

Modélisation et Programmation 3D

Introduction, rappel sur les vecteurs, projection, transformation ...

Cours du 16/01/2017

Plan

- Introduction
- Vecteurs
- Matrices
- Projections
- Transformations
- Processus de gestion des objets 3D
- Scène 3D en *OpenGL*

Plan

- Introduction
- Vecteurs
- Matrices
- Projections
- Transformations
- Processus de gestion des objets 3D
- Scène 3D en *OpenGL*

Introduction

- La 3D est présente de partout de nos jours :
 - de nombreux types de ``3D''

Introduction

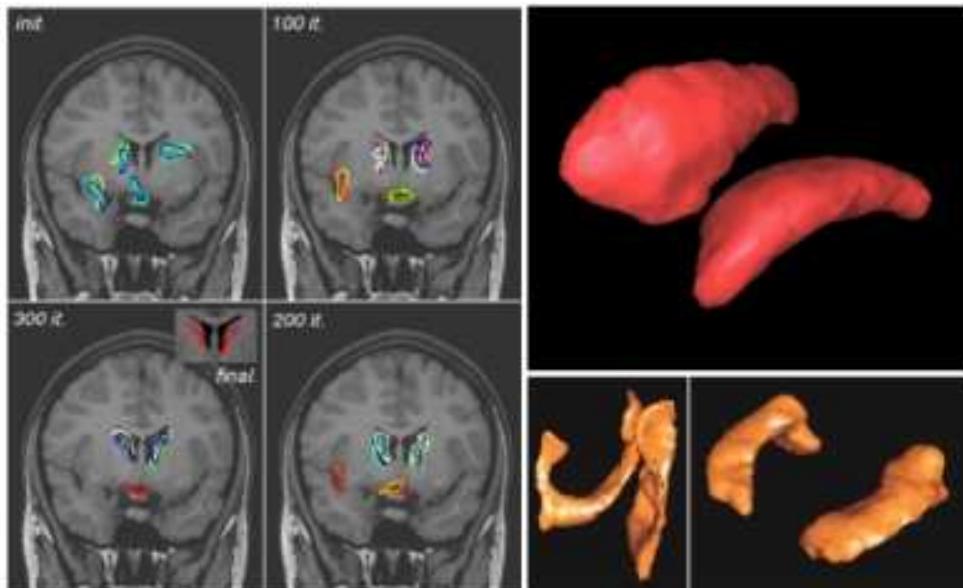
- La 3D est présente de partout de nos jours :
 - de nombreux types de ``3D''
 - la 3D au cinéma



➤ couple d'images 2D,
pas de la ``vraie 3D''

Introduction

→ la 3D dans le domaine médicale



- principalement des images de 2D de “coupe”
- des représentations 3D peuvent en être déduites

Introduction

→ la modélisation



- définition 3D des objets (coordonnées x,y,z) :
 - Jeu vidéo, film d'animation, conception assistée par ordinateur ...

Introduction

→ Représentation 3D possibles :

Modèles 3D

Introduction

→ Représentation 3D possibles :

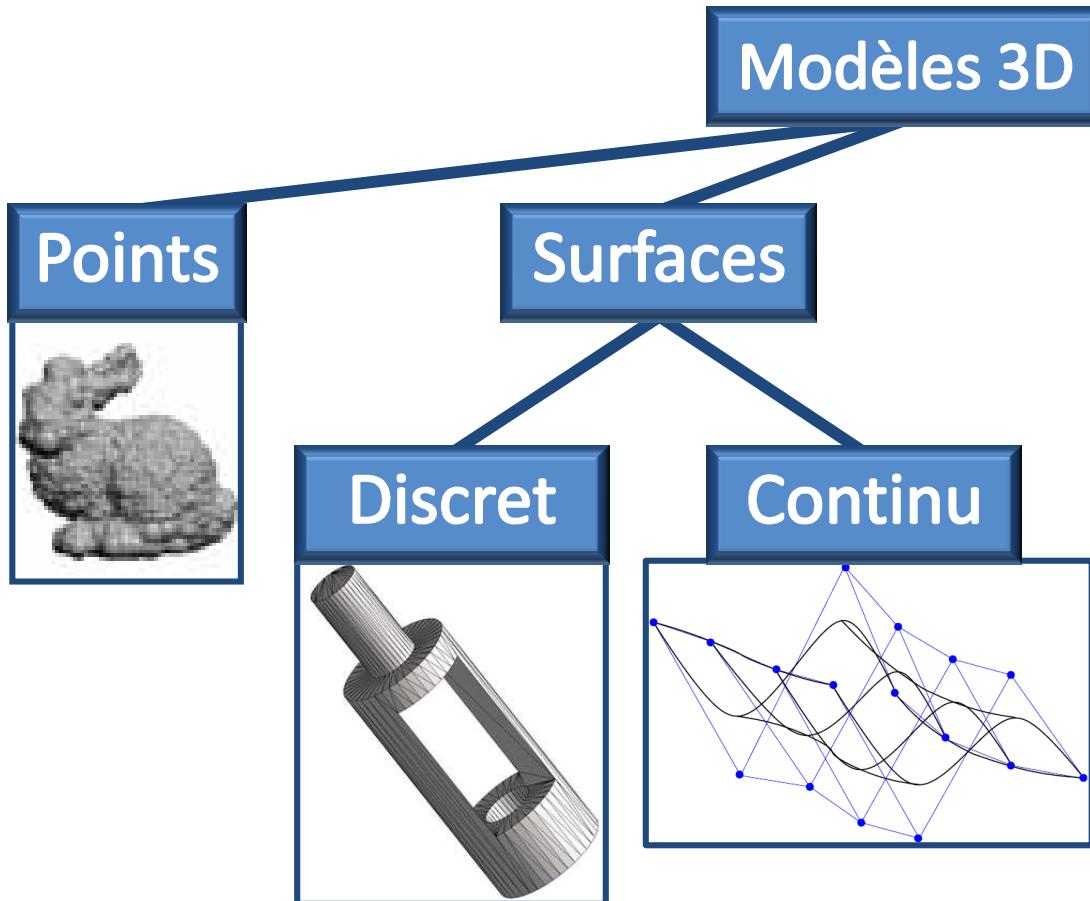
Modèles 3D

Points



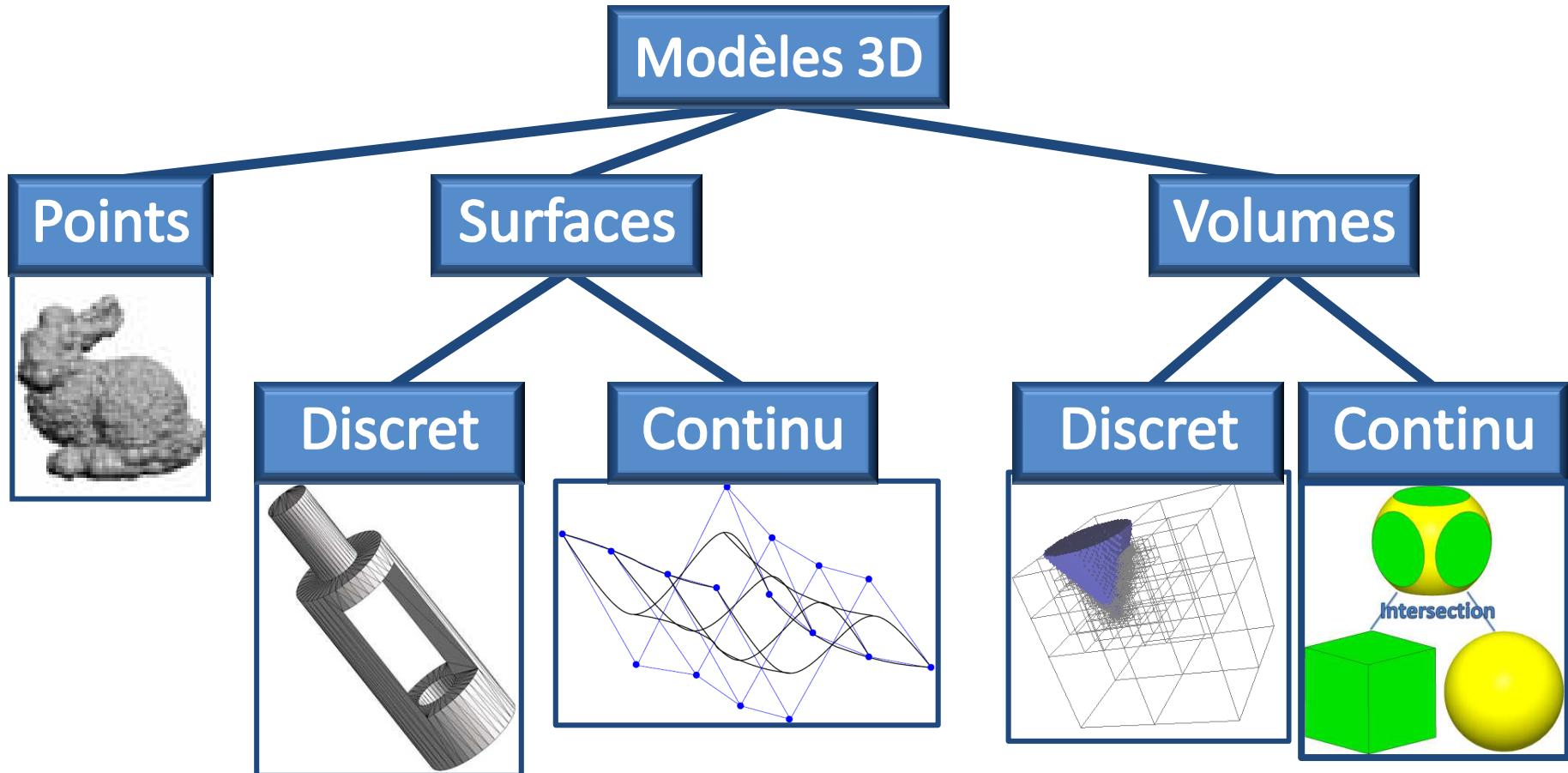
Introduction

→ Représentation 3D possibles :



Introduction

→ Représentation 3D possibles :



Plan

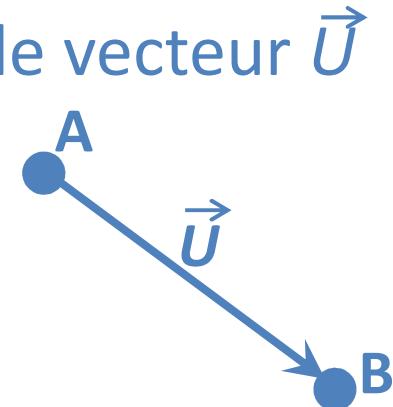
- Introduction
- Vecteurs
- Matrices
- Projections
- Transformations
- Processus de gestion des objets 3D
- Scène 3D en *OpenGL*

Rappel Vecteurs

- **Un vecteur :**

- définit une direction dans l'espace
- est défini par 3 coordonnées, soit le vecteur \vec{U} entre A et B :

$$\begin{aligned} U_x &= B_x - A_x \\ U_y &= B_y - A_y \\ U_z &= B_z - A_z \end{aligned}$$

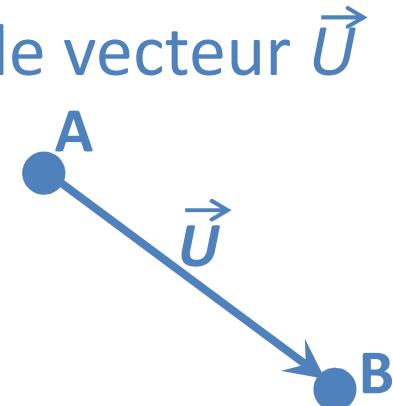


Rappel Vecteurs

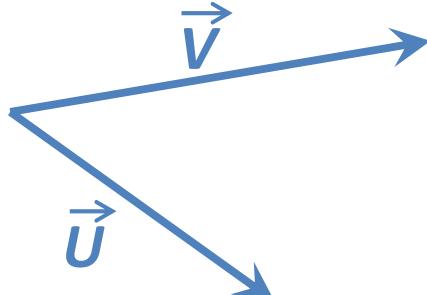
- **Un vecteur :**

- définit une direction dans l'espace
- est défini par 3 coordonnées, soit le vecteur \vec{U} entre A et B :

$$\begin{aligned} U_x &= B_x - A_x \\ U_y &= B_y - A_y \\ U_z &= B_z - A_z \end{aligned}$$



- Addition et Multiplication par scalaire

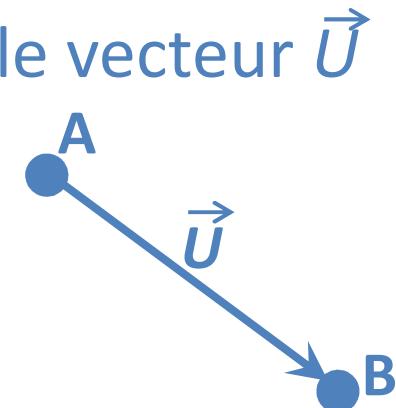


Rappel Vecteurs

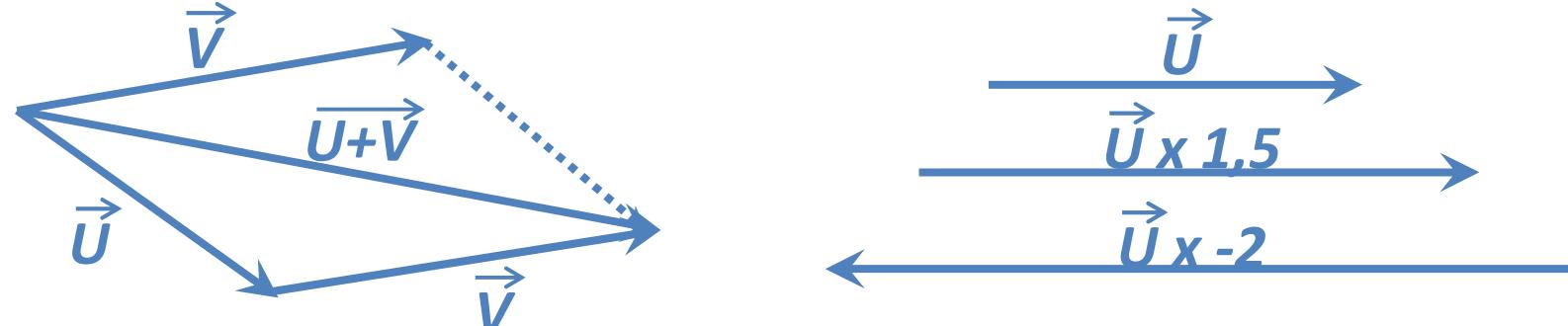
- **Un vecteur :**

- définit une direction dans l'espace
- est défini par 3 coordonnées, soit le vecteur \vec{U} entre A et B :

$$\begin{aligned} U_x &= B_x - A_x \\ U_y &= B_y - A_y \\ U_z &= B_z - A_z \end{aligned}$$



- Addition et Multiplication par scalaire



Rappel Vecteurs

- Opération sur les vecteurs :

- Norme d'un vecteur

- définit par :

$$|U| = \sqrt{U_x^2 + U_y^2 + U_z^2}$$

- si $\vec{U} = \vec{AB}$, alors $|\vec{U}|$ est la distance entre A et B
 - si $|\vec{U}| = 1$ alors \vec{U} est dit **normé**
 - pour tout vecteur non nul il existe un vecteur normé de même direction. Pour normé U :

$$U_x = U_x / |U|$$

$$U_y = U_y / |U|$$

$$U_z = U_z / |U|$$

Rappel Vecteurs

- Opération sur les vecteurs :
 - Produit scalaire entre 2 vecteurs

- définit par :

$$\vec{U} \cdot \vec{V} = U_x \times V_x + U_y \times V_y + U_z \times V_z$$

Rappel Vecteurs

- Opération sur les vecteurs :

- Produit scalaire entre 2 vecteurs

- définit par :

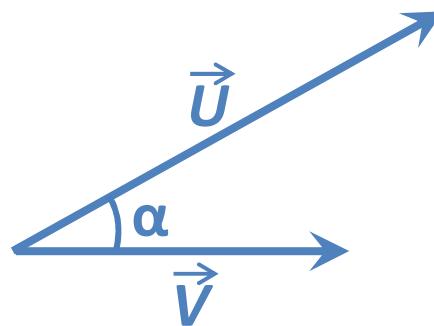
$$\vec{U} \cdot \vec{V} = U_x \times V_x + U_y \times V_y + U_z \times V_z$$

- Propriétés :

- symétrie $\rightarrow \vec{U} \cdot \vec{V} = \vec{V} \cdot \vec{U}$
 - distributivité $\rightarrow \vec{U} \cdot (\vec{V} + \vec{W}) = \vec{U} \cdot \vec{V} + \vec{U} \cdot \vec{W}$
 - homogénéité $\rightarrow (\lambda \vec{U}) \cdot \vec{V} = \lambda (\vec{U} \cdot \vec{V})$
 - lien avec la norme $\rightarrow \vec{U} \cdot \vec{U} = |\vec{U}|^2$

Rappel Vecteurs

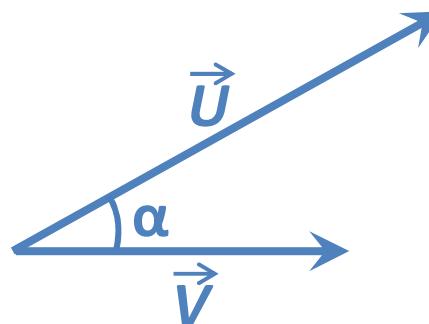
- Opération sur les vecteurs :
 - Produit scalaire entre 2 vecteurs
 - lien entre le scalaire et l'angle entre les vecteurs, soit
α l'angle entre U et V alors :



$$\vec{U} \cdot \vec{V} = \cos(\alpha) \times |\vec{U}| \times |\vec{V}|$$

Rappel Vecteurs

- Opération sur les vecteurs :
 - Produit scalaire entre 2 vecteurs
 - lien entre le scalaire et l'angle entre les vecteurs, soit α l'angle entre U et V alors :



$$\vec{U} \cdot \vec{V} = \cos(\alpha) \times |\vec{U}| \times |\vec{V}|$$

- si $\vec{U} \cdot \vec{V} = 0$ $\rightarrow \alpha = 90^\circ$ ou 270°
 $\rightarrow \vec{U}$ et \vec{V} sont dits **orthogonaux**

Rappel Vecteurs

- Opération sur les vecteurs :
 - Produit vectoriel entre 2 vecteurs
 - définit par :

$$\vec{U} \wedge \vec{V} = \begin{vmatrix} U_y \times V_z - U_z \times V_y \\ U_z \times V_x - U_x \times V_z \\ U_x \times V_y - U_y \times V_x \end{vmatrix}$$

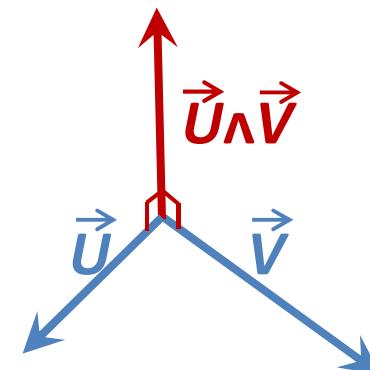
Rappel Vecteurs

- Opération sur les vecteurs :
 - Produit vectoriel entre 2 vecteurs

- définit par :

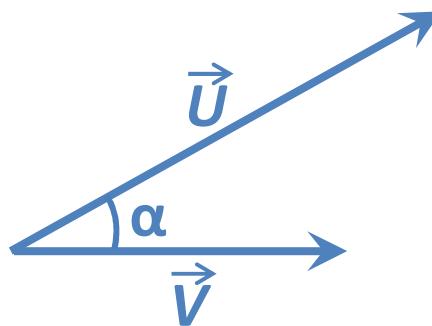
$$\vec{U} \wedge \vec{V} = \begin{vmatrix} U_y \times V_z - U_z \times V_y \\ U_z \times V_x - U_x \times V_z \\ U_x \times V_y - U_y \times V_x \end{vmatrix}$$

- $\vec{U} \wedge \vec{V}$ est un vecteur orthogonal à la fois à \vec{U} et à \vec{V} ,
 - utile pour calculer des normales, des repères orthonormés ...



