Geometric transformations in 3D and coordinate frames

Computer Graphics

CSE 167

Lecture 3

CSE 167: Computer Graphics

- 3D points as vectors
- Geometric transformations in 3D
- Coordinate frames

Representing 3D points using vectors

- 3D point as 3-vector

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

- 3D point using affine homogeneous coordinates as 4-vector

$$\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Geometric transformations

- Translation
- Linear transformations
  - Scale
  - Rotation
- 3D rotations
- Affine transformation
  - Linear transformation followed by translation
- Euclidean transformation
  - Rotation followed by translation
- Composition of transformations
- Transforming normal vectors

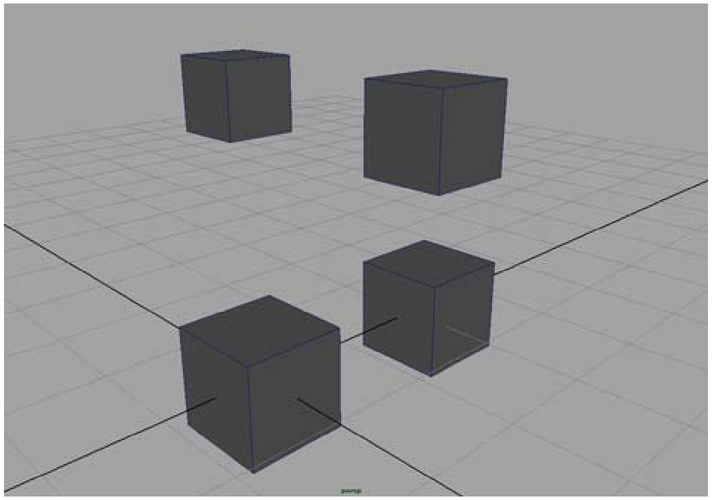
3D translation

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_X \\ t_Y \\ t_Z \end{bmatrix}$$

$$\mathbf{X}' = \mathbf{X} + \mathbf{t}$$

$$\begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

Using homogeneous coordinates



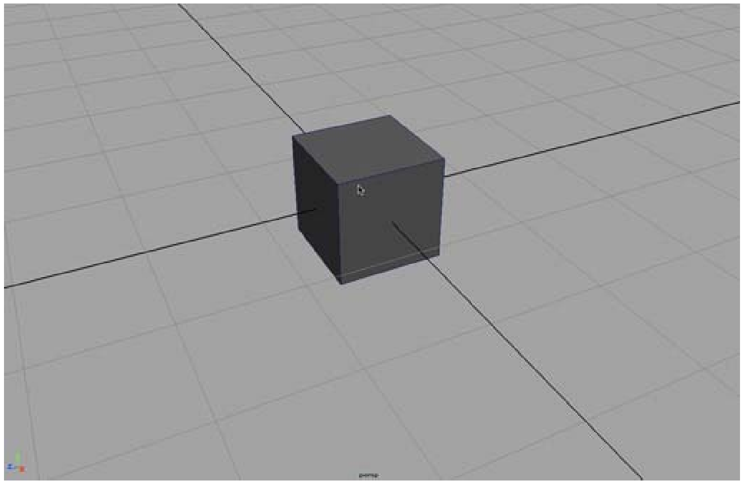
3D nonuniform scale

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} s_X & 0 & 0 \\ 0 & s_Y & 0 \\ 0 & 0 & s_Z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\mathbf{X}' = \mathrm{diag}(s_X, s_Y, s_Z)\mathbf{X}$$

$$\begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{diag}(s_X, s_Y, s_Z) & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

