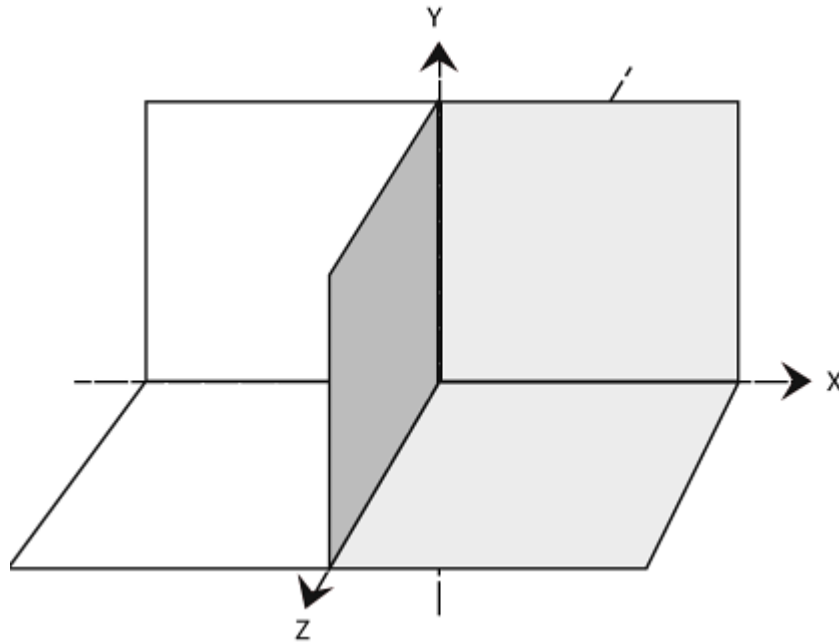


Les transformations géométriques en 3D

Ahlem Bougarradh

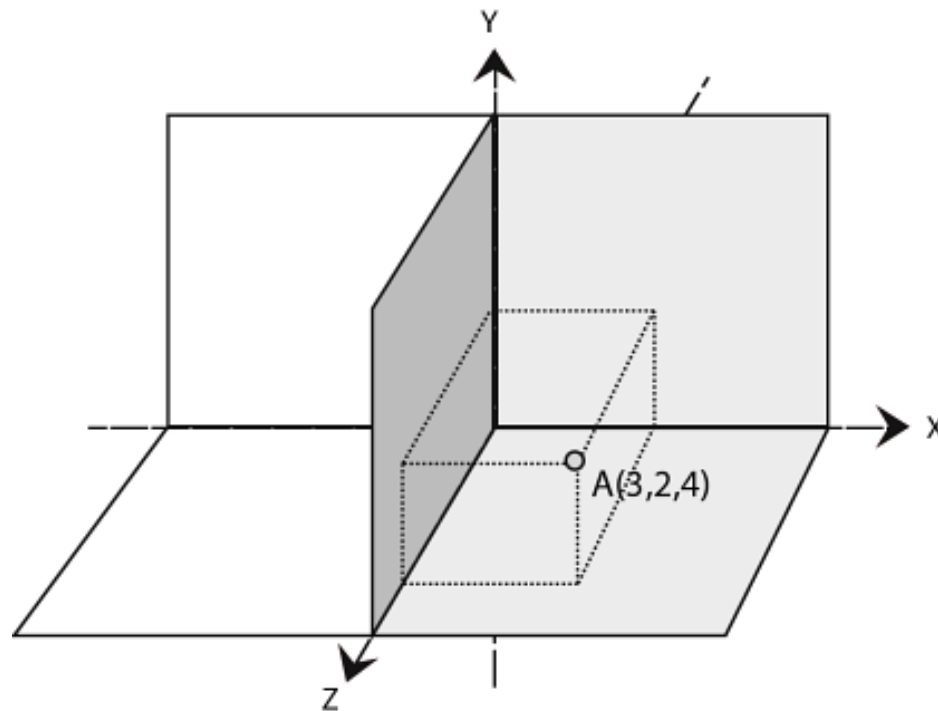
L'espace 3D

- Un point dans l'espace 3d est représenté par ses coordonnées (x,y,z)