Rappel Vecteurs

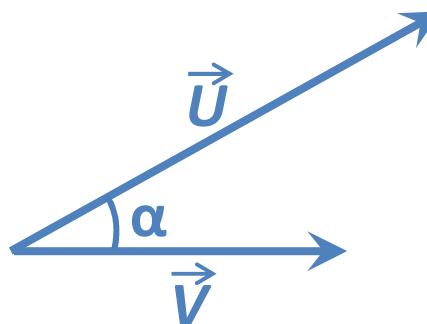
- Opération sur les vecteurs :
 - Produit vectoriel entre 2 vecteurs
 - lien entre le vectoriel et l'angle entre les vecteurs, soit α l'angle entre U et V alors :



$$|\vec{U} \wedge \vec{V}| = \sin(\alpha) \times |\vec{U}| \times |\vec{V}|$$

Rappel Vecteurs

- Opération sur les vecteurs :
 - Produit vectoriel entre 2 vecteurs
 - lien entre le vectoriel et l'angle entre les vecteurs, soit α l'angle entre U et V alors :



$$|\vec{U} \wedge \vec{V}| = \sin(\alpha) \times |\vec{U}| \times |\vec{V}|$$

- si $\vec{U} \wedge \vec{V} = 0$ $\rightarrow \alpha = 0^\circ$ ou 180°
 $\rightarrow \vec{U}$ et \vec{V} sont dits **colinéaires**

Plan

- Introduction
- Vecteurs
- Matrices
- Projections
- Transformations
- Processus de gestion des objets 3D
- Scène 3D en *OpenGL*

Rappel Matrices

- Une matrice est :
 - un ensemble de vecteurs :

$$\begin{array}{|c c c c|} \hline a_0 & a_1 & a_2 & \dots \\ \hline b_0 & b_1 & b_2 & \dots \\ \hline c_0 & c_1 & c_2 & \dots \\ \hline \dots & & & \\ \hline \end{array}$$

Rappel Matrices

- Une matrice est :
 - un ensemble de vecteurs :

$$\begin{array}{|c c c c|} \hline a_0 & a_1 & a_2 & \dots \\ b_0 & b_1 & b_2 & \dots \\ c_0 & c_1 & c_2 & \dots \\ \hline \end{array}$$

- Opérations sur les matrices :

- addition

$$\begin{vmatrix} a_0 & a_1 \\ b_0 & b_1 \end{vmatrix} + \begin{vmatrix} t_0 & t_1 \\ u_0 & u_1 \end{vmatrix} = \begin{vmatrix} a_0 + t_0 & a_1 + t_1 \\ b_0 + u_0 & b_1 + u_1 \end{vmatrix}$$

Rappel Matrices

- Une matrice est :
 - un ensemble de vecteurs :

$$\begin{array}{|c c c c|} \hline a_0 & a_1 & a_2 & \dots \\ \hline b_0 & b_1 & b_2 & \dots \\ \hline c_0 & c_1 & c_2 & \dots \\ \hline \end{array}$$

- Opérations sur les matrices :

- addition

$$\begin{vmatrix} a_0 & a_1 \\ b_0 & b_1 \end{vmatrix} + \begin{vmatrix} t_0 & t_1 \\ u_0 & u_1 \end{vmatrix} = \begin{vmatrix} a_0 + t_0 & a_1 + t_1 \\ b_0 + u_0 & b_1 + u_1 \end{vmatrix}$$

- multiplication par un scalaire

$$\lambda \times \begin{vmatrix} a_0 & a_1 \\ b_0 & b_1 \end{vmatrix} = \begin{vmatrix} \lambda \times a_0 & \lambda \times a_1 \\ \lambda \times b_0 & \lambda \times b_1 \end{vmatrix}$$

Rappel Matrices

- Une matrice est :

➤ un ensemble de vecteurs :

$$\begin{vmatrix} a_0 & a_1 & a_2 & \dots \\ b_0 & b_1 & b_2 & \dots \\ c_0 & c_1 & c_2 & \dots \\ \vdots & & & \end{vmatrix}$$

- Opérations sur les matrices :

➤ addition

$$\begin{vmatrix} a_0 & a_1 \\ b_0 & b_1 \end{vmatrix} + \begin{vmatrix} t_0 & t_1 \\ u_0 & u_1 \end{vmatrix} = \begin{vmatrix} a_0 + t_0 & a_1 + t_1 \\ b_0 + u_0 & b_1 + u_1 \end{vmatrix}$$

➤ multiplication

par un scalaire

$$\lambda \times \begin{vmatrix} a_0 & a_1 \\ b_0 & b_1 \end{vmatrix} = \begin{vmatrix} \lambda \times a_0 & \lambda \times a_1 \\ \lambda \times b_0 & \lambda \times b_1 \end{vmatrix}$$

➤ multiplication par une matrice

$$\begin{vmatrix} a_0 & a_1 \\ b_0 & b_1 \end{vmatrix} \times \begin{vmatrix} t_0 & t_1 \\ u_0 & u_1 \end{vmatrix} = \begin{vmatrix} a_0 \times t_0 + a_1 \times u_0 & a_0 \times t_1 + a_1 \times u_1 \\ b_0 \times t_0 + b_1 \times u_0 & b_0 \times t_1 + b_1 \times u_1 \end{vmatrix}$$

Plan

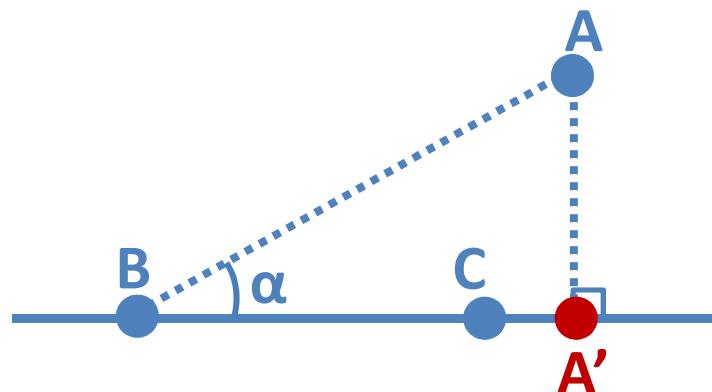
- Introduction
- Vecteurs
- Matrices
- Projections
- Transformations
- Processus de gestion des objets 3D
- Scène 3D en *OpenGL*

Projections

- D'un point sur une droite

➤ soit la droite (d) définie par 2 points B et C et A le point à projeter :

- par Pythagore $|\vec{BA}'| / |\vec{BA}| = \cos(\alpha)$
- or on vient de voir que $\cos(\alpha) = \vec{BA} \cdot \vec{BC} / (|\vec{BA}| \times |\vec{BC}|)$
- d'où :
$$|\vec{BA}'| = \vec{BA} \cdot \vec{BC} / |\vec{BC}|$$



Projections

- D'un point sur une droite

➤ soit la droite (d) définie par 2 points B et C et A le point à projeter :

- par Pythagore $|\vec{BA}'| / |\vec{BA}| = \cos(\alpha)$

- or on vient de voir que $\cos(\alpha) = \vec{BA} \cdot \vec{BC} / (|\vec{BA}| \times |\vec{BC}|)$

- d'où :

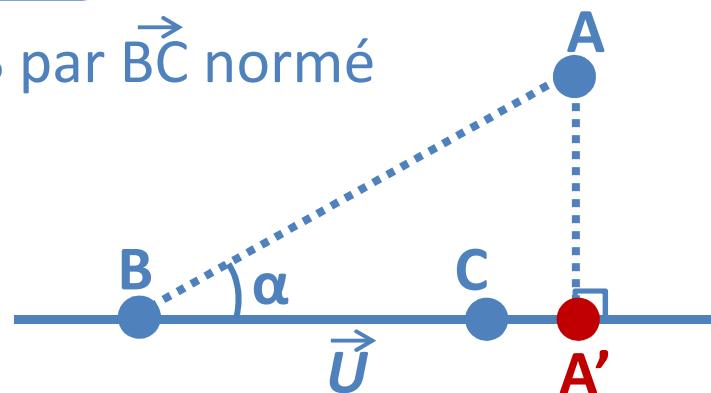
$$|\vec{BA}'| = \vec{BA} \cdot \vec{BC} / |\vec{BC}|$$

- il suffit alors de translater B par \vec{BC} normé avec une distance $|\vec{BA}'|$:

$$A'x = Bx + Ux \times |\vec{BA}'|$$

$$A'y = By + Uy \times |\vec{BA}'|$$

$$A'z = Bz + Uz \times |\vec{BA}'|$$

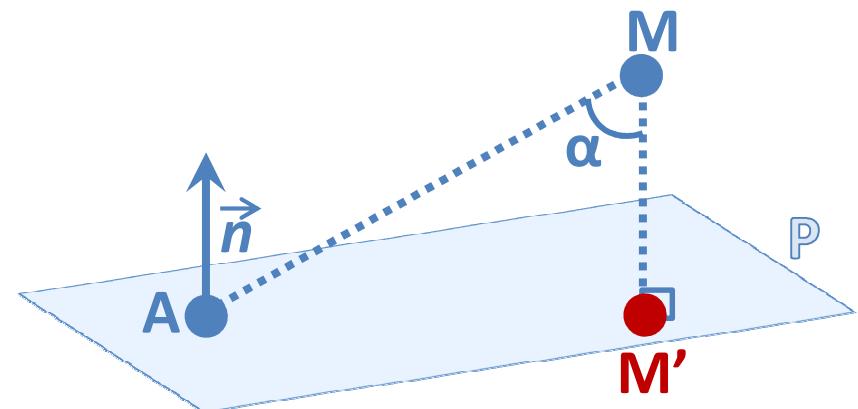


Projections

- D'un point sur un plan

➤ soit le plan P défini par un point A et une normale \vec{n} et M le point à projeter :

- par Pythagore $|\overrightarrow{MM'}| / |\overrightarrow{MA}| = \cos(\alpha)$
- or on vient de voir que $\cos(\alpha) = \overrightarrow{MA} \cdot \vec{n} / (|\overrightarrow{MA}| \times |\vec{n}|)$
- d'où : $|\overrightarrow{MM'}| = \overrightarrow{MA} \cdot \vec{n} / |\vec{n}|$



Projections

- D'un point sur un plan

➤ soit le plan P défini par un point A et une normale \vec{n} et M le point à projeter :

- o par Pythagore $|\overrightarrow{MM'}| / |\overrightarrow{MA}| = \cos(\alpha)$

- o or on vient de voir que $\cos(\alpha) = \overrightarrow{MA} \cdot \vec{n} / (|\overrightarrow{MA}| \times |\vec{n}|)$

- o d'où :

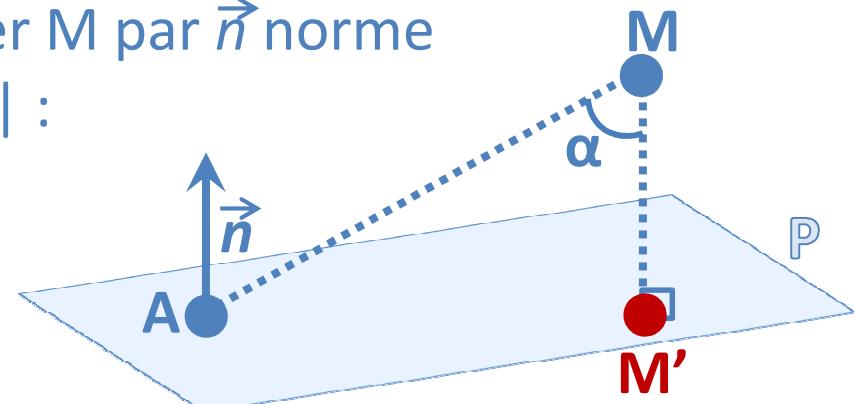
$$|\overrightarrow{MM'}| = \overrightarrow{MA} \cdot \vec{n} / |\vec{n}|$$

- o il suffit alors de translater M par \vec{n} norme avec une distance $|\overrightarrow{MM'}|$:

$$M'x = Mx - nx \times |\overrightarrow{MM'}|$$

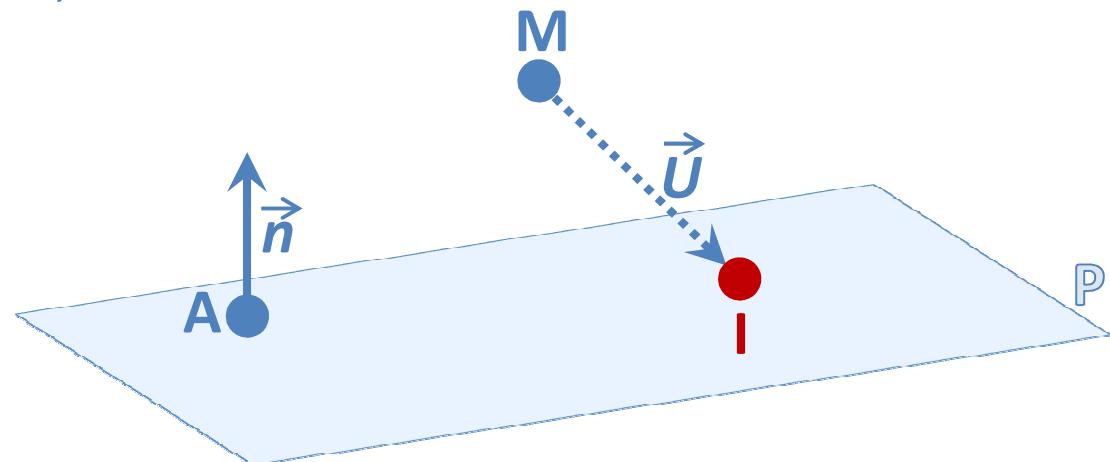
$$M'y = My - ny \times |\overrightarrow{MM'}|$$

$$M'z = Mz - nz \times |\overrightarrow{MM'}|$$



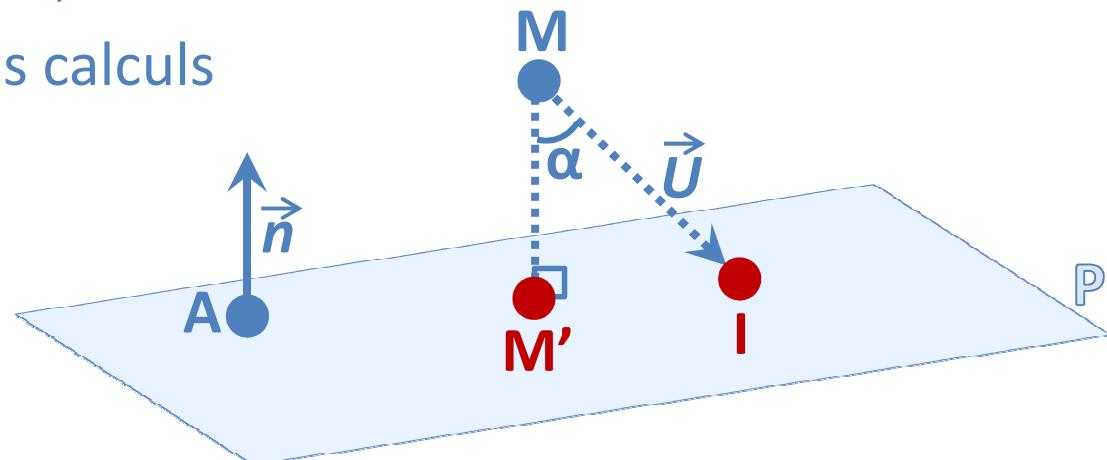
Projections

- Autres projections : non orthogonale
 - Soit un plan P défini par un point A et une normale \vec{n} et M le point à projeter selon la direction \vec{U} .
 - on cherche l'intersection I entre la droite définie par M et \vec{U} et le plan P,



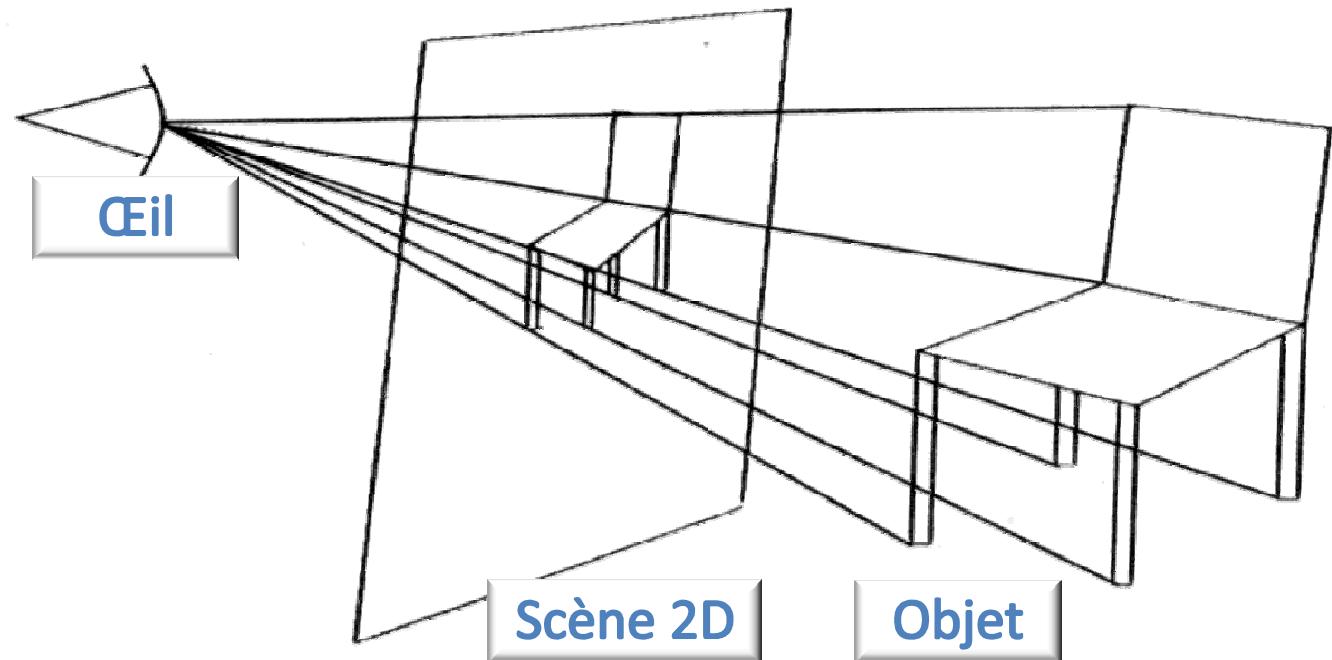
Projections

- Autres projections : non orthogonale
 - Soit un plan P défini par un point A et une normale \vec{n} et M le point à projeter selon la direction \vec{U} .
 - on cherche l'intersection I entre la droite définie par M et \vec{U} et le plan P ,
 - en utilisant les calculs précédents.



Projections

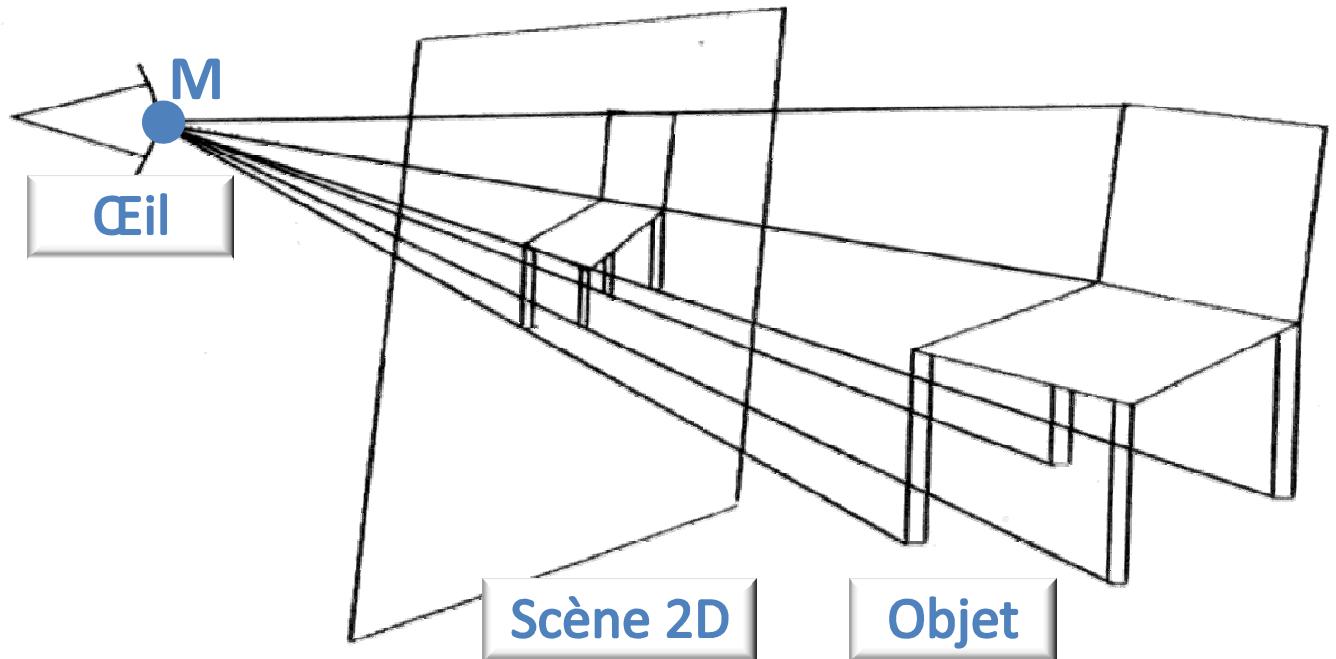
- Application à la visualisation 3D
 - Pour savoir comment on affiche un objet 3D sur un écran 2D, on calcule pour chaque point un point “projeté” sur la scène 2D avec :



Projections

- Application à la visualisation 3D
 - Pour savoir comment on affiche un objet 3D sur un écran 2D, on calcule pour chaque point un point "projeté" sur la scène 2D avec :

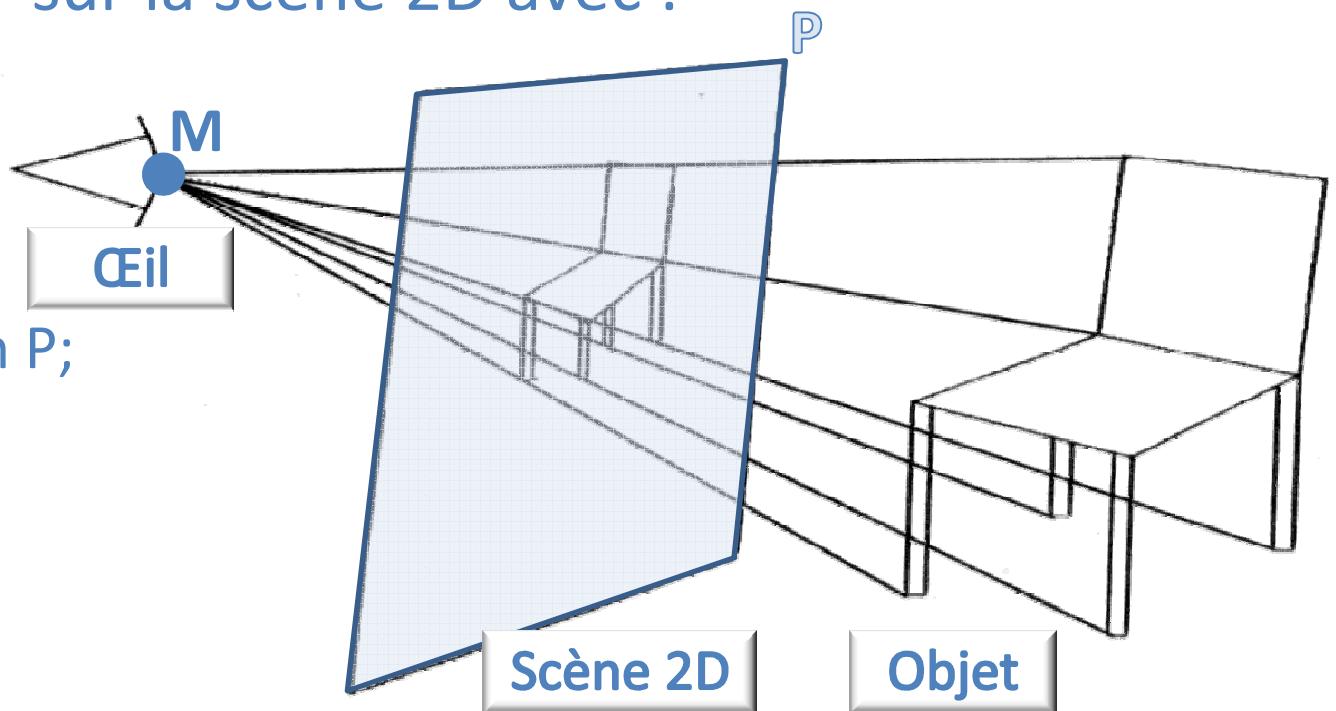
- l'œil -> le point M à projeter;



Projections

- Application à la visualisation 3D
 - Pour savoir comment on affiche un objet 3D sur un écran 2D, on calcule pour chaque point un point "projeté" sur la scène 2D avec :

