Univers Virtuels

Transformations

Alexis NEDELEC

Centre Européen de Réalité Virtuelle
Ecole Nationale d'Ingénieurs de Brest

ÉCOLE NATIONALE D'INGÉNIEURS DE BREST

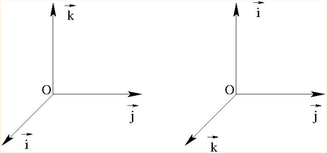
Introduction

Transformations

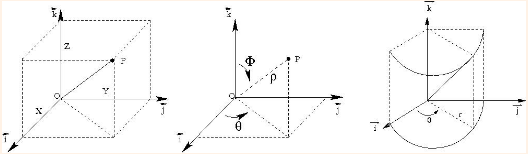
- systèmes de coordonnées
- matrices homogènes (transformations affines)
- repères de caméra
- projections
- déformations

Systèmes de coordonnées

Trièdres directs/indirects



Coordonnées cartésiennes, sphériques, cylindriques



Matrices homogènes

Coordonnées homogènes

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \\ C_{41} & C_{42} & C_{43} & C_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Transposés de vecteurs et produit matriciel

- $[P]$: point de l'espace, $[P]^T$ transposée de $[P]$
- $[M]$: matrice homogène, $[M]^T$: transposée de $[M]$

$[P']$: résultat des transformations successives i $(1 \leq i \leq n)$,

- $[P'] \ = \ [M_n] \ldots [M_i] \ldots [M_1][P]$
- $[P'] \ = \ [P]^T[M_1]^T \ldots [M_i]^T \ldots [M_n]^T$

Transformation de repères

Repères Usuels

- **WCS** : **W**orld **C**oordinate **S**ystem
- **OCS** : **O**bject **C**oordinate **S**ystem
- **CCS** : **C**amera **C**oordinate **S**ystem

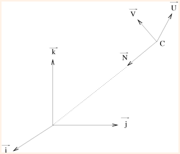
Transformation de repères

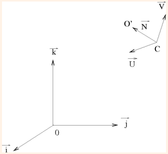
$$\begin{aligned}
\overrightarrow{U} &= c_{11}\overrightarrow{i} + c_{12}\overrightarrow{j} + c_{13}\overrightarrow{k}\\
\overrightarrow{V} &= c_{21}\overrightarrow{i} + c_{22}\overrightarrow{j} + c_{23}\overrightarrow{k}\\
\overrightarrow{N} &= c_{31}\overrightarrow{i} + c_{32}\overrightarrow{j} + c_{33}\overrightarrow{k}
\end{aligned}$$

Transformations de repère de caméra

Repère de Caméra

- une position de caméra (C)
- une direction de visée (point de focalisation : $\overrightarrow{N}$)
- un plan de caméra ($\overrightarrow{U}, \overrightarrow{V}$)





Calcul du plan de caméra ($\overrightarrow{U}, \overrightarrow{V}$)

- position de caméra ($\overrightarrow{OC}$), direction de visée par défaut ($\overrightarrow{CO}$)
- position de caméra ($\overrightarrow{OC}$), on choisit la direction de visée ($\overrightarrow{N}$)

Matrice de caméra : direction de visée par défaut

Définiton du repère de caméra

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \\ C_{41} & C_{42} & C_{43} & C_{44} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ w_w \end{bmatrix}$$

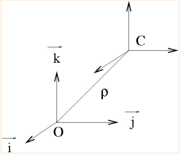
Direction de visée par défaut ($\overrightarrow{N} = \overrightarrow{CO}$)

$[C] = [M_u][R_x][R_z][T]$

- $[T]$: Translation au point d'observation
- $[R_z]$: Rotation autour de l'axe $\overrightarrow{k}$ de WCS
- $[R_u]$: Rotation autour de l'axe $\overrightarrow{U}$ de CCS
- $[M_u]$: passage à un trièdre indirect pour visualisation

Translation au point d'observation

$[T]$: Translation de Repère



$$[T] = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

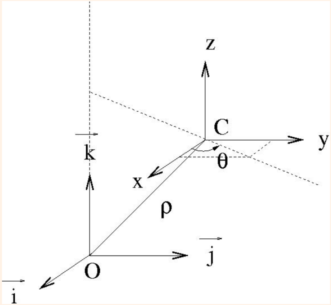
Transformation inverse

$$[T] = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ Au final : } \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 \\ C_{21} & C_{22} & C_{23} & 0 \\ C_{31} & C_{32} & C_{33} & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation autour de l'axe $\overrightarrow{k}$ de WCS