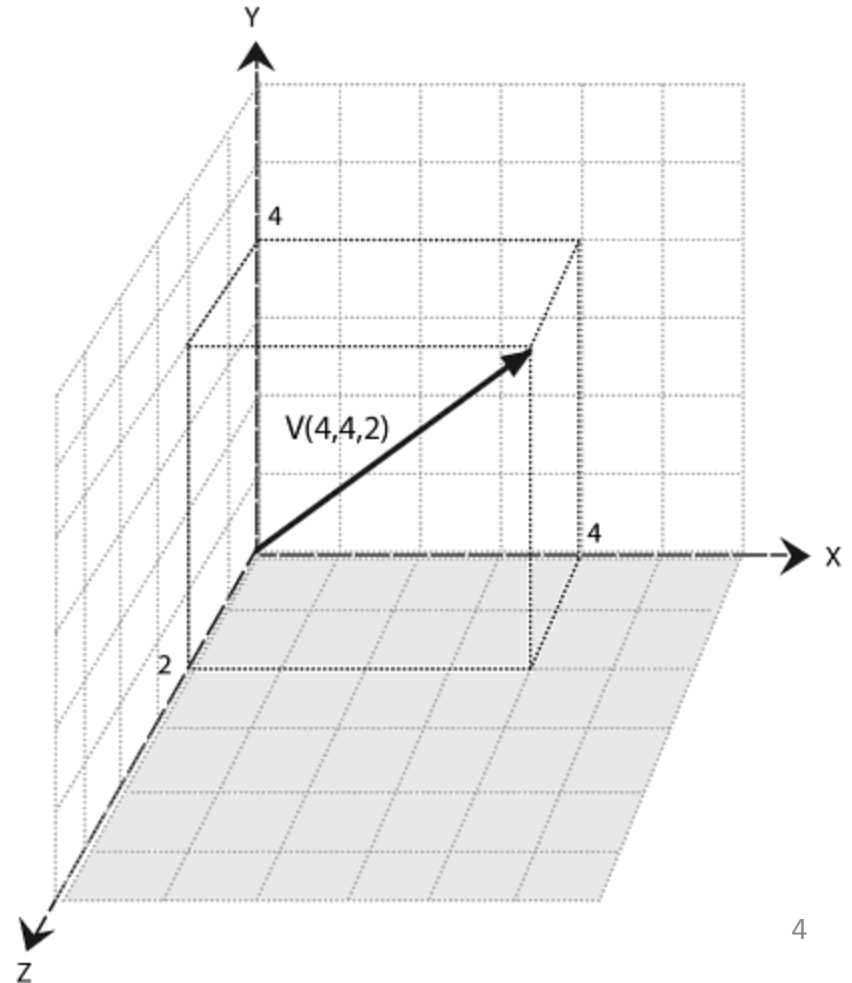
L'espace 3D

- Le point $A(3,2,4)$

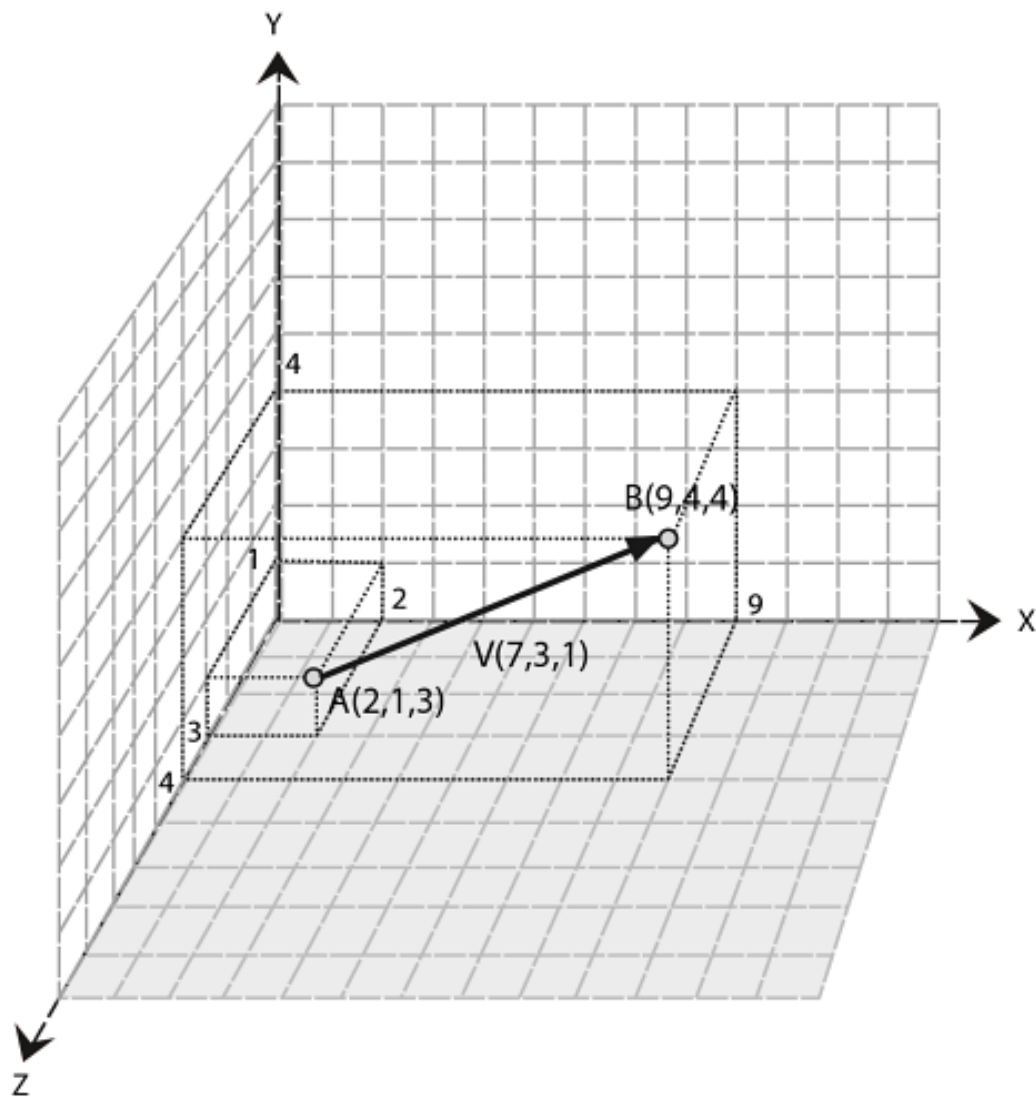


L'espace 3D

- Un vecteur 3D représente un déplacement dans l'espace 3D.
- $V(4,4,2)$
- Exemple: faire un déplacement du point $A(2,1,3)$ par le vecteur $V(7,3,1)$.
- Le résultant $B(?,?,?)$



L'espace 3D



Transformation Affine

- Toute composition de translation, rotation et de mise à l'échelle est appelée transformation affine.
- Soit deux points $P1(x1,y1,z1)$ et $P2(x2,y2,z2)$ de l'espace 3D en coordonnées euclidiennes.
- Si $P2$ est l'image de $P1$ par une transformation alors cette transformation est affine si elle respecte la condition suivante:

Transformation Affine

$$\begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} = \begin{bmatrix} m11 & m12 & m13 \\ m21 & m22 & m23 \\ m31 & m32 & m33 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} + \begin{bmatrix} e \\ f \\ g \end{bmatrix} \rightarrow P2 = M.P1 + T$$

- P2 est obtenu en multipliant P1 par la matrice M puis en lui ajoutant la matrice T.
- La transformation affine préserve le parallélisme des droites
- Ne préserve pas les longueurs et les angles.