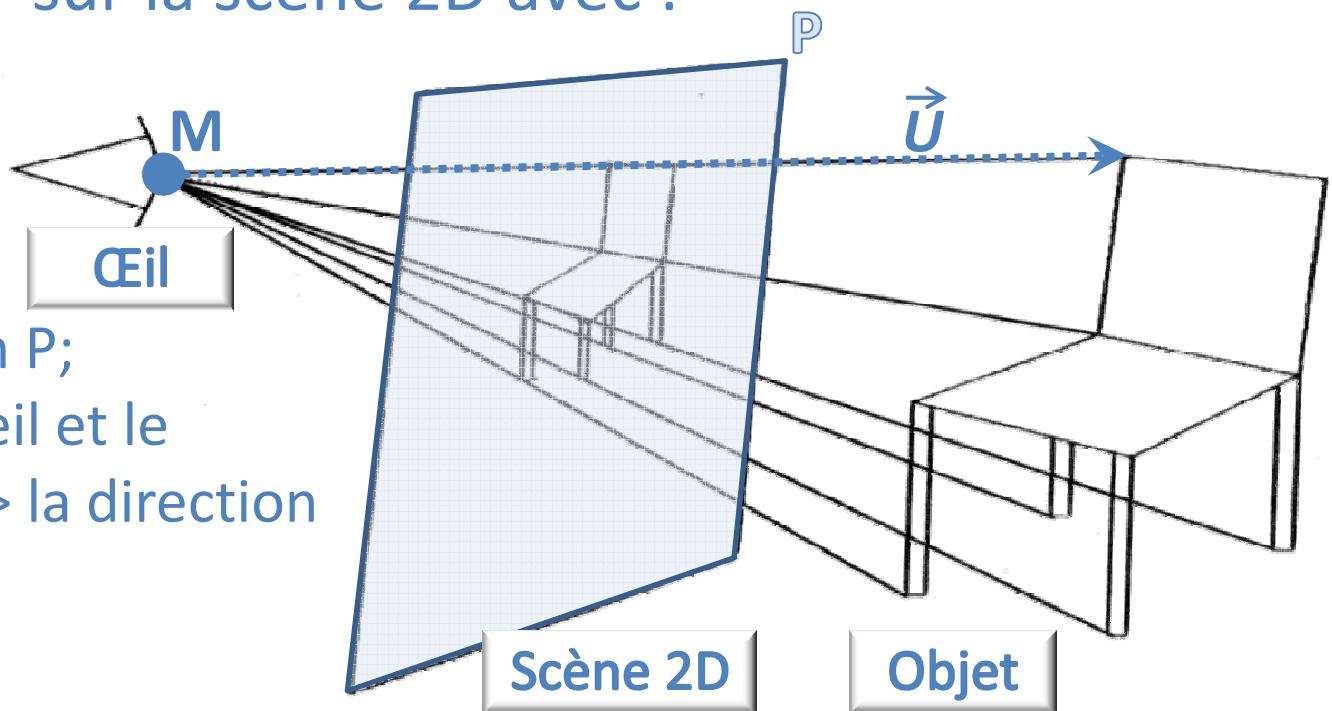
- l'œil -> le point M à projeter;
- la scène -> le plan P;



Projections

- Application à la visualisation 3D
 - Pour savoir comment on affiche un objet 3D sur un écran 2D, on calcule pour chaque point un point “projeté” sur la scène 2D avec :

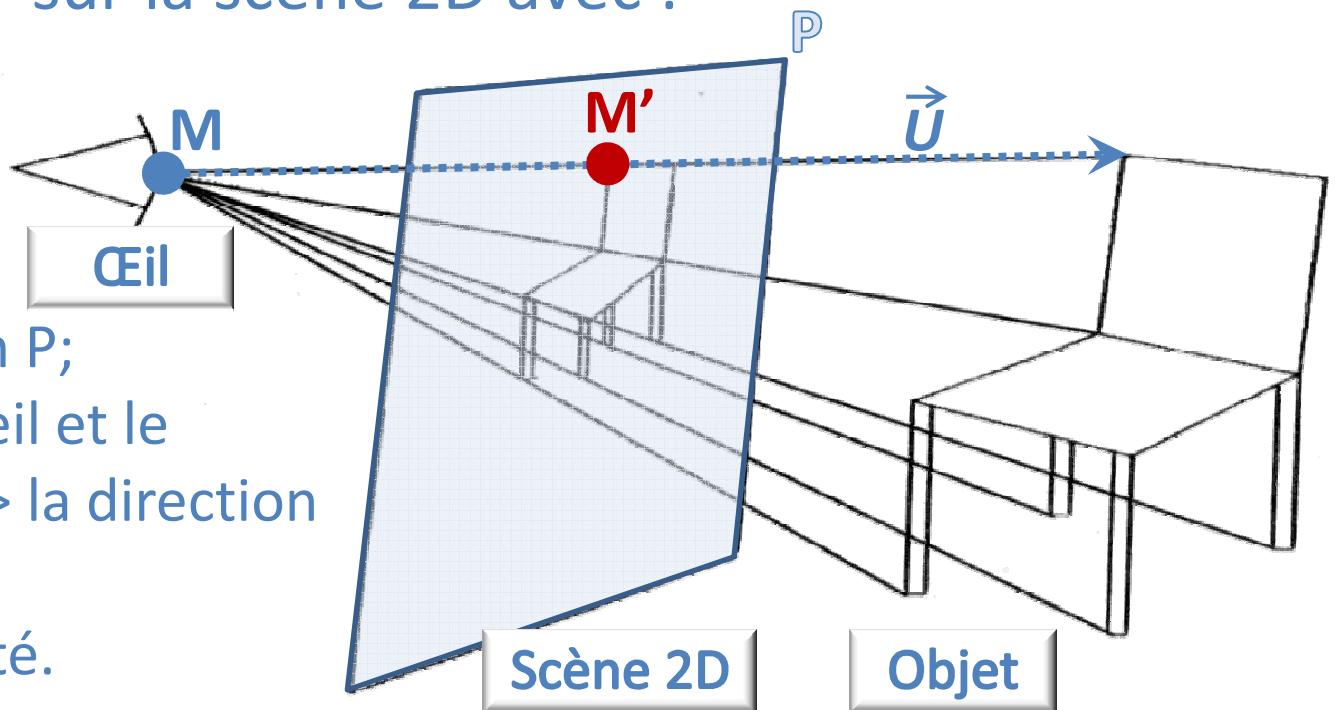
- l’œil -> le point M à projeter;
- la scène -> le plan P;
- la droite entre l’œil et le point de l’objet -> la direction \vec{U} de projection;



Projections

- Application à la visualisation 3D
 - Pour savoir comment on affiche un objet 3D sur un écran 2D, on calcule pour chaque point un point "projeté" sur la scène 2D avec :

- l'œil -> le point M à projeter;
- la scène -> le plan P;
- la droite entre l'œil et le point de l'objet -> la direction \vec{U} de projection;
- M' le point projeté.

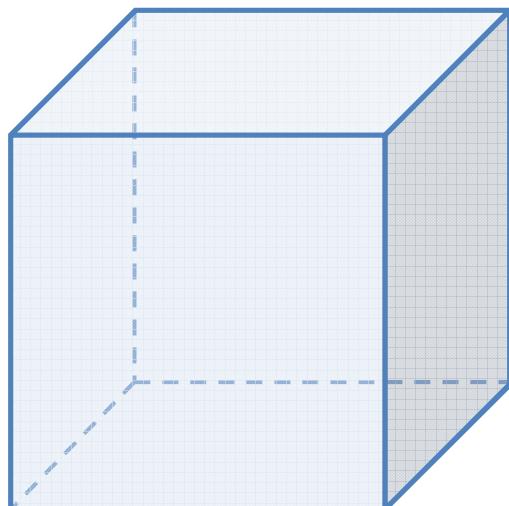


Plan

- Introduction
- Vecteurs
- Matrices
- Projections
- Transformations
- Processus de gestion des objets 3D
- Scène 3D en *OpenGL*

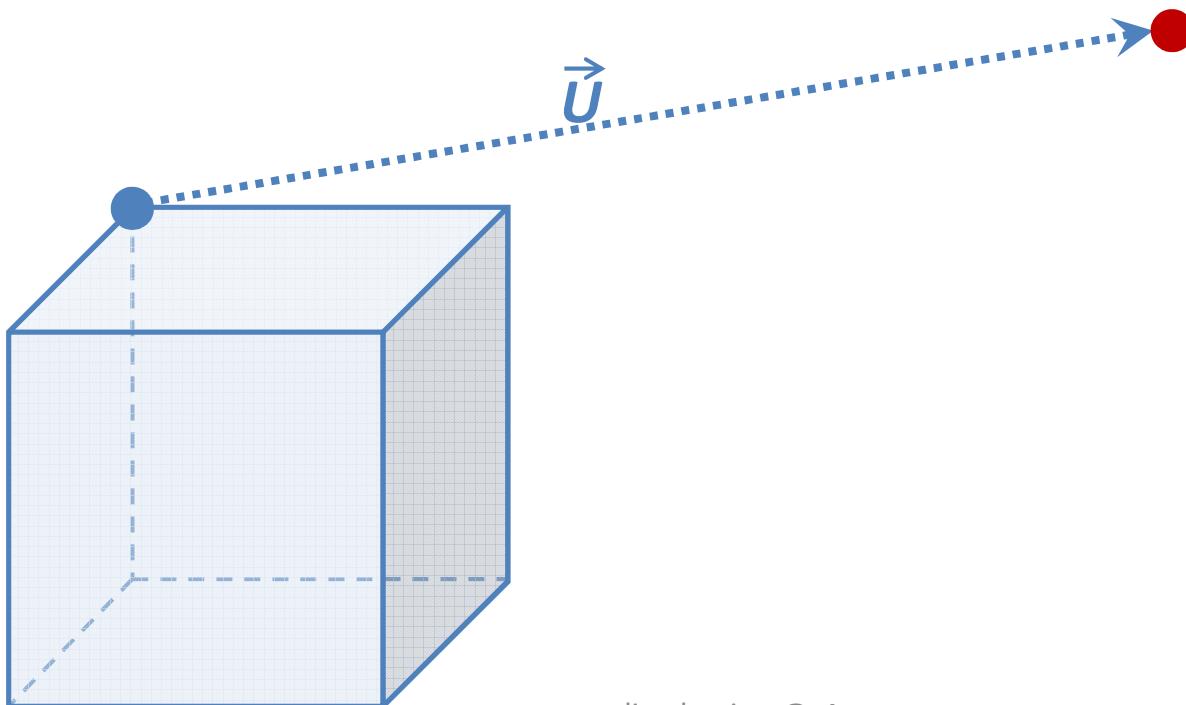
Transformations

- Translation d'un objet 3D
 - A partir d'un vecteur 3D \vec{U} :
 - on applique la translation à chaque point de l'objet,



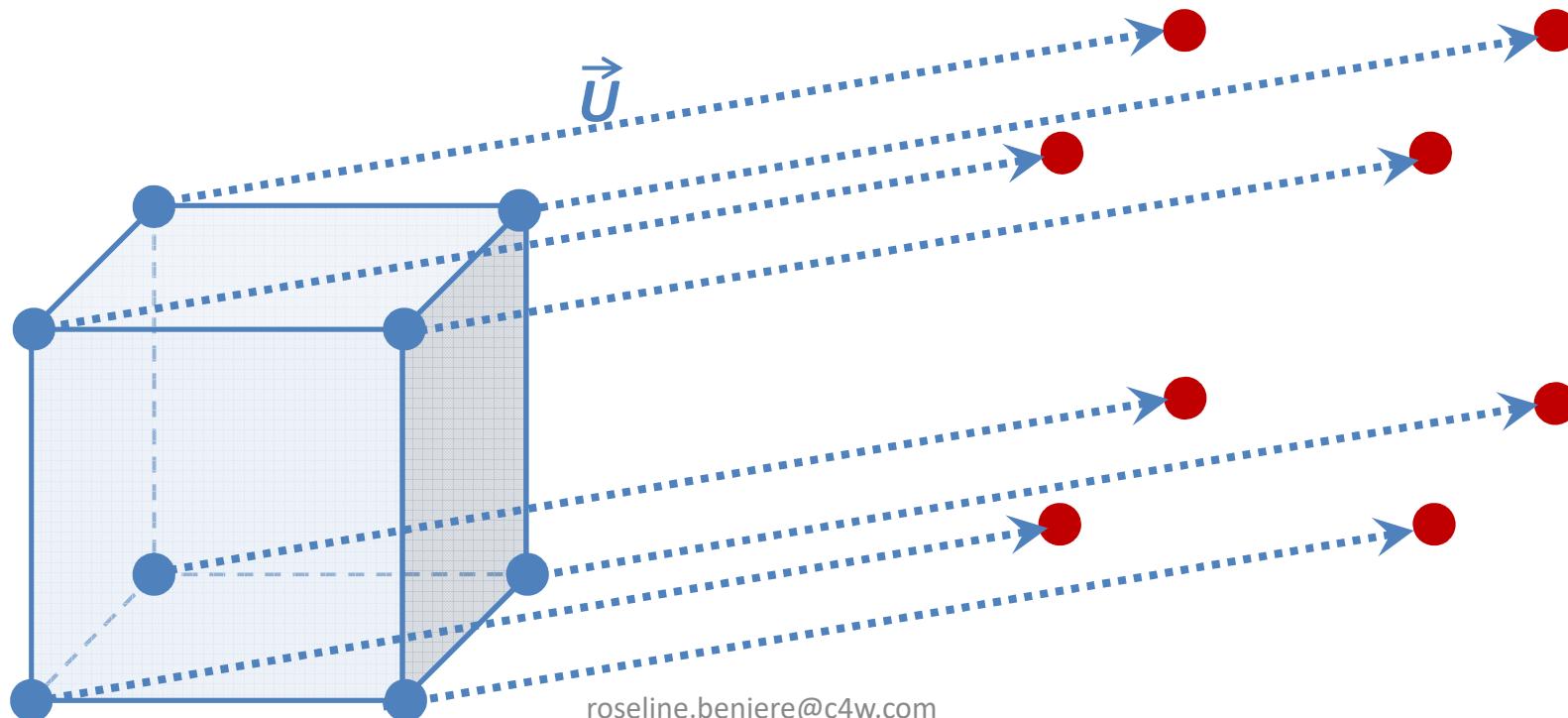
Transformations

- Translation d'un objet 3D
 - A partir d'un vecteur 3D \vec{U} :
 - on applique la translation à chaque point de l'objet,



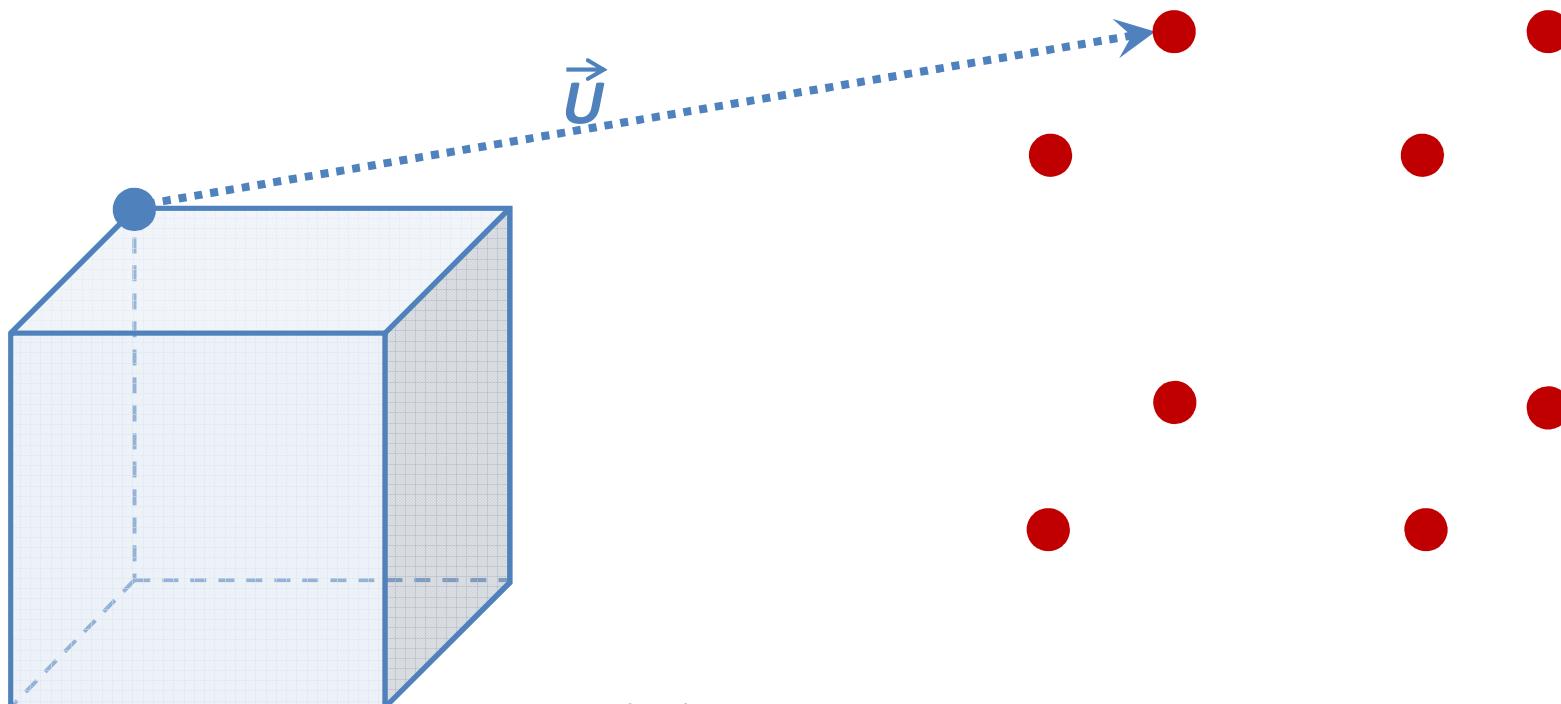
Transformations

- Translation d'un objet 3D
 - A partir d'un vecteur 3D \vec{U} :
 - on applique la translation à chaque point de l'objet,



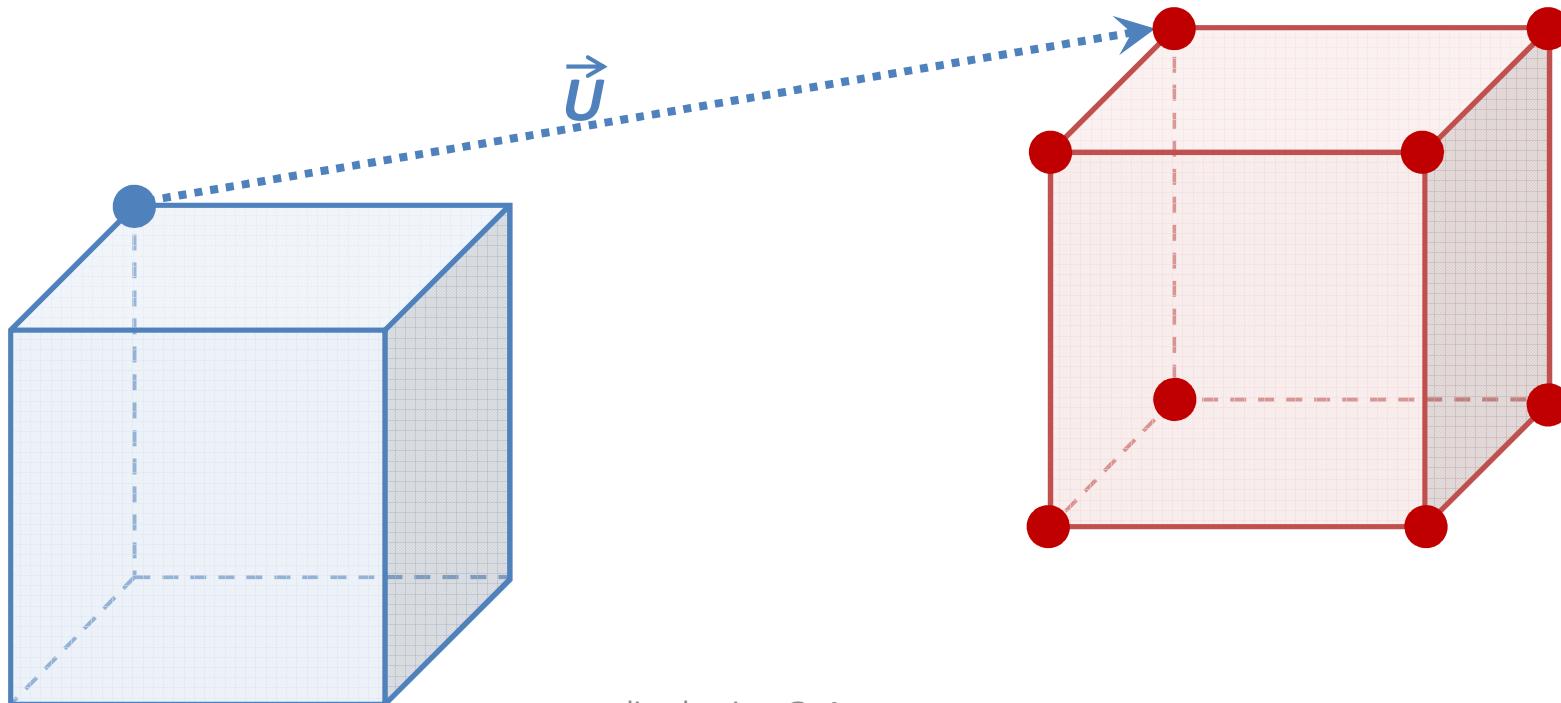
Transformations

- Translation d'un objet 3D
 - A partir d'un vecteur 3D \vec{U} :
 - o on applique la translation à chaque point de l'objet,
 - o les liaisons entre points restent les mêmes.



