Coordonnées Homogènes

Laboratoire d'InfoRmatique en Image et Systèmes d'information

LIRIS **UMR 5205** CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon
Université Claude Bernard Lyon 1, 36 avenue Guy de Collongue - 69134 Ecully Cedex
http://liris.cnrs.fr

Transformations 2 dimensions

- On commence en 2D
  - Plus facile à représenter
- Chaque point est transformé :
  - $x' = f(x,y)$
  - $y' = g(x,y)$
- Comment représenter la transformation ?

Translation

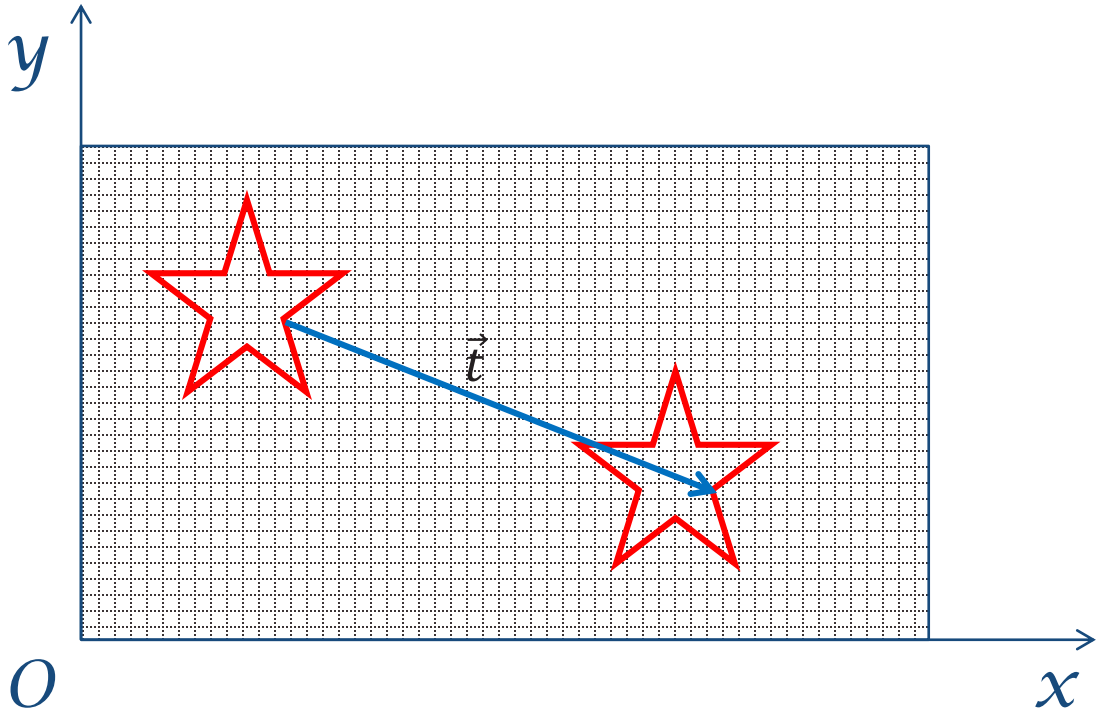
$$x' = x + t\ \ x$$
$$y' = y + t\ \ y$$