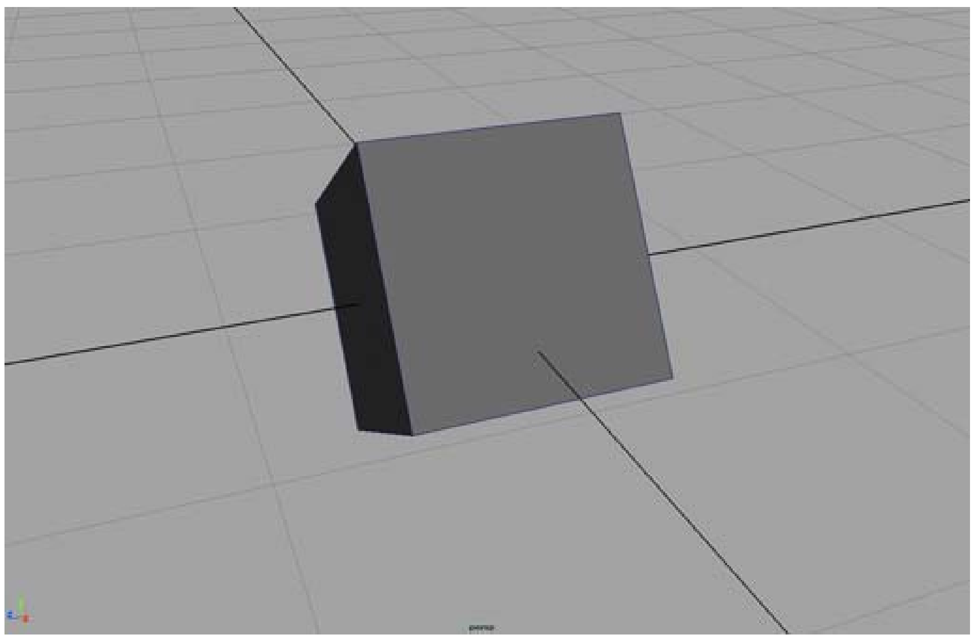
Using homogeneous coordinates



3D rotation about X-axis

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

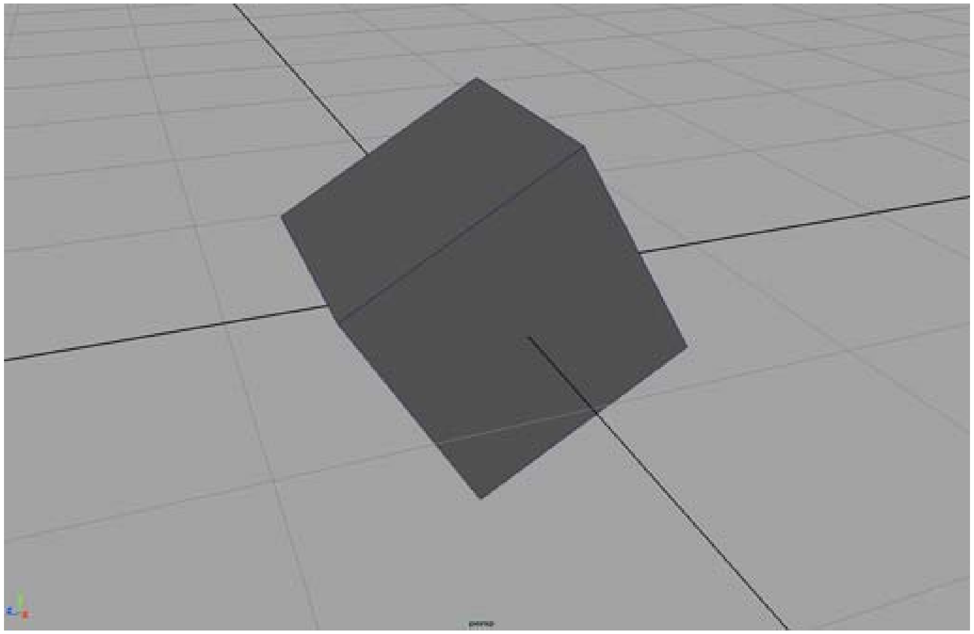
$$\mathbf{X}' = \mathtt{R}_X(\alpha)\mathbf{X}$$



