

Changes of Coordinate Systems

- Coordinate Systems
- Mapping between Coordinate Systems
- Computing the Change of Basis Matrix
- Numerical Example
- Coordinate System for a Virtual Camera

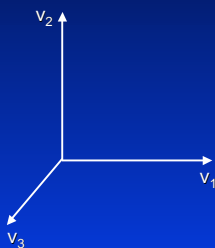
Copyright © Manuel M. Oliveira

Coordinate Systems

- N linearly independent vectors define a basis for \mathbb{R}^n
 - a set of vectors $\{v_1, v_2, \dots, v_k\}$ is LI if $c_1 v_1 + c_2 v_2 + \dots + c_n v_n = 0$ then $c_1 = c_2 = \dots = c_n = 0$
- It is a good idea to select orthogonal vectors
 - v_1 and v_2 are orthogonal if $v_1 \cdot v_2 = 0$
- It is even a better idea to use an orthonormal basis

Copyright © Manuel M. Oliveira

Vectors in \mathbb{R}^3

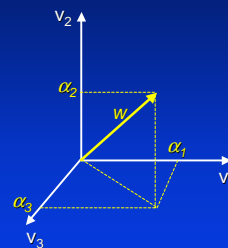


The canonical basis for \mathbb{R}^3 :

$$\begin{aligned} v_1 &= (1, 0, 0) \\ v_2 &= (0, 1, 0) \\ v_3 &= (0, 0, 1) \end{aligned}$$

Copyright © Manuel M. Oliveira

Vectors in \mathbb{R}^3



$$w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$$

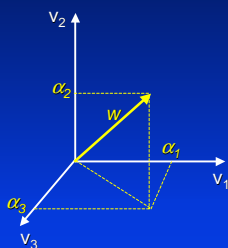
α_i is length of the projection of w onto v_i : $\alpha_i = w \cdot v_i$

$$w = a^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$a = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

Copyright © Manuel M. Oliveira

Vectors in \mathbb{R}^3



Since

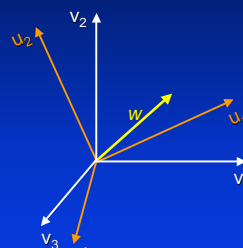
$$V = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = I_{3 \times 3}$$

we can write w simply as

$$w = a = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

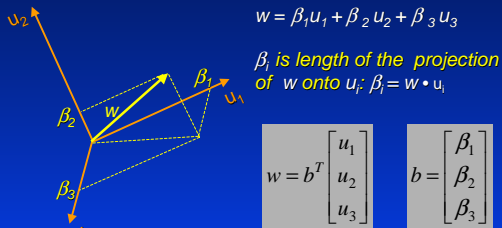
Copyright © Manuel M. Oliveira

Vectors in \mathbb{R}^3



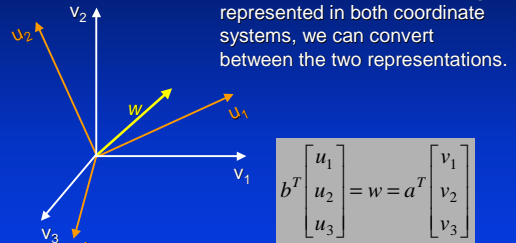
Copyright © Manuel M. Oliveira

Vectors in \mathbb{R}^3



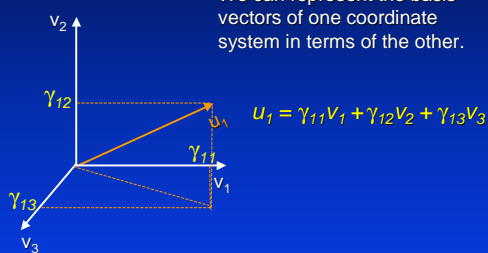
Copyright © Manuel M. Oliveira

Changing Coordinate Systems



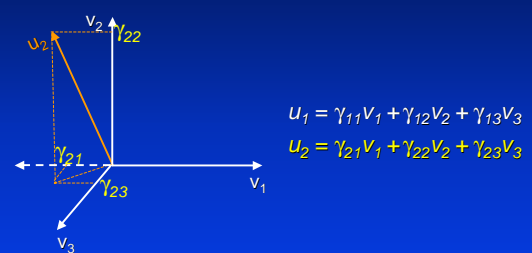
Copyright © Manuel M. Oliveira

How do we do this?



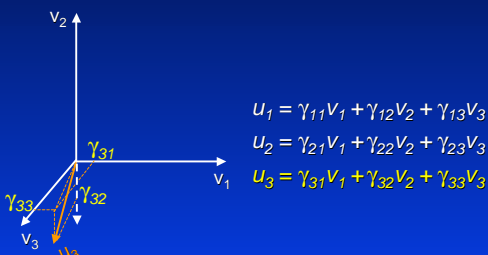
Copyright © Manuel M. Oliveira

How do we do this?



Copyright © Manuel M. Oliveira

How do we do this?



