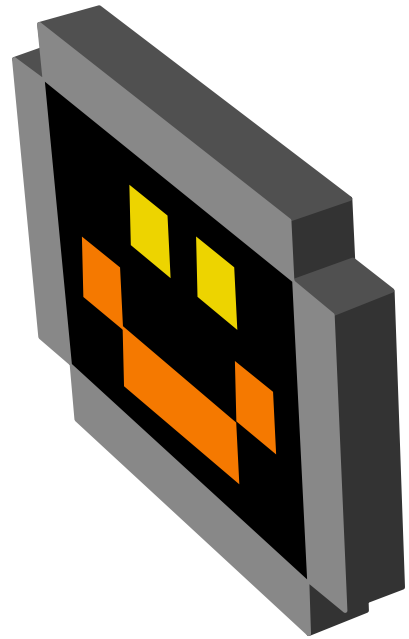
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Introduction à l'infographie 3D

CS211 – Introduction à l'informatique visuelle

24 février 2015

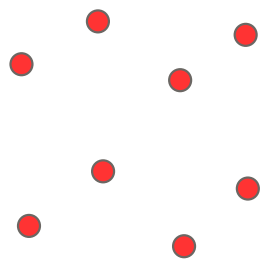


Séverin Lemaignan

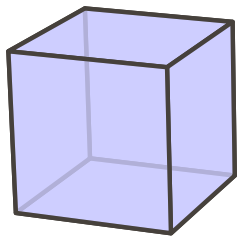
Computer-Human Interaction
for Learning and Instruction **EPFL**

AU COMMENCEMENT, LE POINT...

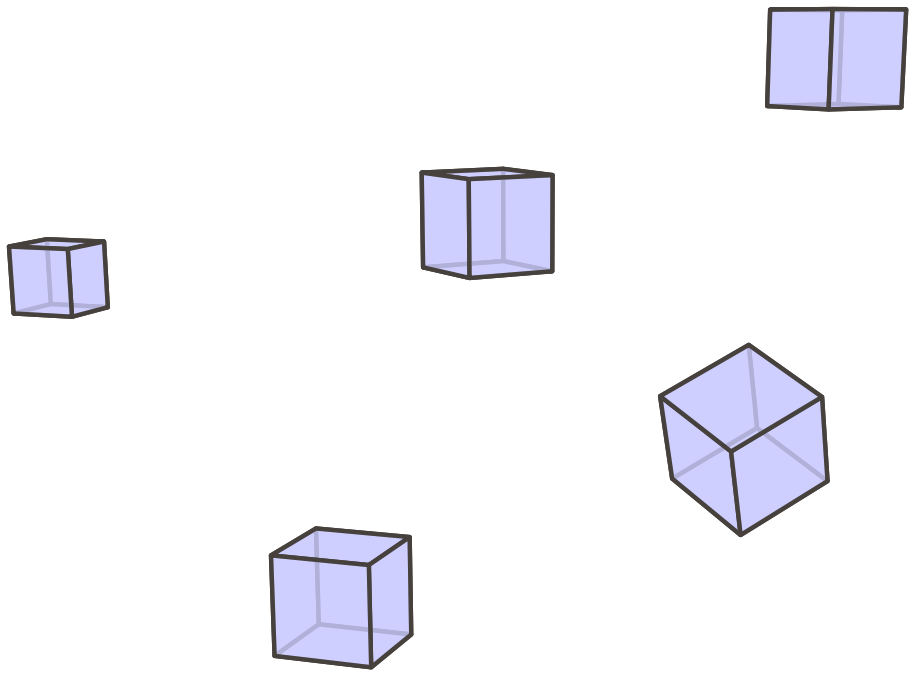
AU COMMENCEMENT, LE POINT...



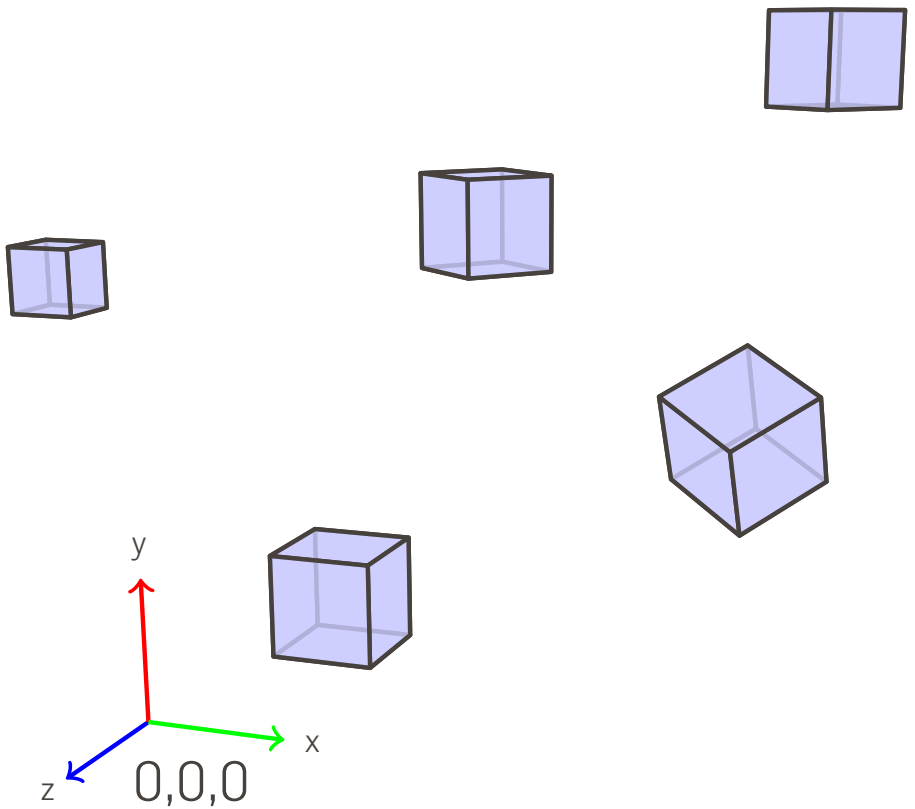
AU COMMENCEMENT, LE POINT...



AU COMMENCEMENT, LE POINT...



AU COMMENCEMENT, LE POINT...



Et Dieu nomma le point...

Et Dieu nomma le point...

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

(et il nota au passage que le point appartenait à $\mathbb{R}^3$)

Et il vit que c'était très réussi.

Alors il nomma le monde...

Et il vit que c'était très réussi.

Alors il nomma le monde...

Base canonique $\{\vec{e_1}, \vec{e_2}, \ldots, \vec{e_n}\}$,

(c'était moins réussi)

Alors il se dit que, en 3D,...

(c'était moins réussi)

Alors il se dit que, en 3D,...

$$\{\vec{u}, \vec{v}, \vec{w}\} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

ça irait bien aussi.

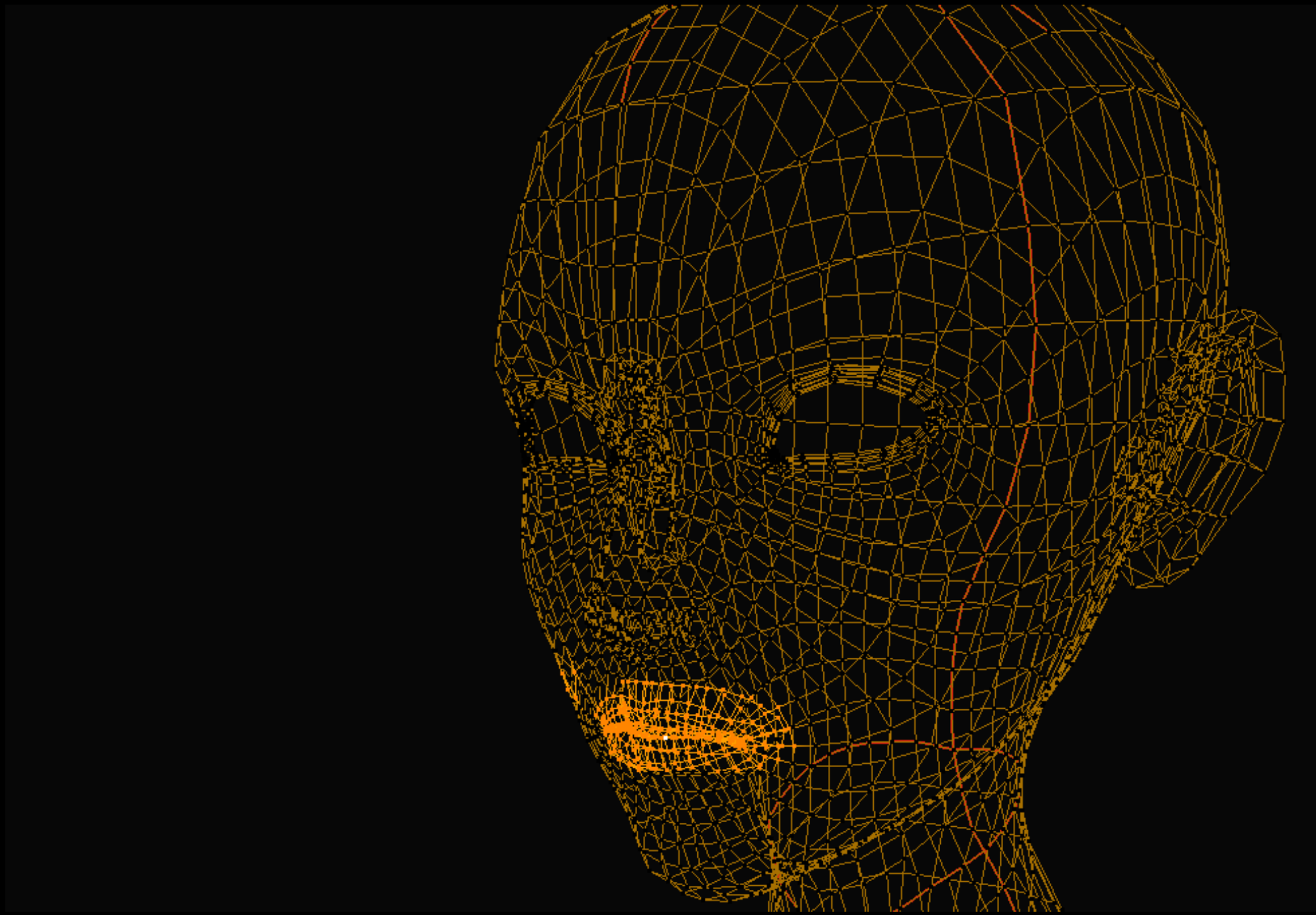
Ensuite, Dieu se gratta le menton :

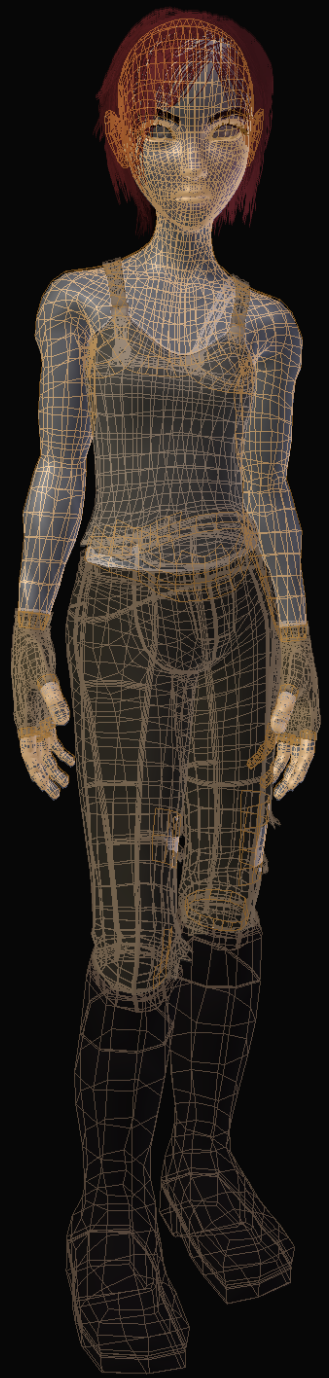
Comment placer tous ces points dans le monde ?

Ensuite, Dieu se gratta le menton :

Comment placer tous ces points dans le monde ?

Alors il créa Sintel...





(**Sintel**, c'est un court métrage d'animation 3D, réalisé avec Blender, et complètement open-source.
Vous pouvez télécharger tout le **source** du film (modèles 3D, textures,...) sur **https://durian.blender.org/**)

Plus sérieusement :

1. Transformations linéaires
2. Coordonnées homogènes
3. Composition de transformations
4. Les projections
5. Rendu 3D

TRANSFORMATIONS

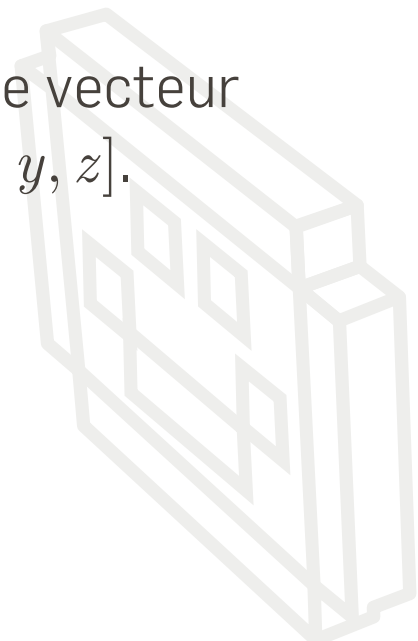
$$T(\mathbf{x}) = A \cdot \mathbf{x}$$

$$T(\mathbf{x}) = A \cdot \mathbf{x} + B$$

$$T(\mathbf{x}) = A \cdot \mathbf{x}$$

$$T(\mathbf{x}) = A \cdot \mathbf{x} + B$$

$\mathbf{x}$ dénote une variable quelconque. Par exemple, le vecteur $\vec{a} = [a_0, a_1, \ldots, a_n]$. Ou bien le vecteur $\vec{x} = [x, y, z]$.

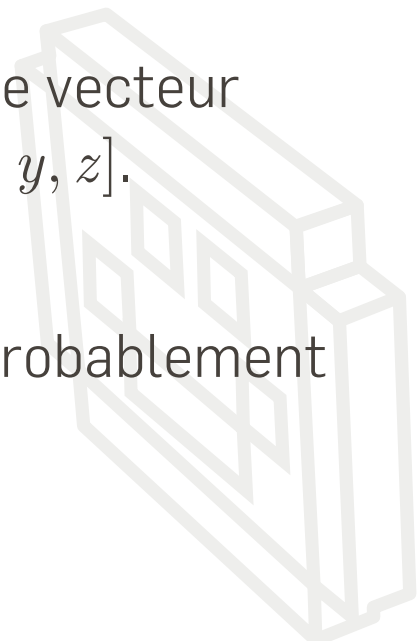


$$T(\mathbf{x}) = A \cdot \mathbf{x}$$

$$T(\mathbf{x}) = A \cdot \mathbf{x} + B$$

$\mathbf{x}$ dénote une variable quelconque. Par exemple, le vecteur $\vec{a} = [a_0, a_1, \ldots, a_n]$. Ou bien le vecteur $\vec{x} = [x, y, z]$.

