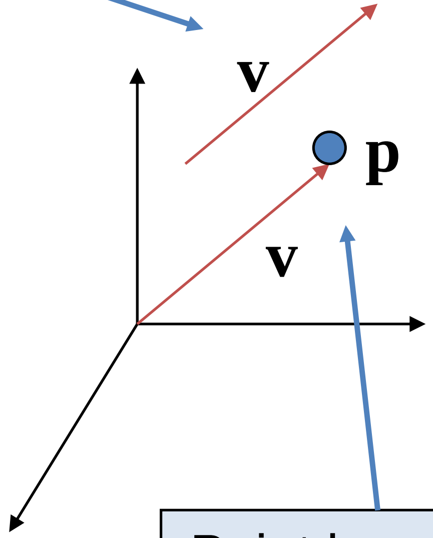
- They appear to have the similar representations:

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

  (which may confuse the point with the vector)

- However, a vector has no position

Vector can be placed anywhere



Point has fixed location

Motivating Homogenous Coordinates

- How to arrive at a single representation that can correctly define both points and vectors?

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

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

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

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

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

Homogenous Coordinates and Computer Graphics

- Homogeneous coordinates are key to all computer graphics systems
  - All standard 3D 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*
  - [Viewing related topics covered Week 06]

Change of Coordinate Systems

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

$\mathbf{a} = [\alpha_1\ \alpha_2\ \ \alpha_3\ ]$

$\mathbf{b} = [\beta_1\ \ \beta_2\ \ \beta_3]$

where:

$$V = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 = [\alpha_1\ \alpha_2\ \alpha_3]\ [v_1\ v_2\ v_3]^T$$

$$= \beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3 = [\beta_1\ \beta_2\ \beta_3]\ [u_1\ u_2\ u_3]^T$$

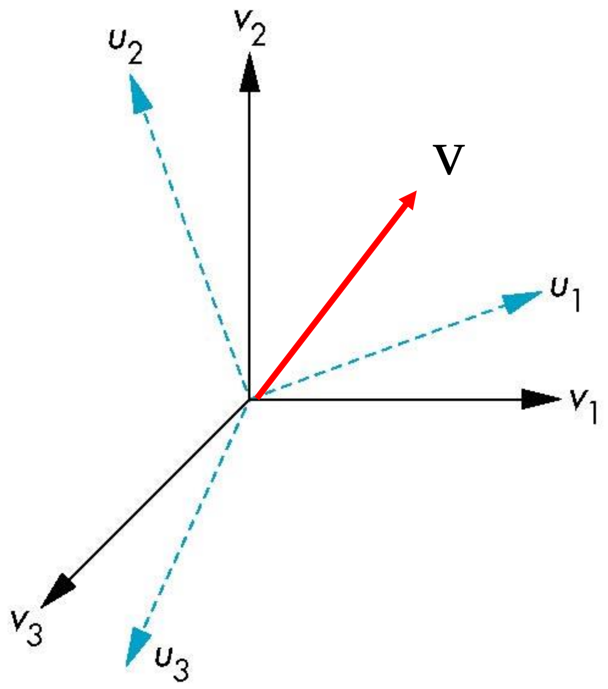
Representing the second basis in terms of the first

- Each of the basis vectors, u1,u2, u3, are vectors that can be represented in terms of the first basis:

$$u_1 = \gamma_{11}v_1+\gamma_{12}v_2+\gamma_{13}v_3$$

$$u_2 = \gamma_{21}v_1+\gamma_{22}v_2+\gamma_{23}v_3$$

$$u_3 = \gamma_{31}v_1+\gamma_{32}v_2+\gamma_{33}v_3$$



Matrix Form of change of basis

- The coefficients define a 3 x 3 matrix

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

- and the bases can be related by:

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

- Textbook: Ch 4 Sec 3.2

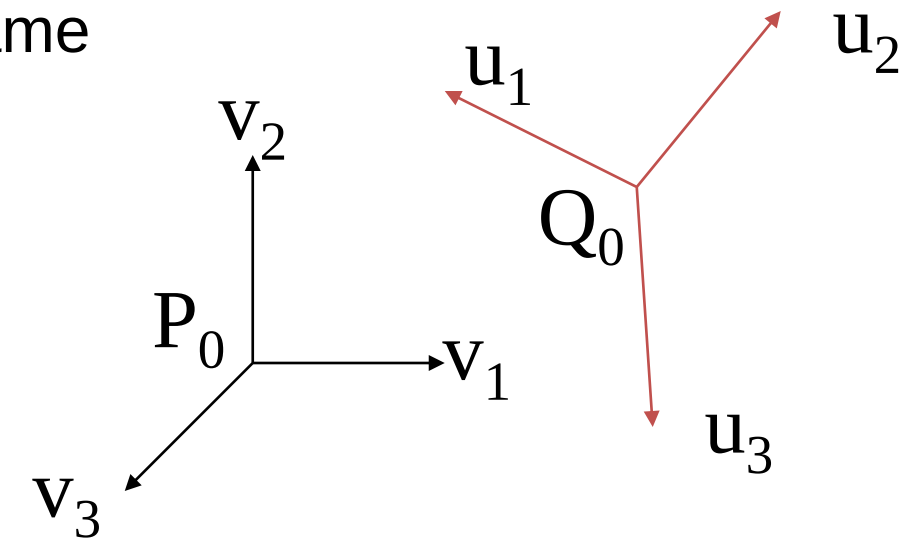
Change of Frames

- 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

  $\alpha_4=\beta_4=1$ 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 (DOF) because 4 of the elements in the matrix are fixed and are a subset of all possible 4 x 4 linear transformations