3D rotation about Y-axis

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

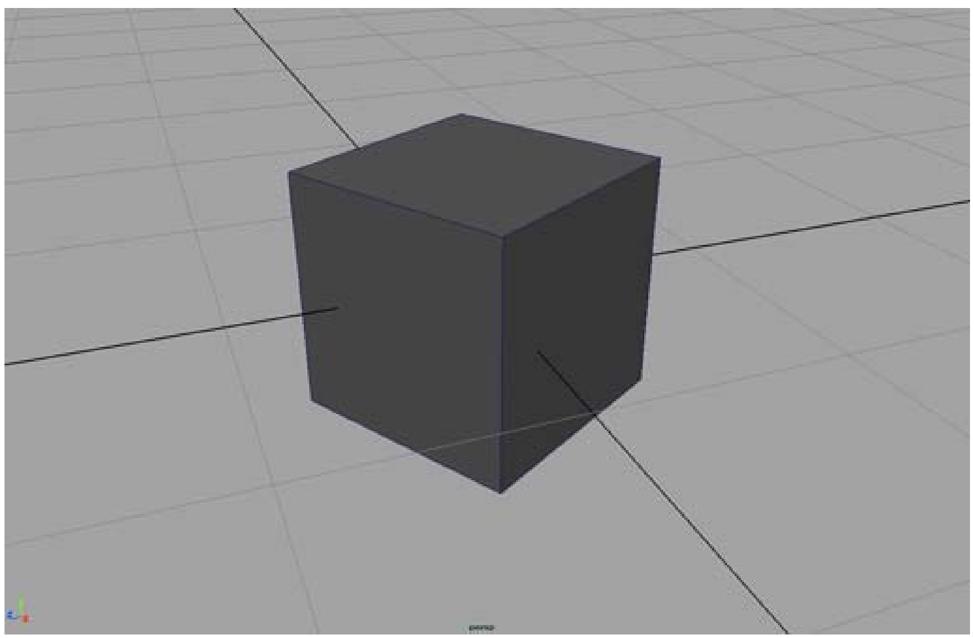
$$\mathbf{X}' = \mathrm{R}_Y(\beta)\mathbf{X}$$



3D rotation about Z-axis

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\mathbf{X}' = \mathtt{R}_Z(\gamma)\mathbf{X}$$



Rotation matrix

- A rotation matrix is a special orthogonal matrix
  - Properties of special orthogonal matrices

$$R^\top R = RR^\top = I$$
$$\det(R) = +1$$

$$R^\top = R^{-1}$$

The inverse of a special orthogonal matrix is also a special orthogonal matrix

- Transformation matrix using homogeneous coordinates

$$\begin{bmatrix} R & 0 \\ 0^\top & 1 \end{bmatrix}$$

3D rotations

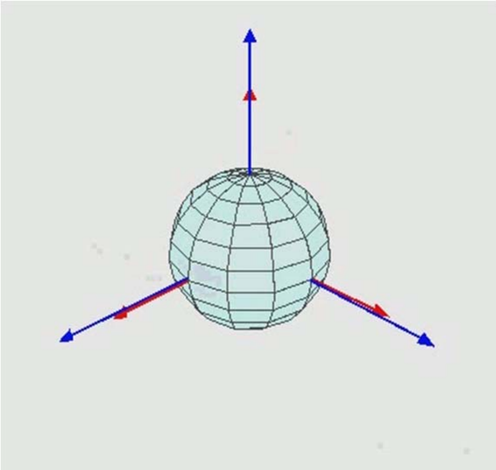
- A 3D rotation can be parameterized with three numbers
- Common 3D rotation formalisms
  - Rotation matrix
    - 3x3 matrix (9 parameters), with 3 degrees of freedom
  - Euler angles
    - 3 parameters
  - Euler axis and angle
    - 4 parameters, axis vector (to scale)
  - Quaternions
    - 4 parameters (to scale)

3D rotation, Euler angles

- A sequence of 3 elemental rotations
- 12 possible sequences

X-Y-X	Y-X-Y	Z-X-Y
X-Y-Z	Y-X-Z	Z-X-Z
X-Z-X	Y-Z-X	Z-Y-X
X-Z-Y	Y-Z-Y	Z-Y-Z

Tait-Bryan angles, also



– Example: Roll-Pitch-Yaw (ZYX convention)

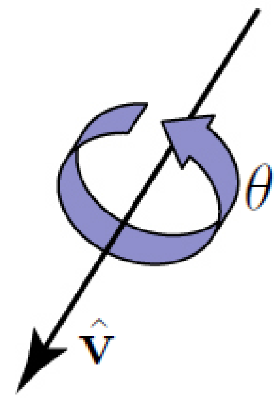
- Rotation about X-axis, followed by rotation about Y-axis, followed by rotation about Z-axis

$R = R_Z(\gamma)R_Y(\beta)R_X(\alpha)$ Composition of rotations

3D rotation, Euler axis and angle

- 3D rotation about an arbitrary axis
  - Axis defined by unit vector
- Corresponding rotation matrix

$$R = \cos(\theta)I + \sin(\theta)[\hat{\mathbf{v}}]_\times + (1 - \cos(\theta))\hat{\mathbf{v}}\hat{\mathbf{v}}^\top$$