Si $\mathbf{x}$ et $T(\mathbf{x})$ sont des vecteurs, alors A (et B) sont probablement des matrices !

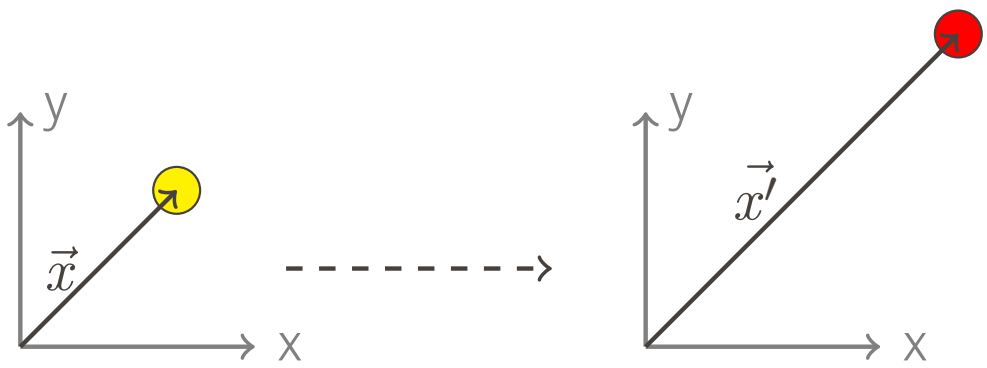


Pour commencer, on se place dans le plan ($\mathbb{R}^2$):

$$\mathbf{x} = \vec{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

EXPRESSION MATRICIELLE DES HOMOTHÉTIES

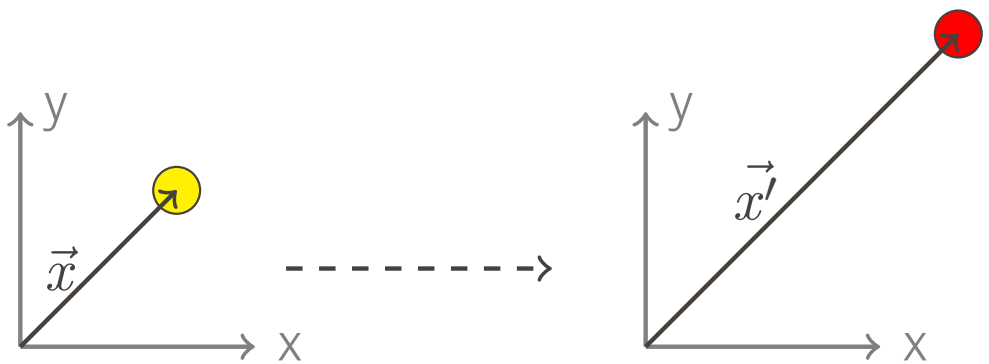
$$\vec{x'} = T(\vec{x}) = 2\vec{x}$$



EXPRESSION MATRICIELLE DES HOMOTHÉTIES

$$\vec{x'} = T(\vec{x}) = 2\vec{x}$$

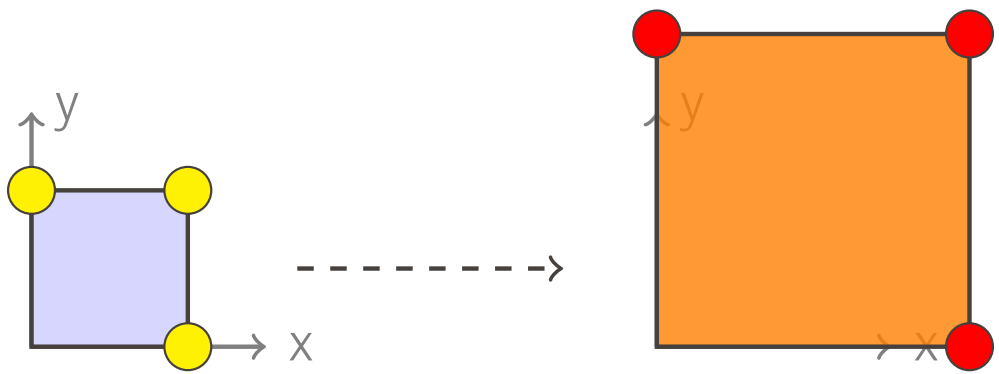
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T(\vec{x}) = \mathbf{A}\vec{x} = 2\mathbf{I}\vec{x} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



EXPRESSION MATRICIELLE DES HOMOTHÉTIES

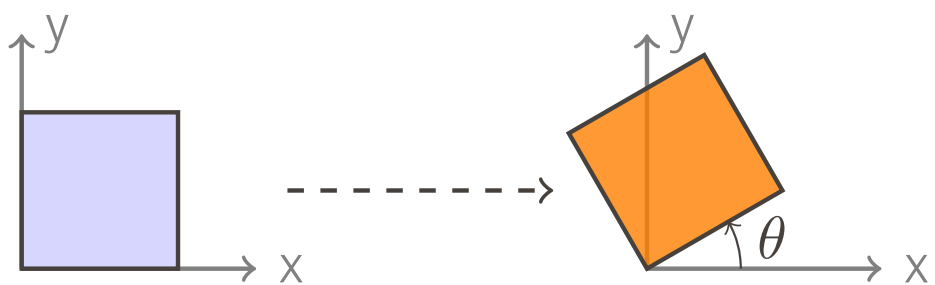
$$\vec{x'} = T(\vec{x}) = 2\vec{x}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T(\vec{x}) = \mathbf{A}\vec{x} = 2\mathbf{I}\vec{x} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



MATRICES DE ROTATION

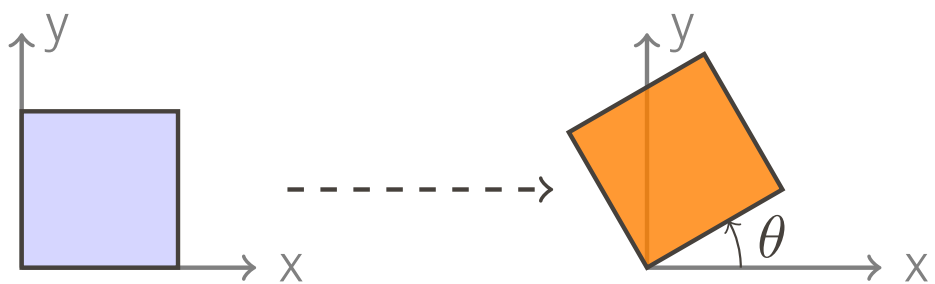
$$\begin{cases} x' = x\cos\theta + y\sin\theta \\ y' = -x\sin\theta + y\cos\theta \end{cases}$$



MATRICES DE ROTATION

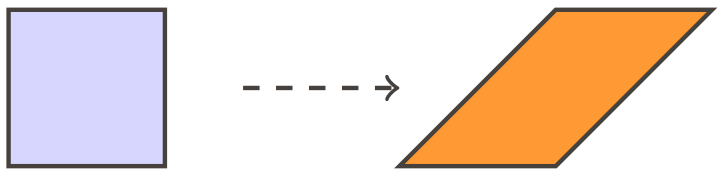
$$\begin{cases} x' = x\cos\theta + y\sin\theta \\ y' = -x\sin\theta + y\cos\theta \end{cases}$$

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

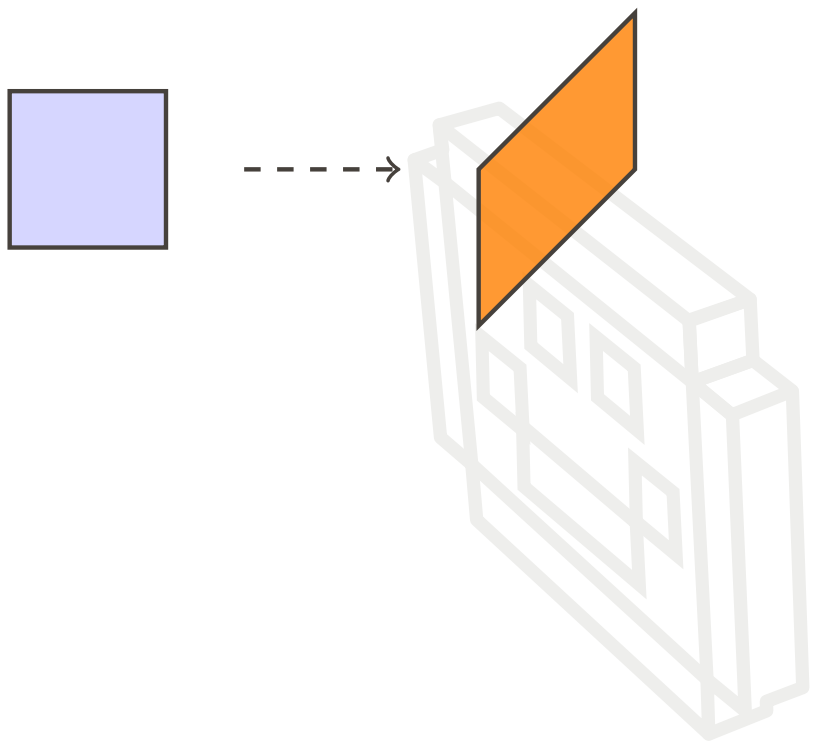


CISAILLEMENT (SHEARING)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Quelle transformation représente la matrice ci-dessous ?

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

Quelle transformation représente la matrice ci-dessous ?

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

Dans $\mathbb{R}^3$,

$$T(\vec{x}) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \vec{x} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Dans $\mathbb{R}^3$,

$$T(\vec{x}) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \vec{x} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

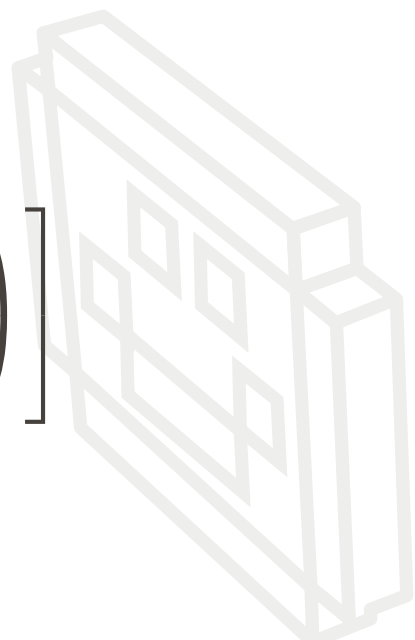
De manière générale, comment construire $\mathbf{A}$ qui représente la transformation linéaire $T(\mathbf{x})$?

Dans $\mathbb{R}^3$,

$$T(\vec{x}) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \vec{x} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

De manière générale, comment construire $\mathbf{A}$ qui représente la transformation linéaire $T(\mathbf{x})$?

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} T(\vec{u}) & T(\vec{v}) & T(\vec{w}) \end{bmatrix} \\ &= \begin{bmatrix} T\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right) & T\left(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right) & T\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right) \end{bmatrix} \end{aligned}$$

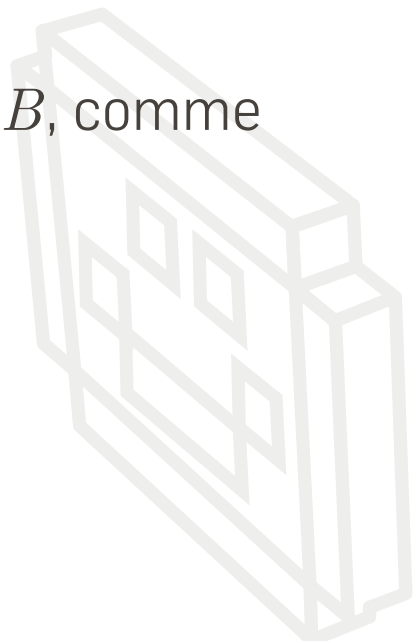


Représenter les transformations sous forme matricielle est particulièrement efficace (opérations ne requiérant que des multiplications et des additions, vectorisation via SSE).

Représenter les transformations sous forme matricielle est particulièrement efficace (opérations ne requiérant que des multiplications et des additions, vectorisation via SSE).

Mais pour les transformations **non linéaires**?

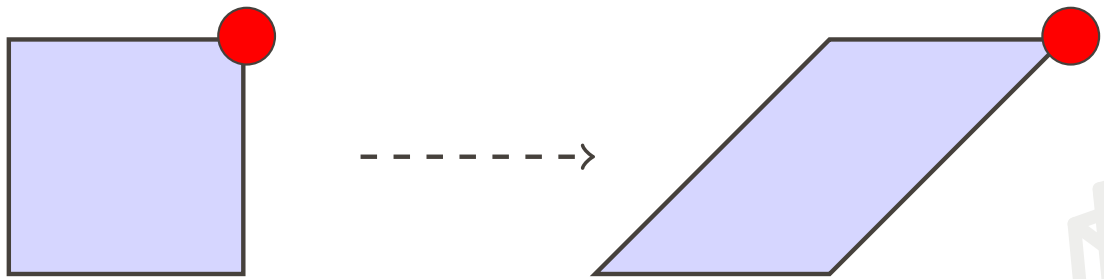
Par ex., les **transformations affines** $T(\mathbf{x}) = A \cdot \mathbf{x} + B$, comme les translations.



