Magnitude of a Vector

- Magnitude of **a**

$$|\mathbf{a}| = \sqrt{a_1^{\,2} + a_2^{\,2} \ldots\ldots\ldots + a_n^{\,2}}$$

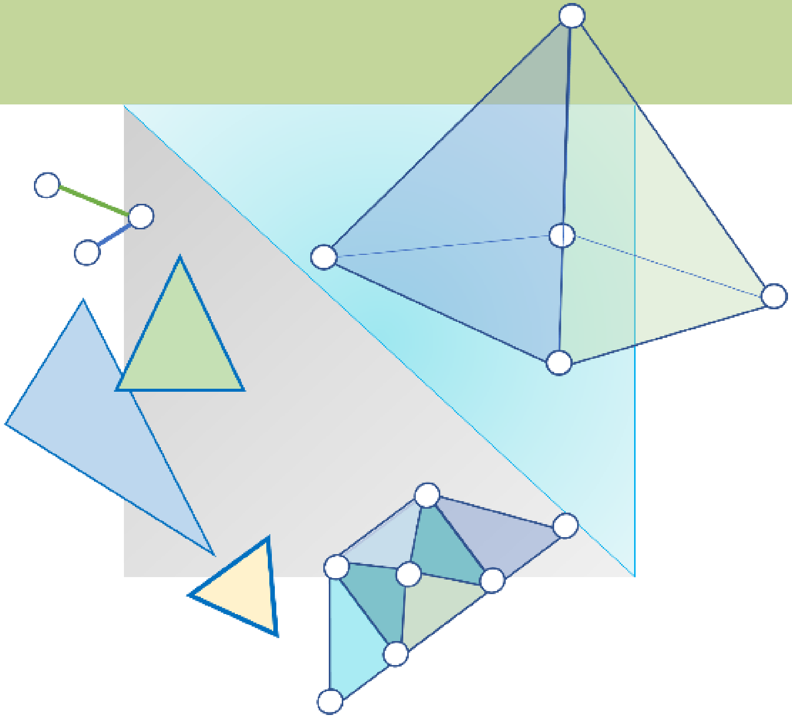
  – If a =(2, 5, 6) $\qquad |\mathbf{a}| = \sqrt{2^2 + 5^2 + 6^2} = \sqrt{65}$

- Normalizing a vector $\qquad \hat{\mathbf{a}} = \dfrac{\mathbf{a}}{|\mathbf{a}|} = \dfrac{vector}{magnitude}$

$$\hat{\mathbf{a}} = \left(\frac{2}{\sqrt{65}}, \frac{5}{\sqrt{65}}, \frac{6}{\sqrt{65}} \right)$$

CITS3003 Graphics & Animation

Lecture 8:
Coordinate Frame Transformations



Breakdown of Lectures

1. Introduction & Image Formation
2. Programming with OpenGL
3. OpenGL: Pipeline Architecture
4. OpenGL: An Example Program
5. Vertex and Fragment Shaders 1
6. Vertex and Fragment Shaders 2
7. Representation and Coordinate Systems
8. Coordinate Frame Transformations
9. Transformations and Homogeneous Coordinates
10. Input, Interaction and Callbacks
11. Mid-semester Test
12. More on Callbacks
13. 3D Hidden Surface Removal
13. Computer Viewing

- Study break

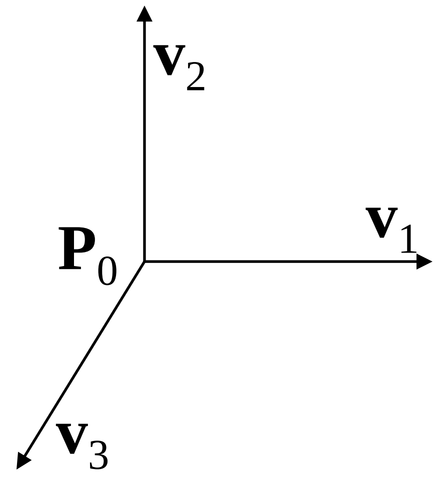
15. Programming Project Discussion
16. Shading
17. Shading Models
18. Shading in OpenGL
19. Texture Mapping
20. Texture Mapping in OpenGL
21. Hierarchical Modelling
22. 3D Modelling: Subdivision Surfaces
23. Animation Fundamentals and Quaternions
24. Skinning

Content

- Learn how to define and change coordinate frames
- Derive homogeneous coordinate transformation matrices
- Introduce standard transformations
  - Rotation, Translation, Scaling, Shear

Coordinate Frame

- Basis vectors alone cannot represent points
- We can add a single point, the *origin*, to the basis vectors to form a *coordinate frame*