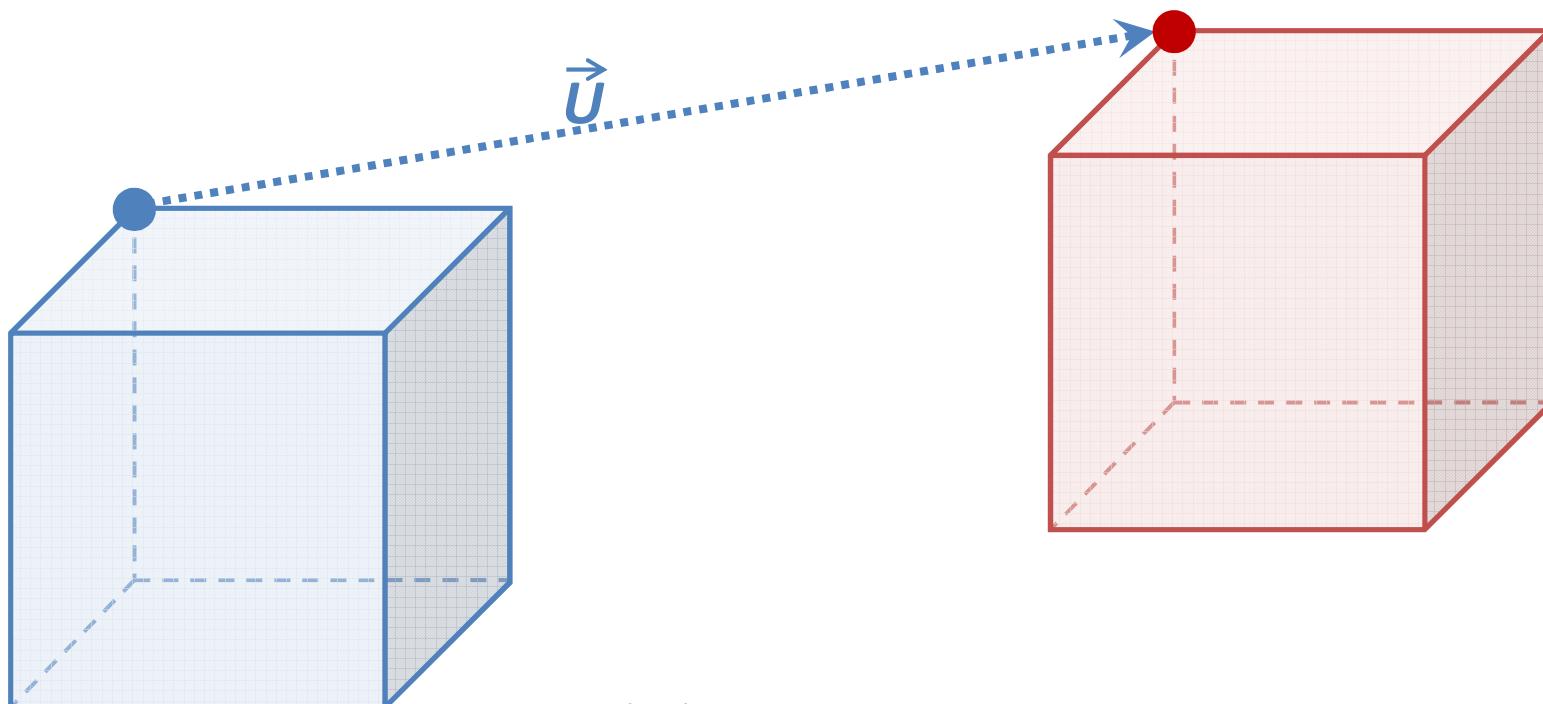
Transformations

- Translation d'un objet 3D
 - A partir d'un vecteur 3D \vec{U} :
 - o on applique la translation à chaque point de l'objet,
 - o les liaisons entre points restent les mêmes.



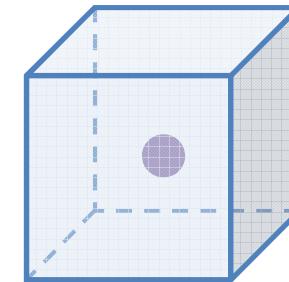
Transformations

- Translation d'un objet 3D
 - A partir d'un vecteur 3D \vec{U} :
 - o on applique la translation à chaque point de l'objet,
 - o les liaisons entre points restent les mêmes.



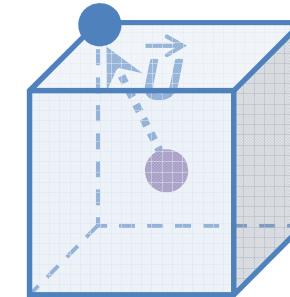
Transformations

- Mise à l'échelle d'un objet 3D
 - A partir d'un centre C et d'un ratio r :



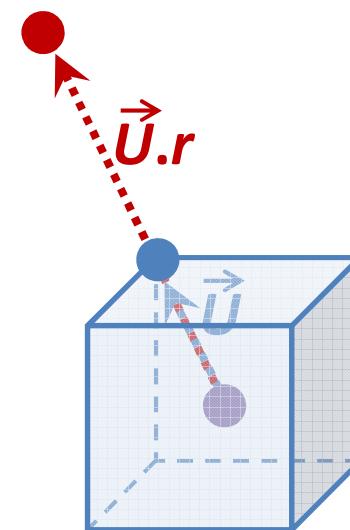
Transformations

- Mise à l'échelle d'un objet 3D
 - A partir d'un centre C et d'un ratio r :
 - pour chaque point, on calcule le vecteur entre le centre et ce point;



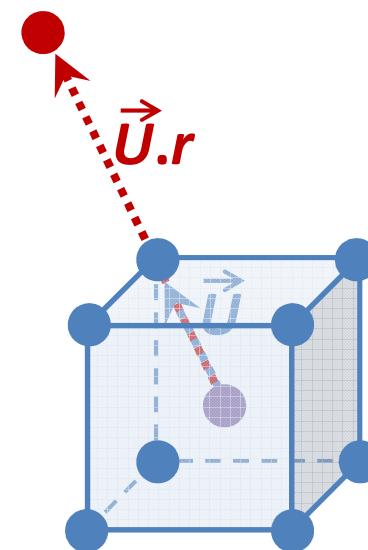
Transformations

- Mise à l'échelle d'un objet 3D
 - A partir d'un centre C et d'un ratio r :
 - pour chaque point, on calcule le vecteur entre le centre et ce point;
 - on translate le centre avec le vecteur multiplier par le ration;



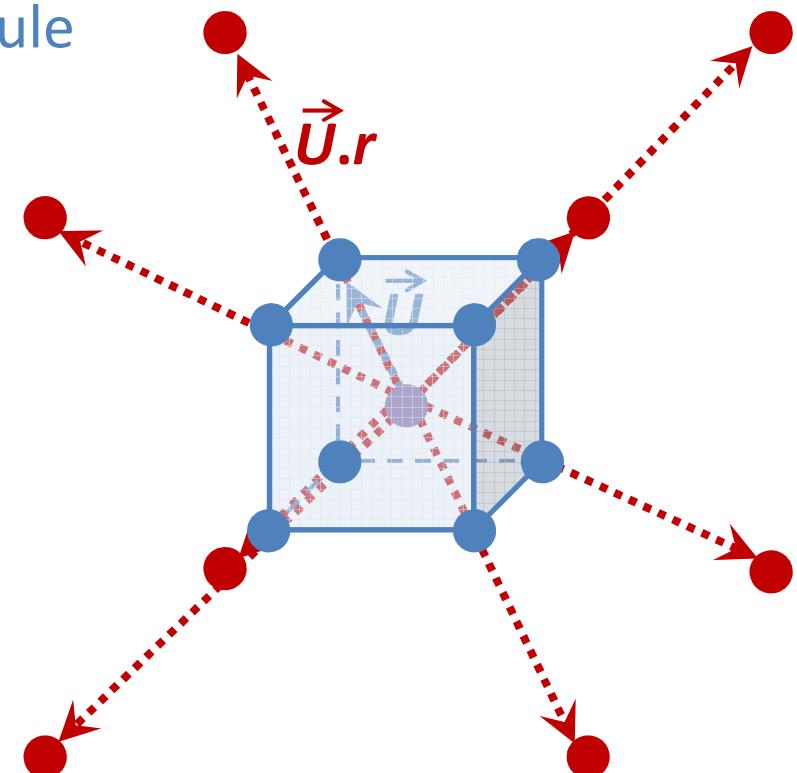
Transformations

- Mise à l'échelle d'un objet 3D
 - A partir d'un centre C et d'un ratio r :
 - pour chaque point, on calcule le vecteur entre le centre et ce point;
 - on translate le centre avec le vecteur multiplier par le ration;



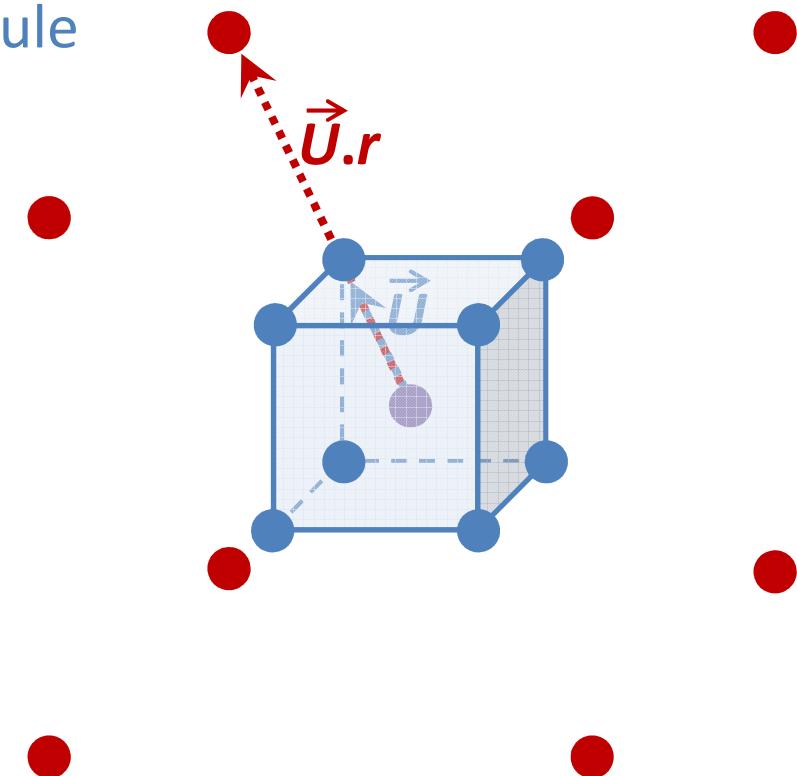
Transformations

- Mise à l'échelle d'un objet 3D
 - A partir d'un centre C et d'un ratio r :
 - pour chaque point, on calcule le vecteur entre le centre et ce point;
 - on translate le centre avec le vecteur multiplier par le ration;



Transformations

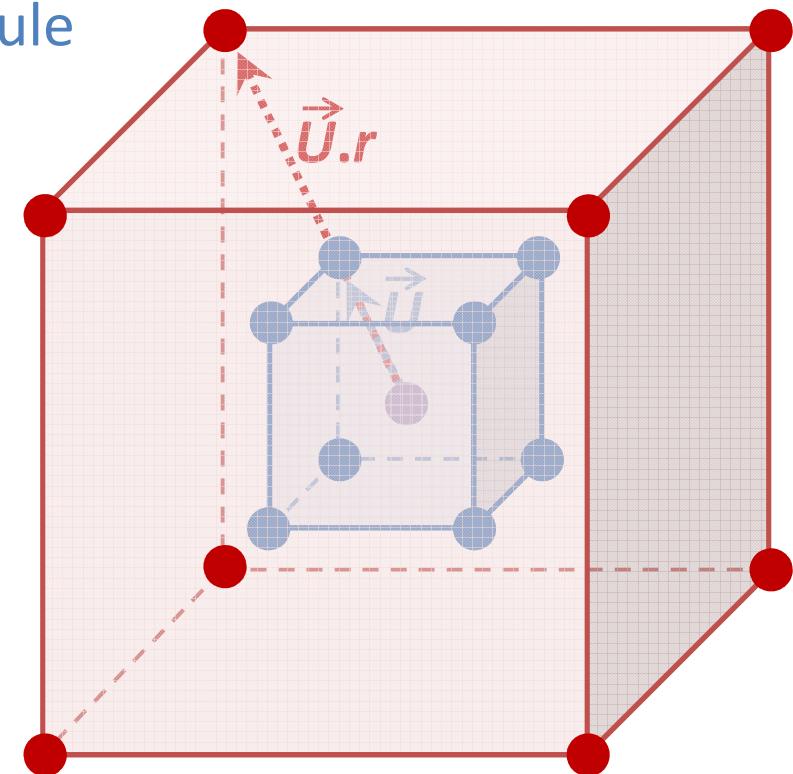
- Mise à l'échelle d'un objet 3D
 - A partir d'un centre C et d'un ratio r :
 - pour chaque point, on calcule le vecteur entre le centre et ce point;
 - on translate le centre avec le vecteur multiplier par le ration;
 - les liaisons entre les points restent les mêmes.



Transformations

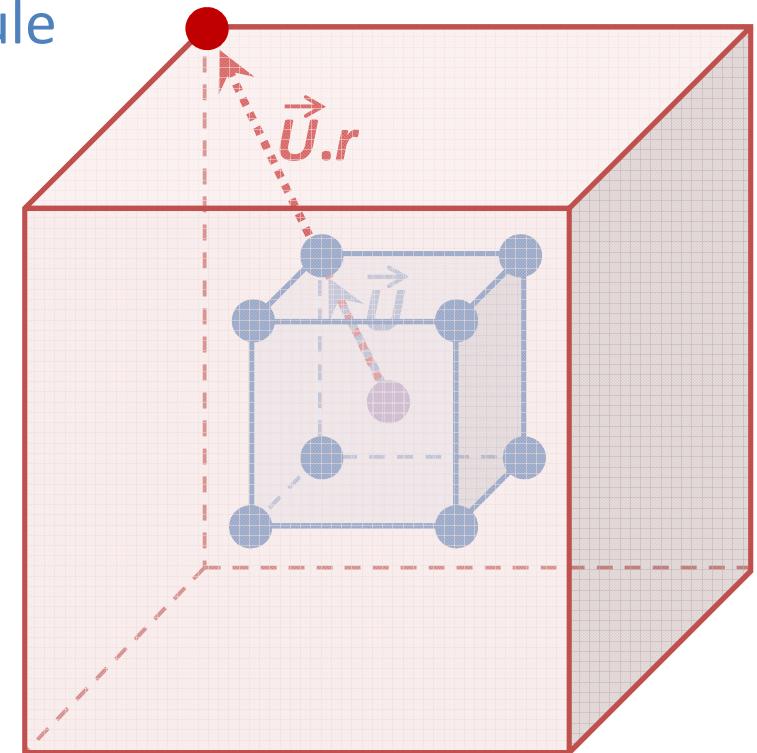
- Mise à l'échelle d'un objet 3D
 - A partir d'un centre C et d'un ratio r :

- pour chaque point, on calcule le vecteur entre le centre et ce point;
 - on translate le centre avec avec le vecteur multiplier par le ration;
 - les liaisons entre les points restent les mêmes.



Transformations

- Mise à l'échelle d'un objet 3D
 - A partir d'un centre C et d'un ratio r :
 - pour chaque point, on calcule le vecteur entre le centre et ce point;
 - on translate le centre avec avec le vecteur multiplier par le ration;
 - les liaisons entre les points restent les mêmes.



Transformations

- Rotation d'un objet 3D
 - A partir d'un axe \vec{A} et d'un angle α :
 - pour chaque point, on applique la matrice de rotation
 - les liaisons entre les points restent les mêmes.

Transformations

- Rotation d'un objet 3D

- A partir d'un axe \vec{A} et d'un angle α :
 - pour chaque point, on applique la matrice de rotation
 - les liaisons entre les points restent les mêmes.
- Matrices de rotation de base

- Atour de l'axe OX:

- $x' = x$
- $y' = y \cdot \cos(\alpha) - z \cdot \sin(\alpha)$
- $z' = y \cdot \sin(\alpha) + z \cdot \cos(\alpha)$

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{vmatrix}$$

Transformations

- Rotation d'un objet 3D

- A partir d'un axe \vec{A} et d'un angle α :
 - pour chaque point, on applique la matrice de rotation
 - les liaisons entre les points restent les mêmes.
- Matrices de rotation de base

- Atour de l'axe OX:

- $x' = x$
- $y' = y \cdot \cos(\alpha) - z \cdot \sin(\alpha)$
- $z' = y \cdot \sin(\alpha) + z \cdot \cos(\alpha)$

- Atour de l'axe OY:

- $x' = x \cdot \cos(\alpha) + z \cdot \sin(\alpha)$
- $y' = y$
- $z' = -x \cdot \sin(\alpha) + z \cdot \cos(\alpha)$

$$\begin{array}{|ccc|} \hline & 1 & 0 & 0 \\ & 0 & \cos(\alpha) & -\sin(\alpha) \\ & 0 & \sin(\alpha) & \cos(\alpha) \\ \hline \end{array}$$

$$\begin{array}{|ccc|} \hline & \cos(\alpha) & 0 & \sin(\alpha) \\ & 0 & 1 & 0 \\ & -\sin(\alpha) & 0 & \cos(\alpha) \\ \hline \end{array}$$

Transformations

- Rotation d'un objet 3D

➤ A partir d'un axe \vec{A} et d'un angle α :

- pour chaque point, on applique la matrice de rotation
- les liaisons entre les points restent les mêmes.

➤ Matrices de rotation de base

○ Atour de l'axe OX:

- $x' = x$
- $y' = y \cdot \cos(\alpha) - z \cdot \sin(\alpha)$
- $z' = y \cdot \sin(\alpha) + z \cdot \cos(\alpha)$

○ Atour de l'axe OY:

- $x' = x \cdot \cos(\alpha) + z \cdot \sin(\alpha)$
- $y' = y$
- $z' = -x \cdot \sin(\alpha) + z \cdot \cos(\alpha)$

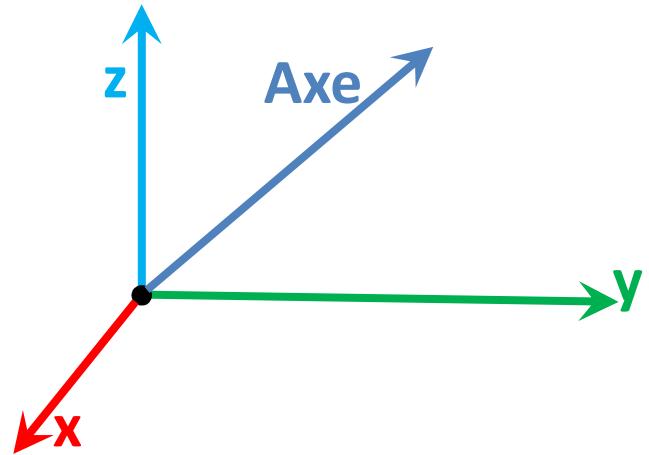
○ Atour de l'axe OZ:

- $x' = x \cdot \cos(\alpha) - y \cdot \sin(\alpha)$
- $y' = x \cdot \sin(\alpha) + y \cdot \cos(\alpha)$
- $z' = z$

$$\begin{array}{|ccc|} \hline & 1 & 0 & 0 \\ & 0 & \cos(\alpha) & -\sin(\alpha) \\ & 0 & \sin(\alpha) & \cos(\alpha) \\ \hline \end{array}$$
$$\begin{array}{|ccc|} \hline & \cos(\alpha) & 0 & \sin(\alpha) \\ & 0 & 1 & 0 \\ & -\sin(\alpha) & 0 & \cos(\alpha) \\ \hline \end{array}$$
$$\begin{array}{|ccc|} \hline & \cos(\alpha) & -\sin(\alpha) & 0 \\ & \sin(\alpha) & \cos(\alpha) & 0 \\ & 0 & 0 & 1 \\ \hline \end{array}$$

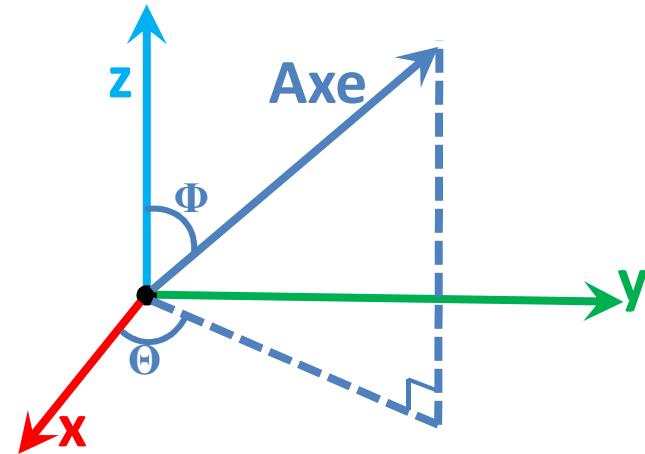
Transformations

- Rotation d'un objet 3D
 - Matrice de rotation d'un axe quelconque
 - on peut la déduire d'une composition de trois matrices de base;



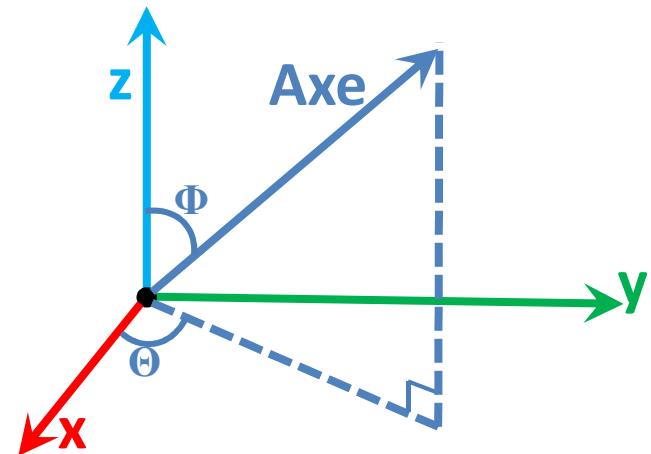
Transformations

- Rotation d'un objet 3D
 - Matrice de rotation d'un axe quelconque
 - on peut la déduire d'une composition de trois matrices de base;
 - 2 rotations pour recaler un axe de base;



Transformations

- Rotation d'un objet 3D
 - Matrice de rotation d'un axe quelconque
 - on peut la déduire d'une composition de trois matrices de base;
 - 2 rotations pour recaler un axe de base;
 - rotation autour de l'axe choisi.



Transformations

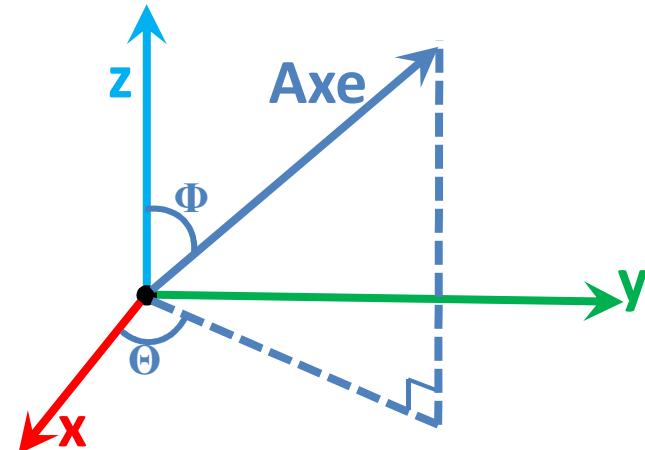
- Rotation d'un objet 3D

- Matrice de rotation d'un axe quelconque

- on peut la déduire d'une composition de trois matrices de base;

- 2 rotations pour recaler un axe de base;
 - rotation autour de l'axe choisi.