Cross product revisited

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}$$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b}$$

$$\text{where } [\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

3D affine transformation

- Linear transformation followed by translation

A is linear transformation matrix

t is translation vector

$$\mathbf{X}' = \mathtt{A}\mathbf{X} + \mathbf{t}$$

$$\begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \mathtt{H}_\mathtt{A} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

Using homogeneous coordinates

$$\text{where } \mathtt{H}_\mathtt{A} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

Notes:

1. Invert an affine transformation using a general 4x4 matrix inverse
2. An inverse affine transformation is also an affine transformation

Affine transformation using homogeneous coordinates

$$\begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

A is linear transformation matrix

- Translation $\begin{bmatrix} \mathtt{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
  – Linear transformation is identity matrix
- Scale $\begin{bmatrix} \mathrm{diag}(s_X, s_Y, s_Z) & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
  – Linear transformation is diagonal matrix
- Rotation $\begin{bmatrix} \mathtt{R} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
  – Linear transformation is special orthogonal matrix

3D Euclidean transformation

- Rotation followed by translation

$$\mathbf{X}' = \mathrm{R}\mathbf{X} + \mathbf{t}$$

$$\begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

Using homogeneous coordinates

$$\begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \mathrm{H}_\mathrm{E} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$\text{where } \mathrm{H}_\mathrm{E} = \begin{bmatrix} \mathrm{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

A Euclidean transformation is an affine transformation where the linear component is a rotation

Inverse Euclidean transformation

Euclidean transformation $\mathbf{X}' = \mathrm{R}\mathbf{X} + \mathbf{t}$

$$\mathbf{X}' - \mathbf{t} = \mathrm{R}\mathbf{X}$$

$$\mathrm{R}^\top(\mathbf{X}' - \mathbf{t}) = \mathbf{X}$$

Inverse Euclidean transformation $\mathrm{R}^\top\mathbf{X}' - \mathrm{R}^\top\mathbf{t} = \mathbf{X}$

$$\begin{bmatrix} \mathrm{R}^\top & -\mathrm{R}^\top\mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

Using homogeneous coordinates

$$\mathrm{H}_\mathrm{E}^{-1} \begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

Use this instead of a general 4x4 matrix inverse

$$\text{where } \mathrm{H}_\mathrm{E}^{-1} = \begin{bmatrix} \mathrm{R}^\top & -\mathrm{R}^\top\mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

An inverse Euclidean transformation is also a Euclidean transformation

Composition of transformations

- Compose geometric transformation by multiplying 4x4 transformation matrices

Composition of two transformations

$$\begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \mathtt{H}_1 \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{X}'' \\ 1 \end{bmatrix} = \mathtt{H}_2 \begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{X}'' \\ 1 \end{bmatrix} = \mathtt{H}_2\mathtt{H}_1 \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

Composition of *n* transformations

$$\begin{bmatrix} \mathbf{X}^{(n)} \\ 1 \end{bmatrix} = \mathtt{H}_n\mathtt{H}_{n-1} \cdots \mathtt{H}_2\mathtt{H}_1 \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

Order of matrices is important!
Matrix multiplication is **not** (in general) commutative

Transforming normal vectors

- Tangent vector **t** at surface point **X** is orthogonal to normal vector **n** at **X**

$$\mathbf{t}^\top \mathbf{n} = \mathbf{n}^\top \mathbf{t} = 0$$

- Transformed tangent vector and transformed normal vector must also be orthogonal

$$\mathbf{t}'^\top \mathbf{n}' = \mathbf{n}'^\top \mathbf{t}' = 0$$

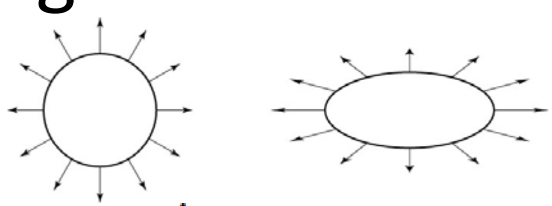
Transforming normal vectors

- Tangent vector can be thought of as a difference of points, so it transforms the same as a surface point

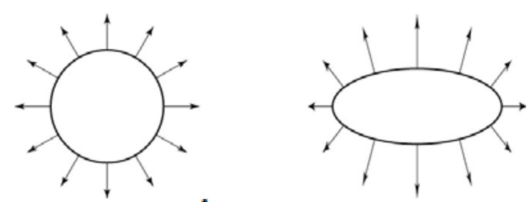
$$\mathbf{t}' = \mathbf{A}\mathbf{t}$$

We are only concerned about direction of vectors, so do not add translation vector

- Normal vector does not transform the same as tangent vector



$$\mathbf{n}' \neq \mathbf{A}\mathbf{n}$$



$$\mathbf{n}' = \mathbf{M}\mathbf{n}$$

How is **M** related to **A**?