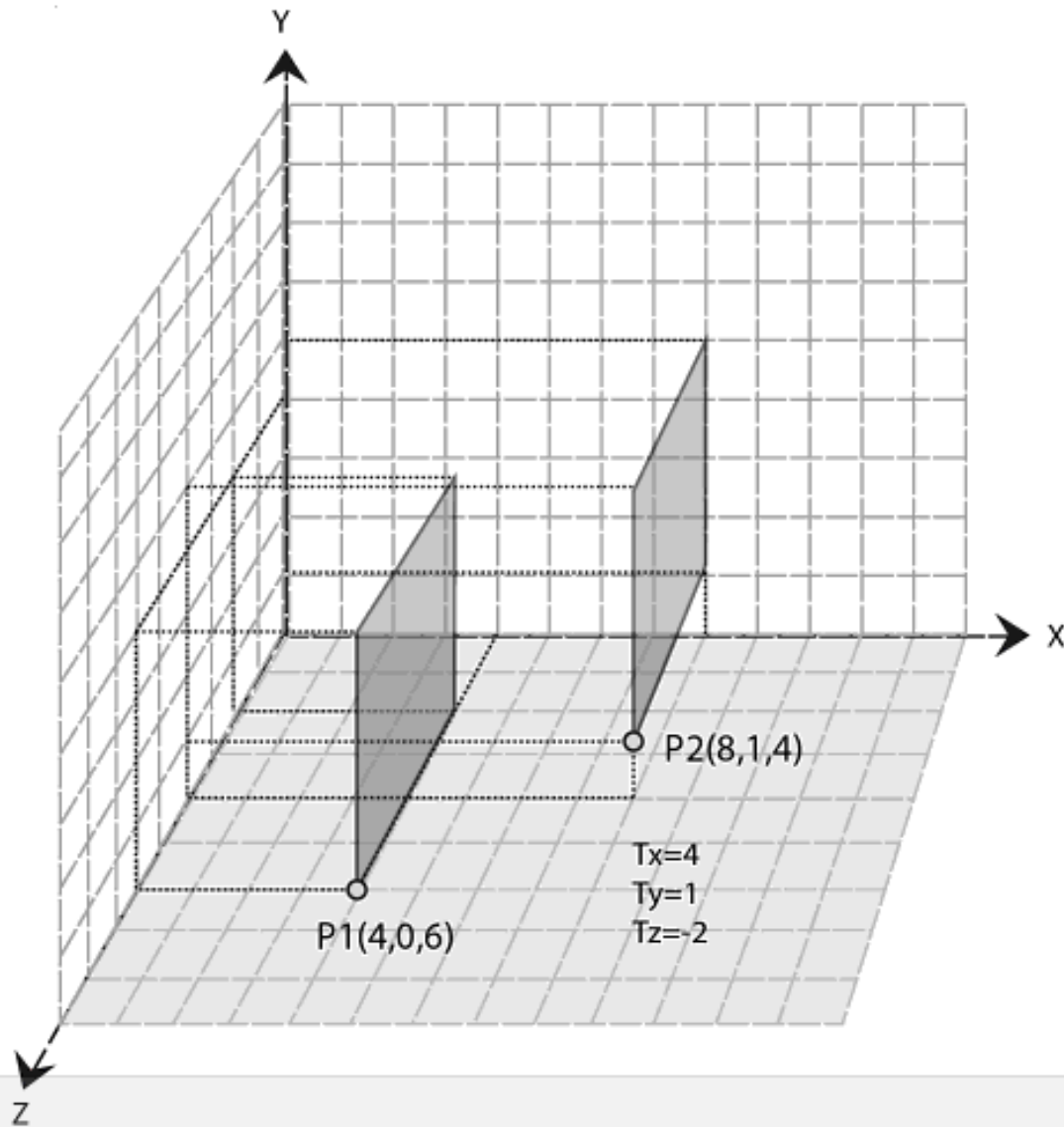
Translation

- Translation: consiste à déplacer un point d'une distance T_x suivant l'axe des X, d'une distance T_y suivant l'axe des Y et d'une distance T_z suivant l'axe des Z.

$$\begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} = \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} + \begin{bmatrix} Tx \\ Ty \\ Tz \end{bmatrix} \rightarrow P2 = P1 + T \quad \begin{cases} x2 = x1 + Tx \\ y2 = y1 + Ty \\ z2 = z1 + Tz \end{cases}$$

- Déterminez les coordonnées du Point P2 transformé de P1(4,0,6) par la matrice translation T($T_x=4$, $T_y=1$, $T_z=-2$)

Translation



Mise à l'échelle