$[R_z]$: Rotation d'angle θ autour de l'axe $\overrightarrow{k}$

- amener l'axe Oy de la caméra
- parallèlement à la projection de $\overrightarrow{OC}$
- sur le plan (Oxy) du repère de scène



Rotation autour de l'axe $\overrightarrow{k}$ de WCS

$[R_z]$: Rotation d'angle θ autour de l'axe $\overrightarrow{k}$

$$[R_z] = \begin{bmatrix} \cos(\theta - \frac{\pi}{2}) & -\sin(\theta - \frac{\pi}{2}) & 0 & 0 \\ \sin(\theta - \frac{\pi}{2}) & \cos(\theta - \frac{\pi}{2}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

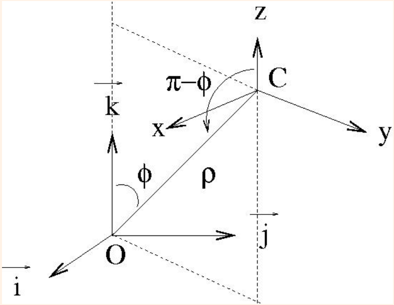
Transformation inverse

$$[R_z] = \begin{bmatrix} \sin\theta & -\cos\theta & 0 & 0 \\ \cos\theta & \sin\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation autour de l'axe $\overrightarrow{U}$ de CCS

$[R_u]$: Rotation d'angle ϕ autour de l'axe $\overrightarrow{U}$

- amener l'axe Oz de la caméra
- dans la direction de $\overrightarrow{CO}$



Rotation autour de l'axe $\overrightarrow{U}$ de CCS

$[R_u]$: Rotation d'angle ϕ autour de l'axe $\overrightarrow{U}$

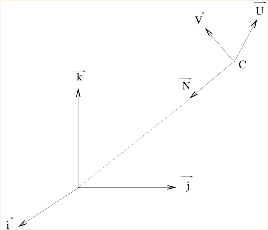
$$[R_u] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\pi - \varphi) & -\sin(\pi - \varphi) & 0 \\ 0 & \sin(\pi - \varphi) & \cos(\pi - \varphi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation inverse

$$[R_x] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & -\cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inversion de l'axe $\overrightarrow{U}$ de CCS

$[M_u]$: Plan de projection

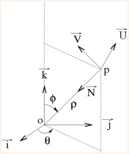


$$[M_u] = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrice de Caméra

Transformation de répère : WCS → CCS

$$[C] = [M_u][R_u][R_z][T]$$



Transformation inverse résultante

$$[C] = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\varphi\cos\theta & -\cos\varphi\sin\theta & \sin\varphi & 0 \\ -\sin\varphi\cos\theta & -\sin\varphi\sin\theta & -\cos\varphi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrice de Caméra

Repère de Transformation $(\overrightarrow{U}, \overrightarrow{V}, \overrightarrow{N})$

$$[C] = \begin{bmatrix} U_x & U_y & U_z & 0 \\ V_x & V_y & V_z & 0 \\ N_x & N_y & N_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrice de Transformation (ρ, θ, φ)

$$[C] = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\varphi\cos\theta & -\cos\varphi\sin\theta & \sin\varphi & 0 \\ -\sin\varphi\cos\theta & -\sin\varphi\sin\theta & -\cos\varphi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrice de Caméra

Repère de Transformation $(\overrightarrow{U}, \overrightarrow{V}, \overrightarrow{N})$

- $\overrightarrow{N}(N_x, N_y, N_z)$: direction d'observation (axe 0_z)
- $\overrightarrow{V}(V_x, V_y, V_z)$: vecteur vertical du plan caméra (axe 0_y)
- $\overrightarrow{U}(U_x, U_y, U_z)$: second vecteur du plan caméra (axe 0_x)