- matrice générale.



$u_x^2 + (1-u_x^2)\cos(\alpha)$	$u_x \cdot u_y(1-\cos(\alpha)) - u_z \cdot \sin(\alpha)$	$u_x \cdot u_z(1-\cos(\alpha)) + u_y \cdot \sin(\alpha)$
$u_x \cdot u_y(1-\cos(\alpha)) + u_z \cdot \sin(\alpha)$	$u_y^2 + (1-u_y^2)\cos(\alpha)$	$u_y \cdot u_z(1-\cos(\alpha)) - u_x \cdot \sin(\alpha)$
$u_x \cdot u_z(1-\cos(\alpha)) - u_y \cdot \sin(\alpha)$	$u_y \cdot u_z(1-\cos(\alpha)) + u_x \cdot \sin(\alpha)$	$u_z^2 + (1-u_z^2)\cos(\alpha)$

Plan

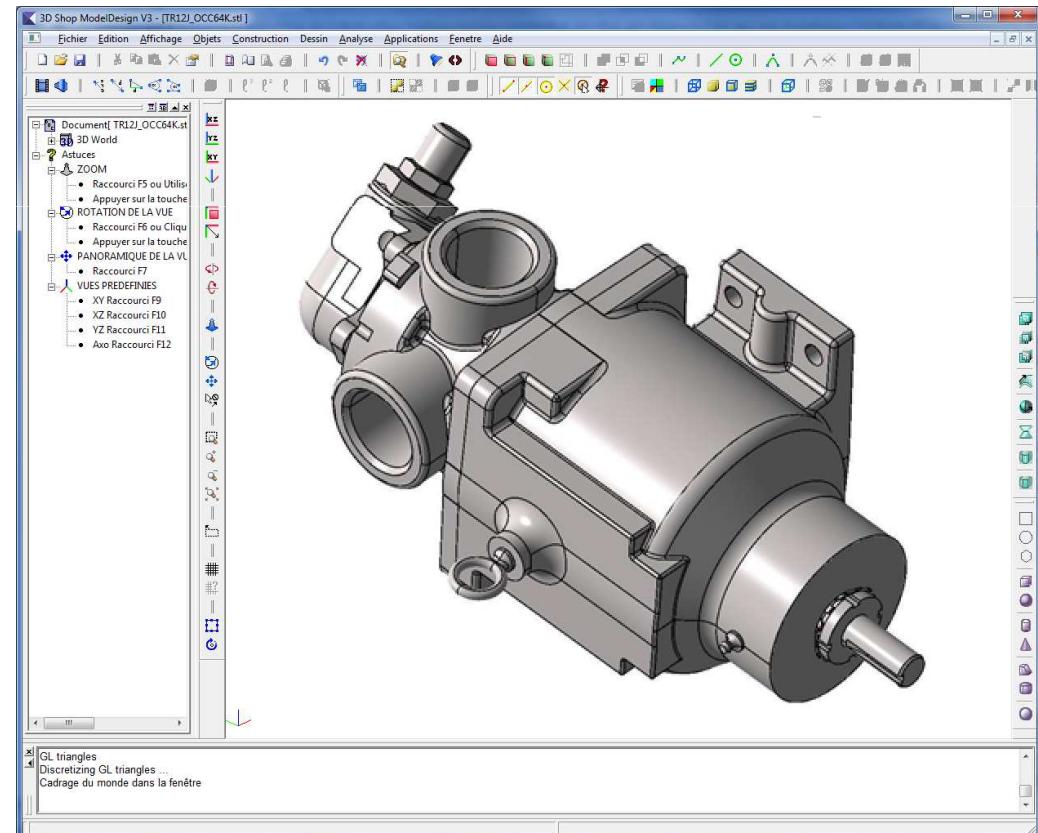
- Introduction
- Vecteurs
- Matrices
- Projections
- Transformations
- Processus de gestion des objets 3D
- Scène 3D en *OpenGL*

Processus de gestion des objets 3D

Création d'objet

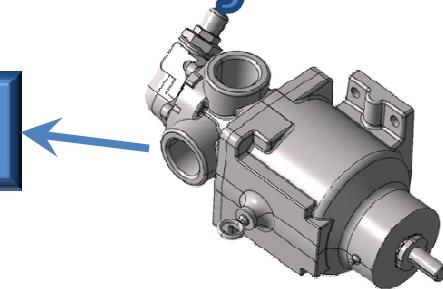
Processus de gestion des objets 3D

- **Création d'objet:**
 - Via un logiciel **CAO** (Conception Assistée par Ordinateur) :
 - 3DS Max,
 - Maya,
 - Blender,
 - ...

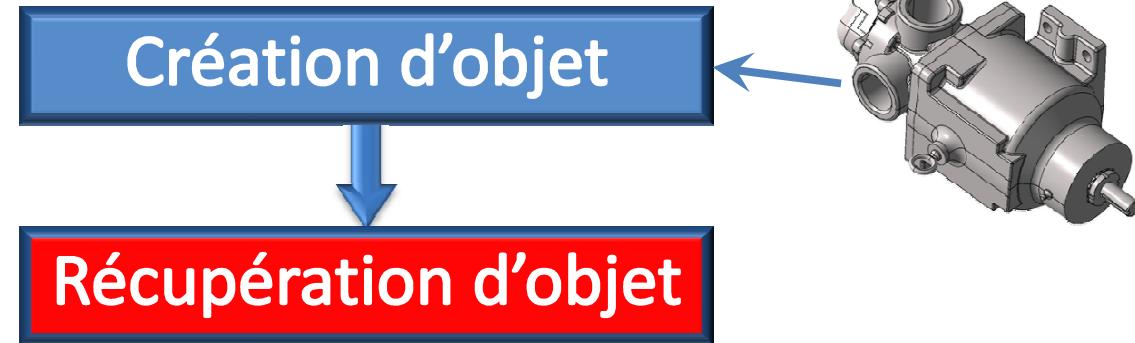


Processus de gestion des objets 3D

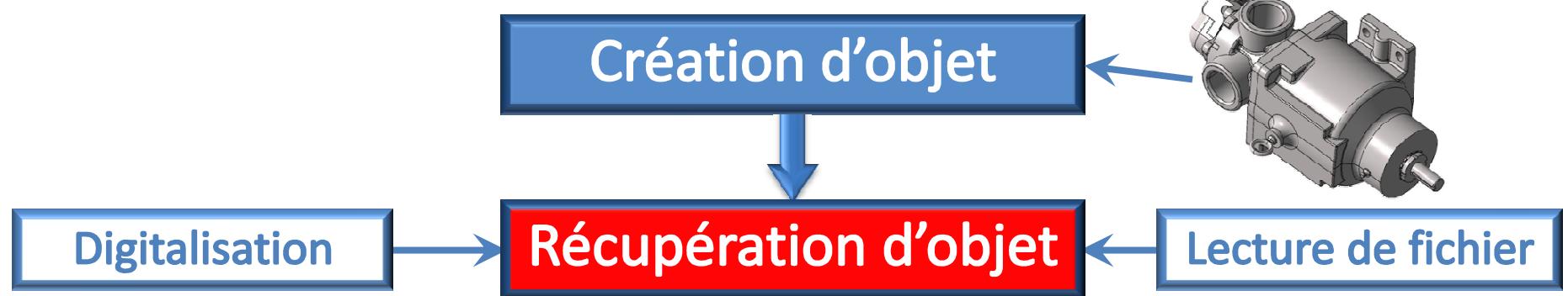
Création d'objet



Processus de gestion des objets 3D



Processus de gestion des objets 3D



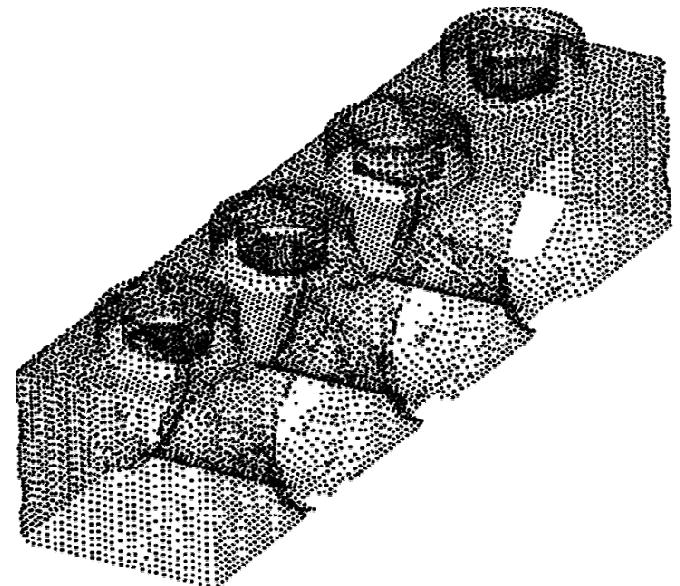
Processus de gestion des objets 3D

- Récupération d'objet:
 - Scan d'un objet



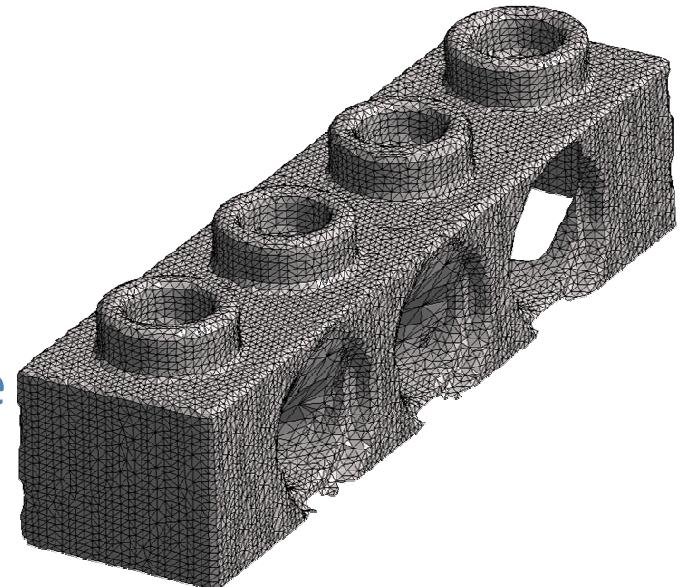
Processus de gestion des objets 3D

- Récupération d'objet:
 - Scan d'un objet
 - nuage de point



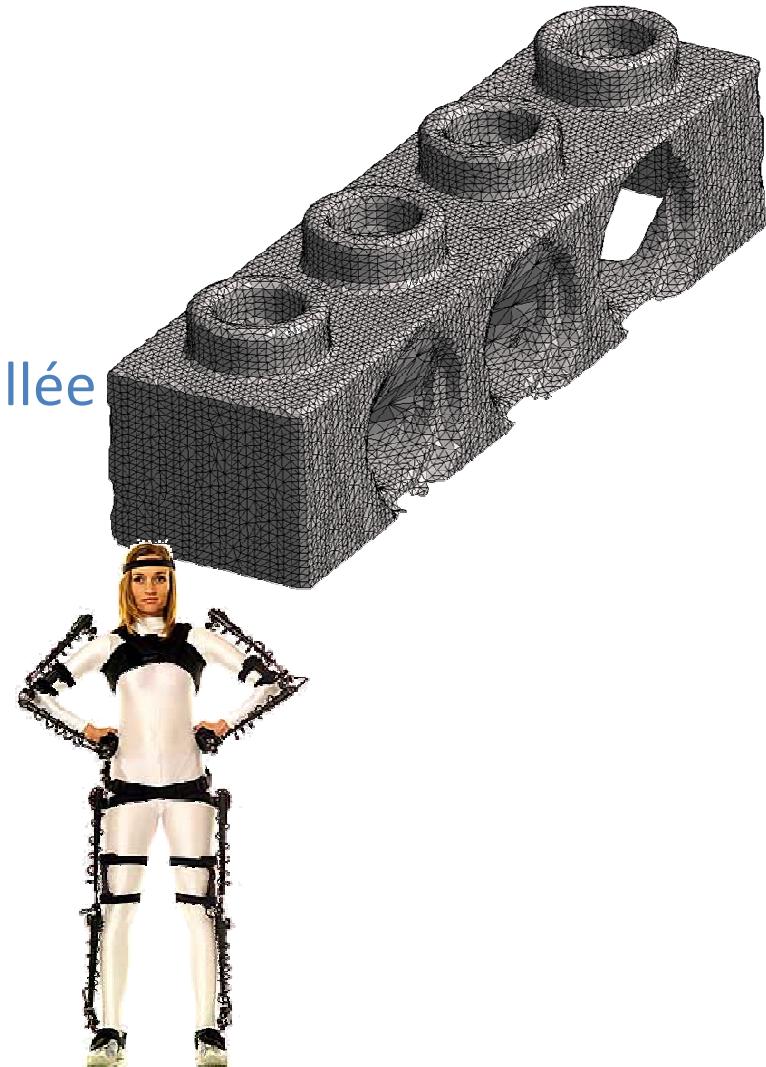
Processus de gestion des objets 3D

- Récupération d'objet:
 - Scan d'un objet
 - nuage de point
 - déduction d'une surface maillée



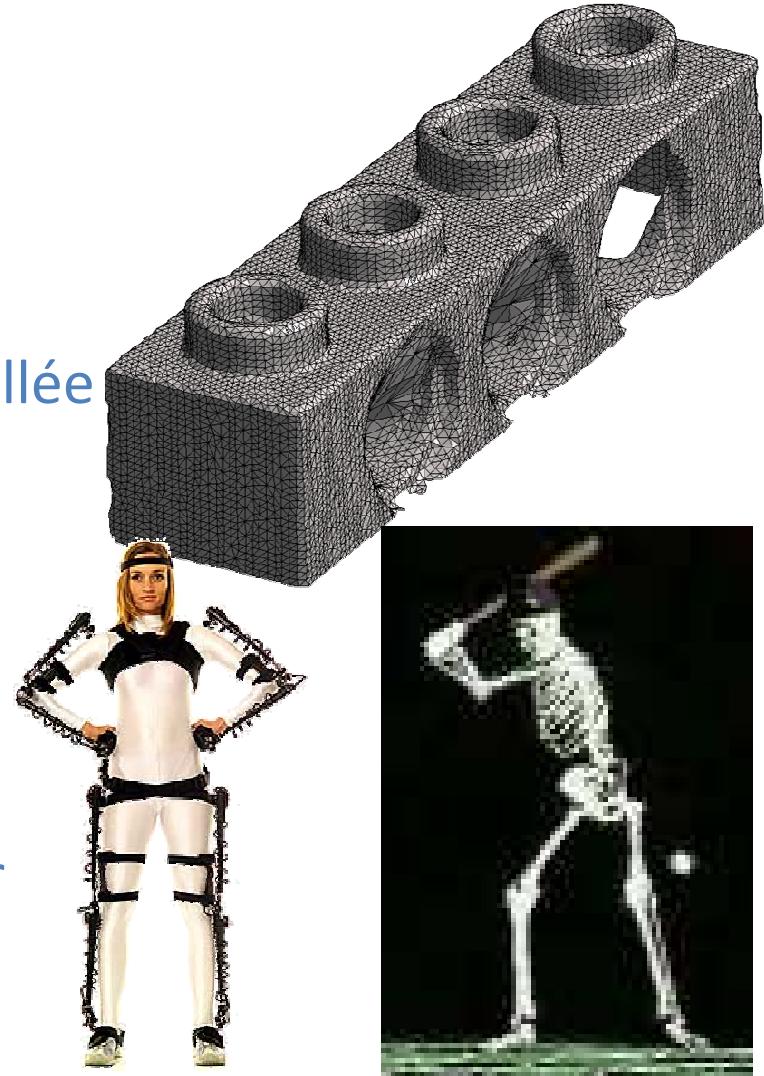
Processus de gestion des objets 3D

- Récupération d'objet:
 - Scan d'un objet
 - nuage de point
 - déduction d'une surface maillée
 - *Motion capture*
 - récupération de points en mouvement.



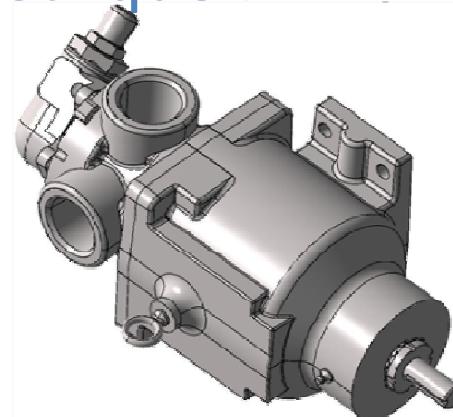
Processus de gestion des objets 3D

- Récupération d'objet:
 - Scan d'un objet
 - nuage de point
 - déduction d'une surface maillée
 - *Motion capture*
 - récupération de points en mouvement.
 - transposition de ces points sur un modèle 3D.



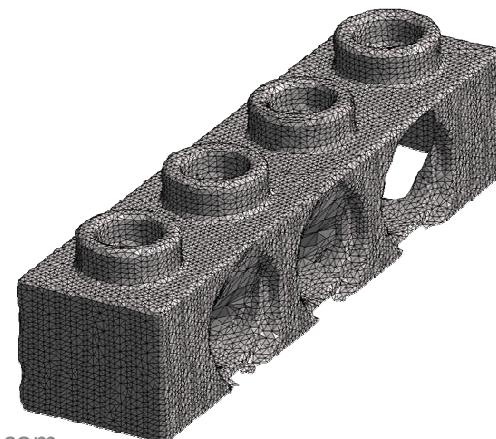
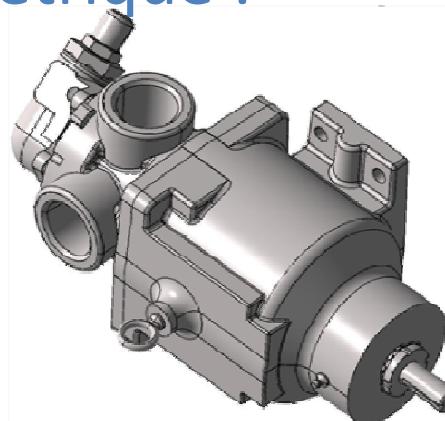
Processus de gestion des objets 3D

- Récupération d'objet:
 - Lecture d'un fichier géométrique :
 - Step
 - Igés
 - Sat
 - ...

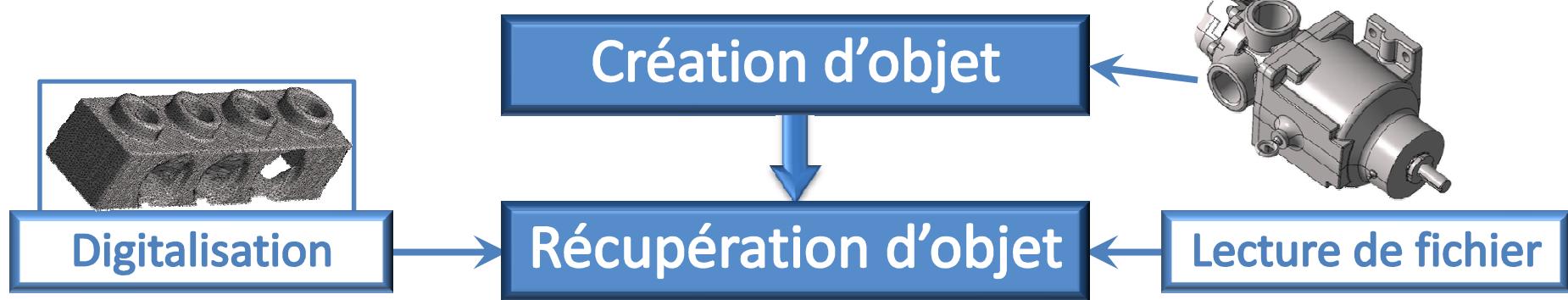


Processus de gestion des objets 3D

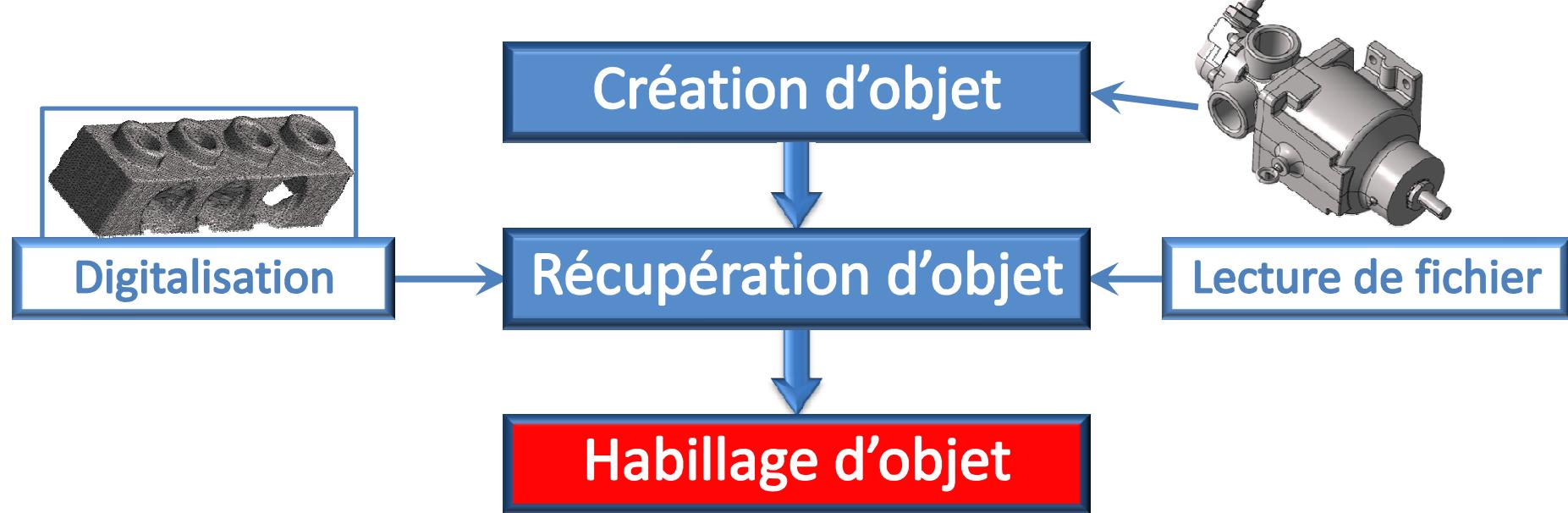
- Récupération d'objet:
 - Lecture d'un fichier géométrique :
 - Step
 - Igés
 - Sat
 - ...
 - Lecture d'un fichier maillé :
 - STL
 - OBJ
 - WRML
 - ...