$$p = {}^{t}[x\,, y]$$

$$t = {}^{t}[u\,, v]$$

$$p' = {}^{t}[x'\,, y']$$

$$p' = p + t$$



Addition par le vecteur de translation

Changement d'échelle

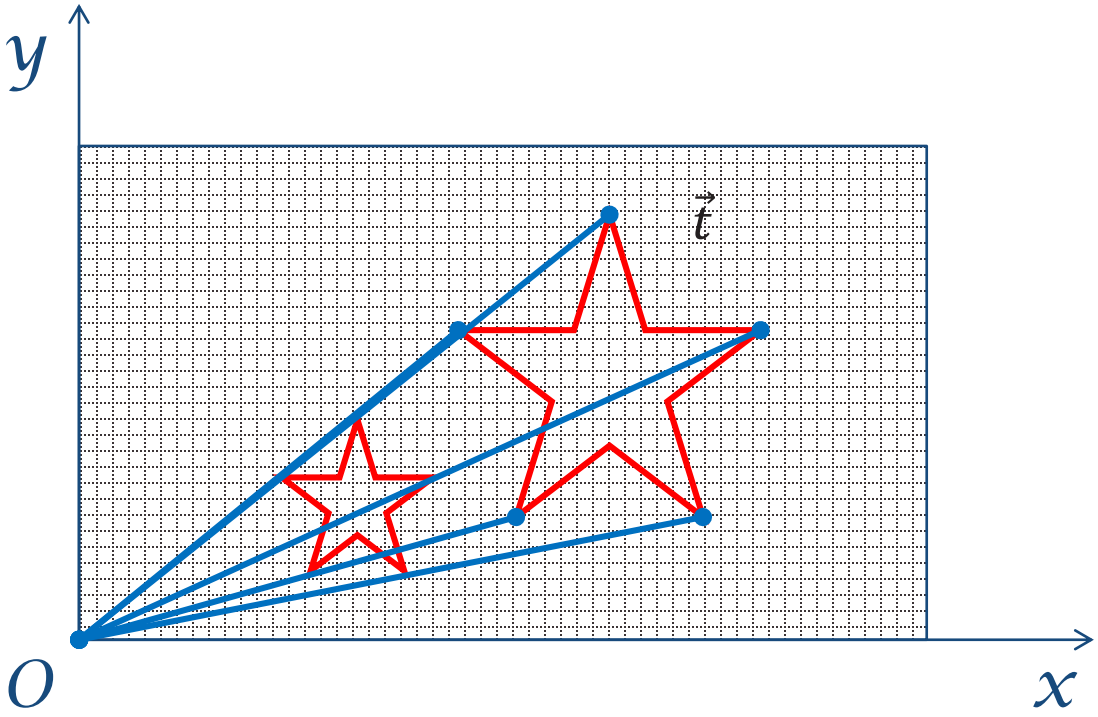
$$x' = s_x\, x$$
$$y' = s_y\, y$$

$$p = {}^t[x\,, y]$$

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

$$p' = {}^t[x'\,, y']$$

$$p' = S\, p$$



Multiplication par le facteur de changement d'échelle

Rotation

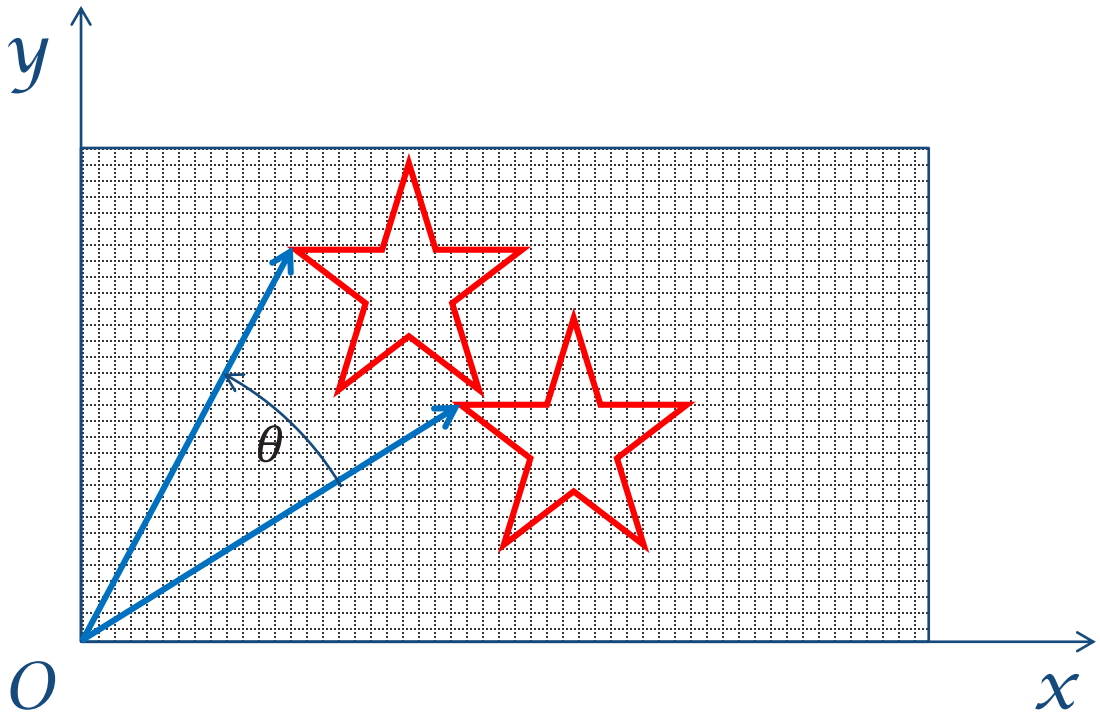
$$x' = \cos\theta\, x - \sin\theta\ \ y$$
$$y' = \sin\theta\, x + \cos\theta\, y$$