Copyright © Manuel M. Oliveira

Change of Basis Matrix

$$\underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}}_{U_{3 \times 3}} = \underbrace{\begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}}_{I_{3 \times 3}}$$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = B \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = B$$

Copyright © Manuel M. Oliveira

Change of Basis Matrix

Recall that since

$$b^T \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = w = a^T$$

and

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = B$$

We have

$$b^T B = a^T$$



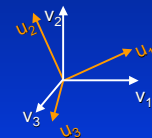
$$b^T = a^T B^{-1} \\ b = (B^{-1})^T a$$

Copyright © Manuel M. Oliveira

Change of Basis Matrix

B (or **U**) is a rotation matrix that maps the canonical basis **V** onto an arbitrary **U** basis. As a rotation matrix, **B** is orthogonal. Therefore $B^{-1} = B^T$ and $b = Ba$

$$B = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = B \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$



Copyright © Manuel M. Oliveira

Orthogonal Matrix

Square matrix with orthonormal rows and columns

$$q_i^T q_j = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$

$$Q^T Q = \begin{bmatrix} - & q_1^T & - & - \\ - & q_2^T & - & - \\ . & . & . & . \\ - & q_n^T & - & - \end{bmatrix} \begin{bmatrix} | & | & . & | \\ q_1 & q_2 & . & q_n \\ | & | & . & | \end{bmatrix} = \begin{bmatrix} 1 & 0 & . & 0 \\ 0 & 1 & . & 0 \\ . & . & . & . \\ 0 & 0 & . & 1 \end{bmatrix}$$

$$Q^T Q = I \rightarrow Q^T = Q^{-1}$$

Copyright © Manuel M. Oliveira

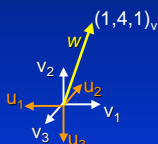
The B Matrix

- B** maps coordinates defined in the canonical **V** basis (WCS) into an arbitrary **U** basis (CCS)!
- This will be very useful for scene visualization
- The **rows** of **B** correspond to the components of the vectors in 3D that represent the axes of the CCS

$$B = \begin{bmatrix} u_1 \cdot v_1 & u_1 \cdot v_2 & u_1 \cdot v_3 \\ u_2 \cdot v_1 & u_2 \cdot v_2 & u_2 \cdot v_3 \\ u_3 \cdot v_1 & u_3 \cdot v_2 & u_3 \cdot v_3 \end{bmatrix} = \begin{bmatrix} u_{1x} & u_{1y} & u_{1z} \\ u_{2x} & u_{2y} & u_{2z} \\ u_{3x} & u_{3y} & u_{3z} \end{bmatrix}$$

Copyright © Manuel M. Oliveira

Numerical Example



Computing B

$$B = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Coordinates of w in the U basis

$$w_u = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -4 \end{bmatrix}$$

Copyright © Manuel M. Oliveira

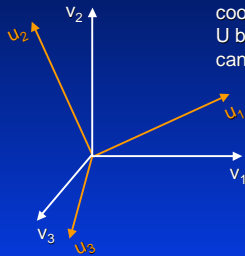
Changing Coordinate Systems

Second Round

Making Things Ready for Graphics Applications

Copyright © Manuel M. Oliveira

Changing Coordinate Systems

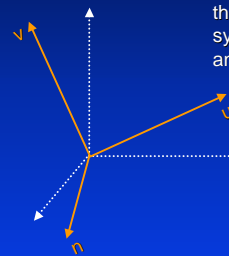


Since we are expressing the coordinates of the vectors in the U basis with respect to the canonical basis for \mathbb{R}^3 ...

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

Copyright © Manuel M. Oliveira

Changing Coordinate Systems

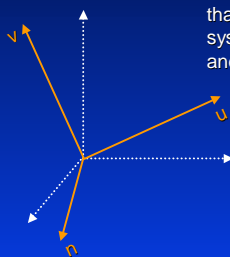


... we can rename the vectors that form the desired coordinate system (for easy of reference) and rewrite them simply as

$$\begin{aligned} u &= (u_x, u_y, u_z) \\ v &= (v_x, v_y, v_z) \\ n &= (n_x, n_y, n_z) \end{aligned}$$

Copyright © Manuel M. Oliveira

Changing Coordinate Systems



... we can rename the vectors that form the desired coordinate system (for easy of reference) and rewrite them simply as

$$B = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{bmatrix}$$

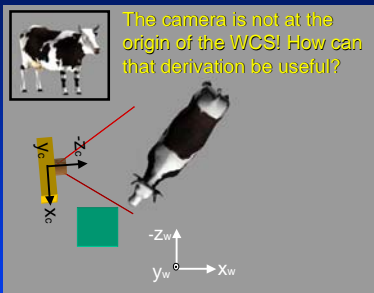
Copyright © Manuel M. Oliveira

B in Homogeneous Coordinates

$$B = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Copyright © Manuel M. Oliveira

What Is This Good for?