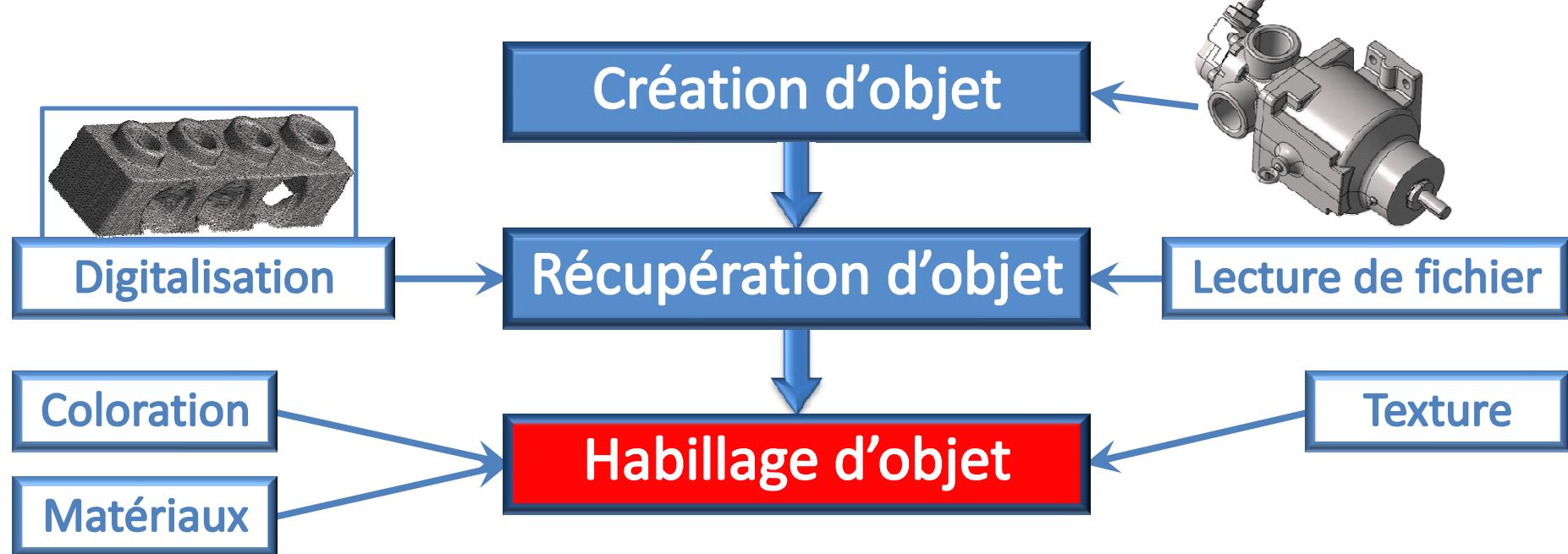
Processus de gestion des objets 3D



Processus de gestion des objets 3D

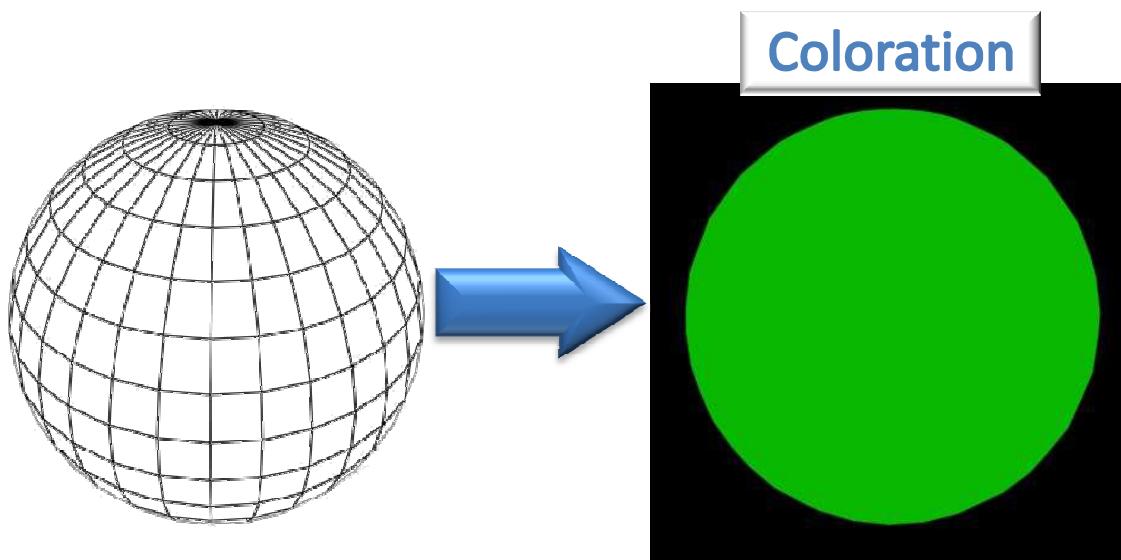


Processus de gestion des objets 3D



Processus de gestion des objets 3D

- Habillage d'objet:
 - Coloration :
 - mettre une **couleur** par point ou par face ou par objet,

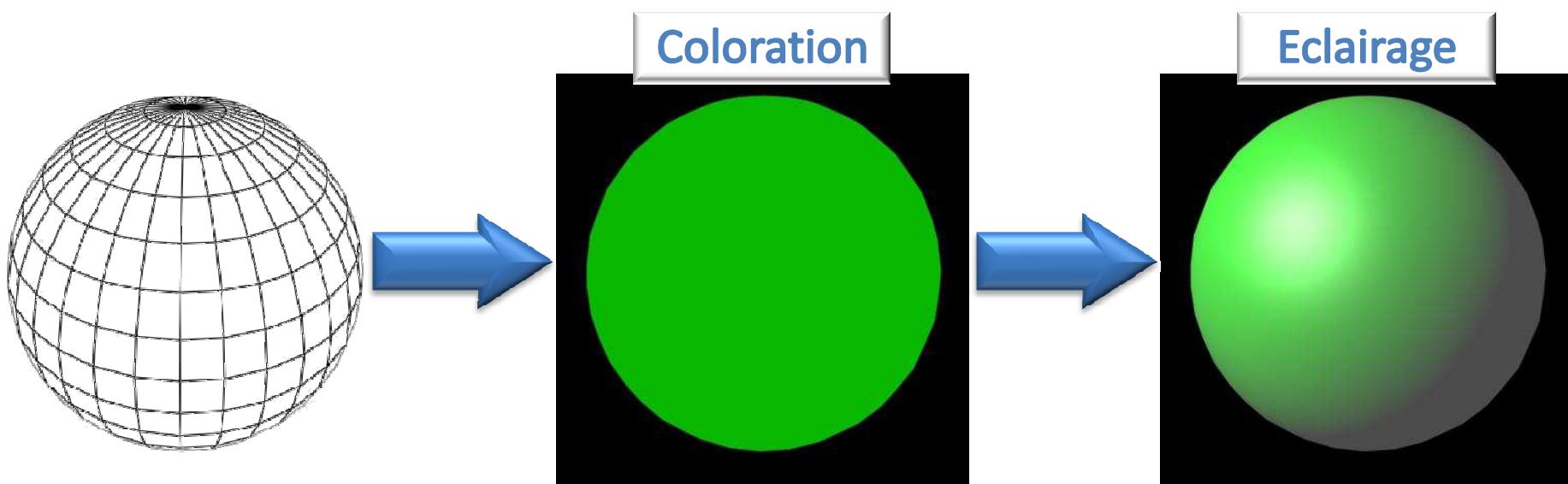


Processus de gestion des objets 3D

- **Habilage d'objet:**

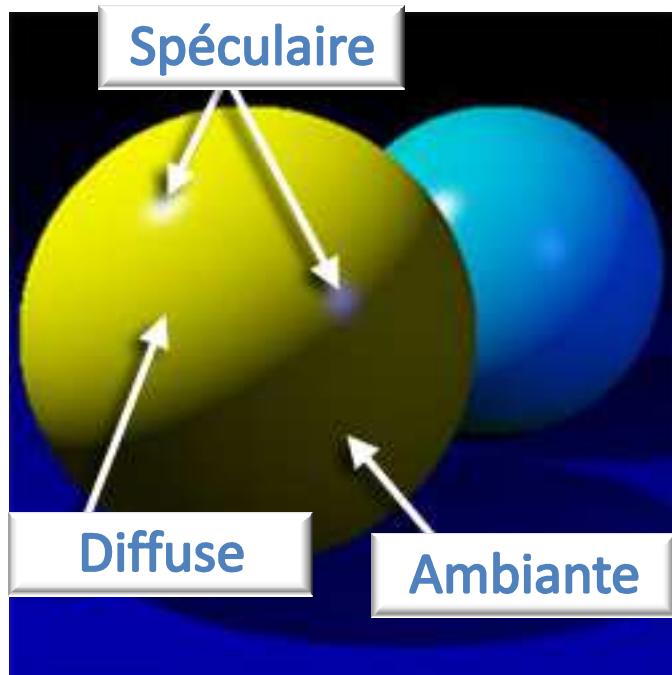
- **Coloration :**

- mettre une **couleur** par point ou par face ou par objet,
 - pour un résultat réaliste de la forme 3D il faut tenir compte de la lumière.



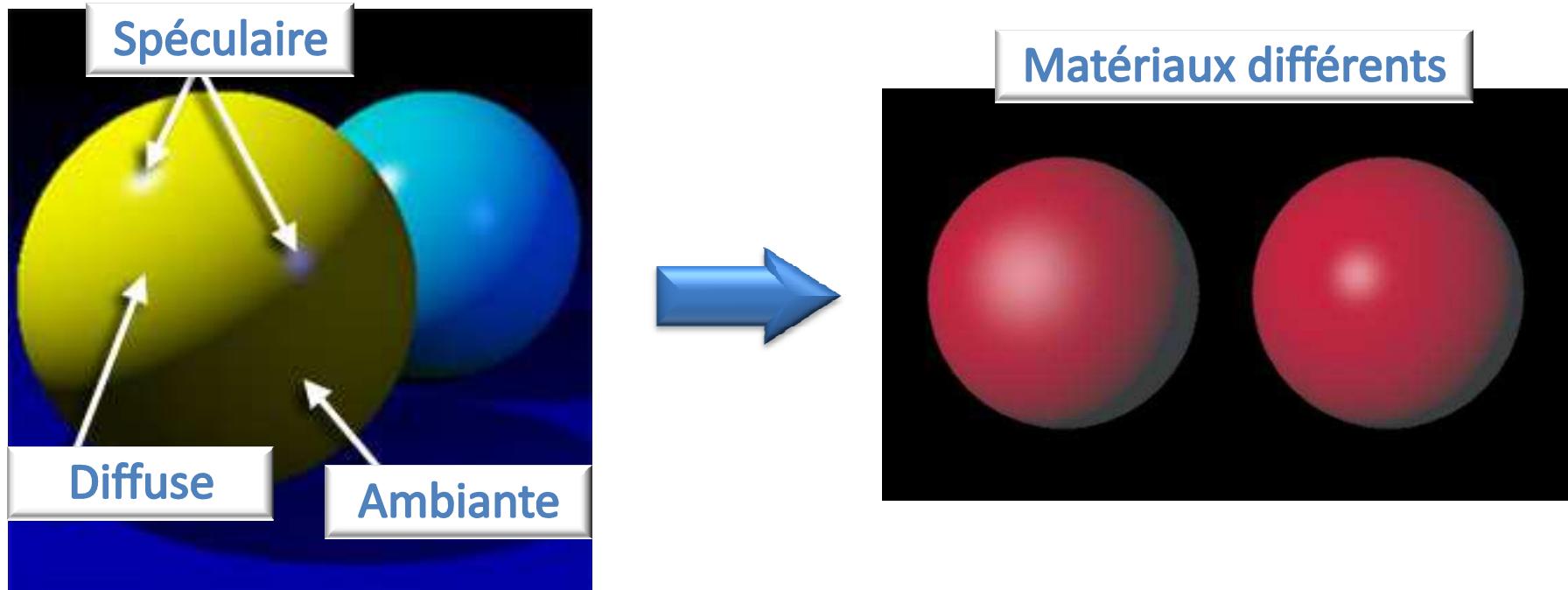
Processus de gestion des objets 3D

- Habillage d'objet:
 - Matériaux
 - permet de modifier les effets de la lumière sur l'objet



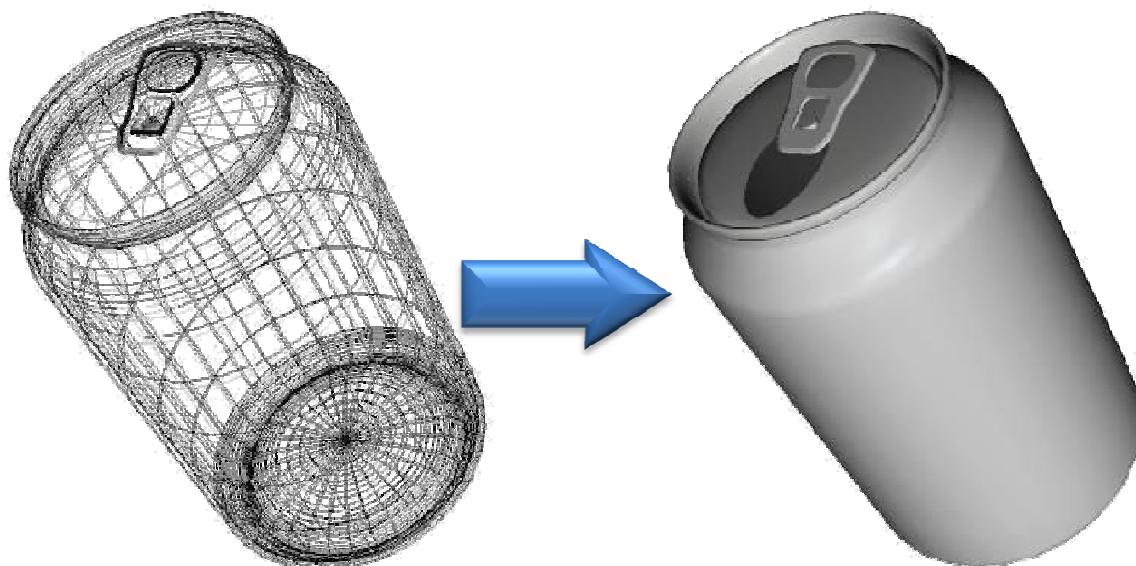
Processus de gestion des objets 3D

- Habillage d'objet:
 - Matériaux
 - permet de modifier les effets de la lumière sur l'objet



Processus de gestion des objets 3D

- Habillage d'objet:
 - Texture
 - après avoir créé la surface

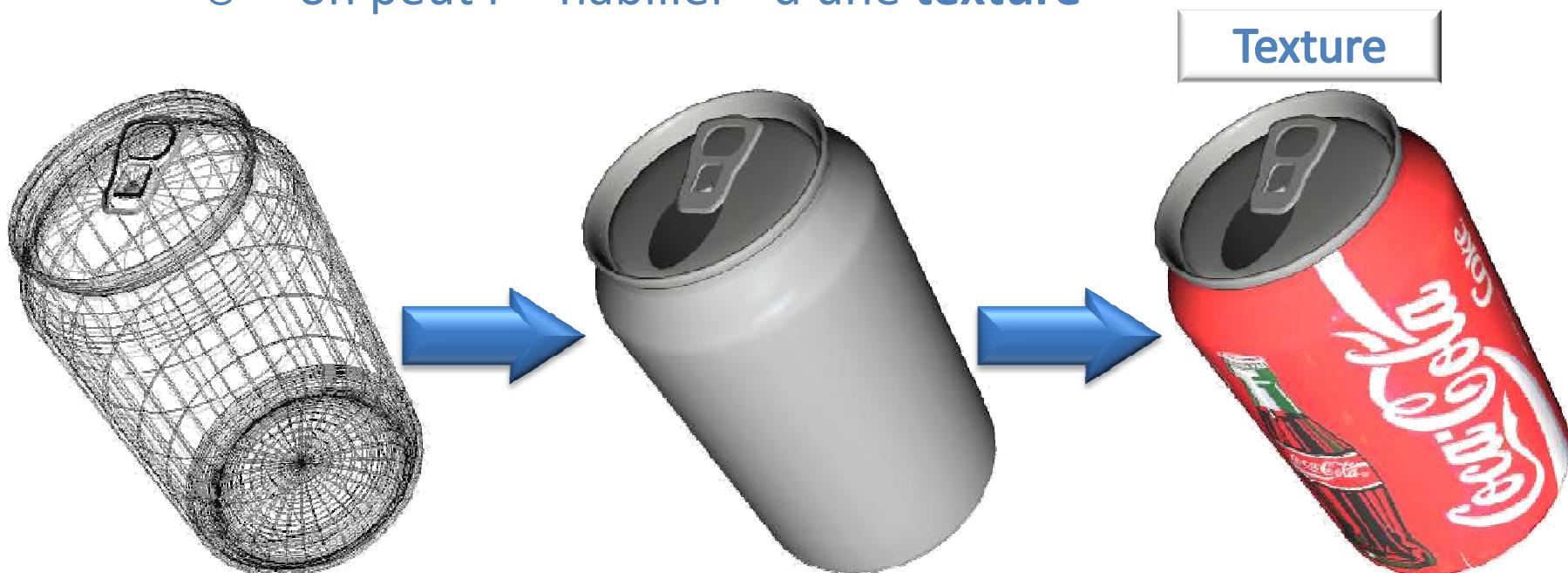


Processus de gestion des objets 3D

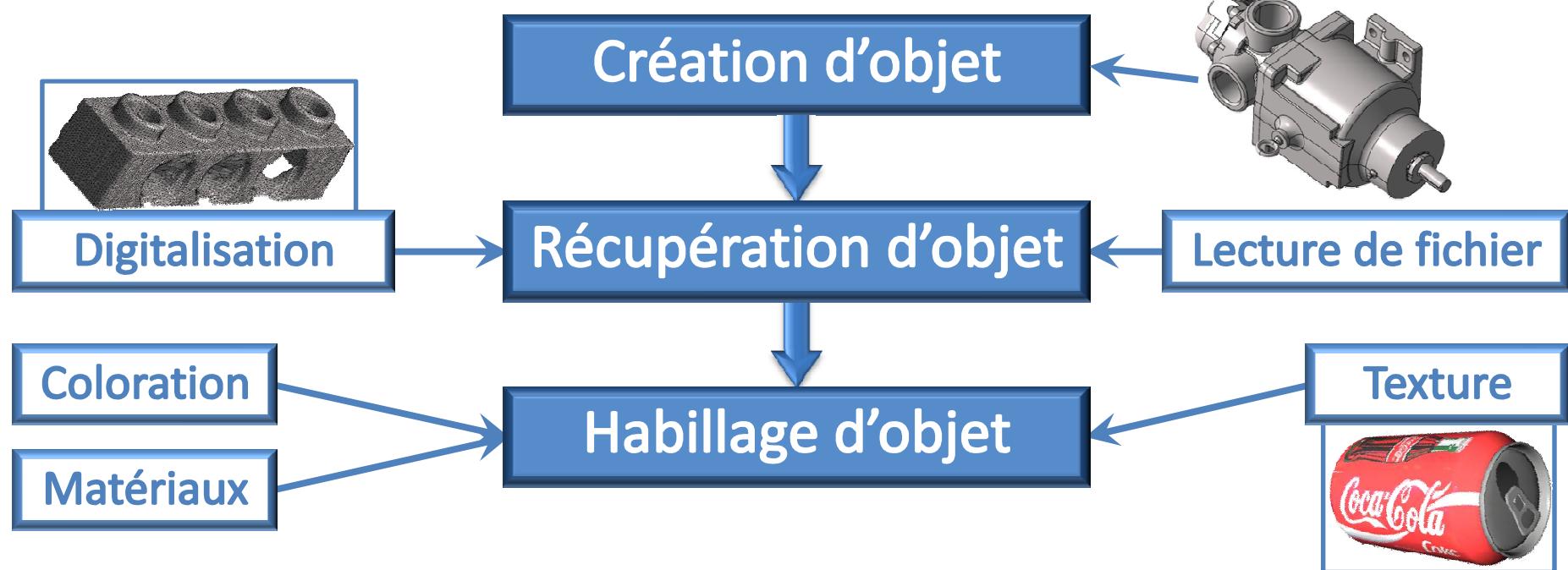
- Habilage d'objet:

- Texture

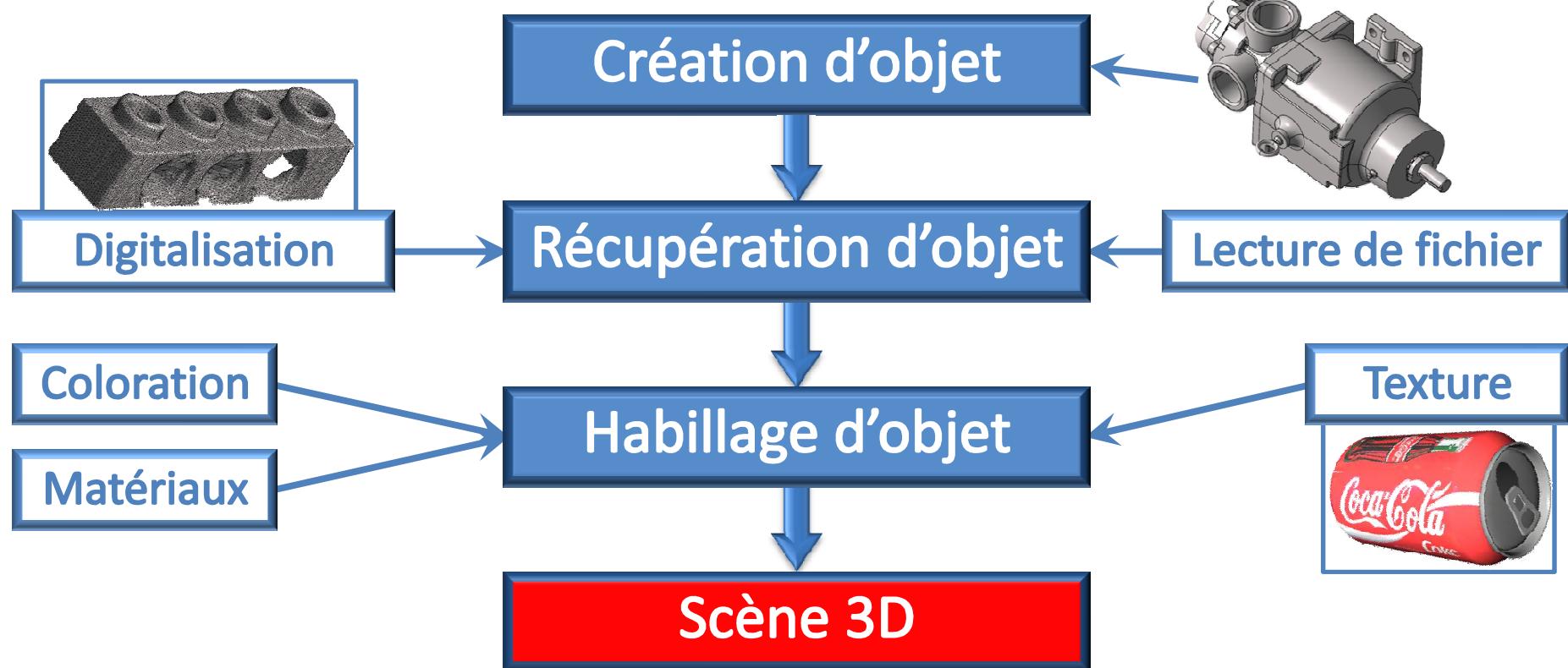
- après avoir crée la surface
 - on peut l' ``habiller'' d'une **texture**