- La mise à l'échelle consiste à agrandir ou à réduire en fonction des facteurs d'échelles S_x selon l'axe X, S_y selon l'axe Y et S_z selon l'axe Z.
- $P_2 = M_s * P_1$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \rightarrow P_2 = M_s . P_1$$

Mise à l'échelle

$$\begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & Sz \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix}$$

$x2 = Sx \cdot x1 + 0 \cdot y1 + 0 \cdot z1$
 donc $x2 = Sx \cdot x1$

$$\begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & Sz \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix}$$

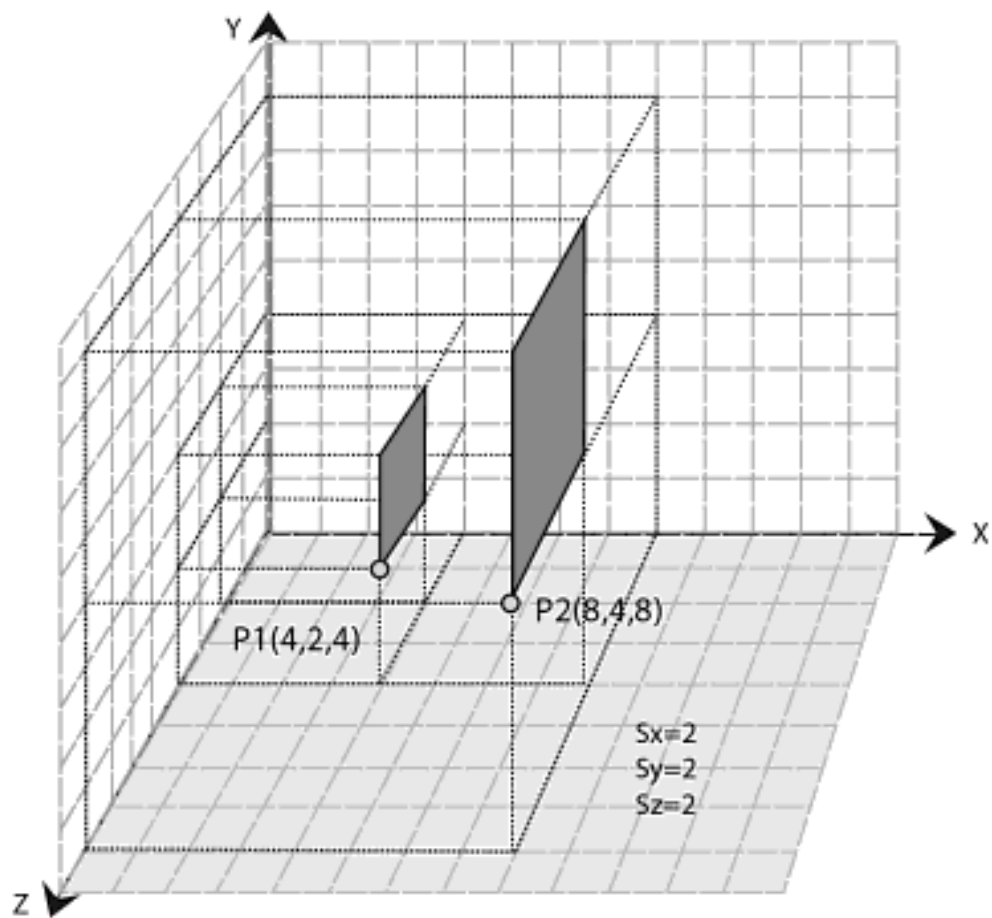
$y2 = 0 \cdot x1 + Sy \cdot y1 + 0 \cdot z1$
 donc $y2 = Sy \cdot y1$