Representation in a Coordinate Frame

- A coordinate system (or coordinate frame) is determined by $(\mathbf{P}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$
- Within this coordinate frame, every vector $\mathbf{v}$ can be written as

$$\mathbf{v} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3$$

Every point can be written as

$$\mathbf{P} = \mathbf{P}_0 + \beta_1\mathbf{v}_1 + \beta_2\mathbf{v}_2 + \beta_3\mathbf{v}_3$$

for some $\alpha_1, \alpha_2, \alpha_3$, and $\beta_1, \beta_2, \beta_3$

Homogeneous Coordinates

- Consider the point **P** and the vector **v**, where

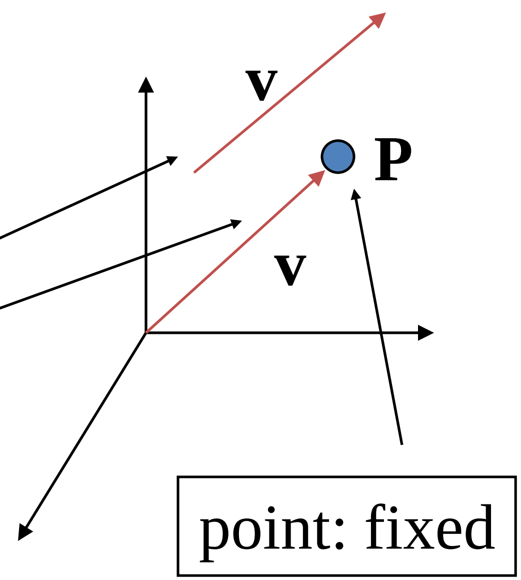
$$\mathbf{P} = \mathbf{P}_0 + \beta_1\mathbf{v}_1 + \beta_2\mathbf{v}_2 + \beta_3\mathbf{v}_3$$

$$\mathbf{v} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3$$

- They appear to have similar representations:

$\mathbf{P} = [\beta_1, \beta_2, \beta_3]^T$, $\mathbf{v} = [\alpha_1, \alpha_2, \alpha_3]^T$ which confuses the point with the vector

A vector has no position



Vector can be placed anywhere

A Single Representation

- Assuming $0 \cdot \mathbf{P} = \mathbf{0}$ and $1 \cdot \mathbf{P} = \mathbf{P}$, we can write

$$\mathbf{v} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 + 0 \cdot \mathbf{P}_0$$
$$\mathbf{P} = \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + \beta_3 \mathbf{v}_3 + \mathbf{P}_0 = \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + \beta_3 \mathbf{v}_3 + 1 \cdot \mathbf{P}_0$$

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

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

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

Homogeneous Coordinates

- The homogeneous coordinate form for a three-dimensional point $[x \quad y \quad z]^{\mathrm{T}}$ is given as

$$\mathbf{p} = [x \quad y \quad z \quad 1]^{\mathrm{T}} \rightarrow [wx \quad wy \quad wz \quad w]^{\mathrm{T}} = [x' \quad y' \quad z' \quad w]^{\mathrm{T}}$$

- 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
- Homogeneous coordinates replace points in three dimensions by lines through the origin in four dimensions
- For $w = 1$, the representation of a point is $[x \quad y \quad z \quad 1]^{\mathrm{T}}$

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*

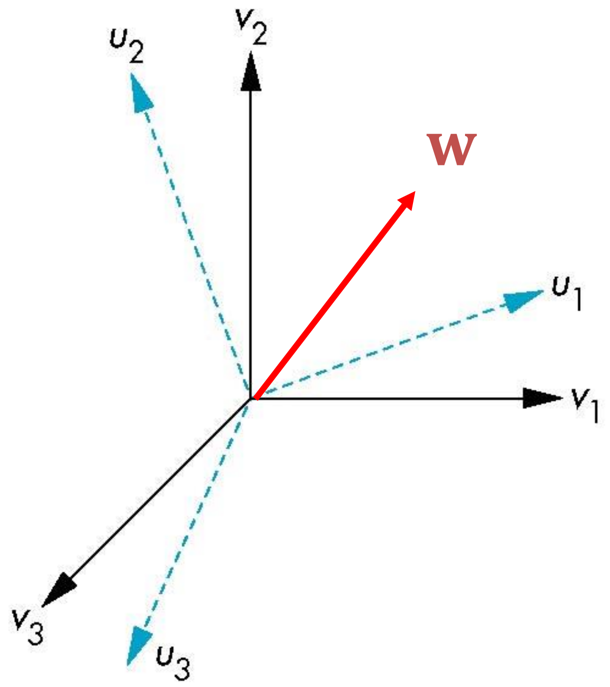
Representing the Second Basis in Terms of the First

- How can we relate **u** with **v**?
- Each of the basis vectors $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ are vectors that can be represented in terms of the first set of basis vectors,

  i.e.,

$$\mathbf{u}_1 = \gamma_{11}\mathbf{v}_1 + \gamma_{12}\mathbf{v}_2 + \gamma_{13}\mathbf{v}_3$$
$$\mathbf{u}_2 = \gamma_{21}\mathbf{v}_1 + \gamma_{22}\mathbf{v}_2 + \gamma_{23}\mathbf{v}_3$$
$$\mathbf{u}_3 = \gamma_{31}\mathbf{v}_1 + \gamma_{32}\mathbf{v}_2 + \gamma_{33}\mathbf{v}_3$$

for some $\gamma_{11}, \dots, \gamma_{33}$



Representing the Second Basis in Terms of the First (cont.)