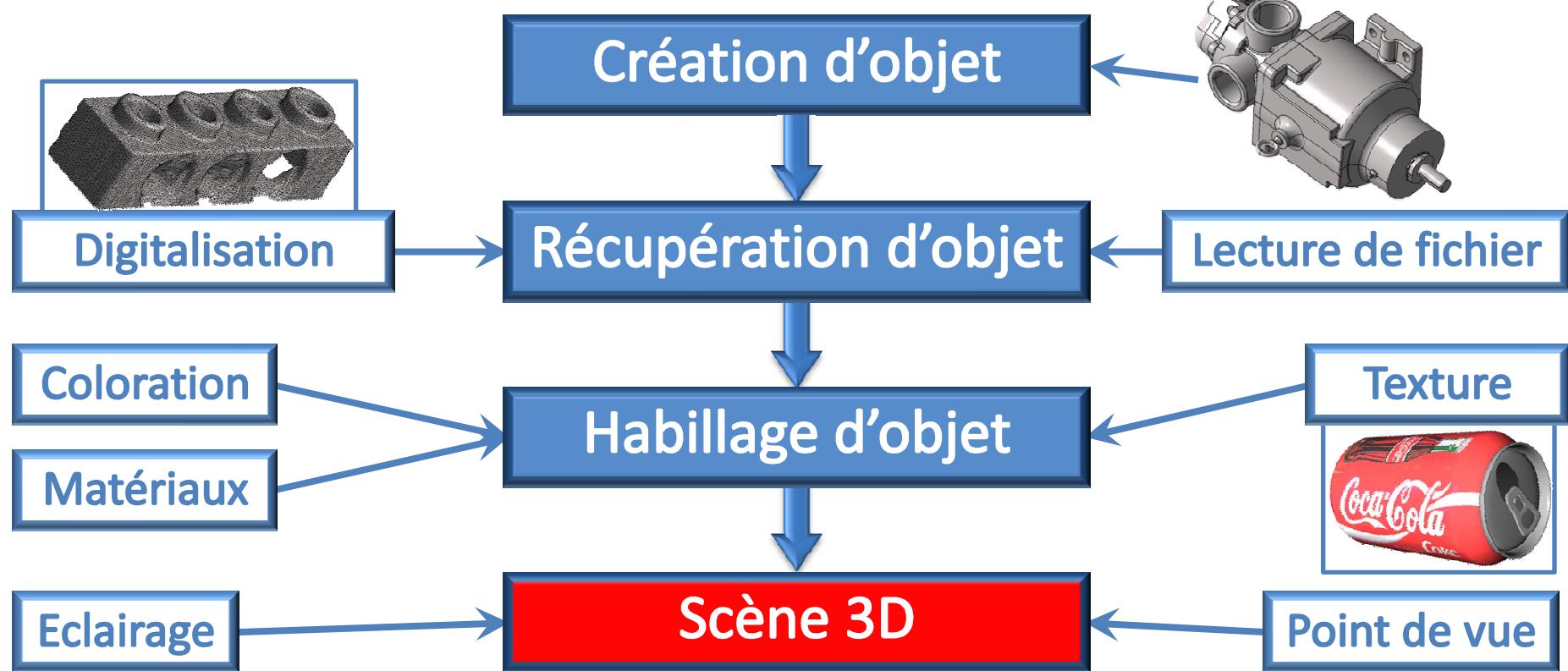
Processus de gestion des objets 3D



Processus de gestion des objets 3D



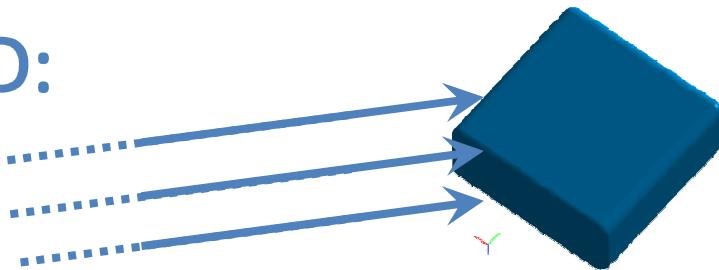
Processus de gestion des objets 3D



Processus de gestion des objets 3D

- Eclairage d'une scène 3D:

- Différents éclairages :
 - source directionnelle
-> pas d'atténuation,

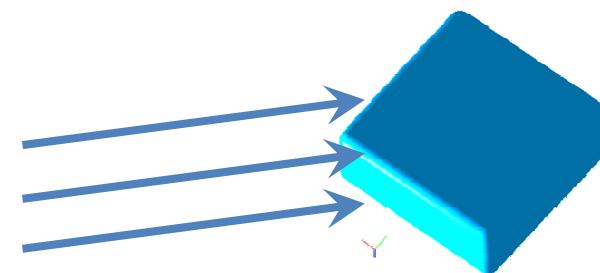
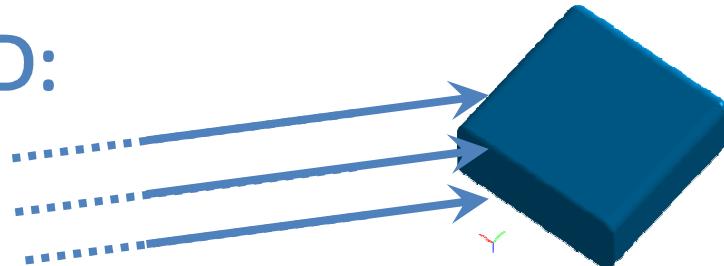


Processus de gestion des objets 3D

- Eclairage d'une scène 3D:

- Différents éclairages :

- source directionnelle
-> pas d'atténuation,

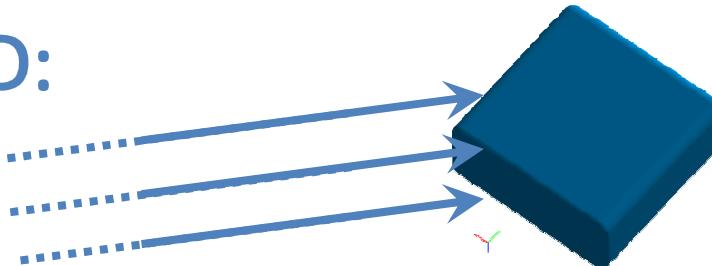


Processus de gestion des objets 3D

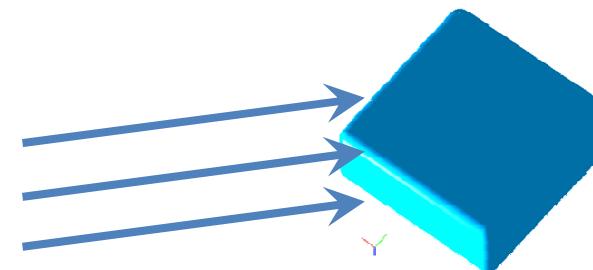
- Eclairage d'une scène 3D:

- Différents éclairages :

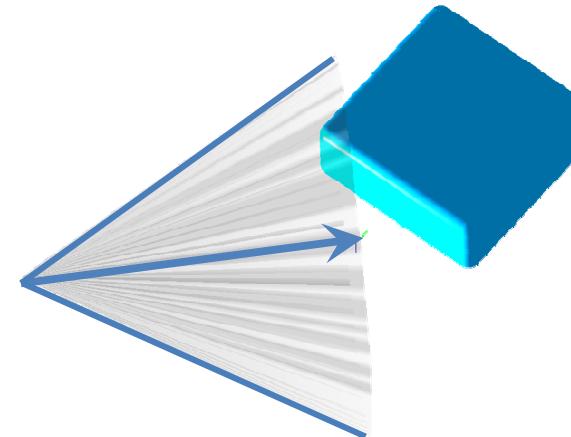
- source directionnelle
-> pas d'atténuation,



- source ponctuelle

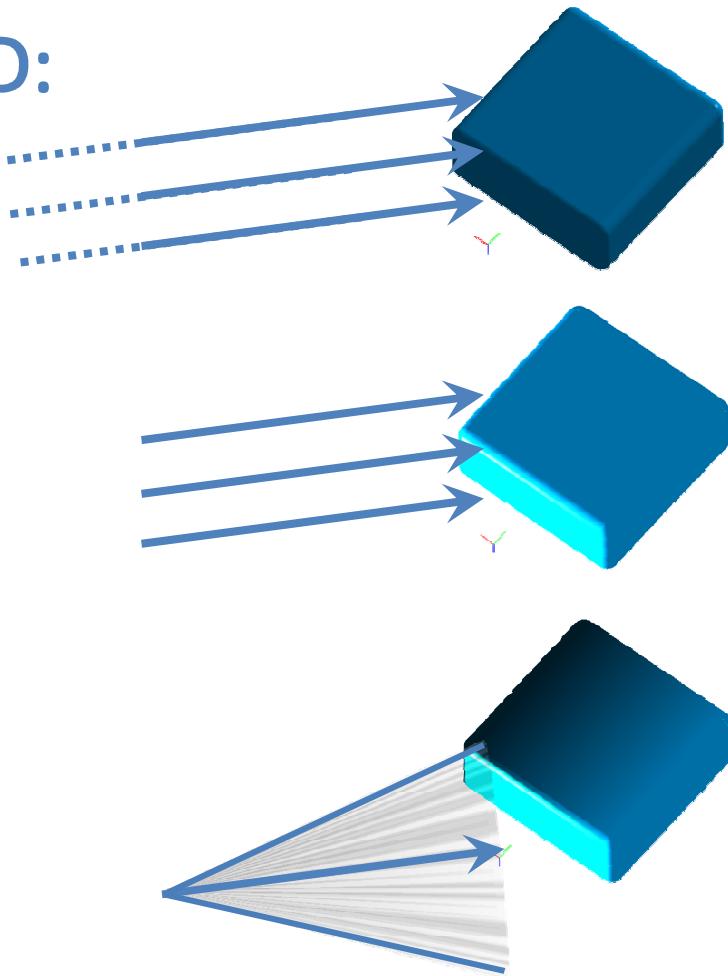


- source projecteur
-> avec un angle



Processus de gestion des objets 3D

- Eclairage d'une scène 3D:
 - Différents éclairages :
 - source directionnelle
-> pas d'atténuation,
 - source ponctuelle
 - source projecteur
-> avec un angle

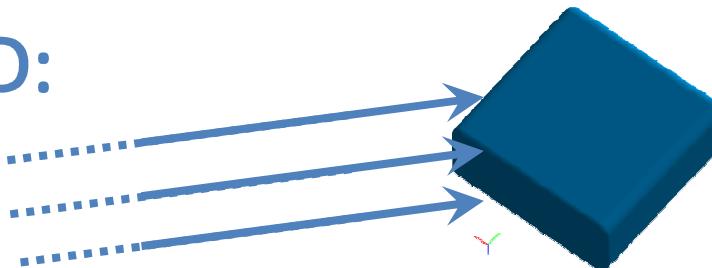


Processus de gestion des objets 3D

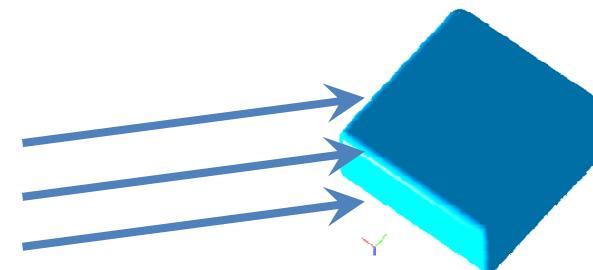
- Eclairage d'une scène 3D:

- Différents éclairages :

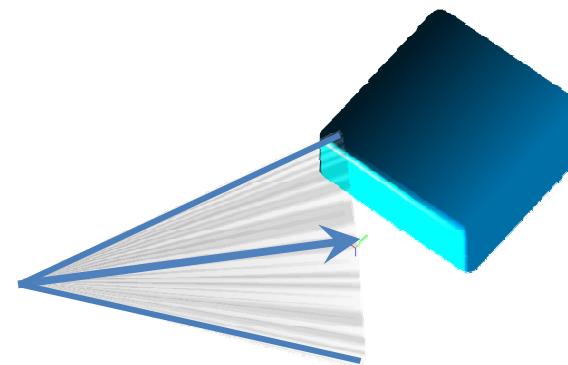
- source directionnelle
-> pas d'atténuation,



- source ponctuelle



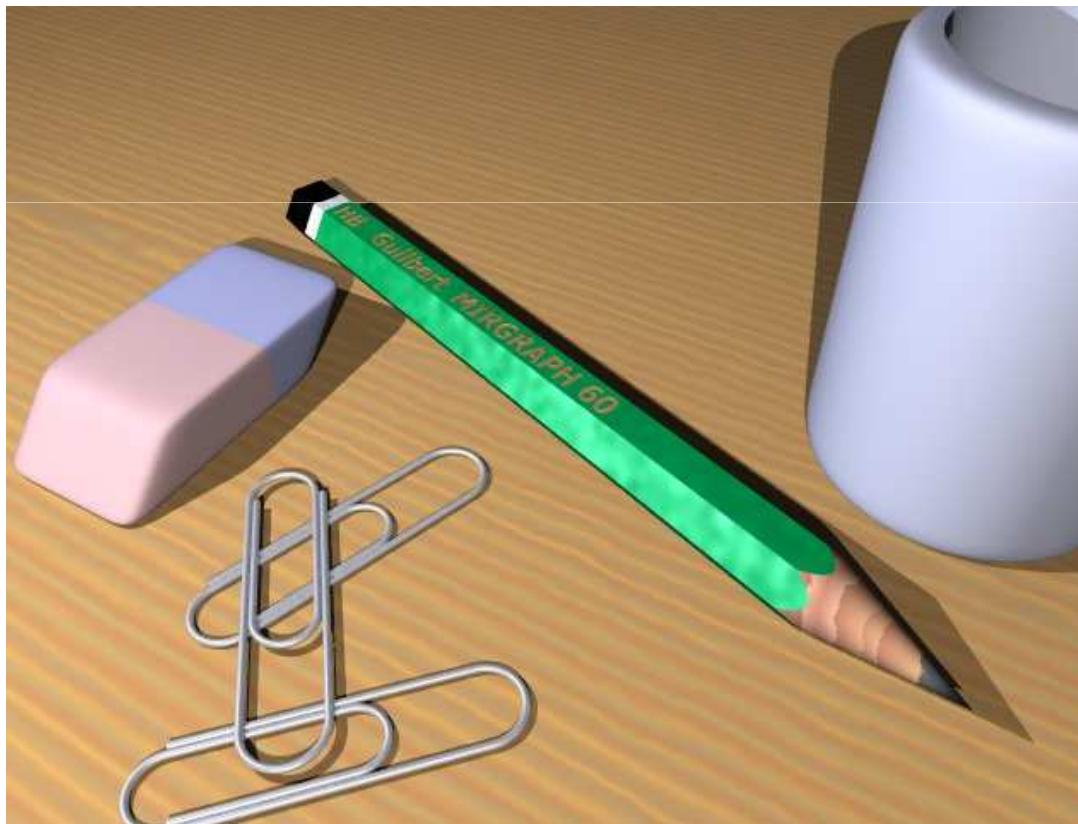
- source projecteur
-> avec un angle



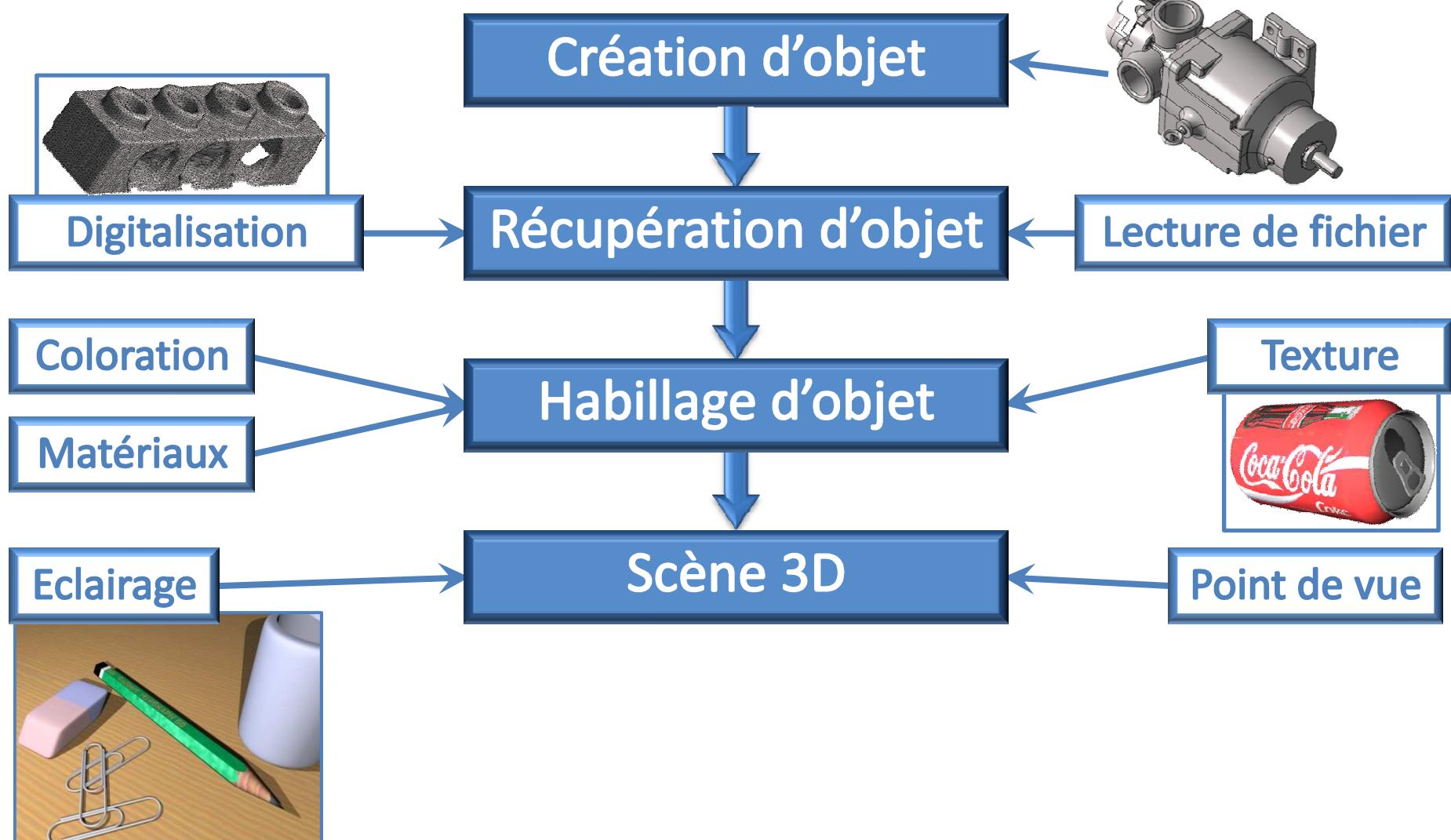
- source = objet

Processus de gestion des objets 3D

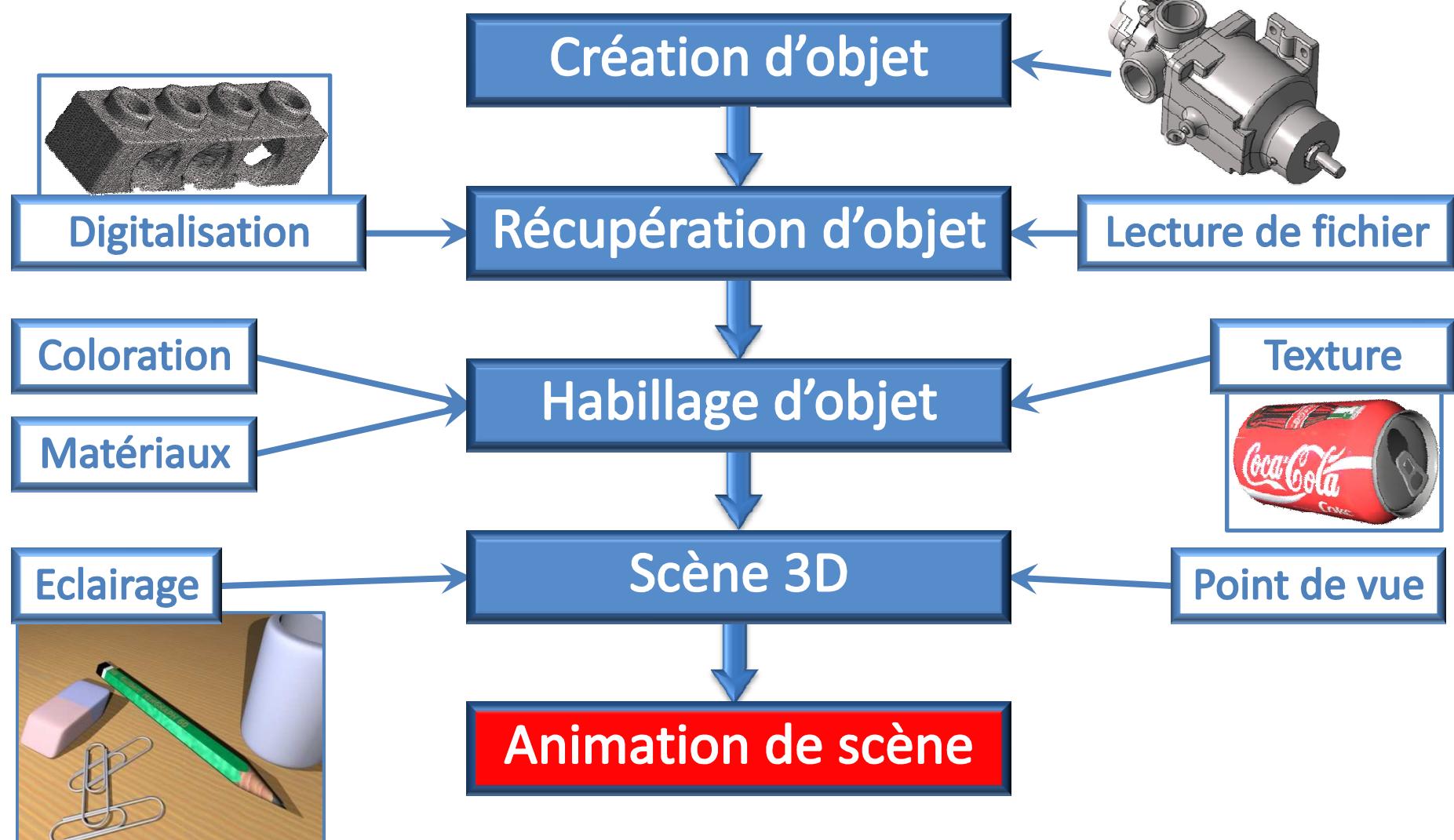
- Eclairage d'une scène 3D :
 - calcul d'ombres à partir des lumières