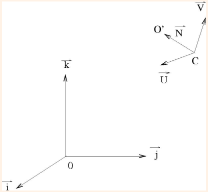
Repère de Orthonormé $(\overrightarrow{U}, \overrightarrow{V}, \overrightarrow{N})$

$$
\begin{array}{ccccccc}
||\overrightarrow{U}|| & = & ||\overrightarrow{V}|| & = & ||\overrightarrow{N}|| & = & 1 \\
\overrightarrow{U} \cdot \overrightarrow{V} & = & \overrightarrow{U} \cdot \overrightarrow{N} & = & \overrightarrow{V} \cdot \overrightarrow{N} & = & 0
\end{array}
$$

Choix de direction de visée

Définition du repère de caméra

- $\overrightarrow{N} = \dfrac{\overrightarrow{O'C}}{||\overrightarrow{O'C}||}$: connu
- si $\overrightarrow{U} \perp \overrightarrow{k}$ alors $\Rightarrow \overrightarrow{U} = \dfrac{\overrightarrow{N} \wedge \overrightarrow{k}}{||\overrightarrow{N} \wedge \overrightarrow{k}||}$ connu
- et donc $\overrightarrow{V} = \dfrac{\overrightarrow{N} \wedge \overrightarrow{U}}{||\overrightarrow{N} \wedge \overrightarrow{U}||}$ connu

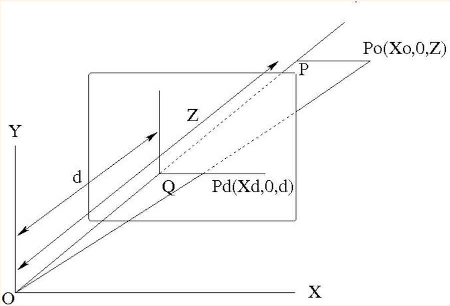


Projections

Types de projections

- projections parallèles : centre de projection à l'infini
- projections perspectives : centre de projection à distance finie

Calcul de projections sur un axe



Triangles semblables (Thalès) : $\frac{P_dQ}{OQ} = \frac{P_oP}{OP}$

Projection sur un axe

Calcul de projection sur un axe

$$\frac{x_d}{d} = \frac{x_0}{z} \; , \; x_d = d\frac{x_0}{z}$$

$$\frac{y_d}{d} = \frac{y_0}{z} \; , \; y_d = d\frac{y_0}{z}$$

Matrice de projection

$$[M_P] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix}$$

Matrice de Projection

Projections sur trois axes

$$[M_P] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{d_x} & \frac{1}{d_y} & \frac{1}{d_z} & 0 \end{bmatrix}$$

Matrice Homogène et transformations affines

$$[M] = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_{14} \\ R_{21} & R_{22} & R_{23} & T_{24} \\ R_{31} & R_{32} & R_{33} & T_{34} \\ P_{41} & P_{42} & P_{43} & E_{44} \end{bmatrix}$$

Quaternions

Définition

Quatre valeurs scalaires : $\mathbf{q} = [q_0, q_1, q_2, q_3]$
Autre représentation :

$$\mathbf{q} = [q_0, \overrightarrow{q}] \, , \; \overrightarrow{q} = q_1 \overrightarrow{i} + q_2 \overrightarrow{j} + q_3 \overrightarrow{k}$$

Hamilton : extension des nombres complexes

$\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ tels que :

- $\mathbf{ii} = \mathbf{jj} = \mathbf{kk} = -1$
- $\mathbf{ij} = -\mathbf{ji} = \mathbf{k}$
- $\mathbf{jk} = -\mathbf{kj} = \mathbf{i}$
- $\mathbf{ki} = -\mathbf{ik} = \mathbf{j}$

Quaternions

Groupe algébrique

- Quaternion : $\mathbf{q} = [q_0, \overrightarrow{q}]$
- Conjugué : $\overline{\mathbf{q}} = [q_0, -\overrightarrow{q}]$
- Module : $||\overline{\mathbf{q}}|| = \sqrt{\mathbf{q}\overline{\mathbf{q}}} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$
- Inverse : $\mathbf{q}^{-1} = \frac{\overline{\mathbf{q}}}{||\mathbf{q}||^2}$
- Quaternions unitaires : $\mathbf{q^{-1}} = \overline{\mathbf{q}}$ $(||\mathbf{q}|| = 1)$
- Elément neutre : $\mathbf{q} = [1, (0, 0, 0)]$