Transforming normal vectors

- How is **M** related to **A**?

$$\mathbf{t}'^{\top}\mathbf{n}' = 0$$

$$(\mathtt{A}\mathbf{t})^{\top}\mathtt{M}\mathbf{n} = 0$$

$$\mathbf{t}^{\top}\mathtt{A}^{\top}\mathtt{M}\mathbf{n} = 0$$

$$\mathbf{t}^{\top}\mathbf{n} = 0 \text{ if } \mathtt{A}^{\top}\mathtt{M} = \mathtt{I}$$

- Solve for **M**

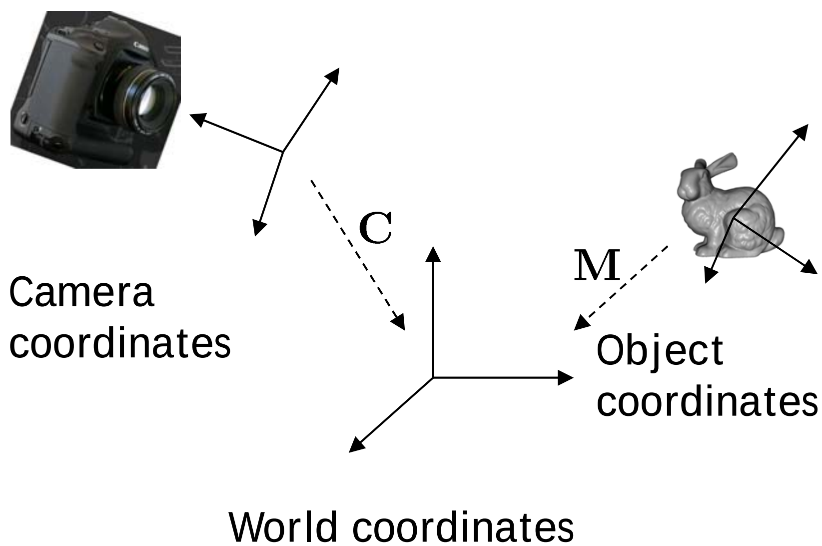
$$\mathtt{M} = (\mathtt{A}^{\top})^{-1} = (\mathtt{A}^{-1})^{\top} = \mathtt{A}^{-\top}$$

- Transform normal vectors using

$$\mathbf{n}' = \mathtt{A}^{-\top}\mathbf{n}$$

Coordinate frames

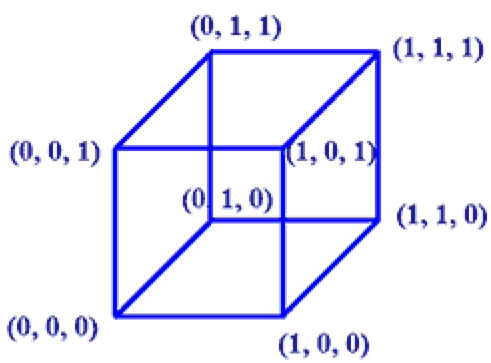
- In computer graphics, we typically use at least three coordinate frames
  - Object coordinate frame
  - World coordinate frame
  - Camera coordinate frame



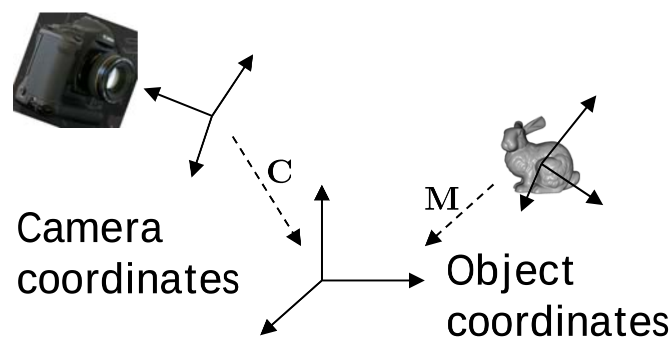
Based on slides courtesy of Jurgen Schulze