Processus de gestion des objets 3D

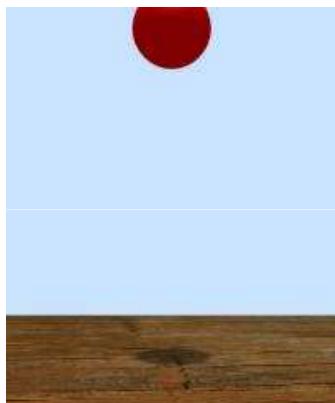


Processus de gestion des objets 3D



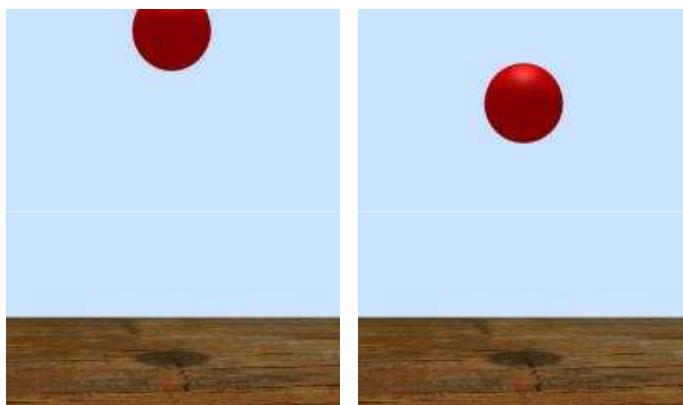
Processus de gestion des objets 3D

- Animation → succession de scènes 2D:
 - Le même objet est déplacé dans la scène



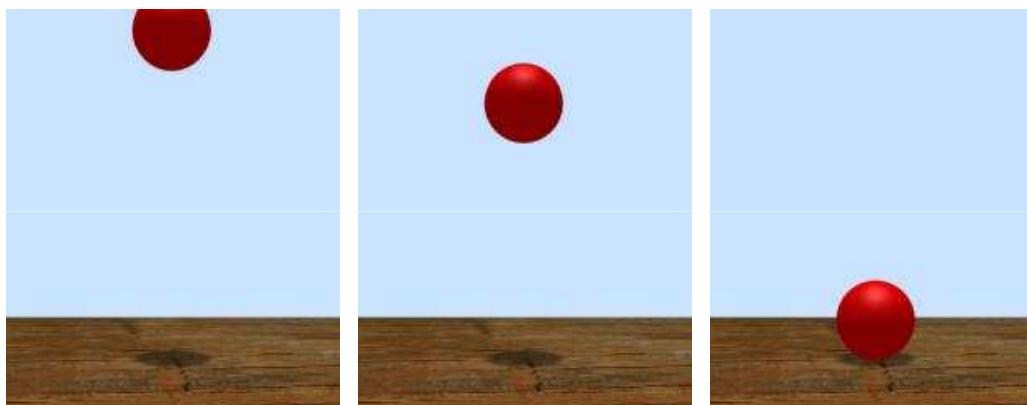
Processus de gestion des objets 3D

- Animation → succession de scènes 2D:
 - Le même objet est déplacé dans la scène



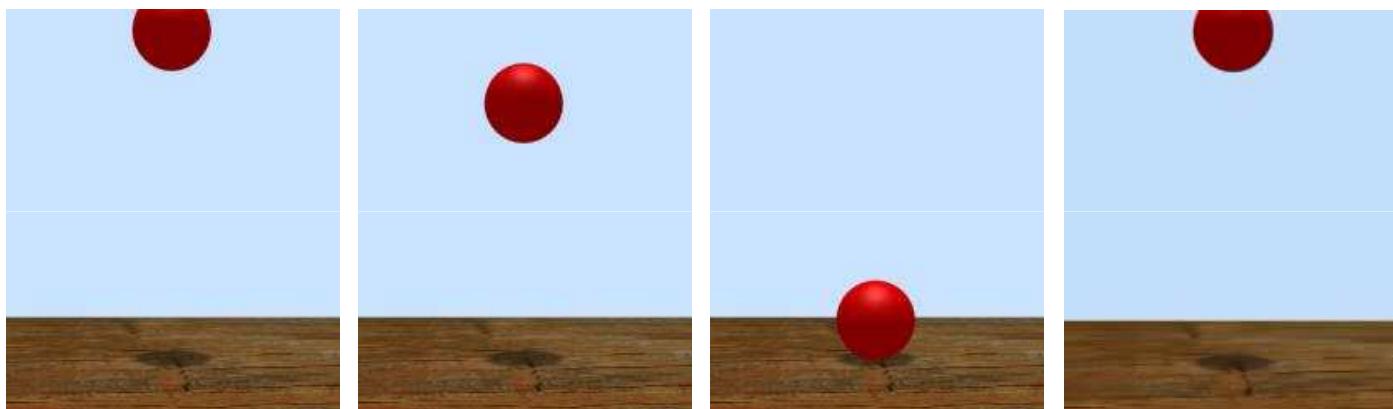
Processus de gestion des objets 3D

- Animation → succession de scènes 2D:
 - Le même objet est déplacé dans la scène



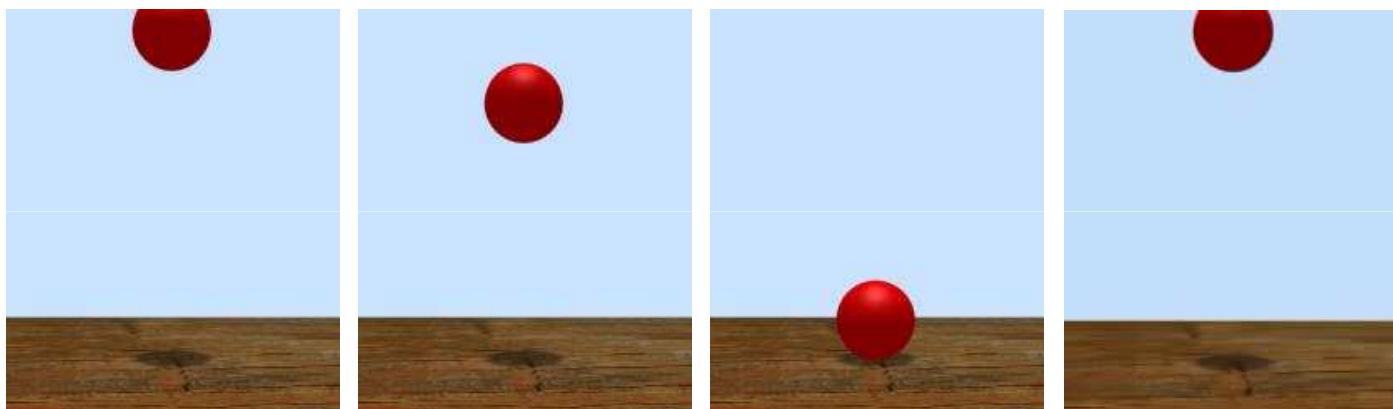
Processus de gestion des objets 3D

- Animation → succession de scènes 2D:
 - Le même objet est déplacé dans la scène

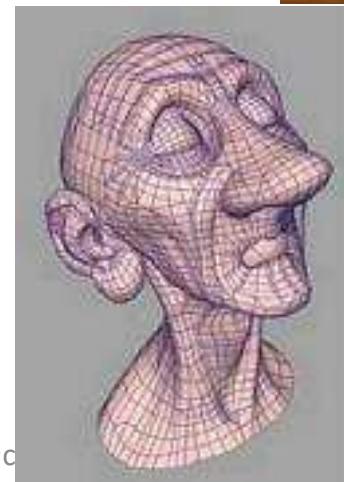


Processus de gestion des objets 3D

- Animation → succession de scènes 2D:
 - Le même objet est déplacé dans la scène

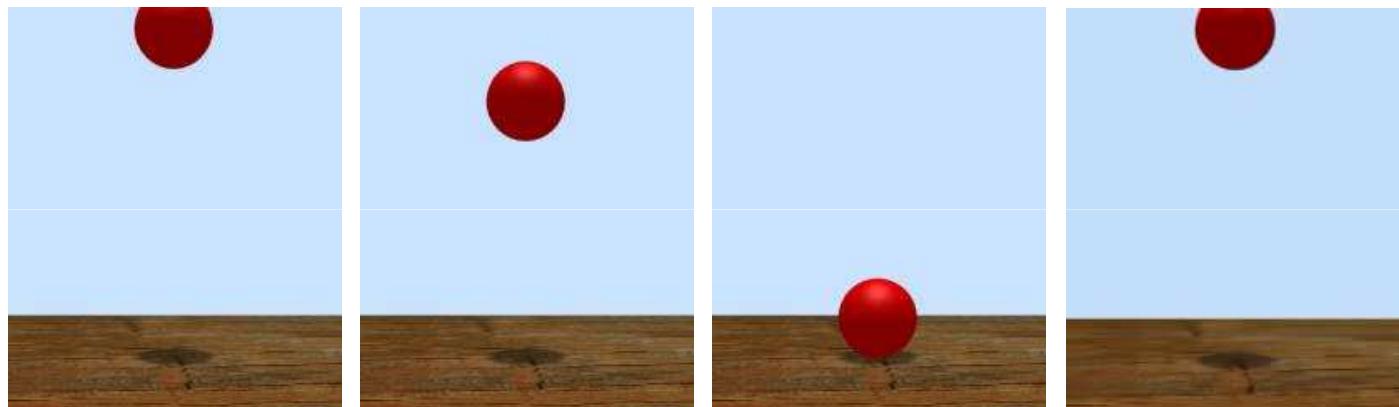


- A partir d'un objet 3D
 - on calcule différents objets 3D par déformation

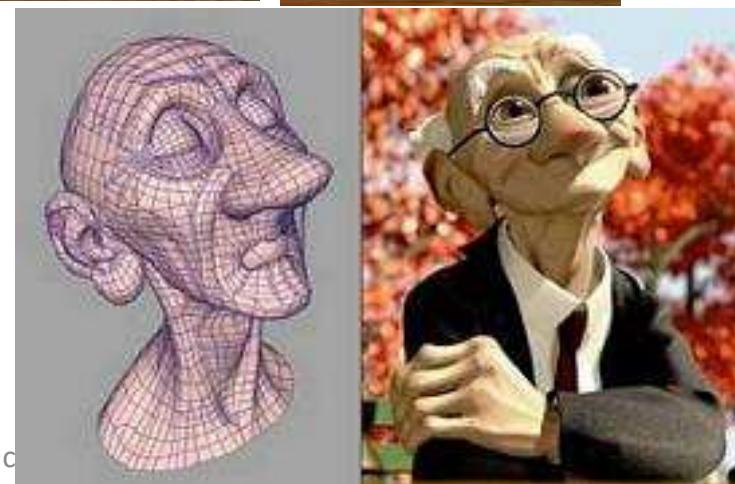


Processus de gestion des objets 3D

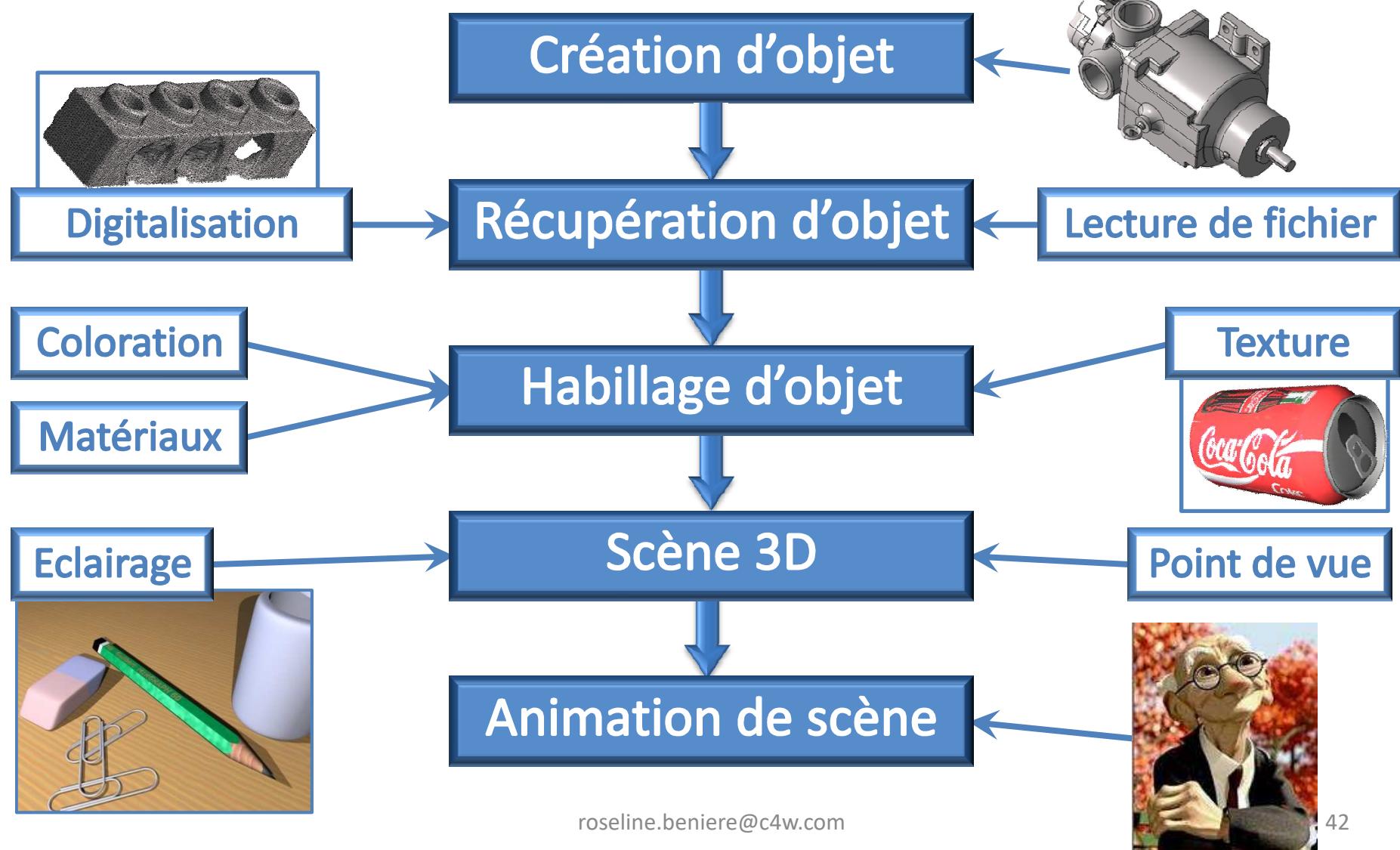
- Animation → succession de scènes 2D:
 - Le même objet est déplacé dans la scène



- A partir d'un objet 3D
 - on calcule différents objets 3D par déformation



Processus de gestion des objets 3D



Plan

- Introduction
- Vecteurs
- Matrices
- Projections
- Transformations
- Processus de gestion des objets 3D
- Scène 3D en *OpenGL*

Scène 3D en *OpenGL*

- Pour dessiner les scènes nous allons utiliser *OpenGL* :
 - *OpenGL* est une **librairie graphique** de bas niveau pour tracer des objets géométriques 2D ou 3D,
 - *OpenGL* est **indépendant du langage** de programmation, on peut l'utiliser avec C ou C++ ou encore JAVA ...
 - *Open GL* est une **machine à état finis**
 - *OpenGL* permet de connaître à tout moment les **paramètres graphiques** courant : transformations, lumières ...

Scène 3D en *OpenGL*

- Etapes du pipeline de rendu :
 - Définition du modèle mathématique
 - Définition d'un point de vue de la scène
 - Calcul des couleurs et lumières

Scène 3D en *OpenGL*

- Etapes du pipeline de rendu :
 - Définition du modèle mathématique
 - Définition d'un point de vue de la scène
 - Calcul des couleurs et lumières
- Nous allons voir comment faire pour créer une scène *OpenGL* avec des exemples en C++.

Scène 3D en *OpenGL*

- **création de la fenêtre**
 - déclaration d'une fenêtre
 -  glutInitWindowSize (400,400);
 - initialisation de la fenêtre
 -  glutInit (argc, argv);
 - type de contenu graphique
 -  glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
 - création de la fenêtre
 -  glutCreateWindow ("Programme OpenGL");
 - fonction pour initialiser *OpenGL*
 -  InitOpenGL ()
- **liaison avec les fonctions callback**
 -  glutReshapeFunc (reshape);
 -  glutKeyboardFunc (keyboard);
 -  glutDisplayFunc (display);
 - boucle sur le programme principale
 -  glutMainLoop ();

Scène 3D en *OpenGL*

- fonction `InitOpenGL`
 - définition de la représentation
 -  `glShadeModel(GL_SMOOTH);`
 - couleur de fond :
 -  `glClearColor(0.0 , 0.0 , 0.0 , 0.0);`
 - activation d'une lumière
 -  `glEnable(GL_LIGHTING);`
 -  `glEnable(GL_LIGHT0);`
 - définition de la lumière
 -  `Glfloat ambient[] = {0.0 , 0.0 , 1.0 , 1.0}`
 - `Glfloat diffuse[] = {0.9 , 0.9 , 0.2 , 1.0}`
 - `Glfloat position[] = {0.0 , 3.0 , 3.0 , -3.0}`
 -  `glLightfv(GL_LIGHT0 , GL_AMBIENT , ambient);`
 - `glLightfv(GL_LIGHT0 , GL_DIFFUSE , diffuse);`
 - `glLightfv(GL_LIGHT0 , GL_POSITION , position);`

Scène 3D en *OpenGL*

- **affichage d'un objet**
 - définition de la couleur
  glColor3f(0.1 , 0.1 , 1.0);
 - définition des transformations de l'objet
  glMatrixMode (GL_MODELVIEW);
 glLoadIdentity ();
  glRotatef (angle, 0 , 1 , 0);
 - ajout de l'objet dans la scène
  glutSolidTeapot (1.0);
 - vidage du buffer
 glFlush ();
 -  glutSwapBuffer ();

Scène 3D en *OpenGL*

- fonction appelée si la fenêtre est déplacée ou modifiée : **reshape (int w , int h)**
 - définition de la zone de dessin
 -  `glViewport(0 , (GLsizei) w , (GLsizei) h);`
 - matrice de projection
 -  `glMatrixMode (GL_PROJECTION);`
 - `glLoadIdentity ();`
 -  `glOrtho(-2 , 2 , -2 , 2 , -2 , 2);`
 - matrice de transformation
 -  `glMatrixMode (GL_MODELVIEW);`
 - `glLoadIdentity ();`

Scène 3D en *OpenGL*

- tracé d'un objet
 - début du tracé
 -  `glBegin (GL_LINES);`
 - définition des points
 -  `glVertex3f (-1 , 1 , 0);`
`glVertex3f (1 , 1 , 0); ...`
 - fin du tracé
 -  `glEnd ();`
- modification taille et style
 - modifier la taille d'un point
 -  `glPointSize (2);`
 - modifier la taille d'une ligne
-  `glLineWidth (2);`
 - mettre une ligne en pointillé
 - `glEnable (GL_LINE_STIPPLE);`
 -  `glLineStipple (2 , 0x0COF);`

Scène 3D en *OpenGL*

- En *OpenGL* les fonctions peuvent prendre de nombreux types:
 - par convention le nom de la fonction donne l'information, on retrouve :
 - le nombre de paramètre
 - leur type : f pour float, d pour double, b pour caractère ...
 - si la dernière lettre est un v alors les paramètres sont sous la forme d'un vecteur.

Scène 3D en *OpenGL*

- En *OpenGL* les fonctions peuvent prendre de nombreux types:
 - par convention le nom de la fonction donne l'information, on retrouve :
 - le nombre de paramètre
 - leur type : f pour float, d pour double, b pour caractère ...
 - si la dernière lettre est un v alors les paramètres sont sous la forme d'un vecteur.
 - exemple : `glColor*` pour définir la couleur des objets
 - `glColor3f (1.0 , 1.0 , 1.0);`
 - double vecteur [4] = {1.0 , 1.0 , 1.0 , 1.0};
`glColor4dv(v);`

Conclusion

- **Représentation d'objet 3D :**
 - différentes représentations possibles,
 - leur processus de visualisation,
 - introduction à *OpenGL*.

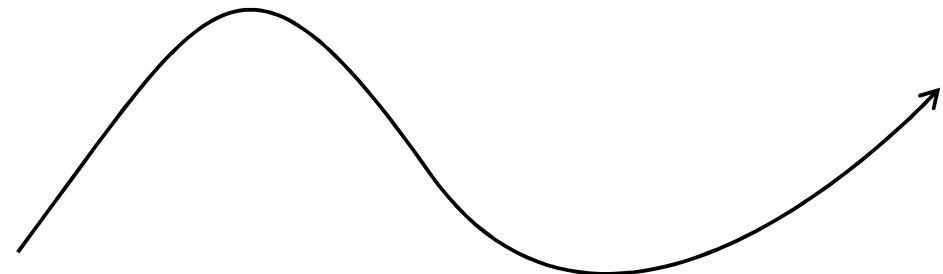
Conclusion

- **Représentation d'objet 3D :**
 - différentes représentations possibles,
 - leur processus de visualisation,
 - introduction à *OpenGL*.
- **Rappel utile pour la visualisation 3D :**
 - vecteurs : définitions et opérations,
 - matrices : opérations
 - projection : sur une droite ou un plan,
 - transformation : translation, mise à l'échelle ...

FIN

Courbes paramétriques

lundi 23/01



Pour récupérer les cours et le TD/TP:

<http://www.lirmm.fr/~beniere/Enseignements.php>

FIN

Courbes paramétriques

lundi 23/01

Surfaces paramétriques

lundi 30/01

Pour récupérer les cours et le TD/TP:
<http://www.lirmm.fr/~beniere/Enseignements.php>

Sources

- Cours utilisés pour ce support :
 - Gilles Gesquière (Gamagora Lyon)
 - Romain Raffin (LSIS Marseille)