$$p = {}^t[x\,, y]$$

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$p' = {}^t[x'\,, y']$$

$$p' = R\, p$$



Multiplication par la matrice de rotation

Coordonnées homogènes

- Outil géométrique puissant :
  - Utilisé vision par ordinateur, synthèse d'image, géométrie projective
- Plan projectif
  - Ajout d'une troisième coordonnée, w
- Un point 2D devient un vecteur à 3 coordonnées

$$p = {}^t[x, y, w]$$

- Egalité entre deux points p et p'

$$p' \frac{1}{w'} = p \frac{1}{w}$$

$${}^t\left[\frac{x'}{w'}, \frac{y'}{w'}, 1\right] = {}^t\left[\frac{x}{w}, \frac{y}{w}, 1\right]$$

- $w = 0$: points « à l'infini »
  - Très utile pour les projections, et pour certaines splines

En 3 dimensions

- Un point 3D devient un vecteur à 4 coordonnées

$$P = {}^t[X, Y\ Z, W]$$

- Egalité entre deux points P et P'

$$P' \frac{1}{W'} = P \frac{1}{W}$$

$${}^t\left[\frac{X'}{W'}, \frac{Y'}{W'}, \frac{Z'}{W'}, 1\right] = {}^t\left[\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W}, 1\right]$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- $w = 0$: points « à l'infini »
- Toutes les transformations sont exprimées sous forme de matrices 4x4

Translation en c. homogènes

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} \frac{x'}{w'} = \frac{x}{w} + t_x \\ \frac{y'}{w'} = \frac{y}{w} + t_y \end{cases}$$

$$\begin{cases} x' = x + wt_x \\ y' = y + wt_y \\ w' = w \end{cases}$$

Changement d'échelle

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} x' = s_x\, x \\ y' = s_y\, y \\ w' = w \end{cases}$$

$$\begin{cases} \dfrac{x'}{w'} = s_x \dfrac{x}{w} \\ \dfrac{y'}{w'} = s_y \dfrac{y}{w} \end{cases}$$

Rotation

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} \dfrac{x'}{w'} = \cos\theta \ \dfrac{x}{w} - \sin\theta \ \dfrac{y}{w} \\ \dfrac{y'}{w'} = \sin\theta \ \dfrac{x}{w} + \cos\theta \ \dfrac{y}{w} \end{cases}$$

$$\begin{cases} x' = \cos\theta \ x - \sin\theta \ y \\ y' = \sin\theta \ x + \cos\theta \ y \\ w' = w \end{cases}$$

Composition des transformations

- Par multiplication matricielle
  - Une rotation suivie d'une translation :

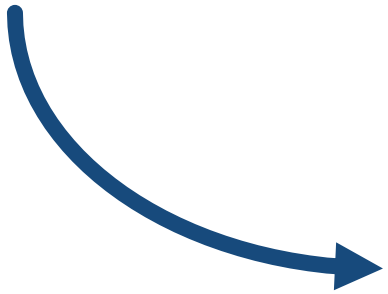
$$M = R\ T$$

- Rotation autour d'un point Q
  - Translater Q à l'origine (TQ),
  - Rotation autour de l'origine (RQ)
  - Translater en retour vers Q $(-\ TQ)$

$$p' = \left(-T_Q\right) R_\Theta\ T_Q\ p$$

Translation en 3D

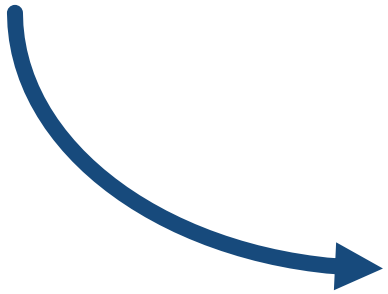
$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\begin{cases} x' = x + w\, t_x \\ y' = y + w\, t_y \\ z' = z + w\, t_z \\ w' = w \end{cases}$$