$$\begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & Sz \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix}$$

$z2 = 0 \cdot x1 + 0 \cdot y1 + Sz \cdot z1$
 donc $z2 = Sz \cdot z1$

- Ex: déterminez P2(x2,y2,y3) transformé du point P1(4,2,4) par la matrice de mise à l'échelle uniforme de valeur 2 S(Sx=2, Sy=2,Sz=2)

Mise à l'échelle



Rotation

- La matrice de Rotation: pour faire pivoter des points d'un angle alpha par rapport à l'axe des X.

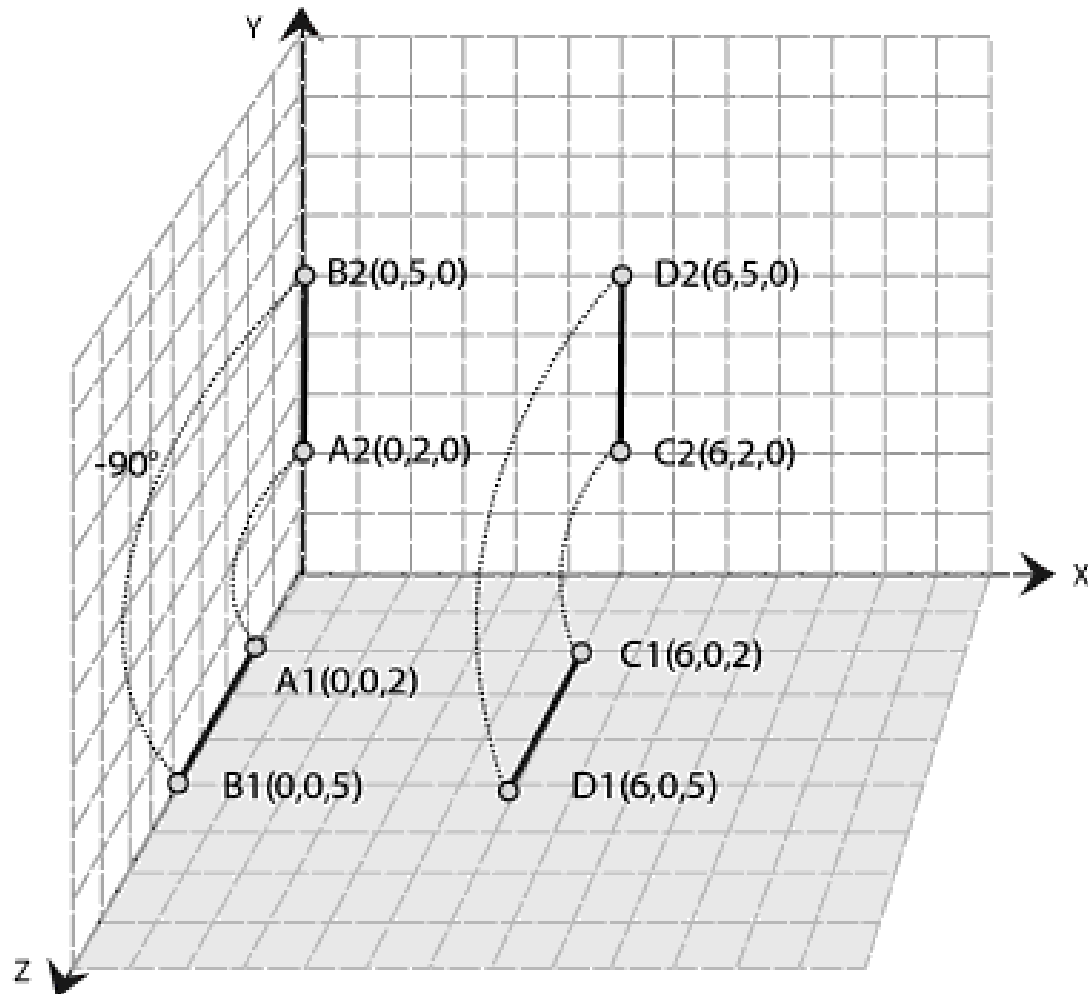
$$\begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} \quad \rightarrow \quad P2 = M_r . P1$$