- $\mathbf{u}_1 = \gamma_{11}\mathbf{v}_1 + \gamma_{12}\mathbf{v}_2 + \gamma_{13}\mathbf{v}_3$ can be written as:

$$\mathbf{u}_1 = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3]\begin{bmatrix}\gamma_{11}\\ \gamma_{12}\\ \gamma_{13}\end{bmatrix} = \mathbf{V}\begin{bmatrix}\gamma_{11}\\ \gamma_{12}\\ \gamma_{13}\end{bmatrix}$$

- Similarly, $\mathbf{u}_2 = \gamma_{21}\mathbf{v}_1 + \gamma_{22}\mathbf{v}_2 + \gamma_{23}\mathbf{v}_3$ and $\mathbf{u}_3 = \gamma_{31}\mathbf{v}_1 + \gamma_{32}\mathbf{v}_2 + \gamma_{33}\mathbf{v}_3$ can be written as:

$$\mathbf{u}_2 = \mathbf{V}\begin{bmatrix}\gamma_{21}\\ \gamma_{22}\\ \gamma_{23}\end{bmatrix}$$

$$\mathbf{u}_3 = \mathbf{V}\begin{bmatrix}\gamma_{31}\\ \gamma_{32}\\ \gamma_{33}\end{bmatrix}$$

Representing the Second Basis in Terms of the First (cont.)

- We can put the terms $\gamma_{11}, \ldots, \gamma_{33}$ into a 3×3 matrix:

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

then we have:

$$[\mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{u}_3] = \mathbf{V}\,\mathbf{M}^{\mathrm{T}}$$

That is,

$$\mathbf{U} = \mathbf{V}\,\mathbf{M}^{\mathrm{T}}$$

The superscript T denotes *matrix transpose*

The same vector w represented in two coordinate systems

- We can write

$$\mathbf{w} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3$$
$$\mathbf{w} = \beta_1\mathbf{u}_1 + \beta_2\mathbf{u}_2 + \beta_3\mathbf{u}_3$$

Let’s call this 3×3 matrix $\mathbf{V}$

as follows:

$$\mathbf{w} = \overbrace{[\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3]}\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix} = \mathbf{V}\,\mathbf{a}$$

$$\mathbf{w} = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{u}_3]\begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\end{bmatrix} = \mathbf{U}\,\mathbf{b}$$

Each $\mathbf{v}_i$ is a column vector of 3 components

Representing the Second Basis in Terms of the First (cont.)

- In this example, we have $\mathbf{w} = \mathbf{V}\,\mathbf{a}$ and $\mathbf{w} = \mathbf{U}\,\mathbf{b}$.
- So

$$\mathbf{V}\,\mathbf{a} = \mathbf{U}\,\mathbf{b}$$

- With $\mathbf{U} = \mathbf{V}\,\mathbf{M}^{\mathrm{T}}$, we have

$$\mathbf{V}\,\mathbf{a} = \mathbf{V}\,\mathbf{M}^{\mathrm{T}}\mathbf{b}$$
$$\Rightarrow \mathbf{a} = \mathbf{M}^{\mathrm{T}}\mathbf{b}$$

- Thus, $\mathbf{a}$ and $\mathbf{b}$ are related by $\mathbf{M}^{\mathrm{T}}$

Representing the Second Basis in Terms of the First (cont.)

- In this example, we have $\mathbf{w} = \mathbf{V}\,\mathbf{a}$ and $\mathbf{w} = \mathbf{U}\,\mathbf{b}$.
- So

$$\mathbf{V}\,\mathbf{a} = \mathbf{U}\,\mathbf{b}$$

- With $\mathbf{U} = \mathbf{V}\,\mathbf{M}^{\mathrm{T}}$, we have

$$\mathbf{V}\,\mathbf{a} = \mathbf{V}\,\mathbf{M}^{\mathrm{T}}\mathbf{b}$$

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

or

$\boldsymbol{b} = \boldsymbol{T}\boldsymbol{a}$

where,

$\boldsymbol{T} = (\boldsymbol{M}^{\boldsymbol{T}})^{-1}$

- Thus, $\mathbf{a}$ and $\mathbf{b}$ are related by $\mathbf{M}^{\mathrm{T}}$

Representation w.r.t the second basis (U)

Representation w.r.t first basis (V)

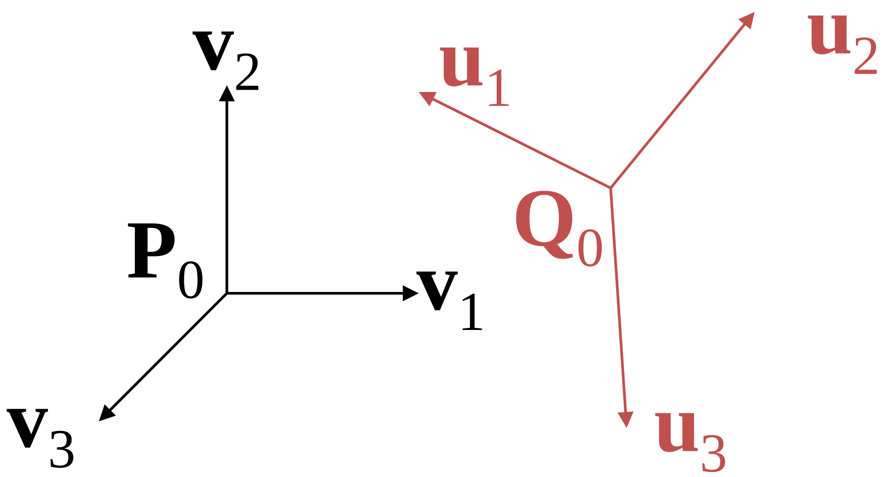
Change of Coordinate Frames

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

Consider two coordinate frames:

$(\mathbf{P}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$

$(\mathbf{Q}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$



- Any point or vector can be represented in either coordinate frame.
- We can represent $(\mathbf{Q}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ in terms of $(\mathbf{P}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$

Representing One Coordinate Frame in Terms of the Other

- We can extend what we did with the change of basis vectors:

$$\mathbf{u}_1 = \gamma_{11}\mathbf{v}_1 + \gamma_{12}\mathbf{v}_2 + \gamma_{13}\mathbf{v}_3$$
$$\mathbf{u}_2 = \gamma_{21}\mathbf{v}_1 + \gamma_{22}\mathbf{v}_2 + \gamma_{23}\mathbf{v}_3$$
$$\mathbf{u}_3 = \gamma_{31}\mathbf{v}_1 + \gamma_{32}\mathbf{v}_2 + \gamma_{33}\mathbf{v}_3$$
$$\mathbf{Q}_0 = \gamma_{41}\mathbf{v}_1 + \gamma_{42}\mathbf{v}_2 + \gamma_{43}\mathbf{v}_3 + \mathbf{P}_0$$

by replacing the 3×3 matrix $\mathbf{M}$ by a 4×4 matrix as follows:

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

Working with Representations

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

  $\mathbf{a} = [\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4]$ in the first frame

  $\mathbf{b} = [\beta_1 \quad \beta_2 \quad \beta_3 \quad \beta_4]$ in the second frame

  where $a_4 = b_4 = 1$ for points and $a_4 = b_4 = 0$ for vectors and

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

or

$\boldsymbol{b} = \boldsymbol{T}\boldsymbol{a}$

where,

$\boldsymbol{T} = (\boldsymbol{M}^{\boldsymbol{T}})^{-1}$

- The matrix $\mathbf{M}^{\mathrm{T}}$ is 4×4 and specifies an affine transformation in homogeneous coordinates

Transformations in Graphics pipeline

We had considered the following coordinate systems

Can be combined in model-view transform

1. Object (or model) coordinates
2. World coordinates
3. Eye (or camera) coordinates
4. Clip coordinates
5. Normalized device coordinates
6. Window (or screen) coordinates

Affine transform

Brings representations in the eye-frame

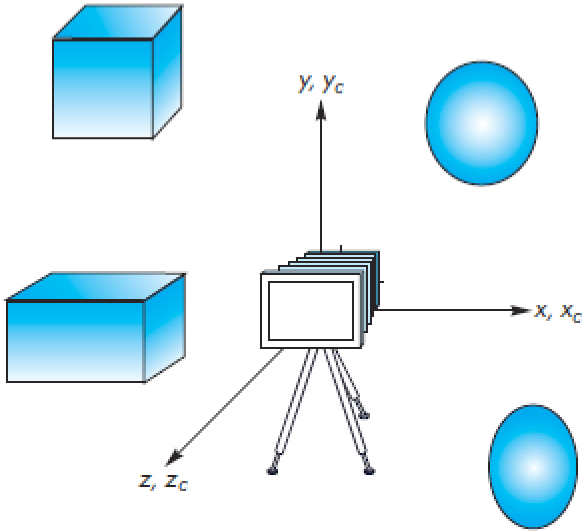
The six frames are w.r.t. immediate-mode rendering

Moving the Camera

Camera and object frame in default positions

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

model-view matrix



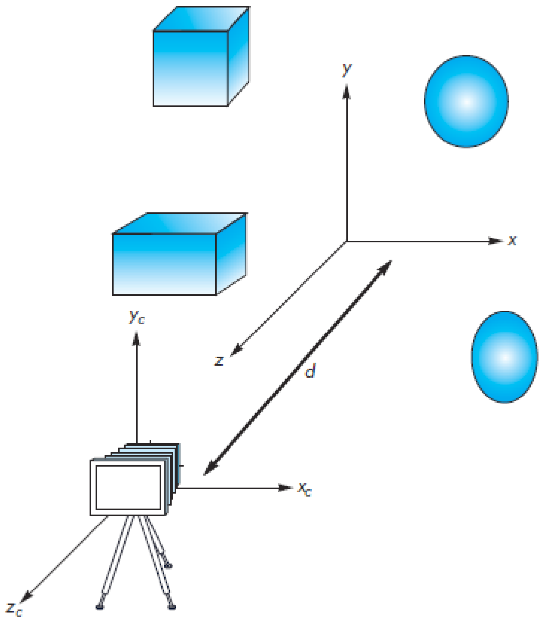
(a)

Moving the Camera

Camera frame is fixed, we are placing object frame relative to the camera frame.