COORDONNÉES HOMOGÈNES

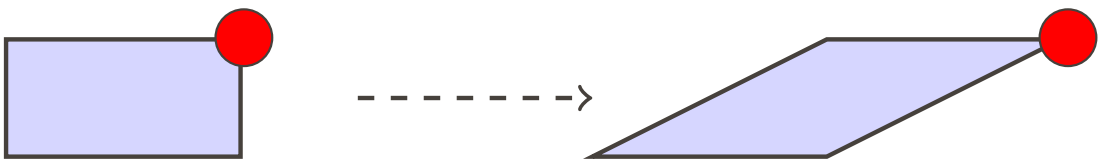
INTUITION

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



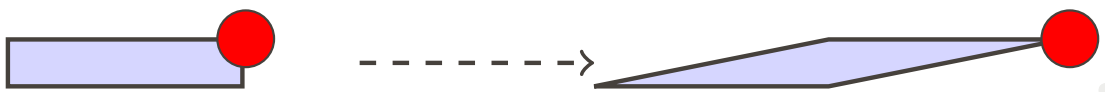
INTUITION

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



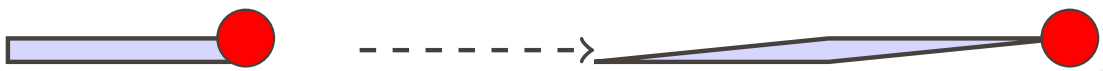
INTUITION

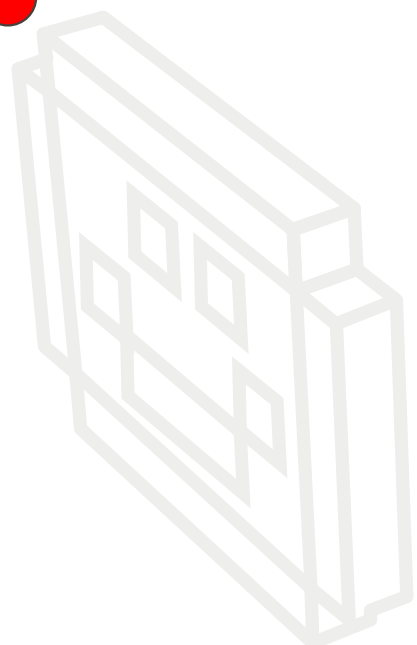
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



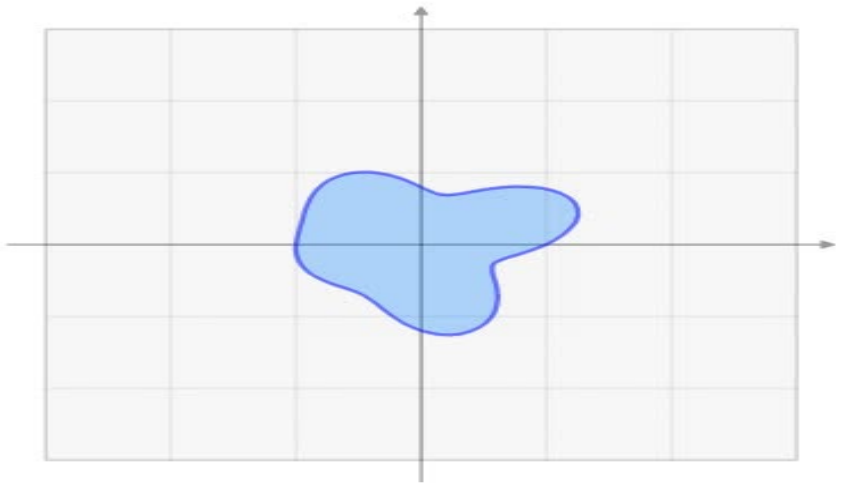
INTUITION

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$





INTUITION



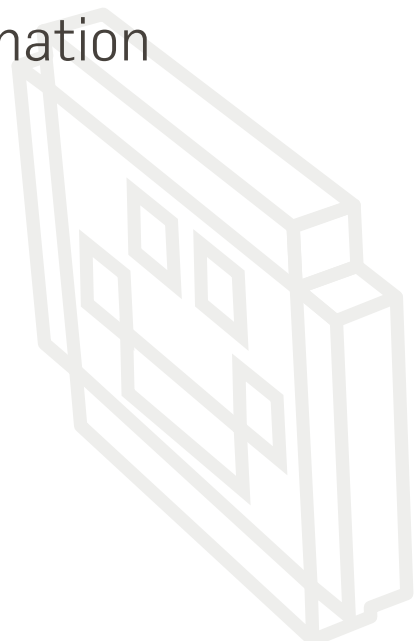
INTUITION

Idée clé : ajouter une dimension (coordonnée w) pour linéariser les **transformations affines** (comme les translations) et les **transformations projectives** (on va y revenir).

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

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

Une translation en 2D (transformation affine) peut être représentée comme un cisaillement en 3D (transformation linéaire).



ESPACE PROJECTIF ET COORDONNÉES HOMOGÈNES

$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ représente **une** des **coordonnées homogènes** du point $\begin{bmatrix} x \\ y \end{bmatrix}$ dans **l'espace projectif** associé à l'espace euclidien $\mathbb{R}^2$.

ESPACE PROJECTIF ET COORDONNÉES HOMOGÈNES

$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ représente **une** des **coordonnées homogènes** du point $\begin{bmatrix} x \\ y \end{bmatrix}$ dans **l'espace projectif** associé à l'espace euclidien $\mathbb{R}^2$.

$\begin{bmatrix} k\,x \\ k\,y \\ k \end{bmatrix}, k \in \mathbb{R} \setminus \{0\}$ représente **l'ensemble des coordonnées homogènes** du point (x, y).

ESPACE PROJECTIF ET COORDONNÉES HOMOGÈNES

$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ représente **une** des **coordonnées homogènes** du point $\begin{bmatrix} x \\ y \end{bmatrix}$ dans **l'espace projectif** associé à l'espace euclidien $\mathbb{R}^2$.

$\begin{bmatrix} k\,x \\ k\,y \\ k \end{bmatrix}, k \in \mathbb{R} \setminus \{0\}$ représente **l'ensemble des coordonnées homogènes** du point (x, y).

Pour repasser en coordonnées euclidiennes, il suffit de diviser par la coordonnée w (**normalisation**) : $\begin{bmatrix} x \\ y \\ w \end{bmatrix} \cong \begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}$.

ESPACE PROJECTIF ET COORDONNÉES HOMOGÈNES

Exemples:

$$\begin{bmatrix} 9 \\ 12 \\ 3 \end{bmatrix} \cong \begin{bmatrix} 45 \\ 60 \\ 25 \end{bmatrix} \cong \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ -2 \\ -3 \end{bmatrix} \cong \begin{bmatrix} 0 \\ 0.66... \\ 1 \end{bmatrix}$$

COORDONNÉES HOMOGÈNES (2)

Si $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ est un point 2D, que penser de $\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$?

COORDONNÉES HOMOGÈNES (2)

Si $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ est un point 2D, que penser de $\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$?

Point à l'infini
ou direction
ou vecteur

COORDONNÉES HOMOGÈNES (2)

Si $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ est un point 2D, que penser de $\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$?

Point à l'infini
ou direction
ou vecteur

L'espace projectif permet d'équiper l'espace euclidien avec des points à l'infini

EN 3D

Coordonnées homogènes: $[x, y, z, w]^\intercal$

Translations :

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

Rotations :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} & R_{3\times3} & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

EN 3D

Homothétie (isotropique si $\alpha = \beta = \gamma$, anisotropique sinon):

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha x \\ \beta y \\ \gamma z \\ 1 \end{bmatrix}$$

COMPOSITION

COMPOSITION DE TRANSFORMATIONS

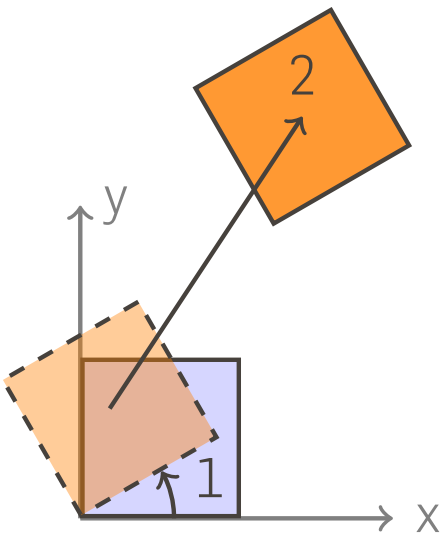
Simple multiplication de matrices ! $\mathbf{B}(\mathbf{A}\vec{x}) = (\mathbf{BA})\vec{x}$

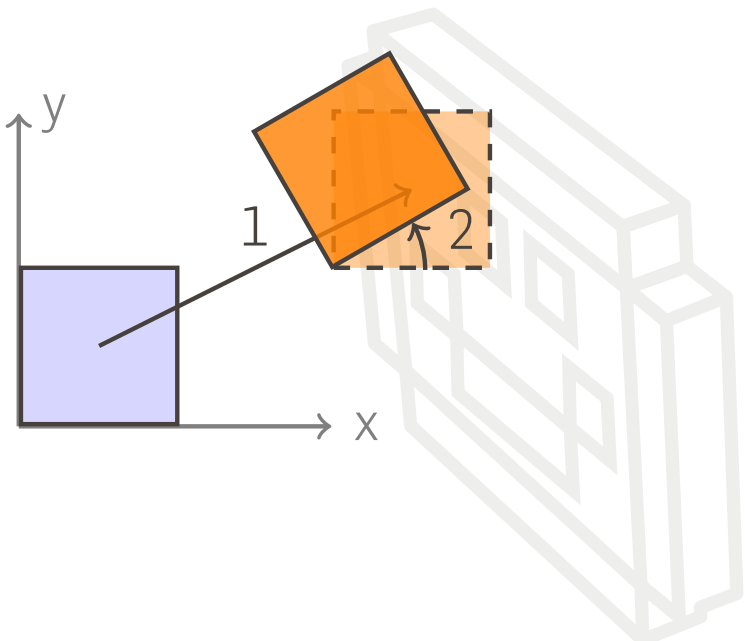
COMPOSITION DE TRANSFORMATIONS

Simple multiplication de matrices ! $\mathbf{B}(\mathbf{A}\vec{x}) = (\mathbf{BA})\vec{x}$

Attention : la multiplication de matrices est associative, mais **pas commutative** en général!

Rotation $\mathbf{A}$ suivie de Translation $\mathbf{B} \equiv \mathbf{BA}\vec{x}$ et non $\mathbf{AB}\vec{x}$!





MATRICES DE TRANSFORMATION GÉNÉRALISÉES

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = ?$$

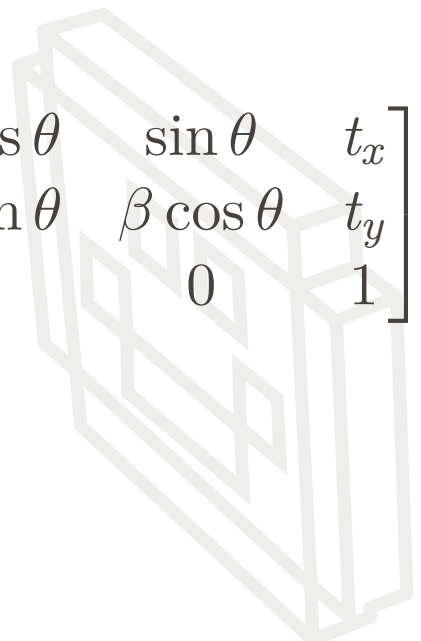
MATRICES DE TRANSFORMATION GÉNÉRALISÉES

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & t_x \\ -\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

MATRICES DE TRANSFORMATION GÉNÉRALISÉES

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & t_x \\ -\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha\cos\theta & \sin\theta & t_x \\ -\sin\theta & \beta\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

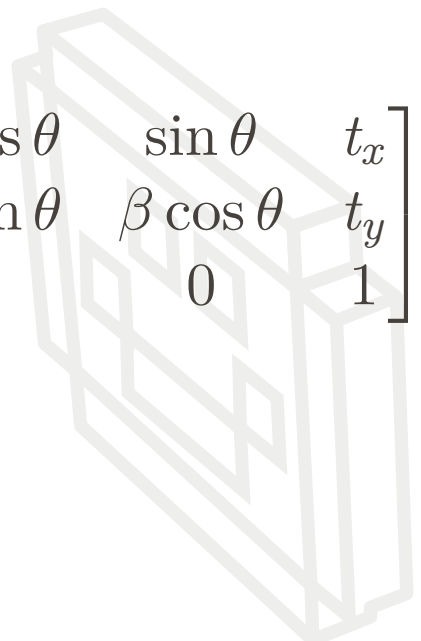


MATRICES DE TRANSFORMATION GÉNÉRALISÉES

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & t_x \\ -\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha\cos\theta & \sin\theta & t_x \\ -\sin\theta & \beta\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Dans quel ordre ?



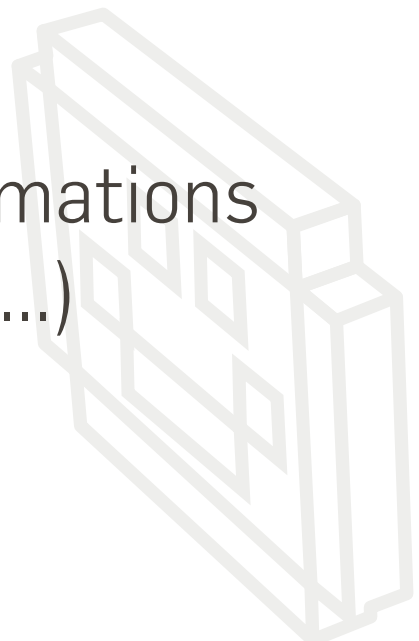
MATRICES DE TRANSFORMATIONS GÉNÉRALISÉES EN 3D

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} & & & T_{1\times 3} \\ & R_{3\times 3} & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

MATRICES DE TRANSFORMATIONS GÉNÉRALISÉES EN 3D

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} & & & T_{1\times 3} \\ & R_{3\times 3} & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Représentation standard des transformations dans l'espace (OpenGL, Direct3D...)

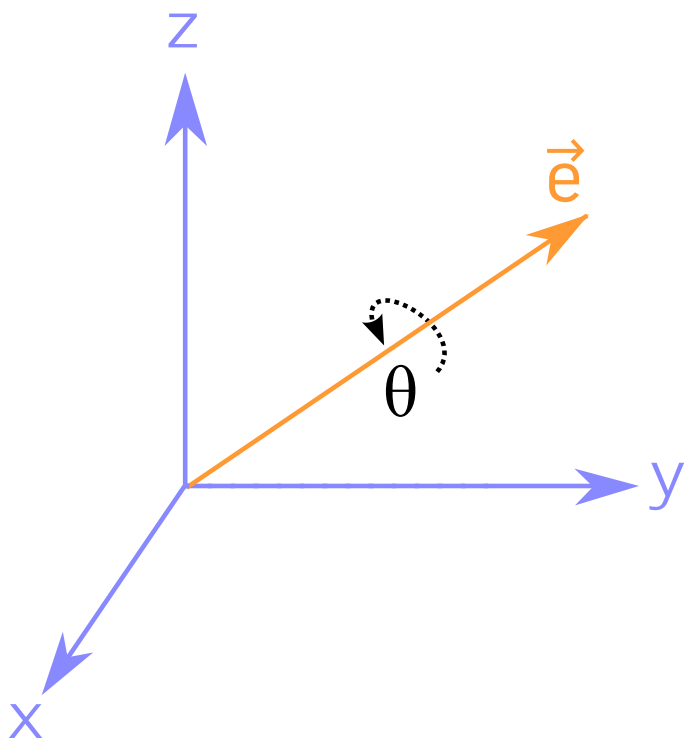


NOTE : ROTATIONS EN 3D

Les rotations dans l'espace sont difficiles à représenter !