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

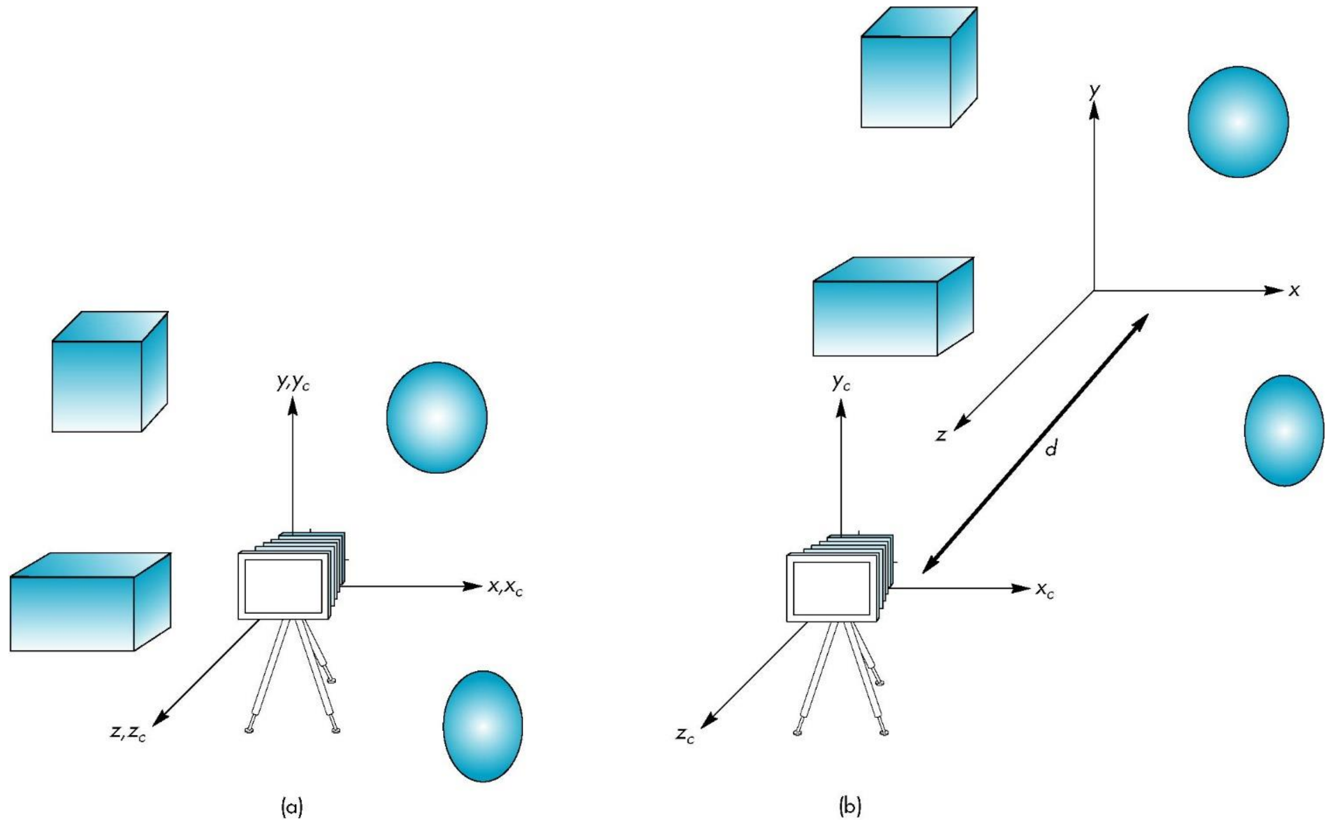
The World and Camera Frames

- 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 WebGL, 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 a 4x4 matrix called the “**model-view**” matrix
- Initially these frames are the same (**M**=**I**) [identity matrix]

Moving the Camera

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

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



CS-537: Interactive Computer Graphics

Six Typical Coordinate Frames of a Graphics Pipeline (I)

1. Model Coordinates
   - Example use case: specify an object as a set of points in an OBJ file
2. Object (or World) Coordinates
   - We want to layout different objects in our virtual world, but most models are saved in OBJ files with vertices centered around the origin. We need to define one common coordinate frame for all objects so we can position them relative to each other.
3. Eye (or Camera) Coordinates
   - The camera center of projection defines its own frame, which is used for rendering in WebGL. We need to define our vertices in this coordinate frame for rendering in shaders.

Six Typical Coordinate Frames of a Graphics Pipeline (II)

4. Clip Coordinates
   - A cube centered at the origin, this frame is used internally by WebGL pipeline to decide which points are clipped and considered out of view during rendering. In Assignment 1, we were defining vertices directly in clip coordinates to delay our coordinate frame discussion.
5. Normalized Device Coordinates (sometimes called NDC)
   - We will discuss this next week, as it has to do with projection (3D -> 2D)
6. Window (or Screen) Coordinates
   - Window coordinates: a location on the screen (device) with retained 3D depth information
   - Screen coordinates: a location on the screen (device) without retained depth information
   - This coordinate frame corresponds to a pixel location

- We'll explore these in more depth next week.