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

Where did we get **A** (model-view matrix) from?



(b)

- The application programmer works in the object/world coordinates (a.k.a. application frame)

Moving the Camera

Camera frame is fixed, we are placing object frame relative to the camera frame.

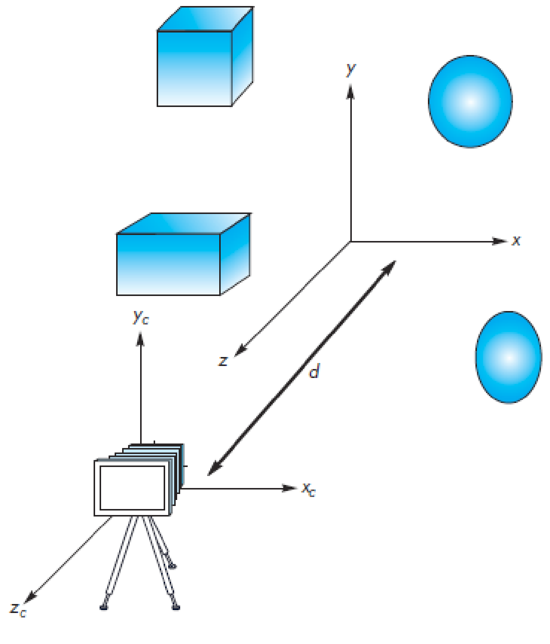
$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Remember?

$$\boldsymbol{b} = \boldsymbol{T}\boldsymbol{a}$$

where,

$$\boldsymbol{T} = (\boldsymbol{M}^{T})^{-1}$$



(b)

- The application programmer works in the object/world coordinates (a.k.a. application frame)

Moving the Camera

Camera frame is fixed, we are placing object frame relative to the camera frame.

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

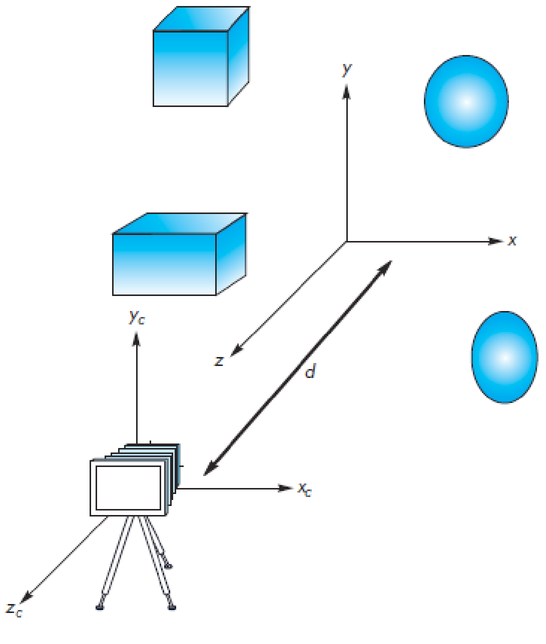
Remember?

$\boldsymbol{b} = \boldsymbol{Ta}$

where,

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

Refer to slide#18



(b)

- The application programmer works in the object/world coordinates (a.k.a. application frame)

Moving the Camera

Camera frame is fixed, we are placing object frame relative to the camera frame.

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

Remember?