Object coordinates

- Local coordinates in which points and other object geometry are given
- Often origin is in geometric center, on the base, or in a corner of the object
  - Depends on how object is generated or used

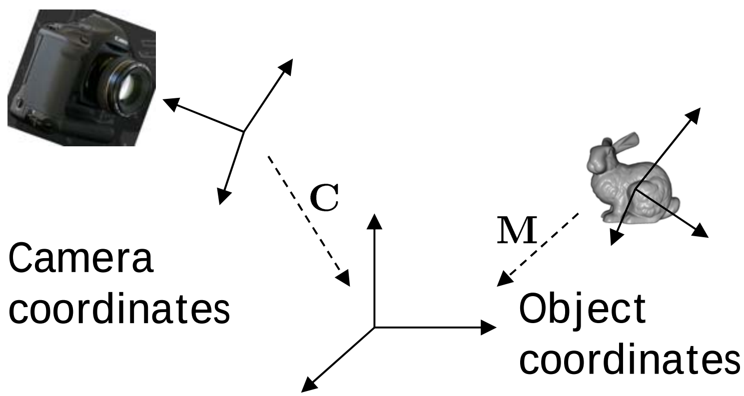


Source: http://motivate.maths.org



World coordinates

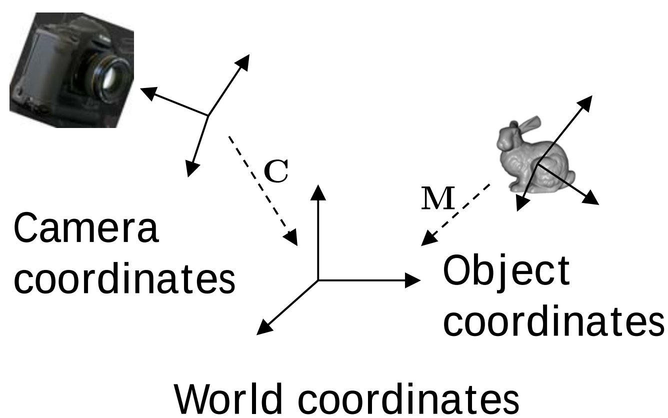
- Common reference frame for all objects in the scene
- No standard for coordinate frame orientation
  – If there is a ground plane, usually X-Y plane is horizontal and positive Z is up
  – Otherwise, X-Y plane is often screen plane and positive Z is out of the screen



World coordinates

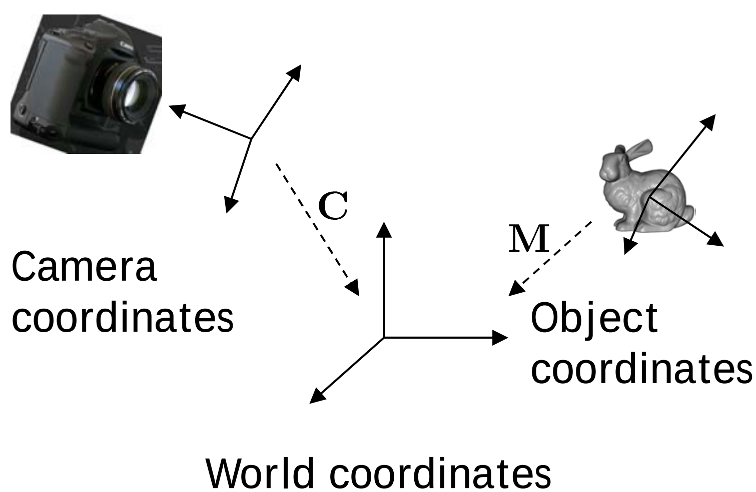
Object transformation

- The transformation from object coordinates to world coordinates is different for each object
- Defines placement of object in scene
- Given by “model matrix” (model-to-world transformation) **M**



