CSE 167: Introduction to Computer Graphics
Lecture #4: Coordinate Systems

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2020
Announcements

- Homework Project 1 due October 25
- Discussion Project 1: Wednesday 1pm
Coordinate System

- Given point \( p \) in homogeneous coordinates:

\[
\begin{bmatrix}
px \\
py \\
pz \\
1
\end{bmatrix}
\]

- Coordinates describe the point's 3D position in a coordinate system with basis vectors \( x, y, z \) and origin \( o \):

\[
p_{xyz} = px\cdot x + py\cdot y + pz\cdot z + o
\]
Rectangular and Polar Coordinates

Point \( p \) can be located relative to the origin by Rectangular Coordinates \((X_p, Y_p)\) or by Polar Coordinates \((r, \theta)\):

\[
X_p = r \cos(\theta) \\
Y_p = r \sin(\theta) \\
r = \sqrt{X_p^2 + Y_p^2} \\
\theta = \tan^{-1}(Y_p / X_p)
\]
Coordinate System Orientation

- Right-handed and left-handed coordinate systems

Unity

OpenCV

Windows Mixed Reality

OpenGL
Coordinate Transformation

Goal: Find coordinates of $p_{xyz}$ in new $uvwq$ coordinate system
Coordinate Transformation

Express coordinates of \( \text{xyz} \) reference frame with respect to \( \text{uvwq} \) reference frame:

- \( \mathbf{x} = \begin{bmatrix} x_u \\ x_v \\ x_w \\ 0 \end{bmatrix} \)
- \( \mathbf{y} = \begin{bmatrix} y_u \\ y_v \\ y_w \\ 0 \end{bmatrix} \)
- \( \mathbf{z} = \begin{bmatrix} z_u \\ z_v \\ z_w \\ 0 \end{bmatrix} \)
- \( \mathbf{o} = \begin{bmatrix} o_u \\ o_v \\ o_w \\ 1 \end{bmatrix} \)
Coordinate Transformation

Point \( p \) expressed in new \( uvwq \) reference frame:

\[
\mathbf{p}_{uvw} = \mathbf{p}_x \begin{bmatrix} x_u \\ x_v \\ x_w \\ 0 \end{bmatrix} + \mathbf{p}_y \begin{bmatrix} y_u \\ y_v \\ y_w \\ 0 \end{bmatrix} + \mathbf{p}_z \begin{bmatrix} z_u \\ z_v \\ z_w \\ 0 \end{bmatrix} + \begin{bmatrix} o_u \\ o_v \\ o_w \\ 1 \end{bmatrix}
\]
Coordinate Transformation

\[
\mathbf{p}_{uvw} = \begin{bmatrix}
x_u & y_u & z_u & o_u \\
x_v & y_v & z_v & o_v \\
x_w & y_w & z_w & o_w \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
px \\
py \\
pz \\
1
\end{bmatrix}
= \begin{bmatrix}
x & y & z & o \\
px \\
py \\
pz \\
1
\end{bmatrix}
\]
Inverse transformation

- Given point $P_{uvw}$ w.r.t. reference frame $uvwq$:
  - Coordinates $P_{xyz}$ w.r.t. reference frame $xyzo$ are calculated as:

\[
p_{xyz} = \begin{bmatrix}
x_u & y_u & z_u & o_u \\
x_v & y_v & z_v & o_v \\
x_w & y_w & z_w & o_w \\
0 & 0 & 0 & 1
\end{bmatrix}^{-1} \begin{bmatrix}
p_u \\
p_v \\
p_w \\
1
\end{bmatrix}
\]
Lecture Overview

- Coordinate Transformation
- Typical Coordinate Systems
Typical Coordinate Systems

- In computer graphics, we typically use at least three coordinate systems:
  - World coordinate system
  - Camera coordinate system
  - Object coordinate system
World Coordinates

- Common reference frame for all objects in the scene
- No standard for coordinate system orientation
  - If there is a ground plane, usually x/y is horizontal and z points up (height)
  - Otherwise, x/y is often screen plane, z points out of the screen
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
Object Transformation

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

![Diagram showing transformation between camera, object, and world coordinates](image)
Camera Coordinate System

- Origin defines center of projection of camera
- x-y plane is parallel to image plane
- z-axis is perpendicular to image plane
Camera Coordinate System

- The Camera Matrix defines the transformation from camera to world coordinates
- Placement of camera in world
Camera Matrix

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

- **Diagram:**
  - Camera coordinates
  - World coordinates

- **Images:**
  - Camera and rabbit model
Camera Matrix

- Construct $\mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c$

Camera coordinates

World coordinates
Camera Matrix

- **Step 1: z-axis**
  \[ z_c = \frac{e - d}{\|e - d\|} \]

- **Step 2: x-axis**
  \[ x_c = \frac{u \times z_c}{\|u \times z_c\|} \]

- **Step 3: y-axis**
  \[ y_c = z_c \times x_c = \frac{u}{\|u\|} \]

- **Camera Matrix:**
  \[ c = \begin{bmatrix} x_c & y_c & z_c & e \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
Transforming Object to Camera Coordinates

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

- Indicate coordinate systems with every point or matrix
  - **Point:** $p_{\text{object}}$
  - **Matrix:** $M_{\text{object} \rightarrow \text{world}}$

- Resulting transformation equation:
  $$p_{\text{camera}} = (C_{\text{camera} \rightarrow \text{world}})^{-1} M_{\text{object} \rightarrow \text{world}} p_{\text{object}}$$

- In C++ code use similar names:
  - **Point:** $p_{\text{object}}$ or $p_{\text{obj}}$ or $p_{o}$
  - **Matrix:** $\text{object2world}$ or $\text{obj2wld}$ or $o2w$

- Resulting transformation equation:
  $$\text{wld2cam} = \text{inverse}(\text{cam2wld});$$
  $$p_{\text{cam}} = p_{\text{obj}} \times \text{obj2wld} \times \text{wld2cam};$$
Inverse of Camera Matrix

- How to calculate the inverse of camera matrix $C^{-1}$?
- Generic matrix inversion is complex and compute-intensive!
- Solution: affine transformation matrices can be inverted more easily
- Observation:
  - Camera matrix consists of translation and rotation: $T \times R$
  - Inverse of rotation: $R^{-1} = R^T$
  - Inverse of translation: $T(t)^{-1} = T(-t)$
  - Inverse of camera matrix: $C^{-1} = R^{-1} \times T^{-1}$
Objects in Camera Coordinates

- We have things lined up the way we like them on screen
  - x points to the right
  - y points up
  - -z into the screen (i.e., z 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