NOTE : ROTATIONS EN 3D

Dans l'espace, une rotation θ autour de l'axe défini par le vecteur $\vec{e} = (e_x, e_y, e_z)$:



$$\begin{bmatrix} e_x e_x(1-\cos\theta)+\cos\theta & e_y e_x(1-\cos\theta)-e_z\sin\theta & e_z e_x(1-\cos\theta)+e_y\sin\theta \\ e_x e_y(1-\cos\theta)+e_z\sin\theta & e_y e_y(1-\cos\theta)+\cos\theta & e_z e_y(1-\cos\theta)-e_x\sin\theta \\ e_x e_z(1-\cos\theta)-e_y\sin\theta & e_y e_z(1-\cos\theta)+e_x\sin\theta & e_z e_z(1-\cos\theta)+\cos\theta \end{bmatrix}$$

NOTE : ROTATIONS EN 3D

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

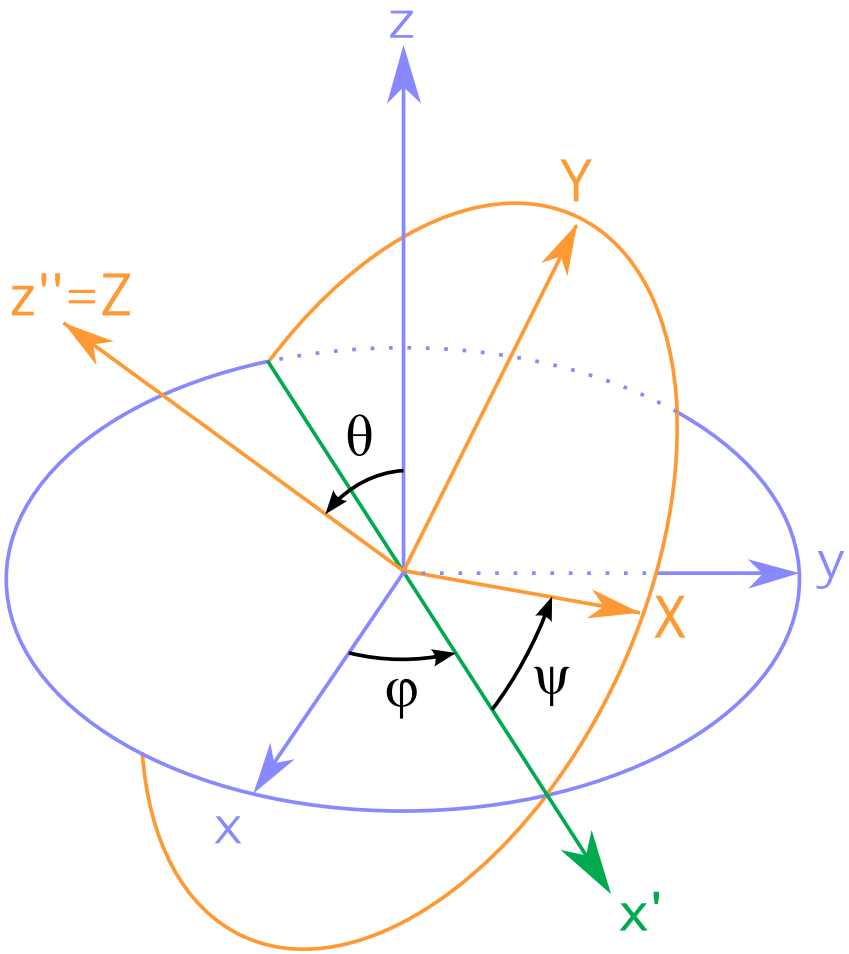
$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

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

$$\mathbf{R} = \mathbf{R}_z(\theta_z)\,\mathbf{R}_y(\theta_y)\,\mathbf{R}_x(\theta_x)?$$

NOTE : ROTATIONS EN 3D

Angles d'Euler (φ, θ, ψ) : $\mathbf{R} = \mathbf{R}_{z''}(\psi)\,\mathbf{R}_{x'}(\theta)\,\mathbf{R}_{z}(\varphi)$



NOTE : ROTATIONS EN 3D

Quaternions (étroitement liés aux nombres complexes) :

$$\mathbf{q} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ w \end{bmatrix}$$

$$q_x = e_x \sin\left(\frac{\theta}{2}\right)$$

$$q_y = e_y \sin\left(\frac{\theta}{2}\right)$$

$$q_z = e_z \sin\left(\frac{\theta}{2}\right)$$

$$w = \cos\left(\frac{\theta}{2}\right)$$

ROTATIONS 3D: EN RÉSUMÉ

Quaternions : très bonnes propriétés mathématiques, peu intuitifs

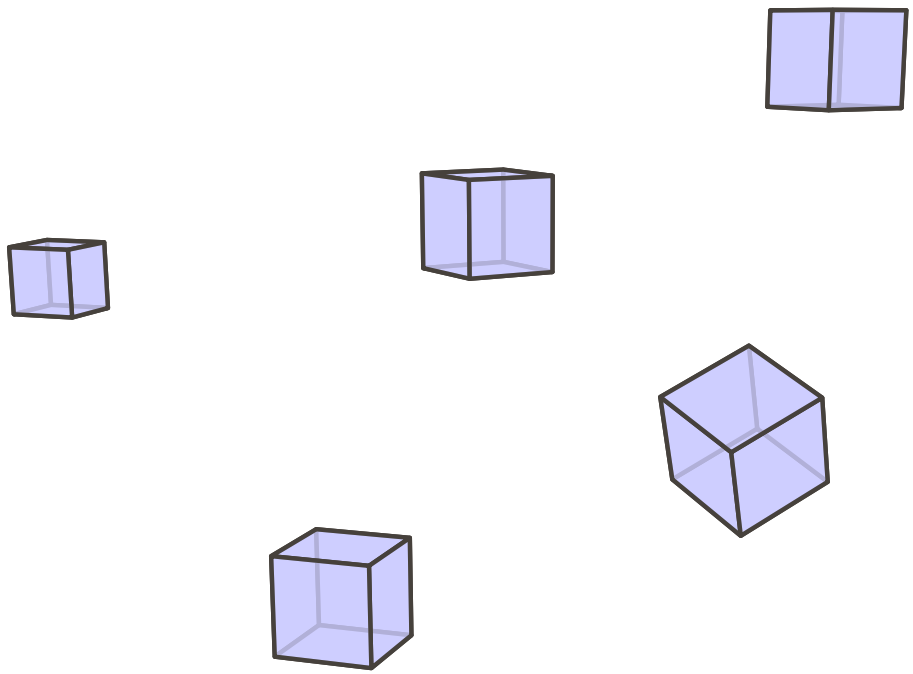
Matrices de rotation : bonnes propriétés mathématiques, courantes, peu intuitives

Angles d'Euler : (parfois) intuitifs, mathématiquement dangereux

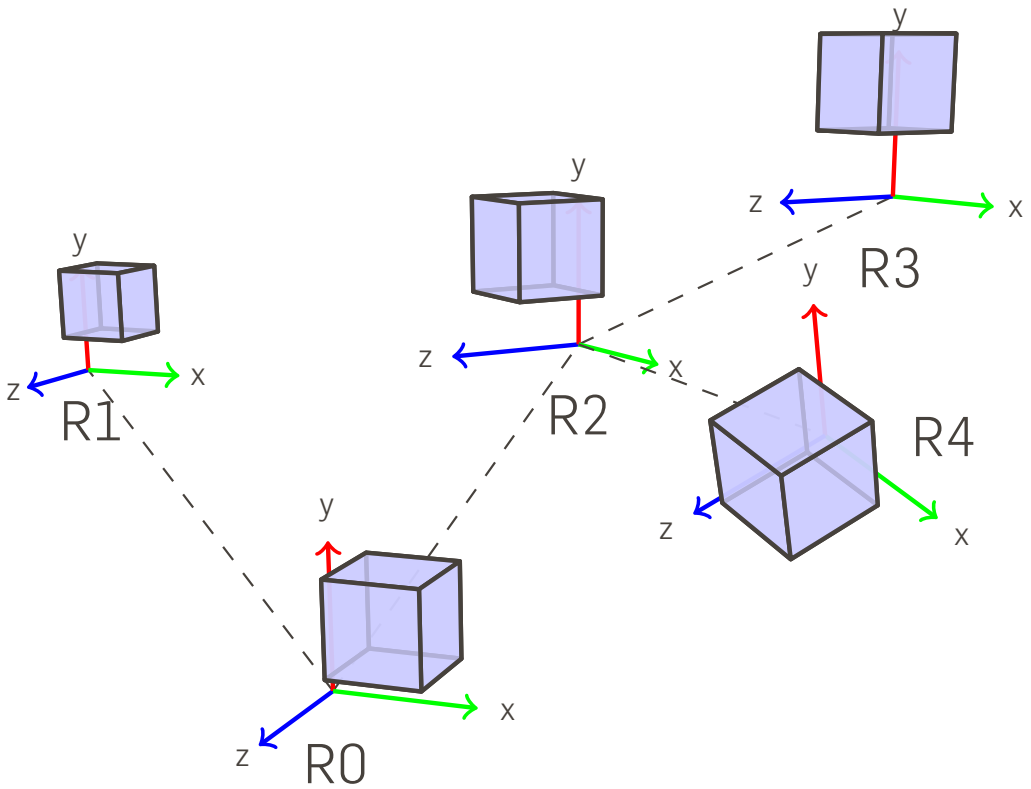
Autres formalismes, souvent domaine-spécifique (yaw-pitch-roll)