Rotation

$$\begin{aligned}
 \begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} &\quad \begin{array}{l} \text{ } \\ \text{ } \\ \text{ } \end{array} \\
 \begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} &\quad \begin{array}{l} x2 = 1*x1 + 0*y1 + 0*z1 \\ \text{donc } x2 = x1 \end{array} \\
 \begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} &\quad \begin{array}{l} \text{ } \\ y2 = 0*x1 + \cos(\alpha)*y1 + (-\sin(\alpha))*z1 \\ \text{donc } y2 = y1*\cos(\alpha) - z1*\sin(\alpha) \end{array} \\
 \begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} &\quad \begin{array}{l} \text{ } \\ \text{ } \\ z2 = 0*x1 + \sin(\alpha)*y1 + \cos(\alpha)*z1 \\ \text{donc } z2 = y1*\sin(\alpha) + z1*\cos(\alpha) \end{array}
 \end{aligned}$$

- Ex: déterminez les coordonnées du segment [A2 B2] transformé du segment [A1B1] par une matrice de rotation d'un angle $\alpha = -90$ autour de l'axe des X. A1(0,0,2) et B1(0,0,5).

Rotation d'un segment



Rotation

- De la même façon, la matrice de rotation pour faire pivoter les points d'un angle alpha autour de l'axe Y est:

$$\begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} \rightarrow P2 = M_r.P1$$

- autour de l'axe Z est:

$$\begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} \rightarrow P2 = M_r.P1$$

Etirement

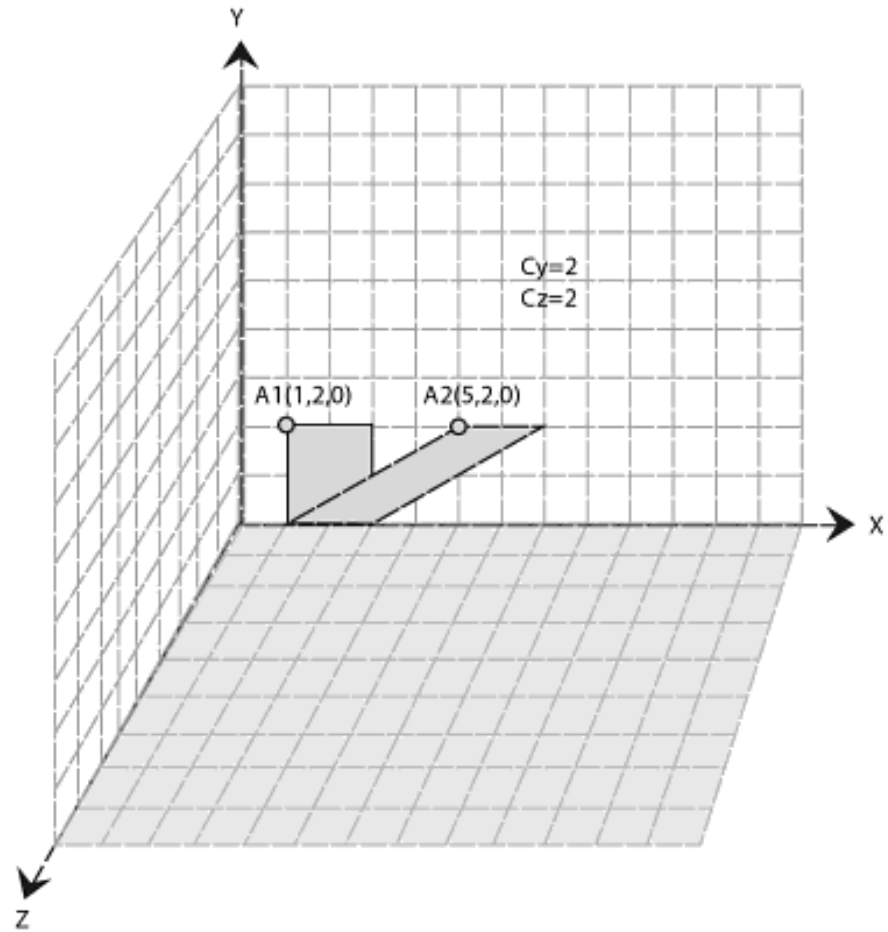
- Etirement ou Cisaillement: c'est une opération qui consiste à déformer les objets.
- On distingue 3 types d'étirements: suivant l'axe X, Y et Z.
- La matrice d'étirement selon l'axe X:

$$\begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} = \begin{bmatrix} 1 & C_y & C_z \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} \rightarrow P2 = M_c . P1$$

Etirement

- $X_2 = x_1 + y_1 * c_y + z_1 * c_z$
- $Y_2 = y_1$
- $Z_2 = z_1$
- Déterminez les coordonnées du point A2 transformé de A1(1,2,0) par un étirement selon l'axe X avec $c_y=2$ et $c_z=2$.