Lois de compositions

- $\mathbf{p} + \mathbf{q} = [p_0 + q_0, \overrightarrow{p} + \overrightarrow{q}]$
- $\mathbf{pq} = [p_0 q_0 - (\overrightarrow{p} \cdot \overrightarrow{q}),\ (q_0 \overrightarrow{p} + p_0 \overrightarrow{q} + \overrightarrow{p} \wedge \overrightarrow{q})]$

On pourra vérifier : $\mathbf{qq^{-1}} = [1, (0, 0, 0)]$

Quaternions et rotations

Rotation autour d'un Vecteur

- un point de l'espace $\overrightarrow{p}(x, y, z)$
- peut-être mis en rotation d'un angle θ
- autour d'un vecteur $\overrightarrow{q}$

par utilisation des quaternions

- $\mathbf{p} = [0, \overrightarrow{p}]$
- $\mathbf{q} = \left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\overrightarrow{q}\right]$

Résultat de la rotation

Quaternion ($\mathbf{p'} = [0, \overrightarrow{p'}]$) et vecteur résultant ($\overrightarrow{p'}$) :

$$\mathbf{p'} = \mathbf{qpq}^{-1}$$

Quaternions et rotations

Enchaînement de rotations

Loi d'associativité de quaternions :

$$\mathbf{q_2}\ (\mathbf{q_1 p q_1^{-1}})\ \mathbf{q_2^{-1}}$$

$$(\mathbf{q_2 q_1})\ \mathbf{p}\ (\mathbf{q_1^{-1} q_2^{-1}})$$

$$(\mathbf{q_2 q_1})\ \mathbf{p}\ (\mathbf{q_2 q_1})^{-1}$$

Rotations successives autour des axes $Pitch, Yaw, Roll$:

$$\mathbf{q} = \mathbf{q_{pitch} q_{yaw} q_{roll}}$$

Quaternions et rotations

Conversion Quaternion/Matrice de rotation

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} & 0 \\ M_{21} & M_{22} & M_{23} & 0 \\ M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & M_{44} \end{bmatrix}$$

$M_{11} = q_0^2 + q_1^2 - q_2^2 - q_3^2$, $M_{12} = 2q_1q_2 - 2q_0q_3$, $M_{13} = 2q_1q_3 + 2q_0q_2$
$M_{21} = 2q_1q_2 + 2q_0q_3$, $M_{22} = q_0^2 - q_1^2 + q_2^2 - q_3^2$, $M_{23} = 2q_2q_3 - 2q_0q_1$
$M_{31} = 2q_1q_3 - 2q_0q_2$, $M_{32} = 2q_2q_3 + 2q_0q_1$, $M_{33} = q_0^2 - q_1^2 - q_2^2 + q_3^2$
$M_{44} = q_0^2 + q_1^2 + q_2^2 + q_3^2$

Quaternions et rotations

Conversion Quaternion/Matrice de rotation

Représentation matricielle avec $||\mathbf{q}|| == \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$

$M_{11} = ||\mathbf{q}||^2 - 2(q_2^2 + q_3^2)$
$M_{22} = ||\mathbf{q}||^2 - 2(q_1^2 + q_3^2)$
$M_{33} = ||\mathbf{q}||^2 - 2(q_1^2 + q_2^2)$
$M_{44} = ||\mathbf{q}||^2$

Quaternions unitaires, $(||\mathbf{q}|| = 1)$

$$\begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) & 0 \\ 2(q_1q_2 + q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3 - q_0q_1) & 0 \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 1-2(q_1^2+q_2^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Quaternions et rotations

Conversion Matrice de rotation/Quaternion

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} & 0 \\ M_{21} & M_{22} & M_{23} & 0 \\ M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- $Trace(M) = M_{11} + M_{22} + M_{33}$
- 4 cas possibles : $max(Trace(M), M_{11}, M_{22}, M_{33})$

Quaternions et rotations

$Trace(M) = max(Trace(M), M_{11}, M_{22}, M_{33})$

- $q_0 = \frac{1}{2}\sqrt{Trace(M) + 1}$
- $q_1 = \frac{M_{32} - M_{23}}{4q_0}, q_2 = \frac{M_{13} - M_{31}}{4q_0}, q_3 = \frac{M_{21} - M_{12}}{4q_0}$