Questions à l'examen : uniquement sur le vocabulaire

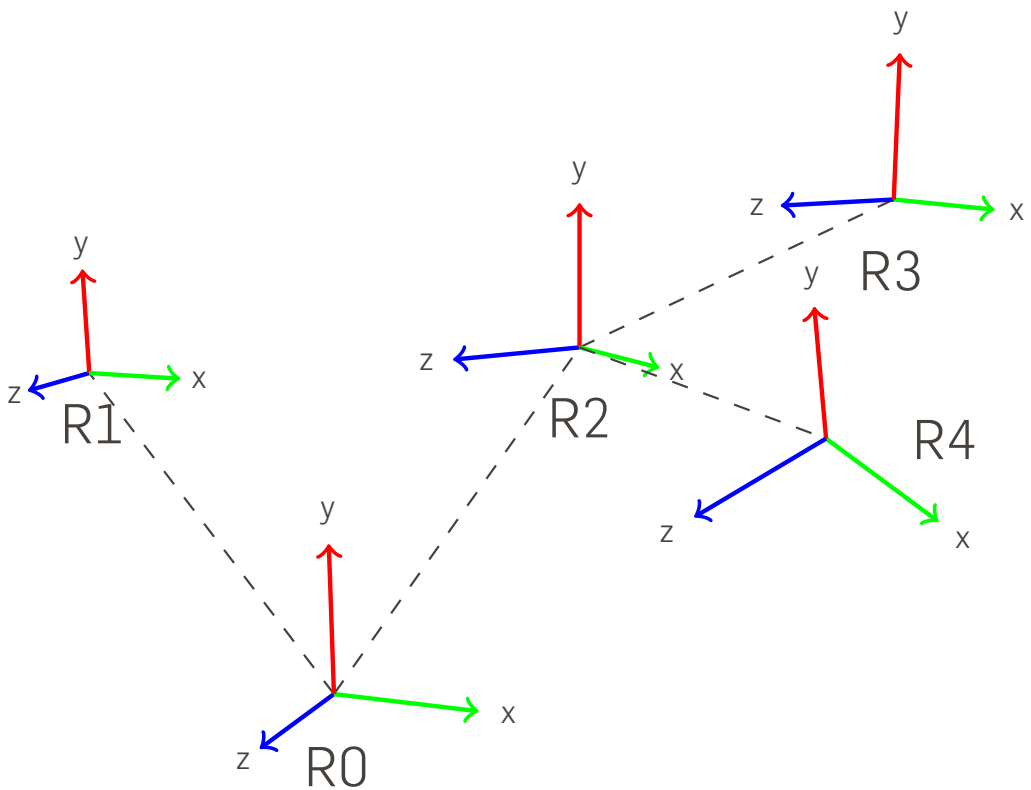
SCENE GRAPH



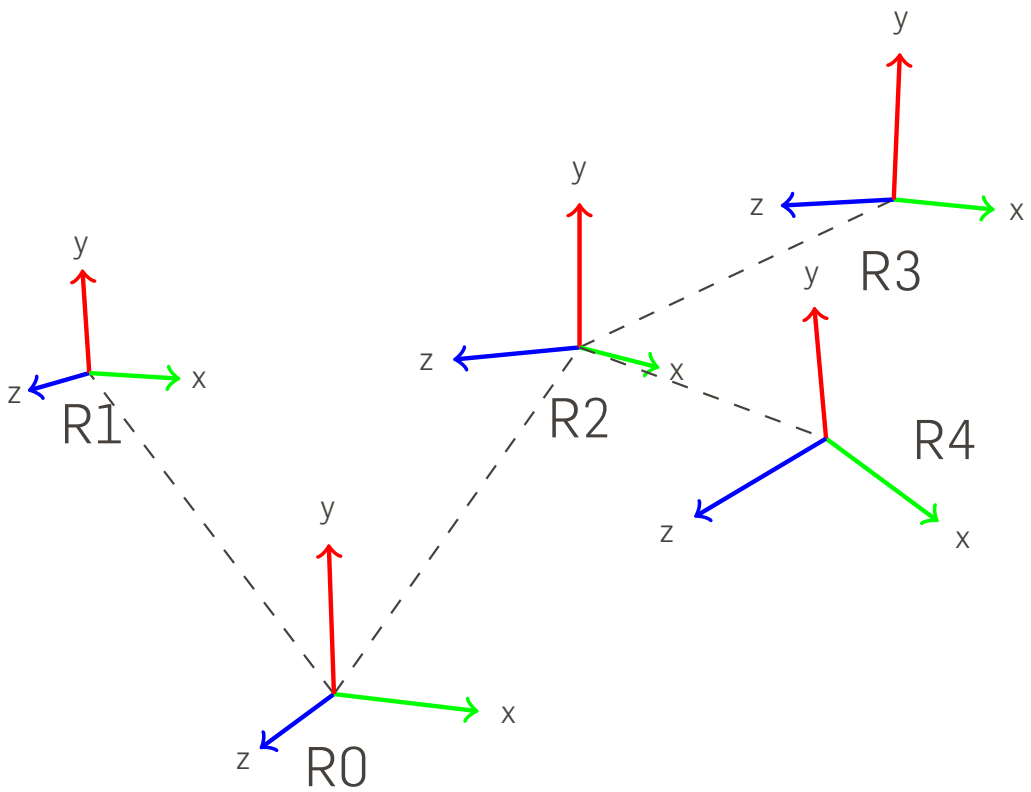
SCENE GRAPH



SCENE GRAPH



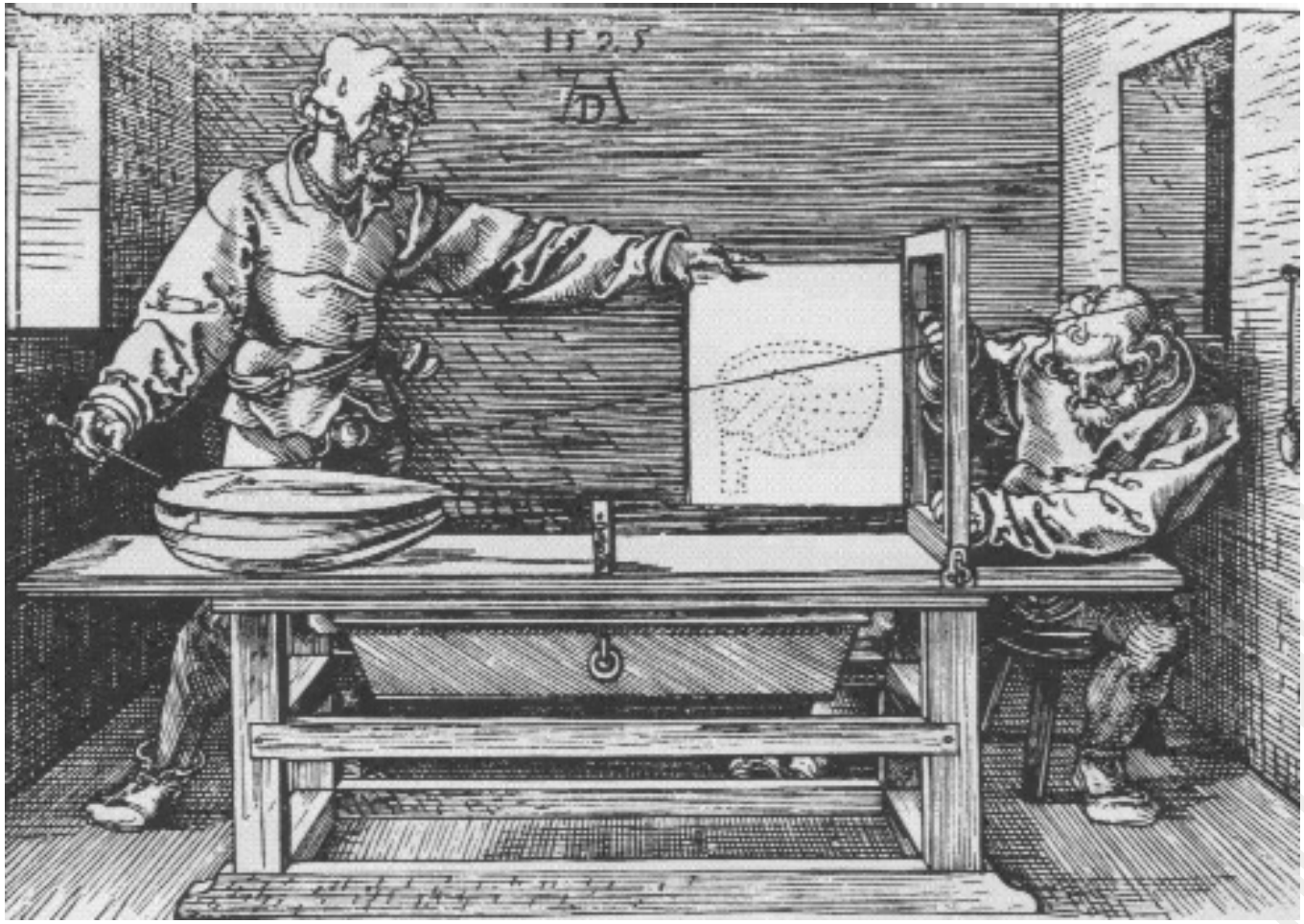
SCENE GRAPH



`pushMatrix()`, `popMatrix()`!

PROJECTIONS

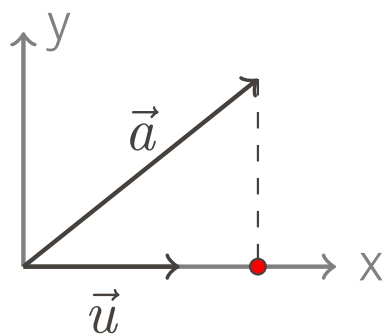
PROJECTION



Projection $\equiv$ fonction idempotente $\equiv P \cdot P = P$

RAPPEL: PRODUIT SCALAIRE

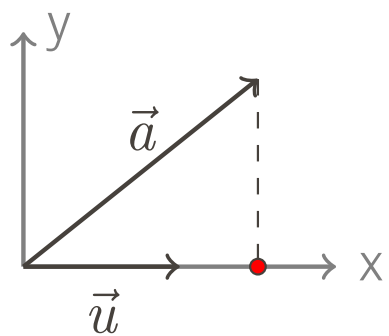
$$\vec{a} \cdot \vec{b} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \cdot \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \|\vec{a}\|\|\vec{b}\| cos\theta = a_x b_x + a_y b_y + a_z b_z$$



Projection de $\vec{a}$ sur $\vec{u}$: $(\vec{a} \cdot \vec{u})\vec{u}$

RAPPEL: PRODUIT SCALAIRE

$$\vec{a} \cdot \vec{b} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \cdot \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \|\vec{a}\|\|\vec{b}\| cos\theta = a_x b_x + a_y b_y + a_z b_z$$



Projection de $\vec{a}$ sur $\vec{u}$: $(\vec{a} \cdot \vec{u})\vec{u}$

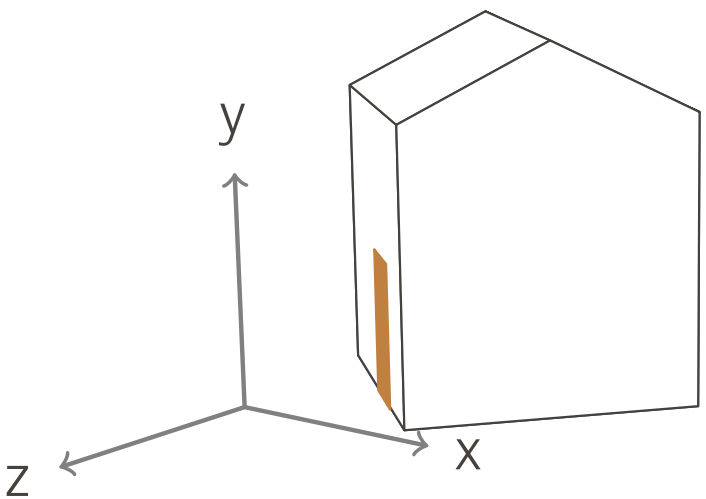
Que se passe-t'il pour des vecteurs orthogonaux ?

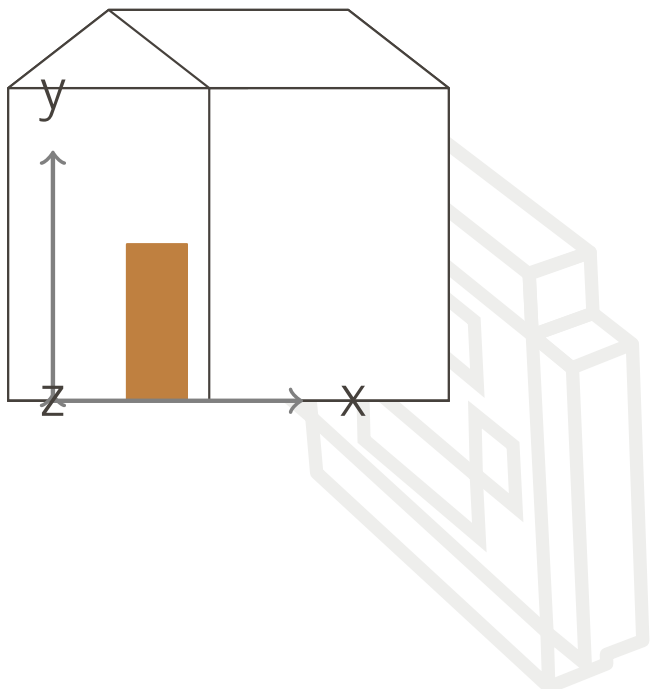
PROJECTION ORTHOGRAPHIQUE

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

PROJECTION ORTHOGRAPHIQUE

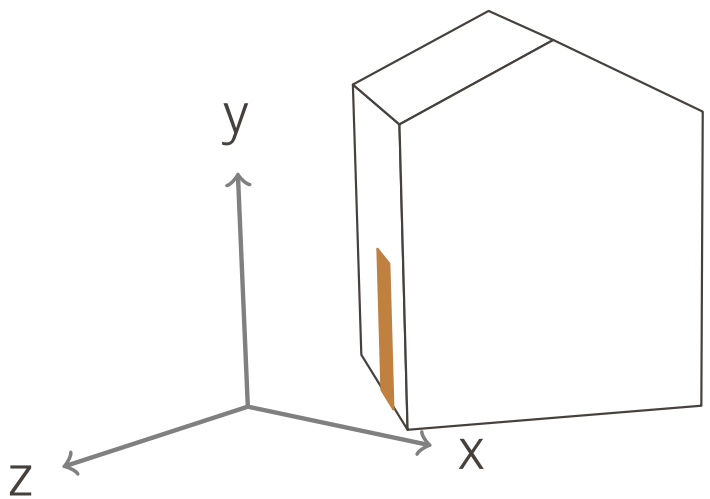
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

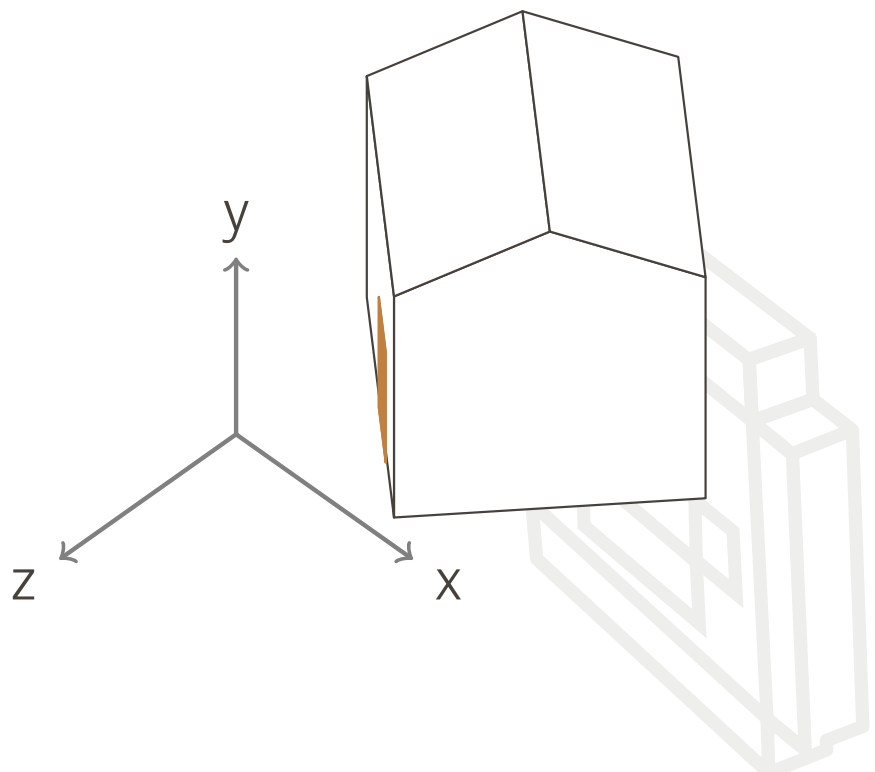




PROJECTION ORTHOGRAPHIQUE

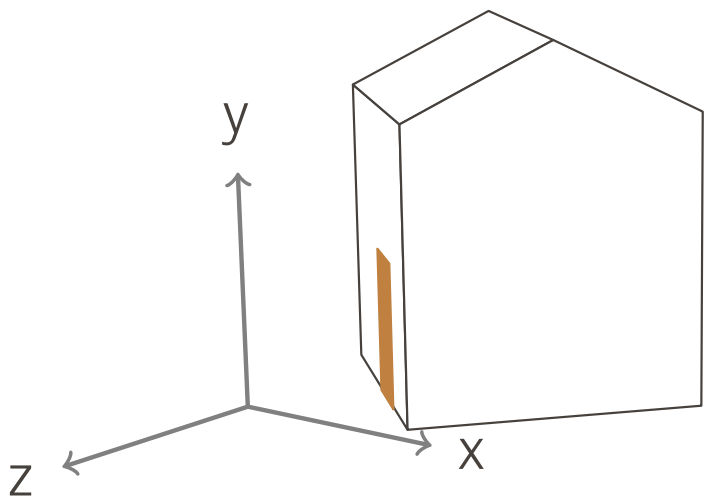
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