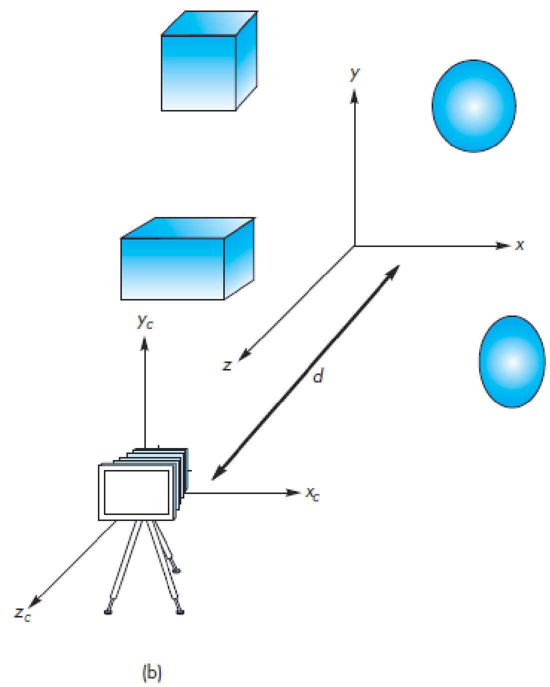
Representation w.r.t camera frame

$$b = Ta$$

where,

$$T = (M^T)^{-1}$$

Representation w.r.t object frame

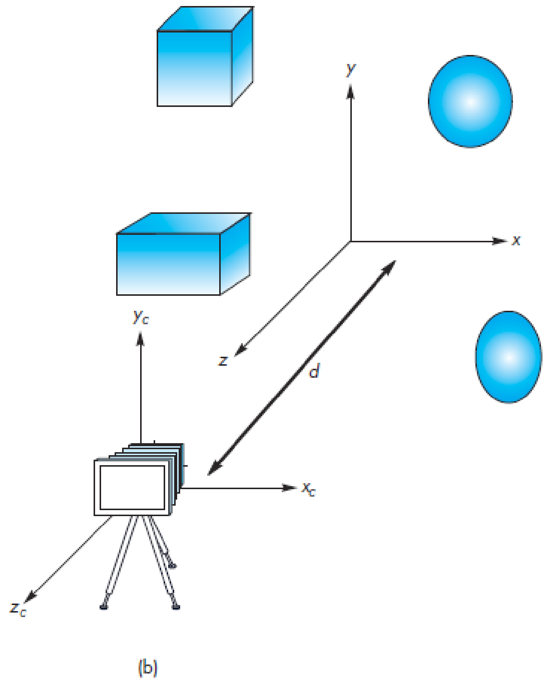


- The application programmer works in the object/world coordinates (a.k.a. application frame)

Moving the Camera

Camera frame is fixed, we are placing object frame relative to the camera frame.

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



Remember?

$\boldsymbol{b} = \boldsymbol{T}\boldsymbol{a}$

where,

$\boldsymbol{T} = (\boldsymbol{M}^{\boldsymbol{T}})^{-1}$

Representation w.r.t camera frame

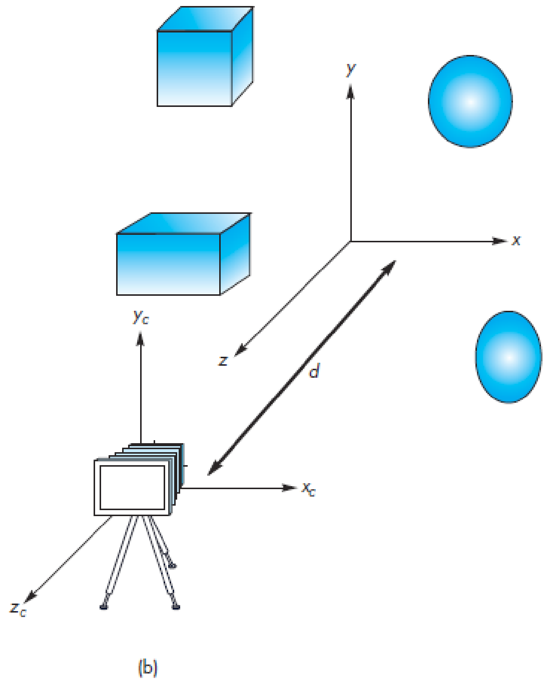
Representation w.r.t object frame

- The application programmer works in the object/world coordinates (a.k.a. application frame)

Moving the Camera

Camera frame is fixed, we are placing object frame relative to the camera frame.

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



Remember?

$b = Ta$

where,

$T = (M^T)^{-1}$

Representation w.r.t camera frame

Representation w.r.t object frame

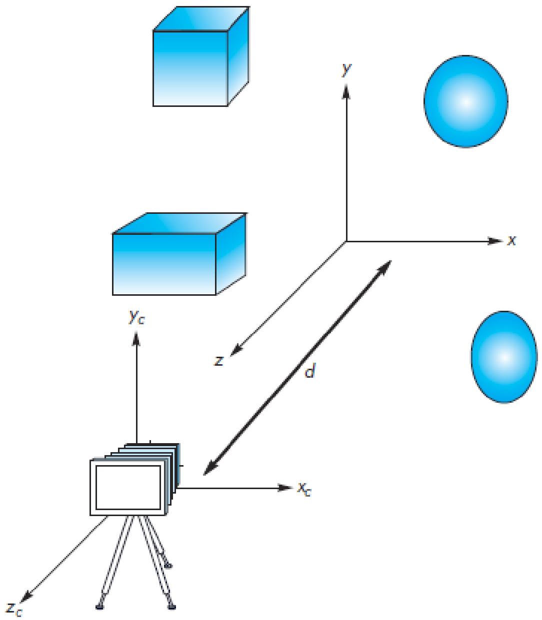
Model View matrix (**A**)

- The application programmer works in the object/world coordinates (a.k.a. application frame)

Moving the Camera

Camera frame is fixed, we are placing object frame relative to the camera frame.

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



(b)

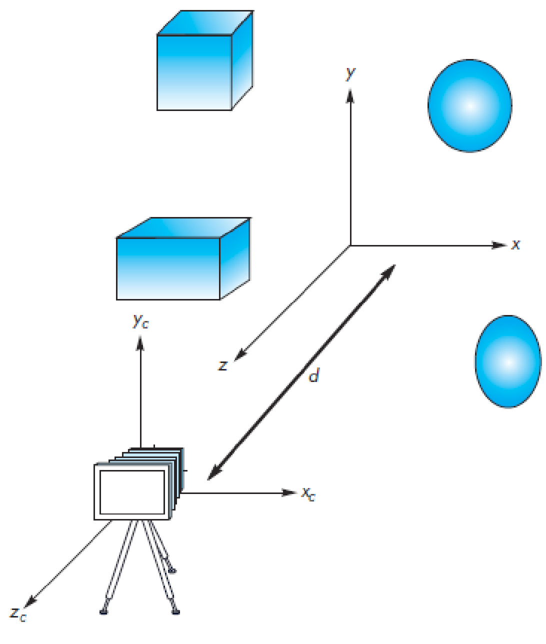
This matrix takes a point (0, 0, d) in the object/world frame, whose representation is:
$\mathbf{p} = [0\ 0\ d\ 1]^T$
to
$\mathbf{p}' = [0\ 0\ 0\ 1]^T$
i.e., the origin in the camera frame

- The application programmer works in the object/world coordinates (a.k.a. application frame)

Moving the Camera

Camera frame is fixed, we are placing object frame relative to the camera frame.