$M_{11} = max(Trace(M), M_{11}, M_{22}, M_{33})$

- $q_1 = \frac{1}{2}\sqrt{2M_{11} - Trace(M) + 1}$
- $q_0 = \frac{M_{32} - M_{23}}{4q_1}, q_2 = \frac{M_{21} + M_{12}}{4q_1}, q_3 = \frac{M_{13} + M_{31}}{4q_1}$

Quaternions et rotations

$M_{22} = max(Trace(M), M_{11}, M_{22}, M_{33})$

- $q_2 = \frac{1}{2}\sqrt{2M_{22} - Trace(M) + 1}$
- $q_0 = \frac{M_{13} - M_{31}}{4q_2}, q_1 = \frac{M_{21} + M_{12}}{4q_2}, q_3 = \frac{M_{32} + M_{23}}{4q_2}$

$M_{33} = max(Trace(M), M_{11}, M_{22}, M_{33})$

- $q_3 = \frac{1}{2}\sqrt{2M_{33} - Trace(M) + 1}$
- $q_0 = \frac{M_{21} - M_{12}}{4q_3}, q_1 = \frac{M_{13} + M_{31}}{4q_3}, q_2 = \frac{M_{32} + M_{23}}{4q_3}$

Déformations

Transformations affines

- homothéties, translations, rotations

Déformations

- déformations globales linéaires ou non (modèles de Barr)
- déformations souples, élastiques (modèles de Sedeberg)

Modèles de Barr et Sedeberg

Avantages :

- modélisation d'objets complexes
- technique d'animation par interpolation sur la déformation

Inconvénients :

- contrôle de la déformation

Déformations

Modèles de Barr

Appliquer une perturbation sur la transformation

- $\overrightarrow{P}(x, y, z)$: point de l'objet avant déformation
- $\overrightarrow{P}_{def}(X, Y, Z)$: point correspondant après déformation

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f(x, y, z) \\ g(x, y, z) \\ h(x, y, z) \end{bmatrix}$$

Modèles de Barr

Transformation d'échelle

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} e_x & 0 & 0 \\ 0 & e_y & 0 \\ 0 & 0 & e_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} e_x x \\ e_y y \\ e_z z \end{bmatrix}$$

Déformation d'échelle (tapering) sur un axe

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} rx \\ ry \\ z \end{bmatrix}$$

avec : $r = f(z)$

Modèles de Barr

Rotation autour d'un axe

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x\cos\theta & -y\sin\theta \\ x\sin\theta & +y\cos\theta \\ & z \end{bmatrix}$$

Déformation, torsion autour d'un axe

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x\cos\theta & -y\sin\theta \\ x\sin\theta & +y\cos\theta \\ & z \end{bmatrix}$$

avec : $\theta = f(z)$

Modèles de Barr

Problème : calcul de normales après déformation

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f(x,y,z) \\ g(x,y,z) \\ h(x,y,z) \end{bmatrix}$$

Solution : Jacobien de la déformation

$$\begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \\ \frac{\partial h}{\partial x} & \frac{\partial h}{\partial y} & \frac{\partial h}{\partial z} \end{bmatrix}$$

Modèles de Barr

Calcul de la normale au point déformé

- $\overrightarrow{n}$: normale au point $\overrightarrow{P}(x, y, z)$ avant déformation
- $\overrightarrow{N}$: normale au point $\overrightarrow{P_{def}}(X, Y, Z)$ après déformation

$$\overrightarrow{N} = \det(J)\ {}^{t}[J^{-1}]\ \overrightarrow{n}$$

Remarques sur le calcul de déformation

- $\det(J) = 1$: conservation du volume après déformation
- Calcul de $\det(J)$: inutile pour connaître uniquement la direction

Modèles de Barr

Calcul de normales après torsion autour de l'axe O_z

$$\begin{array}{rcl} X & = & x\cos\theta - y\sin\theta \\ Y & = & x\sin\theta + y\cos\theta \\ Z & = & z \end{array}$$

Jacobien de la transformation

$$[J] = \begin{bmatrix} \cos\theta & -\sin\theta & -\theta'(x\sin\theta + y\cos\theta) \\ \sin\theta & \cos\theta & \theta'(x\cos\theta - y\sin\theta) \\ 0 & 0 & 1 \end{bmatrix}$$

On en déduit

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

Modèles de Barr

Matrice des cofacteurs

$$[\widetilde{J}] = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ \theta' y & -\theta' x & 1 \end{bmatrix}$$

avec :