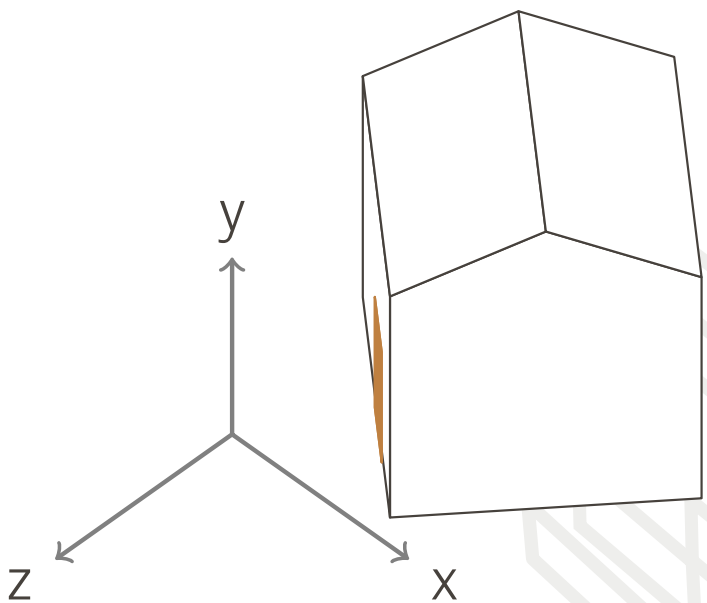




PROJECTION ORTHOGRAPHIQUE

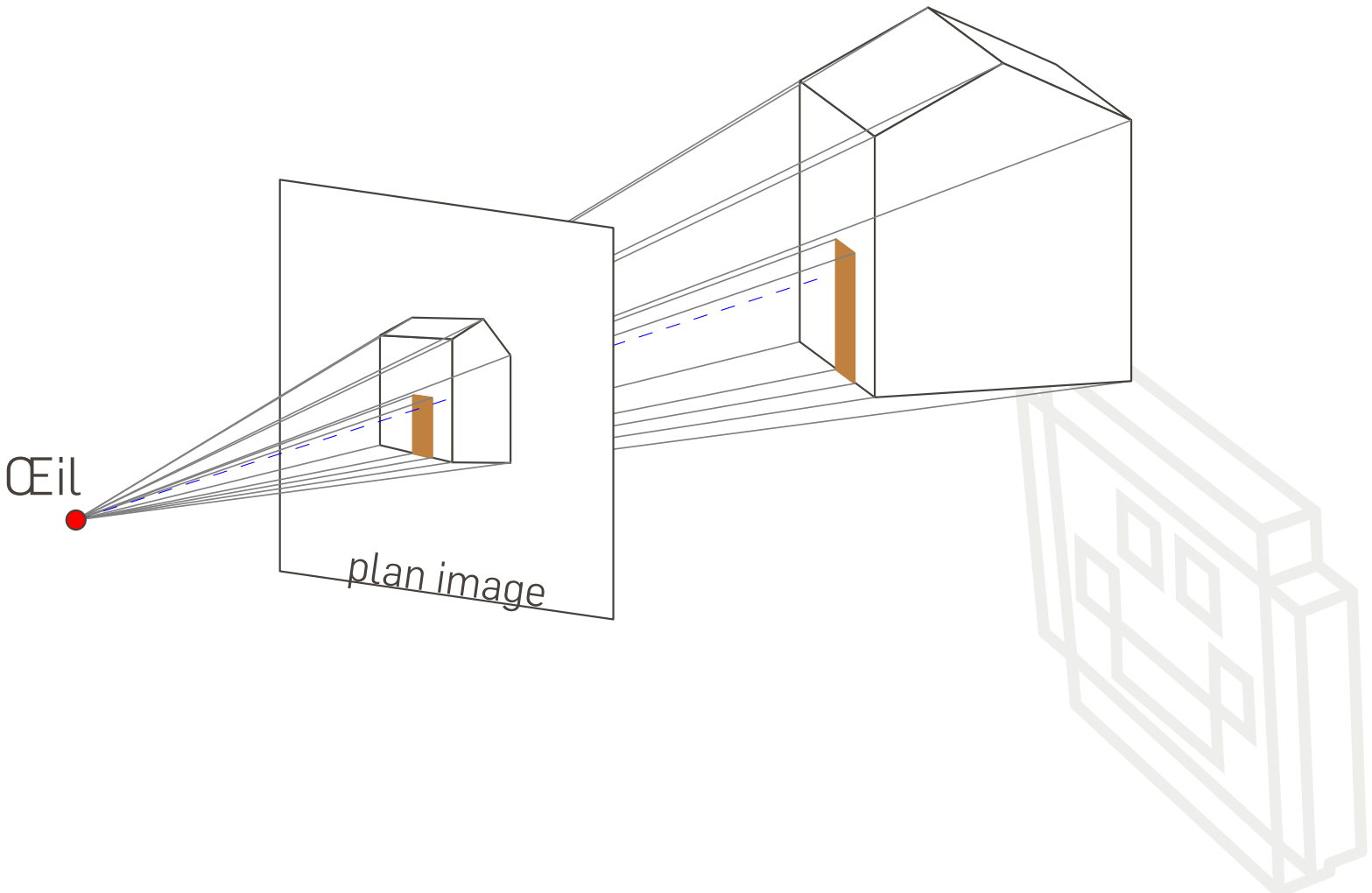
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$



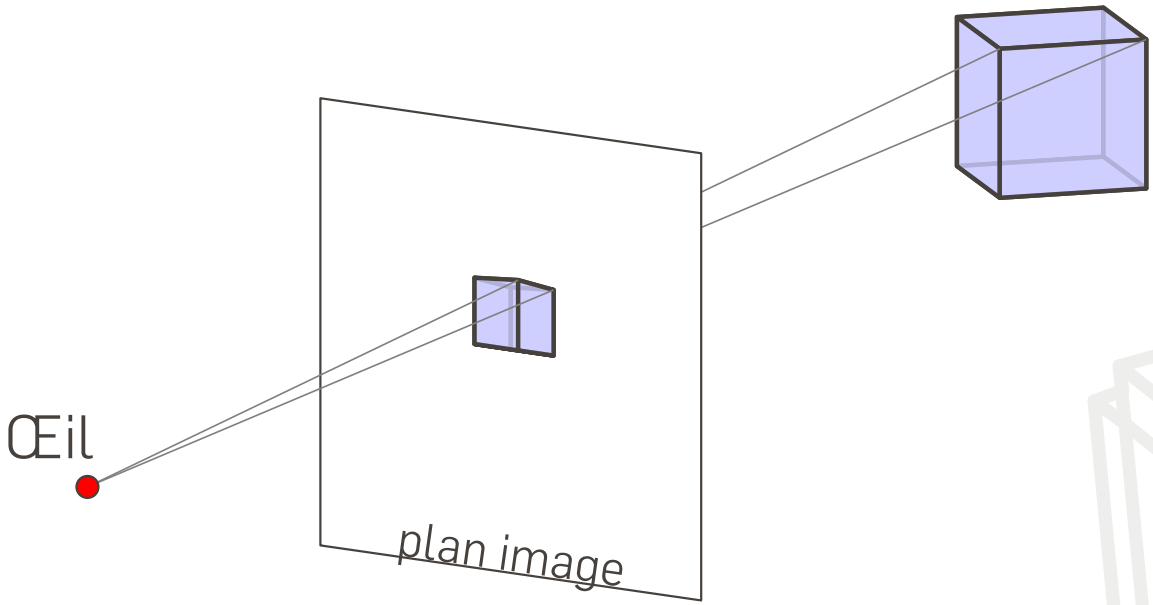


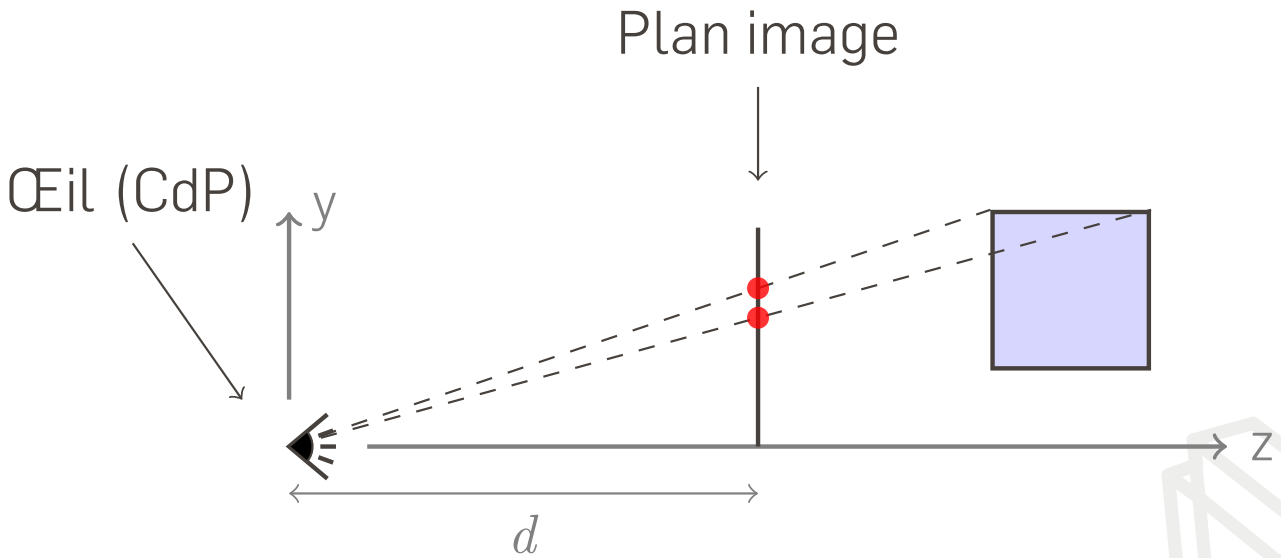
Transformation linéaire

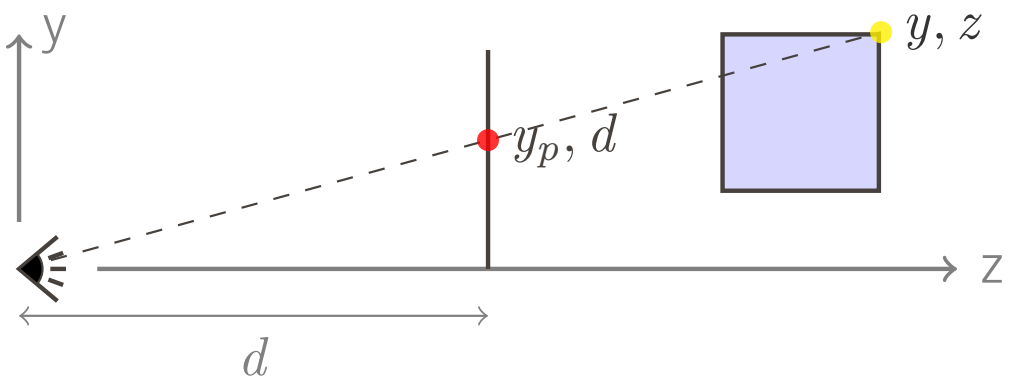
PROJECTION PERSPECTIVE



PROJECTION PERSPECTIVE



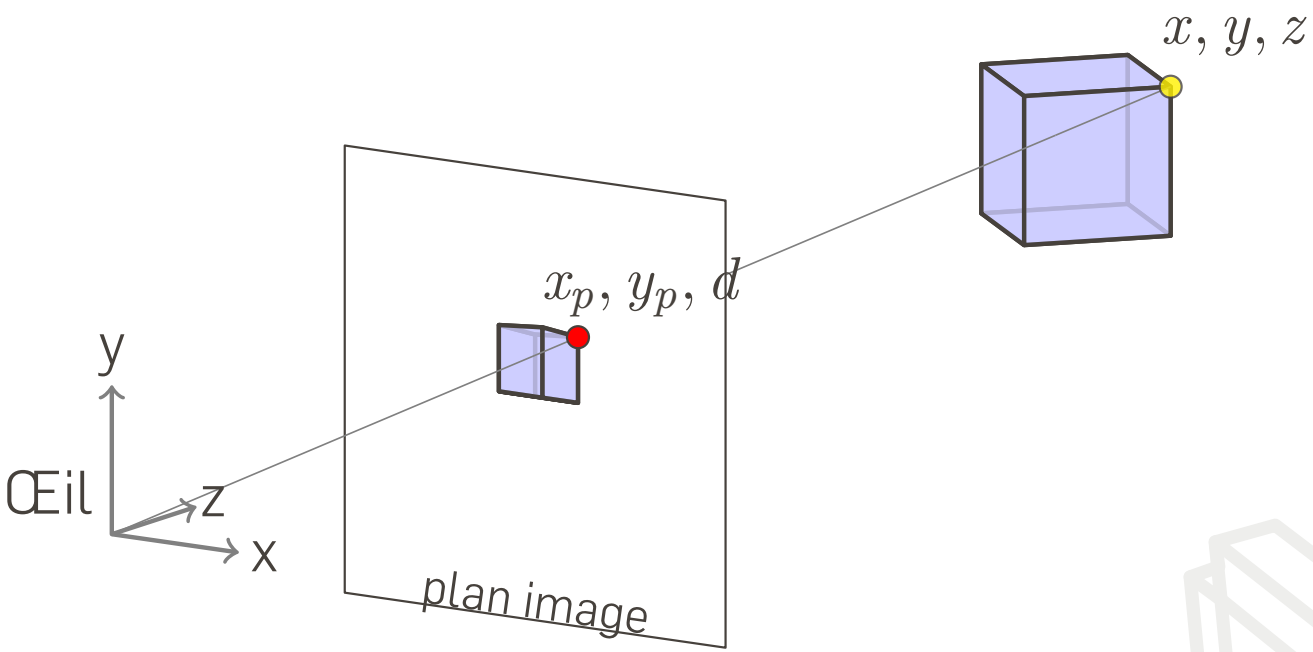

Plan image
Œil (CdP)
y
z
d



$$\frac{y_p}{y} = \frac{d}{z} \rightarrow y_p = \frac{d}{z} y$$

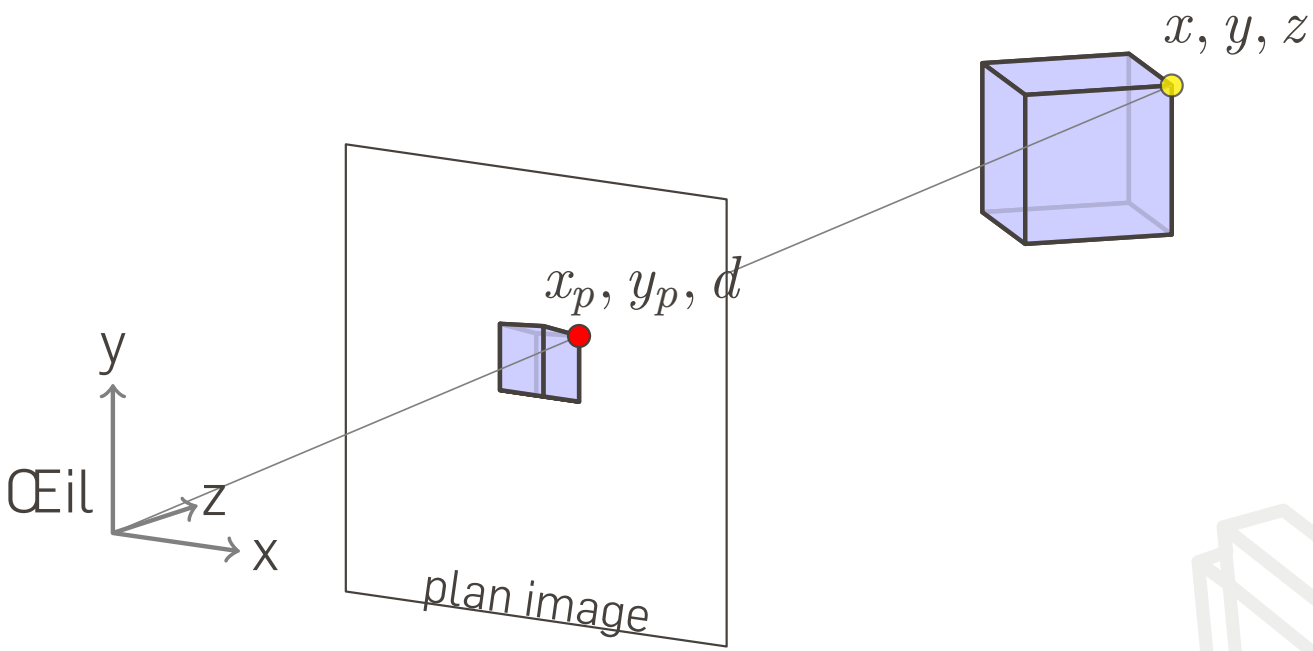
$$\begin{bmatrix} y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} \frac{d}{z} y \\ d \end{bmatrix}$$

EN 3D



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \alpha \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \alpha \end{bmatrix} \cong \begin{bmatrix} x/\alpha \\ y/\alpha \\ z/\alpha \\ 1 \end{bmatrix}$$

EN 3D



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{z}{d} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix} \cong \begin{bmatrix} \frac{dx}{z} \\ \frac{dy}{z} \\ d \\ 1 \end{bmatrix}$$

EN 3D

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{z}{d} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix} \cong \begin{bmatrix} \frac{dx}{z} \\ \frac{dy}{z} \\ d \\ 1 \end{bmatrix}$$

Quel est le problème ?