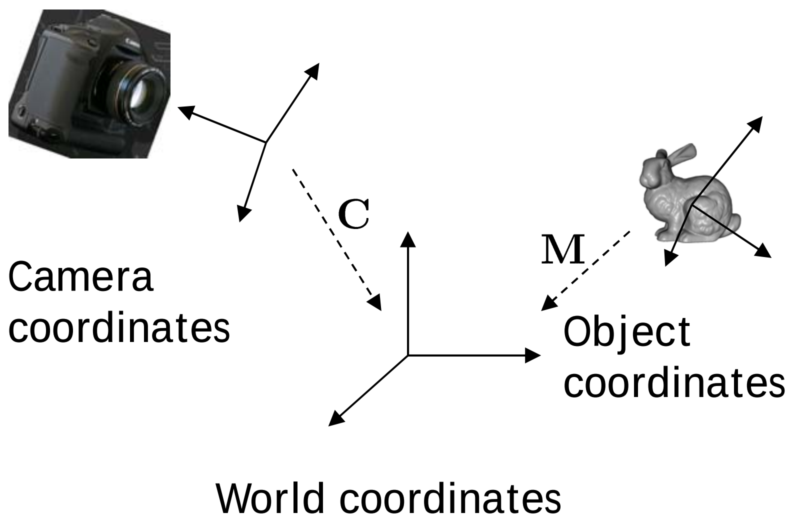
Camera coordinates

- Origin defines center of projection of camera
- X-Y plane is parallel to image plane
- Z-axis is orthogonal to image plane



Camera coordinates

- The “camera matrix” defines the transformation from camera coordinates to world coordinates
  - – Placement of camera in world



Camera matrix

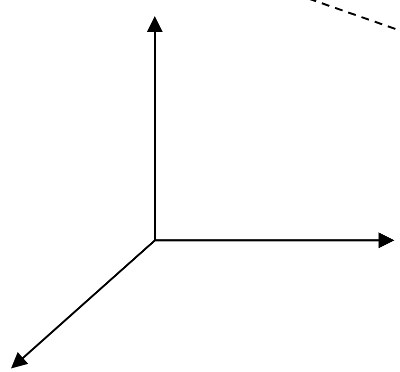
u

e

Camera coordinates

- Given:
  - Center point of projection **e**
  - Look at point **d**
  - Camera up vector **u**