Etirement



Etirement de la figure présentant un carré

Les coordonnées homogènes

- Les mathématiques des images par ordinateur sont liées à la multiplication matricielle.
- La translation d'un point ne correspond pas directement à une multiplication matricielle. (parce que la translation n'est pas une transformation linéaire).
- Pour contourner cette difficulté ,il est classique d'introduire ce qu'on appelle les coordonnées homogènes.
- On Introduit une quatrième coordonnée différente de zéro W .

Les coordonnées homogènes

- Un point $P(x,y,z)$ en coordonnées euclidiennes s'écrit $P(x/w, y/w, z/w, w)$ en coordonnées homogènes.
- Si $w=1$ alors $P(x,y,z,1)$.
- Si $w=0$ représente les points à l'infini
- Les transformations 3d sont représentées par des matrices 4×4
- Une matrice unique 4×4 pourra effectuer toutes les transformations affines (translation, rotation, échelle, étirement).
- Rassembler en une seule matrice, une composition de transformations en effectuant une multiplication de matrices.


Matrice de transformation

- Un point $P1(x1,y1,z1,1)$ donnera un point $P2(x2,y2,z2,1)$ par une matrice de transformation M :

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix}$$


Matrice de transformation

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix}$$




$$x2 = m11*x1 + m12*y1 + m13*z1 + m14*1$$

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix}$$




$$y2 = m21*x1 + m22*y1 + m23*z1 + m24*1$$

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix}$$



$$z2 = m31*x1 + m32*y1 + m33*z1 + m34*1$$

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix}$$



$$1 = 0*x1 + 0*y1 + 0*z1 + 1*1$$

ce qui donne bien 1=1

Matrice de transformation

composantes pour la rotation, la mise à l'échelle et le cisaillement



composantes pour la translation



$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix}$$

- La matrice de transformation de translation:

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix} \rightarrow P2 = Mt \cdot P1$$

Matrice de transformation

**Matrice de Rotation autour
de l'axe X**

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix} \rightarrow P2 = Mr \cdot P1$$

**Matrice de Rotation autour
de l'axe Y**

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix} \rightarrow P2 = Mr \cdot P1$$

**Matrice de Rotation autour
de l'axe Z**

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix} \rightarrow P2 = Mr \cdot P1$$

Matrice de transformation

- La mise à l'échelle modifie la taille d'un objet par rapport à l'origine en l'agrandissant ou en la réduisant.
- La matrice de transformation Ms:

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix} \rightarrow P2 = Ms \cdot P1$$

Matrice de transformation

Matrice d'étirement autour de l'axe X

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & C_y & C_z & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix} \rightarrow P2 = M_c \cdot P1$$

Matrice d'étirement autour de l'axe Y

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ C_x & 1 & C_z & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix} \rightarrow P2 = M_c \cdot P1$$

Matrice d'étirement autour de l'axe Z

$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ C_x & C_y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix} \rightarrow P2 = M_c \cdot P1$$

Composition de matrices

- Exemple de composition de transformation:

$$P' = M_r.M_t.P$$

$$P' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1.5 \\ 10 \\ 2.1 \\ 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 3.5 \\ -6.1 \\ 11 \\ 1 \end{bmatrix}$$

Les transformations géométriques en Opengl

- **void glTranslatef(float x,float y,float z);**
- Cette fonction (et sa variante glTranslated()) dont les paramètres sont de type 'double') multiplie (à droite) la matrice active par une matrice de translation de vecteur (x,y,z).
- **void glRotatef(float theta,float x,float y,float z);**
- Cette fonction multiplie la matrice active par une matrice de rotation d'angle θ autour de l'axe passant par l'origine et porté par le vecteur (x,y,z).
- **void glScalef(float hx,float hy,float hz);**
- Cette fonction multiplie la matrice active par une matrice d'homothétie dont les facteurs suivant les axes X, Y et Z sont respectivement hx, hy et hz.

Matrice Modelview

- De l'objet à l'image, les coordonnées subissent un certain nombre de transformations géométriques (rotation, translation, perspective). Toutes sont modélisées par des matrices 4x4 opérant sur les coordonnées homogènes des points (x,y,z,w).
- OpenGL gère 3 piles de matrices, sélectionnées par **glMatrixMode** avec les arguments: **GL_ModelView**, **GL_Projection** et **GL_Texture**.
 - ModelView passe de l'objet au point de vue,
 - Projection effectue la perspective,
 - Texture opère sur les coordonnées des textures.
- Ces transformations sont directement appliquées par OpenGL lors de la visualisation

Matrice Modelview

- La méthode générale pour afficher chaque objet (rendu) est la suivante :
 - préparer la matrice "**modelview**" chargée de placer correctement l'objet dans la scène (basiquement : une translation, une rotation et une mise à la bonne taille).
 - définir l'objet (ensemble de facettes dont on donne les sommets et les arêtes): Les sommets sont donnés par rapport à l'origine du repère, "modelview" se chargeant de transformer leurs coordonnées et donc du placement dans la scène.
 - l'opération de projection de la scène sur l'écran (par exemple une projection perspective) la matrice « **projection** ».