EN 3D

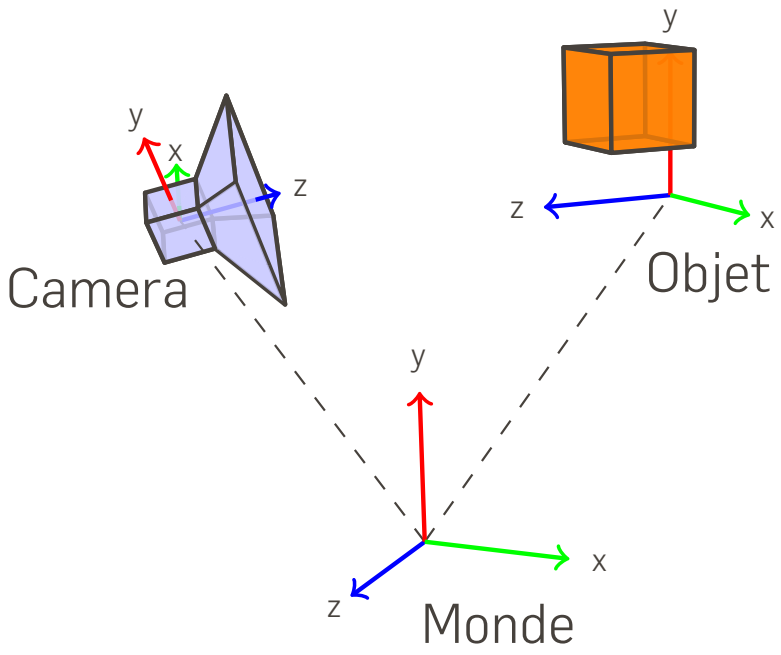
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{z}{d} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix} \cong \begin{bmatrix} \frac{dx}{z} \\ \frac{dy}{z} \\ d \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix} \cong \begin{bmatrix} \frac{dx}{z} \\ \frac{dy}{z} \\ d \\ 1 \end{bmatrix}$$

Que représente d ?

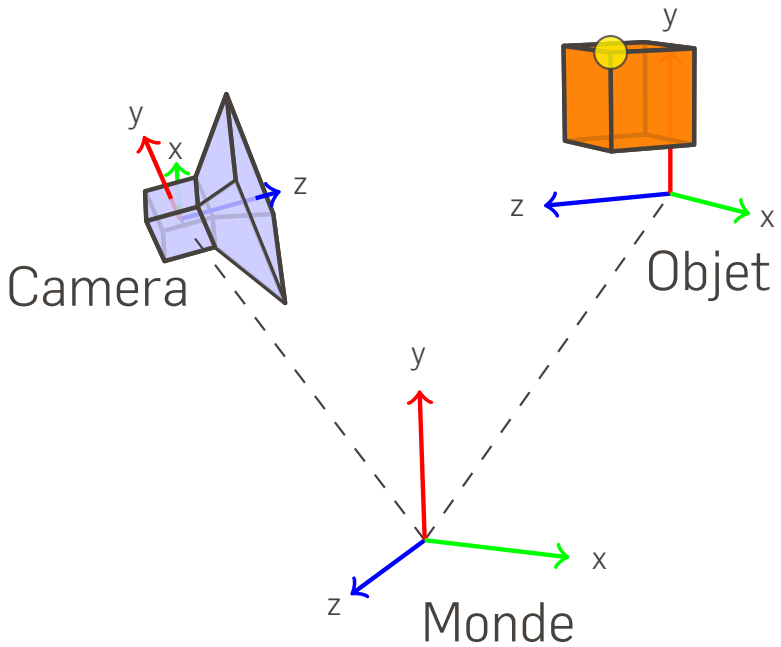
Que représente d ? La **distance focale** f de notre caméra

MATRICE MODEL-VIEW-PROJECTION



$$\mathcal{M}_{\text{MVP}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix}}_{\text{Matrice de projection}} \underbrace{\begin{bmatrix} & & & T_{cam} \\ & R_{cam} & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}}_{\text{Matrice vue}} \underbrace{\begin{bmatrix} & & & T_{obj} \\ & R_{obj} & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Matrice objet}}$$

MATRICE MODEL-VIEW-PROJECTION



$$\begin{bmatrix} x_p \\ y_p \\ f \\ 1 \end{bmatrix} \cong \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} & & & T_{cam} \\ & R_{cam} & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} & & & T_{obj} \\ & R_{obj} & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Vous voilà équipés pour le projet.