Changement d'échelle en 3D

$$S(s_x, s_y, s_z) = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\begin{cases} x' = x + w\, t_x \\ y' = y + w\, t_y \\ z' = z + w\, t_z \\ w' = w \end{cases}$$

Rotation en 3D

- Définie par un axe et un angle
- La matrice dépend de l'axe et de l'angle
- Expression directe possible, en partant de l'axe et de l'angle, et quelques produits vectoriels
  - Passage par les quaternions
- Fait par la librairie graphique :
  - glRotatef(angle, x, y, z)

Quaternion

- Soit la matrice de rotation R. Il existe quatre nombres $\{q_1, q_2, q_3, q_4\}$ tels que $q_0^2+q_1^2+q_2^2+q_3^2=0$, et

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_2 q_1 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_3 q_1 - q_0 q_2) & 2(q_3 q_2 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}$$

Quaternion

La représentation quaternion $\{q_1, q_2, q_3, q_4\}$ d'une rotation autour d'un axe l et d' un angle Ω, $0 \leq \Omega \leq \pi$, est donnée par

$$q_0 = \varepsilon \cos\frac{\Omega}{2}, \qquad \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} = \varepsilon\, l \sin\frac{\Omega}{2}, \qquad \varepsilon = \pm 1$$

$$\Omega = 2\cos^{-1} q_0, \qquad l = \frac{\begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}}{\left\| \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} \right\|}, \qquad si\ q_0 \geq 0,$$

.

$$\Omega = 2(\pi - \cos^{-1} q_0), \qquad l = -\frac{\begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}}{\left\| \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} \right\|}, \qquad si\ q_0 < 0.$$

Prérequis

- Soit la matrice de corrélation *K*. Soit *K'* la matrice symétrique de dimension quatre, définie à partir de *K* de la manière suivante

$$K' = \begin{pmatrix} K_{11}+K_{22}+K_{33} & K_{32}-K_{23} & K_{13}-K_{31} & K_{21}-K_{12} \\ K_{32}-K_{23} & K_{11}-K_{22}-K_{33} & K_{12}+K_{21} & K_{31}+K_{13} \\ K_{13}-K_{31} & K_{12}+K_{21} & -K_{11}+K_{22}-K_{33} & K_{23}+K_{32} \\ K_{21}-K_{12} & K_{31}+K_{13} & K_{23}+K_{32} & -K_{11}-K_{22}+K_{33} \end{pmatrix}$$

- Soit *q'* le vecteur propre unité de dimension quatre de la matrice *K'*, associé à la plus grande valeur propre
- Alors, *trace*($^tR\,K$) est maximisée par la matrice de rotation *R* définie par *q'*
- La solution est unique si la plus grande valeur propre est une racine simple du polynôme caractéristique

Décomposition de la matrice épipolaire

- Calculer h, le vecteur propre unité de la matrice ${}^tM\,M$ pour la plus petite valeur propre
- Poser $K = \sim h \times M$, et par la méthode de représentation quaternion calculer la matrice de rotation R qui maximise :

$$\text{trace}({}^tR\,K)$$

- Retourner les paramètres du mouvement (l, Ω, h)
- Calcul du résidu de l'équation épipolaire

Toutes les transformations 3D

- Toute transformation 3D s'exprime comme combinaison de translations, rotations, changement d'échelle
  - Et donc comme une matrice en coordonnées homogènes
- Fournies par la librairie graphique :
  - glTranslatef(x, y, z);
  - glRotatef(angle, x, y, z);
  - glScalef(x, y, z);

Transformations 3D (suite)

- On peut faire ses transformations soi-même :
  - glLoadIdentity();
    - Remplace la matrice de transformation courante par la matrice identité
  - glLoadMatrixf(pm);
    - Remplace la matrice de transformation courante par la matrice pm exprimée en floats
  - glMultMatrixf(pm);
    - Multiplie la matrice de transformation courante par la matrice pm exprimée en floats
- Pile de transformations :
  - glPushMatrix();
    - Pousse la matrice courante dans le stack et la duplique. Après l'appel à cette fonction la matrice au dessus du stack est dupliquée.
  - glPopMatrix();
    - Déplie la matrice courante.