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



(b)

This matrix takes a point (0, 0, d) in the object/world frame, whose representation is:
p=$[0\ 0\ d\ 1]^T$
to
p'= $[0\ 0\ 0\ 1]^T$
i.e., the origin in the camera frame

$$\mathbf{p}' = \mathbf{A}\mathbf{p}$$

- The application programmer works in the object/world coordinates (a.k.a. application frame)

The World and Camera Coordinate Frames

- When we work with representations, we work with *n*-tuples or arrays of scalars
- Changes in coordinate frame are then defined by 4×4 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 (i.e., **M=I**)

An Example

We consider two reference frames that have basis vector relation

$$u_1 = v_1,$$

$$u_2 = v_1 + v_2,$$

$$u_3 = v_1 + v_2 + v_3.$$

Let's say the reference point does not change, so

$$Q_0 = P_0.$$

Our matrix M^T would be:

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \longrightarrow \mathbf{M}^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Only accounting for rotation

An Example

Now, we want our frames to have different reference point….
Let's say, to the point Q_0 that has the following representation in the original system.

$$Q_0 = P_0 + v_1 + 2v_2 + 3v_3,$$

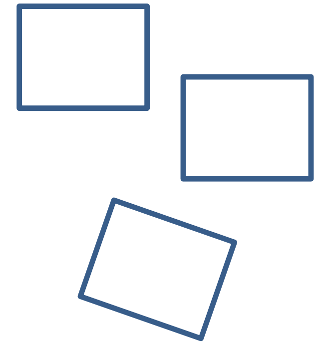
The M^T for such a setting will be:

$$\mathbf{M}^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Also accounting for translation

A Few Common Transformations

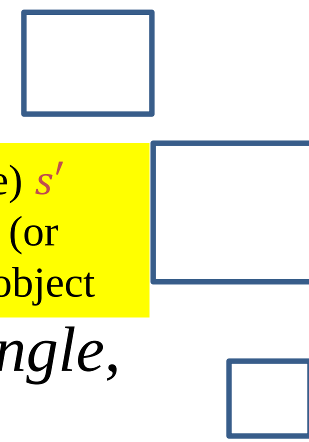
- **Rigid transformation:** The 4×4 matrix has the form:



$$\begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$$

where R is a 3×3 rotation matrix and $\mathbf{t} \in \mathbb{R}^{\mathbf{3x1}}$ is a translation vector. Rigid transformation preserves everything (*angle* (this means the *shape*), *length*, *area*, etc.,)

- **Similarity transformation:** The matrix has the form:



Large (or small) s values enlarge (or diminish) the object

$$\begin{bmatrix} sR & \mathbf{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \text{ or } \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^{\mathrm{T}} & s' \end{bmatrix}$$

Small (or large) s' values enlarge (or diminish) the object

where $s, s' \neq 1$. Similarity transformation preserves *angle*, ratios of *lengths* and of *areas*.

A Few Common Transformations (cont.)

- **Affine transformation:** The 4×4 matrix has the form:

$$\begin{bmatrix} A & \mathbf{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$$

where A can be any 3×3 non-singular matrix and $\mathbf{t} \in \mathbb{R}^{\mathbf{3}}$ is a translation vector. Affine transformation preserves *parallelism*, ratios of lengths.

- **Perspective transformation:** The matrix can be any non-singular 4×4 matrix. Perspective transformation matrix preserves *cross ratios* (i.e., ratio of ratios of lengths).

A Few Common Transformations (cont.)

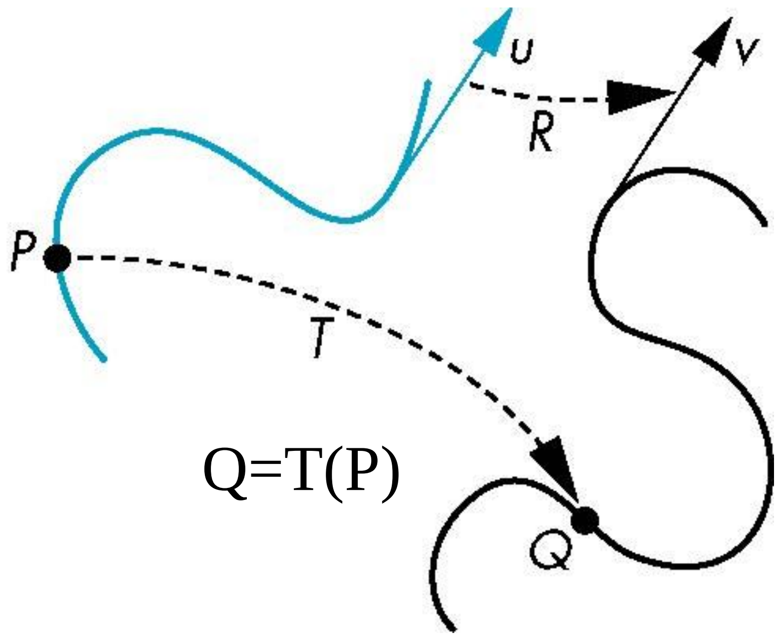
- Rigid transformation is equivalent to a change in coordinate frames. It has 6 degrees of freedom (dof) i.e., 3 rotations + 3 translations (along each of the three axes)