Les transformations par objet

- Le chargement et l'initialisation de la matrice de Modelview:
 - `glMatrixMode(GL_MODELVIEW);`
 - `glLoadIdentity();`
- La matrice active à un moment donné est la première de la pile.
- `glLoadMatrix`, `glLoadIdentity`, `glMultMatrix`, `glScale`, `glTranslate`, `glRotate` modifient la matrice active.
- `glPushMatrix` `glPopMatrix` manipulent la pile.
- `glPushMatrix()` & `glPopMatrix()`: pour isoler les transformations appliquées à des objets différents.

Exemple

- Exemple: effectuer une transformation composée d'une translation de vecteur $(0,1,0)$ suivie d'une rotation d'angle 45° autour de l'axe Z.
- En notant M_t la matrice de translation et M_r la matrice de rotation, la matrice résultante est le produit de $M_r.M_t$.
- L'ordre d'écriture des matrices est inversé par rapport à l'ordre d'application des transformations.
- La portion de code OpenGL pour accomplir cette tâche est la suivante :

```
glLoadIdentity();
```

```
glRotatef(45.0,0.0,0.0,1.0);
```

```
glTranslatef(0.0,1.0,0.0);
```

- A priori, on ne sait pas ce qui se trouve dans la matrice, et donc avant d'effectuer la moindre opération, il convient de la réinitialiser.

Exemple

- Exemple de composition de transformation:
- L'objectif est de dessiner un carré 2D centré à l'origine, auquel on a appliqué une rotation d'angle 'a' autour de l'axe Z, suivie d'une translation de vecteur (0.5,0.0,0.0) et d'une rotation d'angle 'b' autour de Z.

Exemple

- Le code correspondant:

```
glLoadIdentity();  
glRotatef(b,0.0,0.0,1.0);  
glTranslatef(0.5,0.0,0.0);  
glRotatef(a,0.0,0.0,1.0);  
  
glBegin(GL_POLYGON);  
glVertex3f(-0.2,-0.2, 0.0);  
glVertex3f( 0.2,-0.2, 0.0);  
glVertex3f( 0.2, 0.2, 0.0);  
glVertex3f(-0.2, 0.2, 0.0);  
glEnd();
```

Exercice

- 1- Modélisation de la scène.
- 2- Tester (pas en même temps) les trois translations de valeur 0.5 respectivement suivant les axes X, Y, Z.
- 3- Tester les trois Rotations d'angle 90° du teapot (pas en même temps) respectivement suivant les axes X, Y, Z.
- 4- Résultat de la transformation
RoT et ToR pour l'objet teapot
- 5- Ecrire une fonction clavier effectuant une rotation de 5° autour de l'axe Y à chaque frappe de la touche :
 - 'a': rotation directe
 - 'b': rotation indirecte

Les objets 3D en opengl

Objet

Sphère

Rayon
Nombre sections
Nombre coupes

```
glutWireSphere(double r,  
                int ns,  
                int nc);  
glutSolidSphere(double r),  
                int ns,  
                int nc);
```

Cube

Coté

```
glutWireCube(double c);  
glutSolidCube(double c);
```

Tore

Rayon intérieur
Rayon extérieur
Nombre côtés
Nombre anneaux

```
glutWireTorus(double rint,  
              double rext,  
              int ns,  
              int nr);  
glutSolidTorus(double rint,  
              double rext,  
              int ns,  
              int nr));
```

Cône

Rayon
Hauteur
Nombre sections
Nombre coupes

```
glutWireCone(double r,  
             double h,  
             int ns,  
             int nc);  
glutSolidCone(double r,  
             double h,  
             int ns,  
             int nc);
```

Les objets 3D en opengl

Tétraèdre
4 faces

```
glutWireTetrahedron(void) ;  
glutSolidTetrahedron(void) ;
```

Octaèdre
8 faces

```
glutWireOctahedron(void) ;  
glutSolidOctahedron(void) ;
```

Icosaèdre
12 faces

```
glutWireIcosahedron(void) ;  
glutSolidIcosahedron(void) ;
```

Dodécaèdre
20 faces

```
glutWireDodecahedron(void) ;  
glutSolidDodecahedron(void) ;
```

Teapot
Rayon

```
glutWireTeapot(double s) ;  
glutSolidTeapot(double s) ;
```