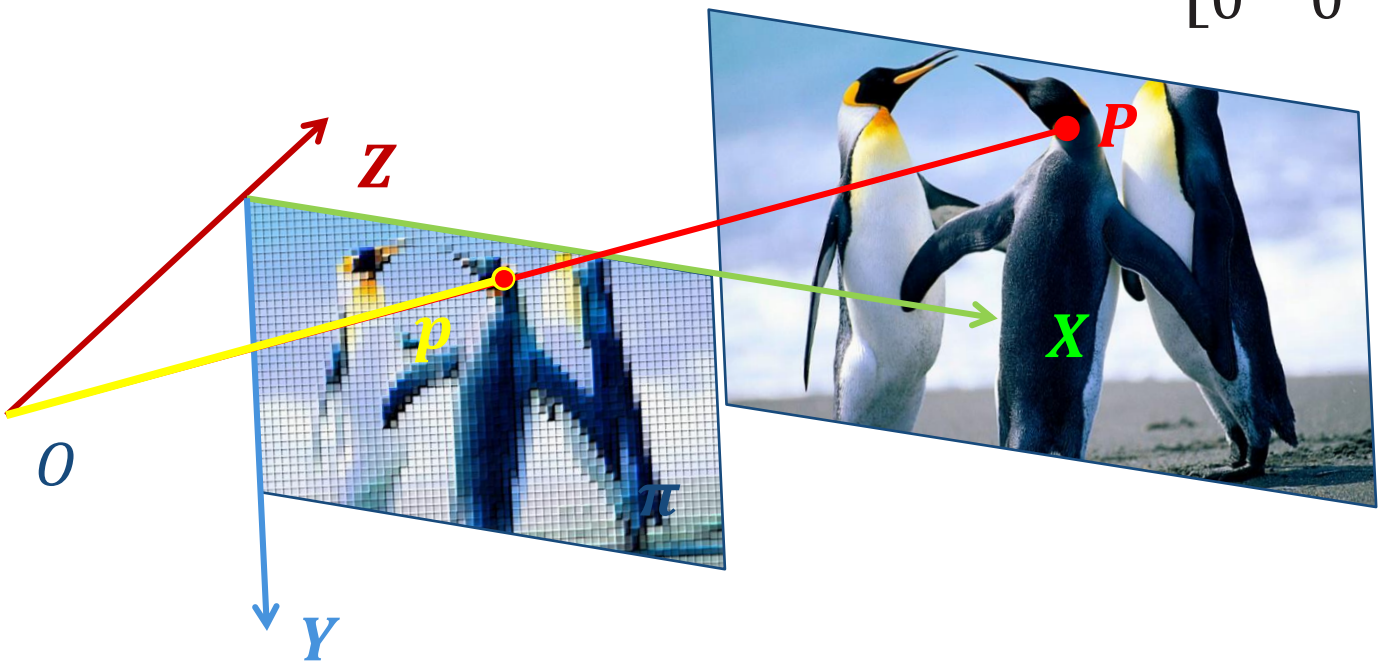
Exemple

```
drawHighLevelObject(parameters) {
                                    glPushMatrix()
                                      glRotate(…)
                                     glTranslate(…)
                                      glScale(…)
                                    drawSimpleShape()
  gllPopMatrix()
}
drawModel()                                                          {
                                    glPushMatrix()
                                  drawHighLevelObject1(…)
                                    glTranslate(…)
                                  drawHighLevelObject2(…)
                                       [etc…]
  gllPopMatrix()
}
```

Supplément : projection perspective

- Projection sur le plan z = 0, avec le centre de projection placé à z = – d

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{bmatrix}$$



Supplément : perspective (suite)

- Projection perspective s'appuie sur coordonnées homogènes
- La rétrécissement des objets utilise w

$$W' = \frac{Z}{d} + W$$

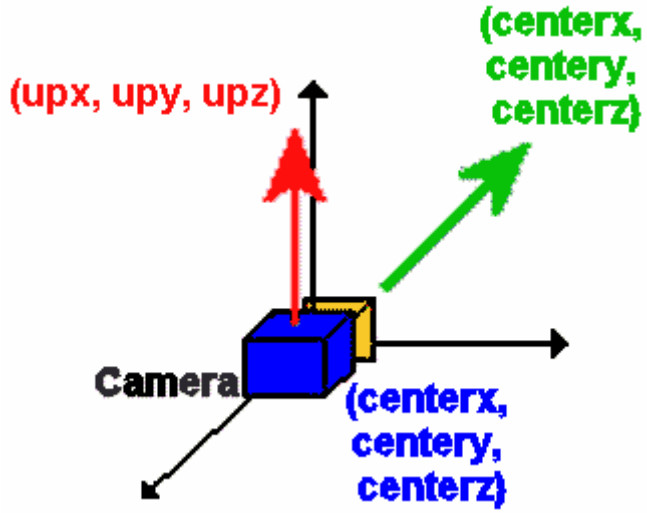
$$\frac{X'}{W'} = \frac{X}{\frac{Z}{d} + W}$$