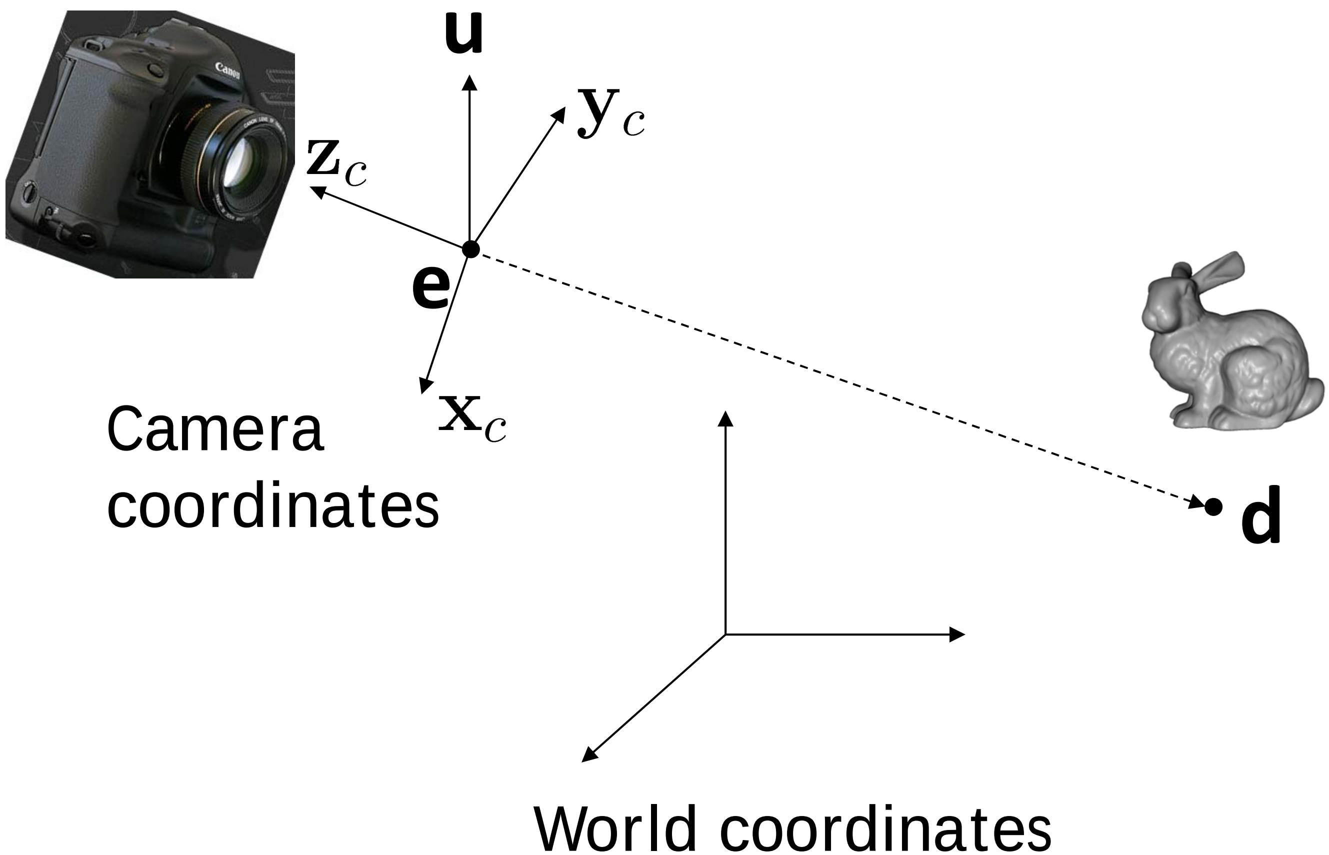
d



World coordinates

Camera matrix

- Construct $\mathbf{x_c}$, $\mathbf{y_c}$, $\mathbf{z_c}$

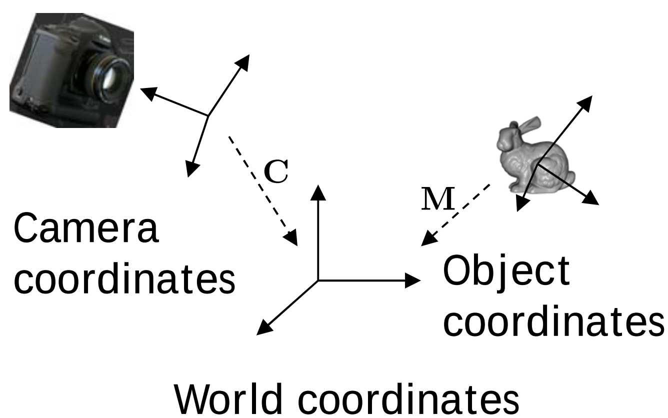


Camera matrix

- Step 1: Z-axis $$\boldsymbol{z}_C = \frac{\boldsymbol{e} - \boldsymbol{d}}{\|\boldsymbol{e} - \boldsymbol{d}\|}$$
- Step 2: X-axis $$\boldsymbol{x}_C = \frac{\boldsymbol{u} \times \boldsymbol{z}_C}{\|\boldsymbol{u} \times \boldsymbol{z}_C\|}$$
- Step 3: Y-axis $$\boldsymbol{y}_C = \boldsymbol{z}_C \times \boldsymbol{x}_C = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$$
- Camera Matrix: $$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{x}_C & \boldsymbol{y}_C & \boldsymbol{z}_C & \boldsymbol{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transforming object coordinates to camera coordinates

- Object to world coordinates: **M**
- Camera to world coordinates: **C**
- Point to transform: **p**
- Resulting transformation equation **p'** = $\mathbf{C}^{-1}$ **M p**



Use inverse of Euclidean transformation (slide 17) instead of a general 4x4 matrix inverse

Tips for notation

- Indicate coordinate systems with every point or matrix
  - Point: $\mathbf{p}_{\text{object}}$
  - Matrix: $\mathbf{M}_{\text{object}\rightarrow\text{world}}$
- Resulting transformation equation:

$$\mathbf{p}_{\text{camera}} = (\mathbf{C}_{\text{camera}\rightarrow\text{world}})^{-1}\, \mathbf{M}_{\text{object}\rightarrow\text{world}}\, \mathbf{p}_{\text{object}}$$

- In source code use similar names:
  - Point: `p_object` or `p_obj` or `p_o`
  - Matrix: `object2world` or `obj2wld` or `o2w`
- Resulting transformation equation:

```
wld2cam = inverse(cam2wld);
p_cam = p_obj * obj2wld * wld2cam;
```

Objects in camera coordinates

- We have things lined up the way we like them on screen
  - The positive X-axis points to the right
  - The positive Y-axis points up
  - The negative Z-axis points into the screen (positive Z-axis points out of the screen)
  - Objects to look at are in front of us, i.e., have negative Z values
- But objects are still in 3D
- Next step: project scene to 2D plane