- $x = X\cos\theta + Y\sin\theta$
- $y = -X\sin\theta + Y\cos\theta$

Modèles de Barr

Calcul de normale après déformation

- $\overrightarrow{N} =^t [J^{-1}]\, \overrightarrow{n} = [\widetilde{J}]\, \overrightarrow{n}$

avec :

- $det(J) \; = \; 1$
- $[J^{-1}] =^t [\widetilde{J}]$

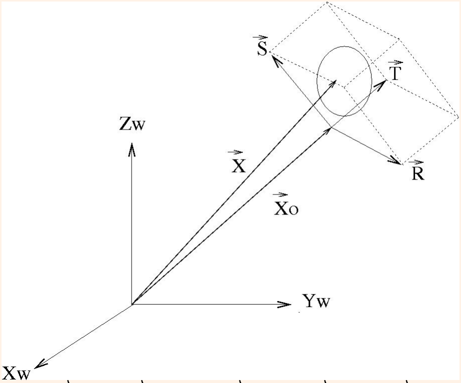
On trouve donc :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ \theta' y & -\theta' x & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Modèles de Sedeberg/Parry

FFD : Free Form Deformation

- déformation d'objets dans une région de l'espace



$$\overrightarrow{X} = \overrightarrow{X}_0 + r\overrightarrow{R} + s\overrightarrow{S} + t\overrightarrow{T}$$

Modèles de Sedeberg/Parry

FFD : Free Form Deformation

- $\overrightarrow{X_0}$: vecteur définissant l'origine du repère de déformation.
- $\overrightarrow{R}, \overrightarrow{S}, \overrightarrow{T}$: vecteurs du repère de déformation.

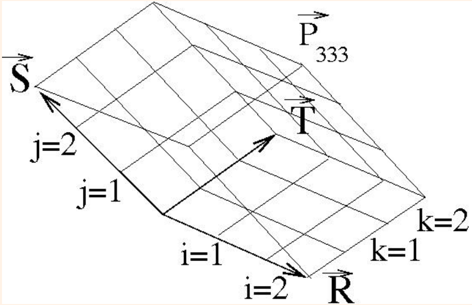
Calcul des paramètres $0 \leq (r, s, t) \leq 1$

$$
\begin{aligned}
r &= \frac{(\overrightarrow{S} \wedge \overrightarrow{T}) \cdot (\overrightarrow{X} - \overrightarrow{X}_0)}{\overrightarrow{R} \cdot (\overrightarrow{S} \wedge \overrightarrow{T})} \\
s &= \frac{(\overrightarrow{R} \wedge \overrightarrow{T}) \cdot (\overrightarrow{X} - \overrightarrow{X}_0)}{\overrightarrow{S} \cdot (\overrightarrow{R} \wedge \overrightarrow{T})} \\
t &= \frac{(\overrightarrow{R} \wedge \overrightarrow{S}) \cdot (\overrightarrow{X} - \overrightarrow{X}_0)}{\overrightarrow{T} \cdot (\overrightarrow{R} \wedge \overrightarrow{S})}
\end{aligned}
$$

Modèles de Sedeberg/Parry

Grille de points de contrôles

- $(l+1)$ plans suivant $\overrightarrow{R}$ (paramètre i)
- $(m+1)$ plans suivant $\overrightarrow{S}$ (paramètre j)
- $(n+1)$ plans suivant $\overrightarrow{T}$ (paramètre k)



Modèles de Sedeberg/Parry

Calcul de déformation

- $\overrightarrow{P}_{ijk}$: points de contrôle du cube de déformation
- Polynômes de Bernstein : $B_{n,i}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$

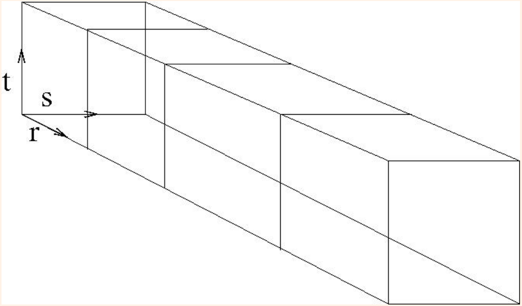
Point déformé $\overrightarrow{X}_{def}(r,s,t)$

$$\overrightarrow{X}_{def} = \sum_{i=0}^{l} B_{l,i}(r) \left[\sum_{j=0}^{m} B_{m,j}(s) \left(\sum_{k=0}^{n} B_{n,k}(t) \overrightarrow{P}_{ijk} \right) \right]$$