$$\frac{Y'}{W'} = \frac{Y}{\frac{Z}{d} + W}$$

Perspective en pratique

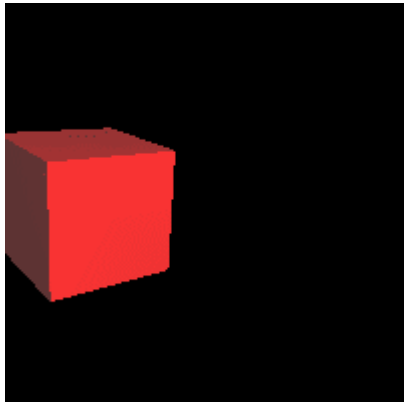
void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz);

- Définit la position et l'orientation de la caméra.
- (eyex, eyey, eyez) spécifies la position de la caméra,
- (centerx, centery, centerz) spécifies la position vers laquelle la caméra pointe (aussi appelé la ligne de mire)
- (upx, upy, upz) spécifies un vecteur qui nous informe de la direction du haut de la caméra
- Cette commande est contenu dans la GLU (OpenGL Utility Library).
- Inclure le header (fichier d'en-tête) suivant : glu.h

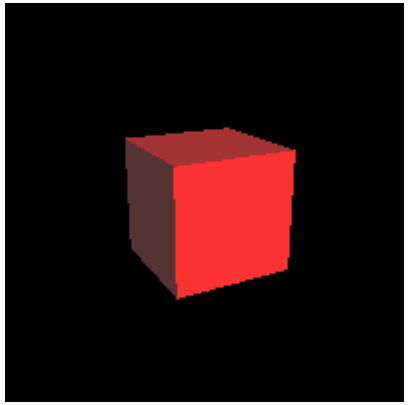


Perspective : en pratique

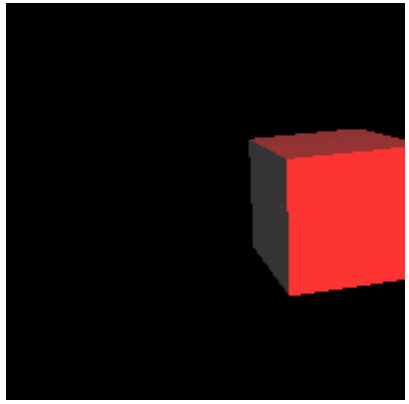
```
...
glLoadIdentity(); // Restaure la matrice de modélisation-visualisation
gluLookAt(0.0, 0.0, 4.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); // Positionne et
   oriente la caméra Dessin d'un cube
...
```



gluLookAt(0, 0, 4, 1, 0, 0, 0, 1, 0)



gluLookAt(0, 0, 4, 0, 0, 0, 0, 1, 0)



gluLookAt(0, 0, 4, -1, 0, 0, 0, 1, 0)

Perspective : en pratique

void gluPerspective(GLdouble fovy, GLdouble aspect,
GLdouble zNear, GLdouble zFar);

- Assure la projection en perspective.
- fovy spécifie l'angle du champs de vision, en degrés, dans la direction de y,
- aspect spécifie l'aspect ratio qui détermine le champs de vision dans la direction de x. aspect ratio est le rapport de x (largeur) sur y (hauteur) ,
- zNear spécifie la distance de l'observateur au plus proche plan (toujours positif),
- zFar spécifie la distance de l'observateur au plus loin plan (toujours positif),
- Inclure le header (fichier d'en-tête) suivant : glu.h.

$$f = \cot angent\left(\frac{fovy}{2}\right)$$

$$\begin{pmatrix} \frac{f}{aspect} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{zFar + zNear}{zFar - zNear} & \frac{2 \times zFar \times zNear}{zFar - zNear} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$