The camera is not at the origin of the WCS! How can that derivation be useful?

Copyright © Manuel M. Oliveira

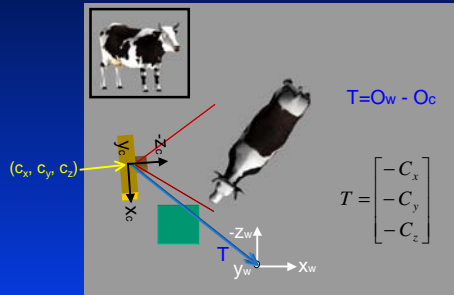
Modeling the Visualization Problem

Interpretation I

Translation of the camera to the origin followed by the actual change of coordinate system

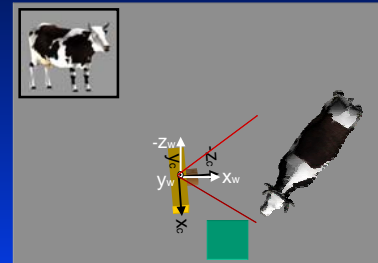
Copyright © Manuel M. Oliveira

Translating the world (before)



Copyright © Manuel M. Oliveira

Translating the world (after)



Copyright © Manuel M. Oliveira

Coordinates in the Camera CS

$$p_{ccs} = B(T p_{wcs})$$

- Back in homogeneous coordinates

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 1 & -C_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Copyright © Manuel M. Oliveira

M=BT: Model/View Matrix

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & -O_c \cdot u \\ v_x & v_y & v_z & -O_c \cdot v \\ n_x & n_y & n_z & -O_c \cdot n \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

- Where $O_c = (c_x, c_y, c_z)$

Copyright © Manuel M. Oliveira

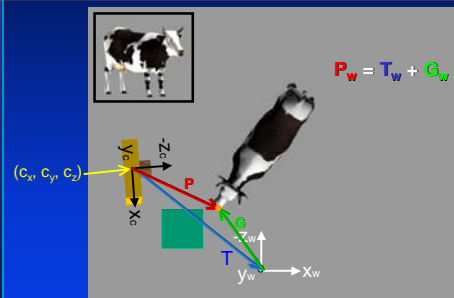
Modeling the Visualization Problem

Interpretation II

Computing a vector from the point in 3D to the camera origin and projecting it into the camera axes

Copyright © Manuel M. Oliveira

Projecting Coordinates in the CCS



Copyright © Manuel M. Oliveira

Coordinates in the Camera CS

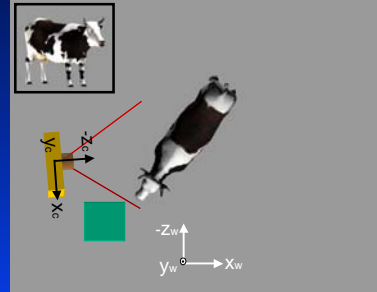
$$p_{ccs} = B(T_u + G_w)$$

- Back in homogeneous coordinates

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -c_x + x_w \\ -c_y + y_w \\ -c_z + z_w \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & -O_c \cdot u \\ v_x & v_y & v_z & -O_c \cdot v \\ n_x & n_y & n_z & -O_c \cdot n \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Copyright © Manuel M. Oliveira

How do we specify the CCS?



Copyright © Manuel M. Oliveira

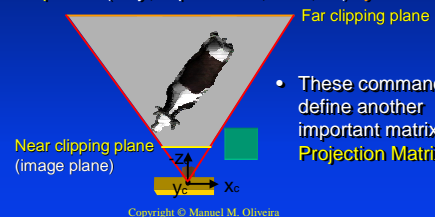
gluLookAt

- `gluLookAt(eye_x, eye_y, eye_z, look_at_x, look_at_y, look_at_z, up_x, up_y, up_z)`
- These parameters provide all the information needed to create the **Model View Matrix**.
- $n = \text{normalize}(\text{eye} - \text{look_at})$ (right hand coord system)
- $u = \text{up} \times n$
- $v = n \times u$
- $O_c = (\text{eye_x}, \text{eye_y}, \text{eye_z})$

Copyright © Manuel M. Oliveira

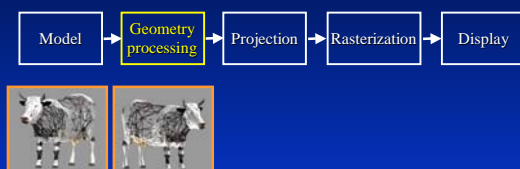
View Frustum in OpenGL

- `glFrustum(left, right, bottom, top, near, far)`
 - The frustum does not need to be symmetrical
- `gluPerspective(fovy, aspect ratio, near, far)` **symmetrical**
- These commands define another important matrix: the **Projection Matrix**.



Copyright © Manuel M. Oliveira

Where Are We in the Pipeline?



- Map from WCS to CCS
- Culling
- Lighting

Copyright © Manuel M. Oliveira