- A chaque valeur (r,s,t) correspond un point $\overrightarrow{X}$
- $\overrightarrow{X}_{def}(r,s,t) = f(\overrightarrow{P}_{ijk})$

Modèles de Sedeberg/Parry

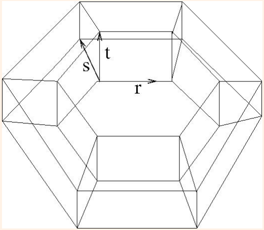
Extended Free Form Deformation



- limitations sur la déformation : box parallélépipédique
- solutions : utiliser des box non–cubiques de déformation

EFFD : Extended Free Form Deformation

Extended Free Form Deformation



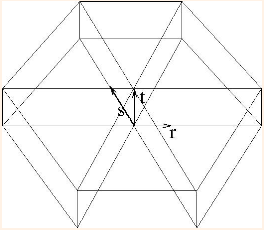
$$\overrightarrow{X_{def}} = \sum_{i=0}^{l} B_{l,i}(r) \left[\sum_{j=0}^{m} B_{m,j}(s) \left(\sum_{k=0}^{n} B_{n,k}(t) \overrightarrow{P_{ijk}} \right) \right]$$

On fait coïncider les points de contrôle

- $r = 0 : \overrightarrow{P_{0jk}}$
- $r = 1 : \overrightarrow{P_{ljk}} = \overrightarrow{P_{0jk}}$

EFFD : Extended Free Form Deformation

Extended Free Form Deformation



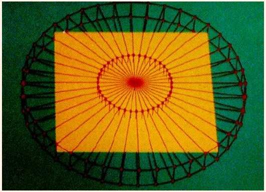
$$\overrightarrow{X_{def}} = \sum_{i=0}^{l} B_{l,i}(r) \left[\sum_{j=0}^{m} B_{m,j}(s) \left(\sum_{k=0}^{n} B_{n,k}(t) \overrightarrow{P_{ijk}} \right) \right]$$

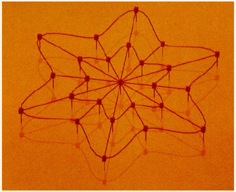
On fait coïncider les points de contrôle

- $s = 0$ et $t = 0$: $\overrightarrow{P_{i00}} = \overrightarrow{P_{i'00}}$
- $s = 0$ et $t = 1$: $\overrightarrow{P_{i0n}} = \overrightarrow{P_{i'0n}}$

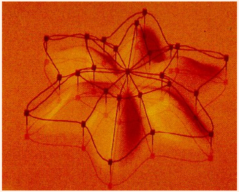
EFFD : Extended Free Form Deformation

Extended Free Form Deformation





Extended Free Form Deformation



Conclusion

Calculs matriciel

- coordonnées homogènes
- matrice de rotations (angles d'Euler)
- transformations de repères (caméra)

Quaternions

- calculs de rotations autour d'un axe
- expression matricielle de quaternions
- conversion quaternions / rotations

Déformations

- Rigides (Barr), souples (Sedeberg)
- calculs de normales après déformation

Bibliographie

Livres, articles

- **B. Péroche, D. Bechmann** :
  "Informatique graphique et rendu"
  Collection Hermès, Lavoisier (2007)
- **R. Malgouires** :
  "Algorithmes pour la synthèse d'images et l'animation 3D"
  Éditions Dunod (2002)
- **Samuel R. Buss** :
  "3D Computer Graphics :
  A mathematical introduction with OpenGL"
  Collection Cambridge University Press (2003)

Bibliographie

Articles

- **T. Sederberg and S.R. Parry** : "Free-form deformation of solid geometric models"SIGGRAPH '86
- **S. Coquillart** : "Extended Free-Form Deformation : A Sculpturing Tool for 3D Geometric Modeling" SIGGRAPH'90

Adresses "au Net"

- `http://paulbourke.net` :
  la 3D en long et en large (et en hauteur)
- `http://malgouyres.org`
- `http://culturemath.ens.fr/maths/pdf/logique/quaternions.pdf`
- `www.developpez.com` : entre autre de la 2D/3D