- Similarity transformation has 7 *dof* (an additional scaling)

- Affine transformation has 12 *dof*
  - 3 rotations + 3 translations + 3 scaling + 3 shear

General Transformations

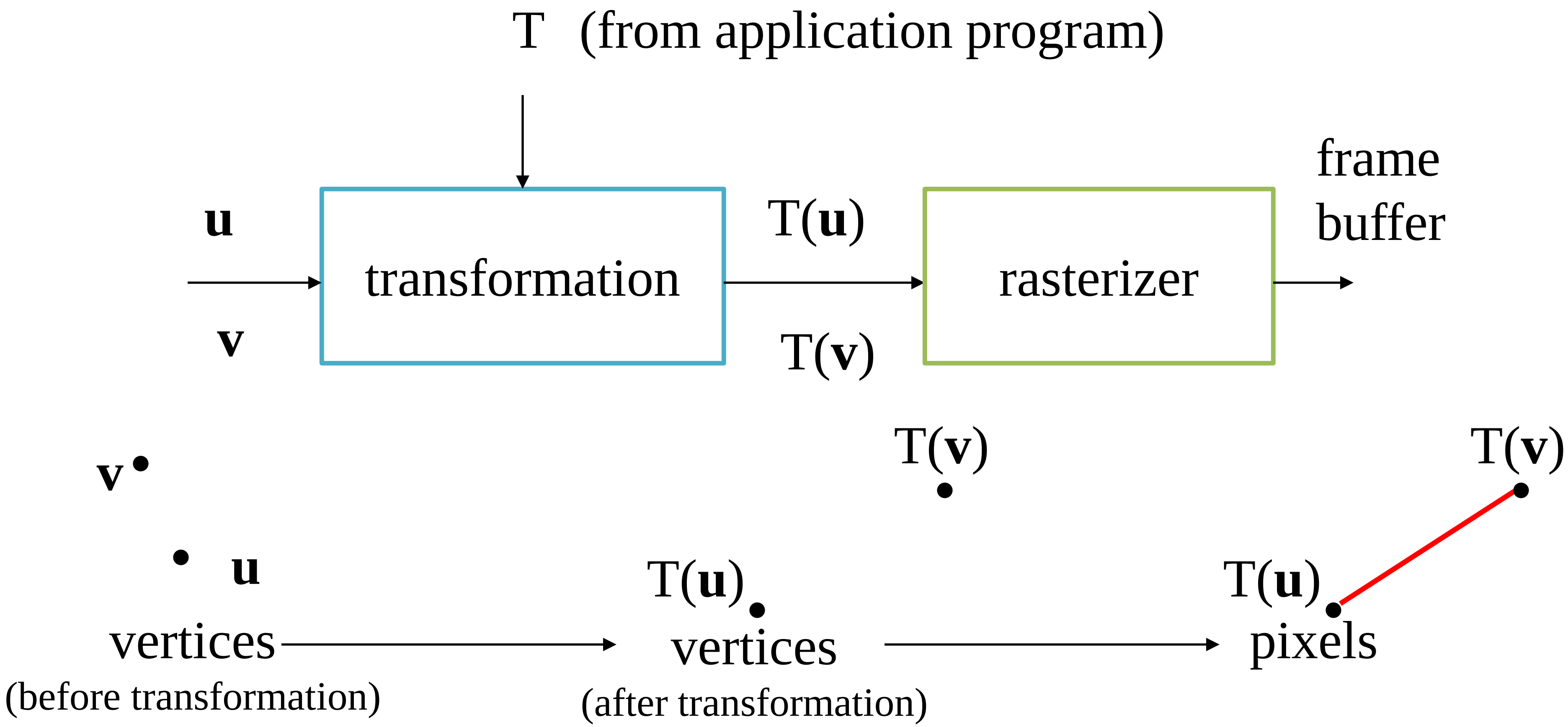
- A transformation maps points to other points and/or vectors to other vectors



$\mathbf{v} = T(\mathbf{u})$

$Q=T(P)$

Pipeline Implementation



Further Reading

“Interactive Computer Graphics – A Top-Down Approach with Shader-Based OpenGL” by Edward Angel and Dave Shreiner, 6th Ed, 2012

- Sec 3.7 to 3.9