Annexes

Quaternion : Application PyOpenGL

```
def display() :
  ...
  glPushMatrix()              #  sphere to animate
  x,y,z=q_to_translate      #  translate quaternion
  glColor3f(0.0,1.0,0.0)
  glTranslatef(x,y,z)
  latitude,longitude=10,10
  create_sphere(size/5.,latitude,longitude)
  glPopMatrix()
  glutSwapBuffers()
```

Annexes

Quaternion : Application PyOpenGL

```
def animation() :
  global theta
  global q_to_translate # rotation to compute
  theta=theta+0.0001
  delta=0.5
  point=[0,delta,delta,delta]
  p=point_to_quaternion(point)
```

Annexes

Quaternion : Application PyOpenGL

```
axe=[sqrt(2)/2,-sqrt(2)/2,0.0]
q1=rotation_to_quaternion(theta,axe)
resultat=(q1*p)*q1.inverse()
q_to_translate=resultat.get_point()
q_to_translate=xyz_to_zxy(q_to_translate)
glutPostRedisplay()
```

Annexes

Quaternion : Application PyOpenGL

```
# OPenGL WCS point conversion
def xyz_to_zxy(point) :
  x,y,z=point
  z,x,y=x,y,z
  return x,y,z
```

Annexes

Quaternions : Classe

```
from math import hypot,pi,sin,cos,sqrt
class Quaternion:
  def __init__(self,a,b) :
    self.a=a
    self.b=b
  def __neg__(self):
    return Quaternion(-self.a,-self.b)
  def __add__(self,other):
    return Quaternion(self.a+other.a,self.b+other.b)
```

wikibooks : mathématiques avec Python et Ruby

Annexes

Quaternions : Classe

```
def __sub__(self,other):
  return Quaternion(self.a-other.a,self.b-other.b)
def __mul__(self,other):
  c=self.a*other.a-self.b*other.b.conjugate()
  d=self.a*other.b+self.b*other.a.conjugate()
  return Quaternion(c,d)
def __rmul__(self,k):
  return Quaternion(self.a*k,self.b*k)
def __abs__(self):
  return hypot(abs(self.a),abs(self.b))
def conjugate(self):
  return Quaternion(self.a.conjugate(),-self.b)
```

Annexes

Quaternions : Classe

```
def inverse(self):
  q=self.conjugate()
  q.set_a(q.get_a()/abs(self)**2)
  q.set_b(q.get_b()/abs(self)**2)
  return q
def __div__(self,other):
  return self*(1./abs(other)**2*other.conjugate())
def __pow__(self,n):
  r=1
  for i in range(n):
    r=r*self
  return r
```

Annexes

Quaternions : Tests

```
if __name__ == "__main__" :
  p0,p1,p2,p3=0,1,1,1
  a=complex(p0,p1)
  b=complex(p2,p3)
  p=Quaternion(a,b)          # quaternion to rotate
  theta=pi
  q0=cos(theta/2.)                    # rotation angle
  q1,q2,q3=0.0,-sqrt(2)/2,sqrt(2)/2  # rotation axis
  q1=sin(theta/2.)*q1
  q2=sin(theta/2.)*q2
  q3=sin(theta/2.)*q3
  a=complex(q0,q1)
  b=complex(q2,q3)
  q=Quaternion(a,b)         #  quaternion axis
```

Annexes

Quaternions : Tests

```
qp=q*p
print("Q*P : ",qp)
result=qp*q.inverse()
print("Q*P*Q^1 : ",result)
print("result : ",result.get_point())
```

Quaternions : Affichage

```
{logname@hostname} python quaternions.py
Q*P:(0.0)+(-1.4142135623730951)i+(0.7071067811865477)j+(0.7071067811865477)k
Q*P*Q^1:(0.0)+(-1.0000000000000002)i+(-1.0000000000000002)j+(-1.0000000000000002)k
result:(-1.0000000000000002, -1.0000000000000002, -1.0000000000000002)
```

Annexes

Quaternions : Affichage

```
class Quaternion:
  ...
  def __repr__(self):
    to_display='('
    to_display+=str(self.a.real)+')+('
    to_display+=str(self.a.imag)+')i+('
    to_display+=str(self.b.real)+')j+('
    to_display+=str(self.b.imag)+')k'
    return to_display
  ...
```