RENDU 3D

RENDU 3D

Visibilité

Illumination des objets

Ombres

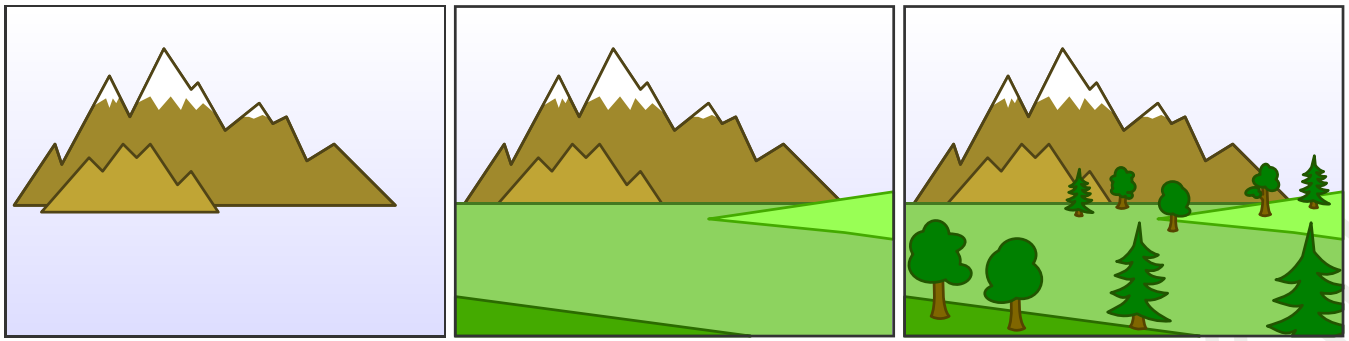
Textures

Pipeline de rendu

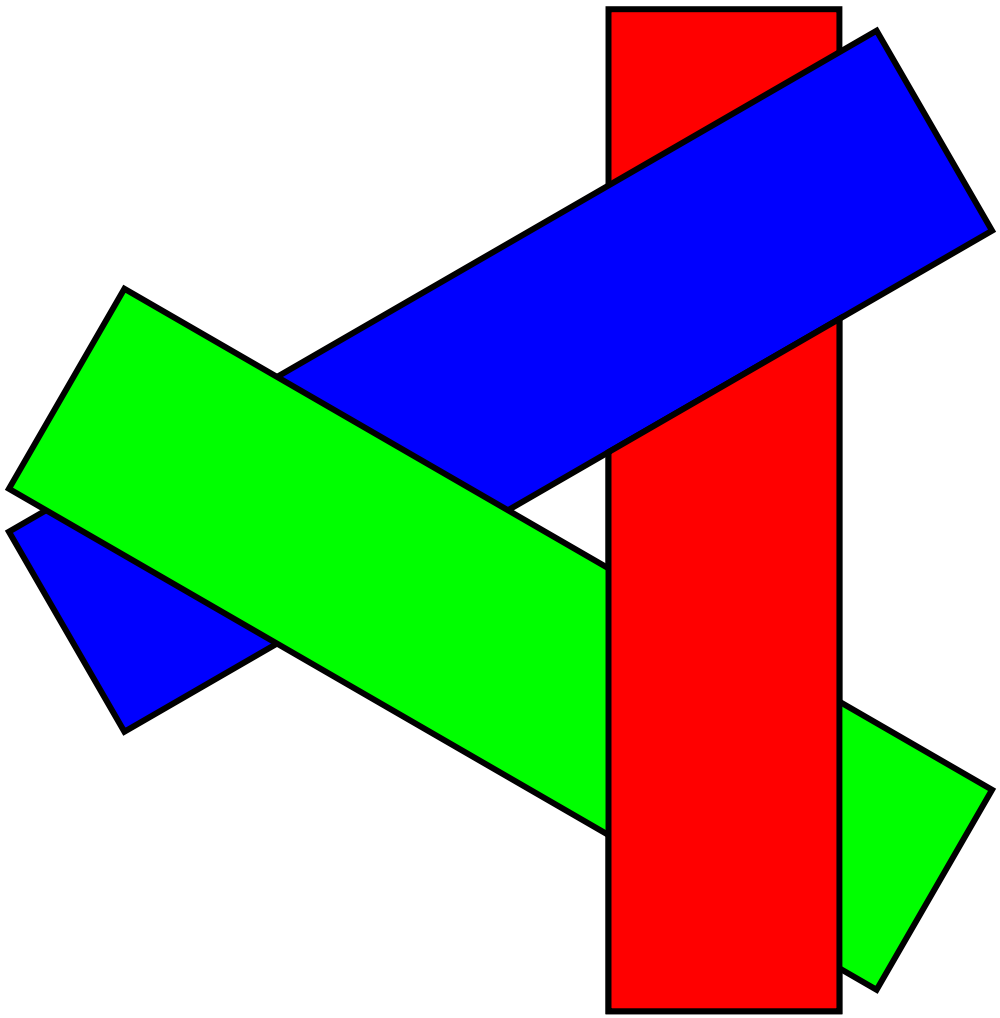
Shaders

PROBLÈME DE LA VISIBILITÉ

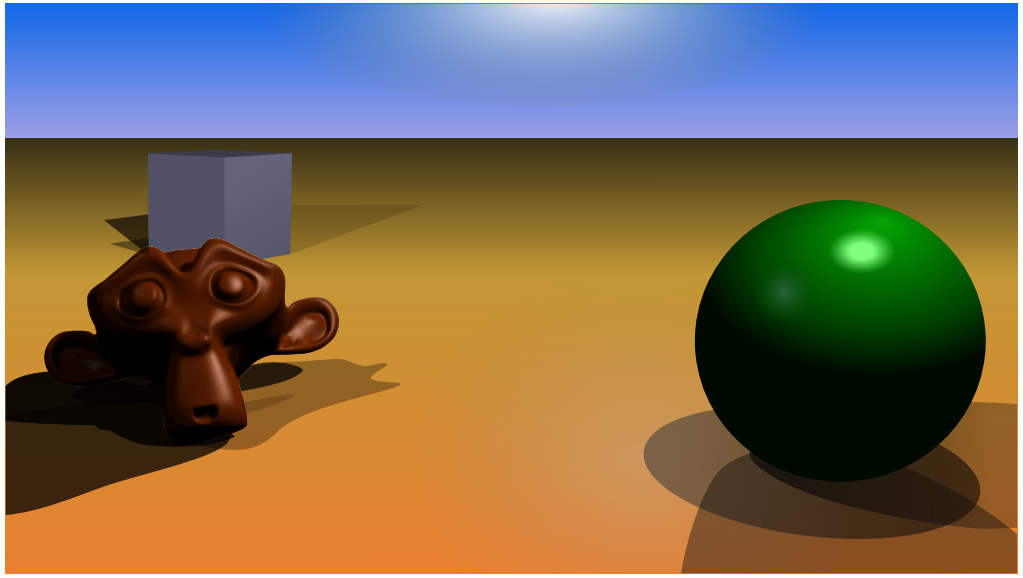
Approche naïve: algorithme du peintre

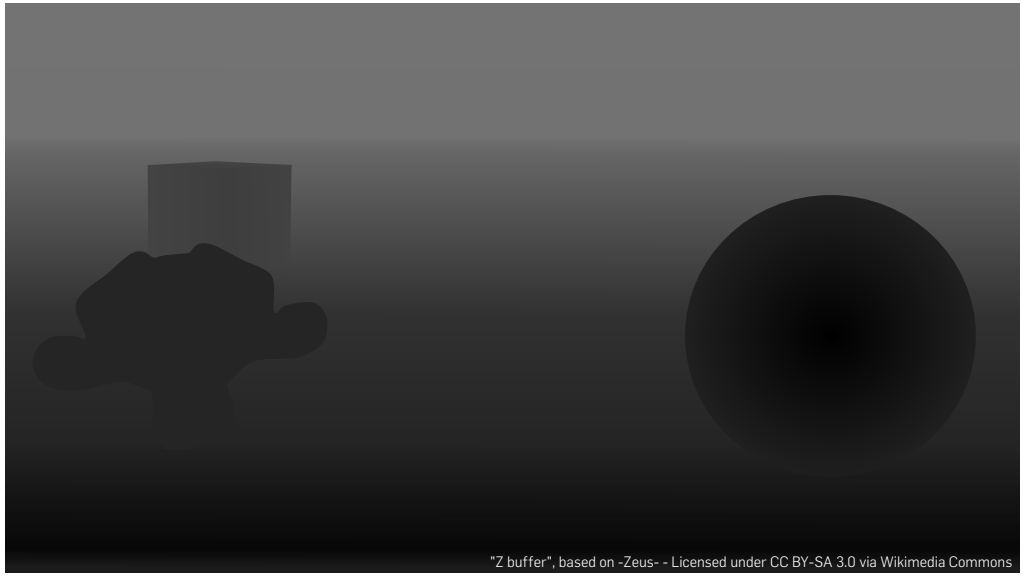


PROBLÈME DE LA VISIBILITÉ



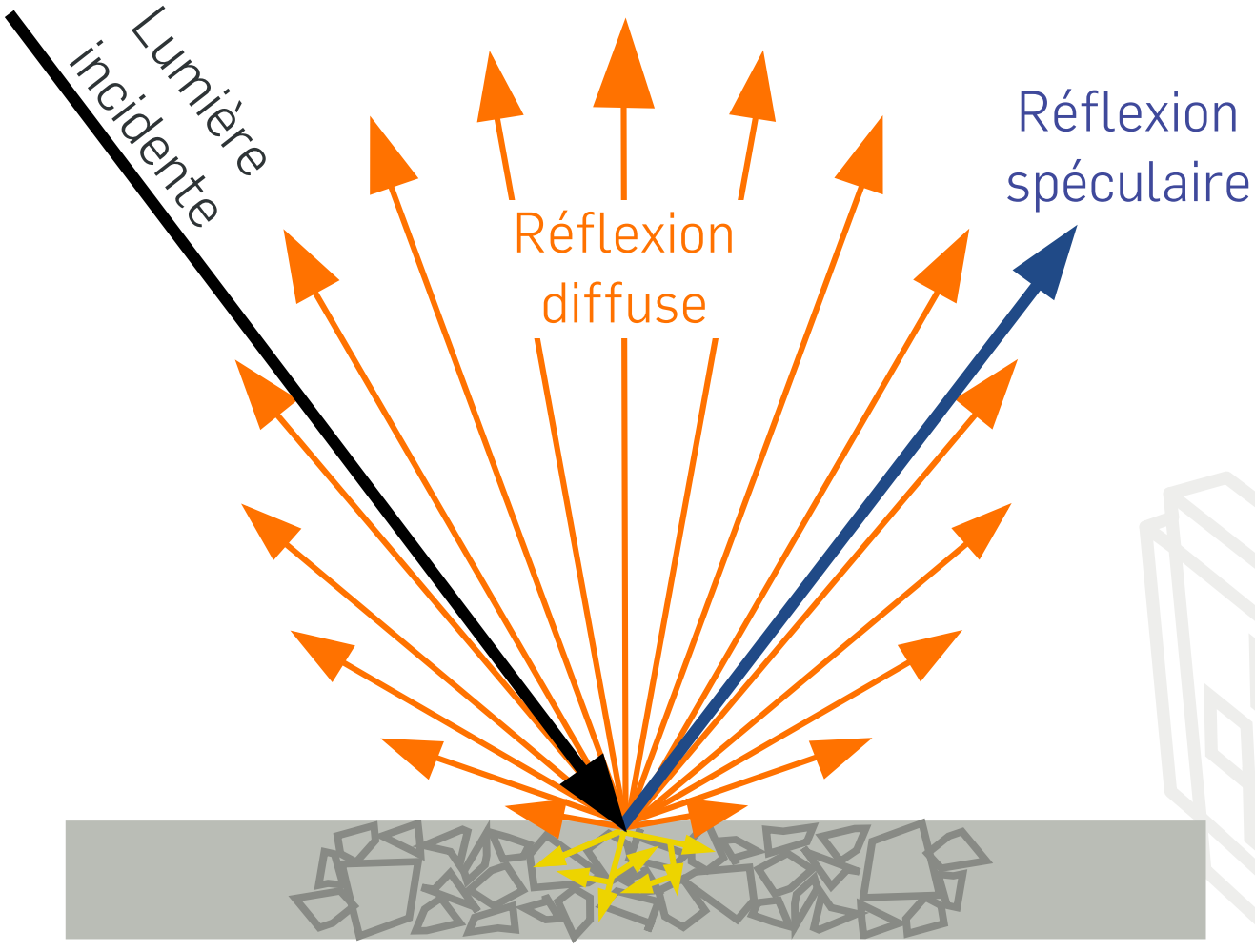
Z-BUFFERING





"Z buffer", based on -Zeus- - Licensed under CC BY-SA 3.0 via Wikimedia Commons

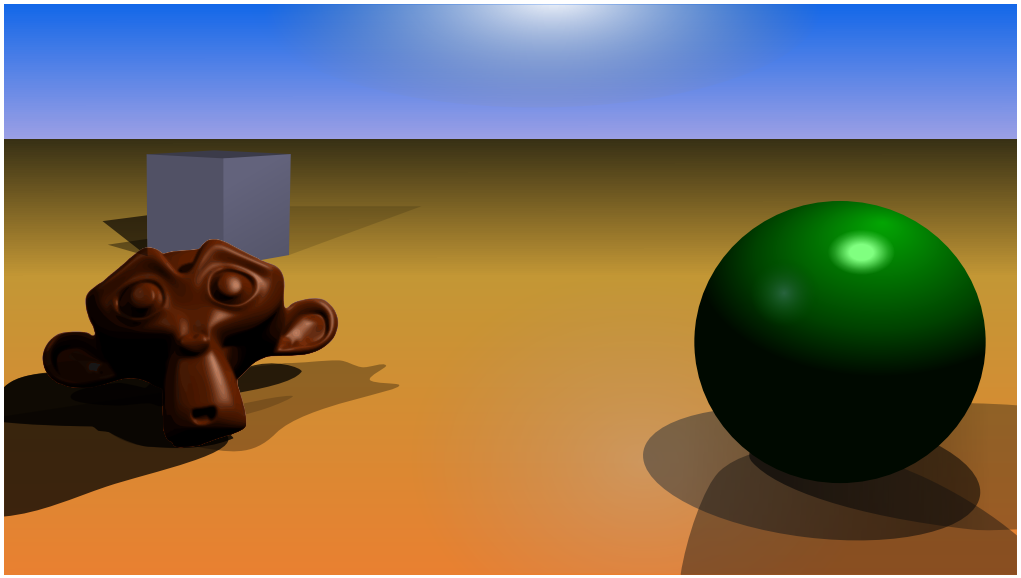
ILLUMINATION (1)



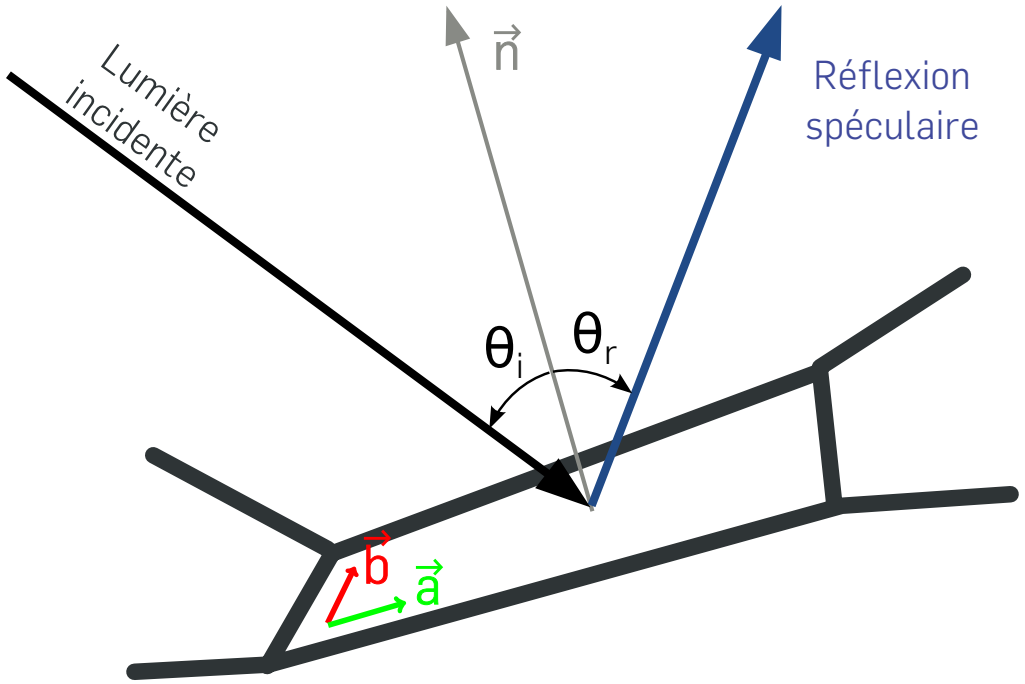
ILLUMINATION (2)

Quatre sources principales d'illumination :

ambiante
diffuse (couleur)
spéculaire (reflets)
émissive



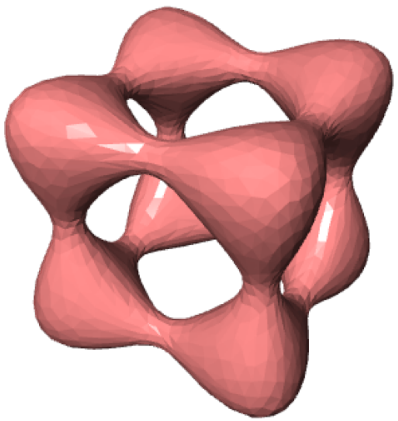
ILLUMINATION (3)



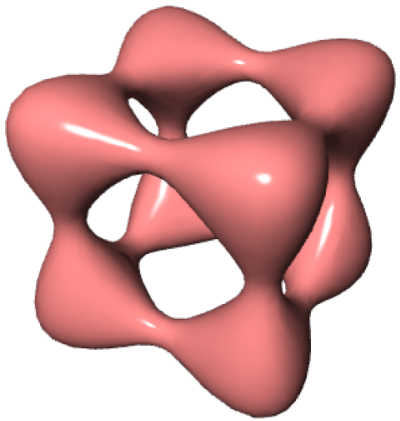
Si la surface est définie par deux vecteurs $\vec{a}$ et $\vec{b}$,

$$\vec{n} = \vec{a} \times \vec{b} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \times \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix}$$

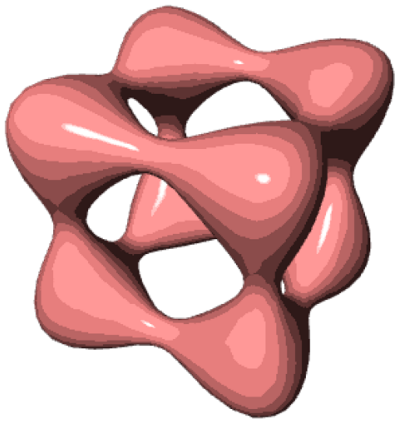
ILLUMINATION (4)



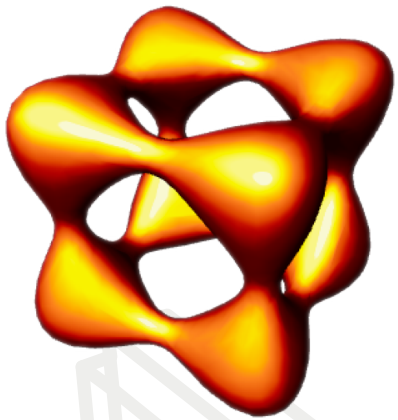
flat



phong



cartoon

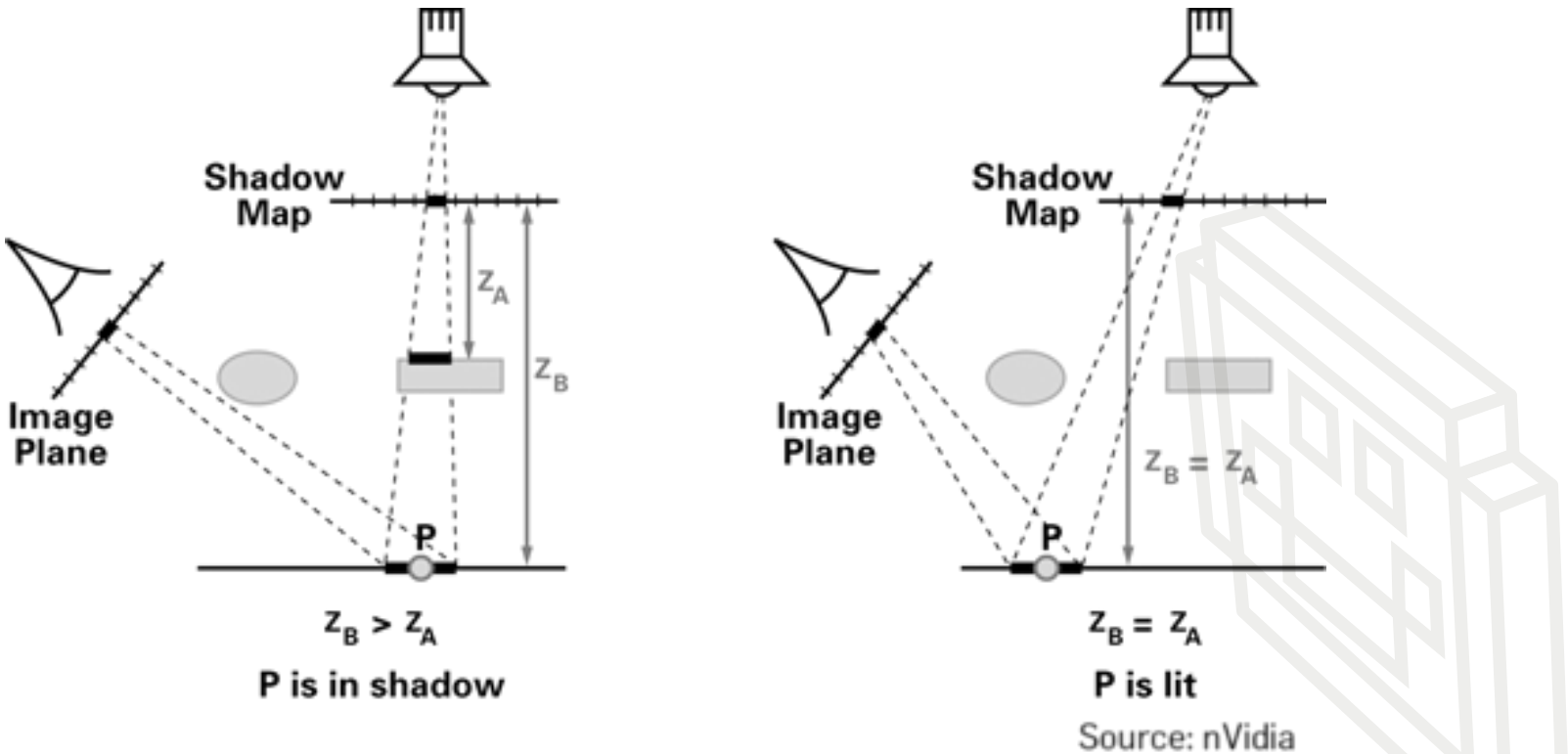


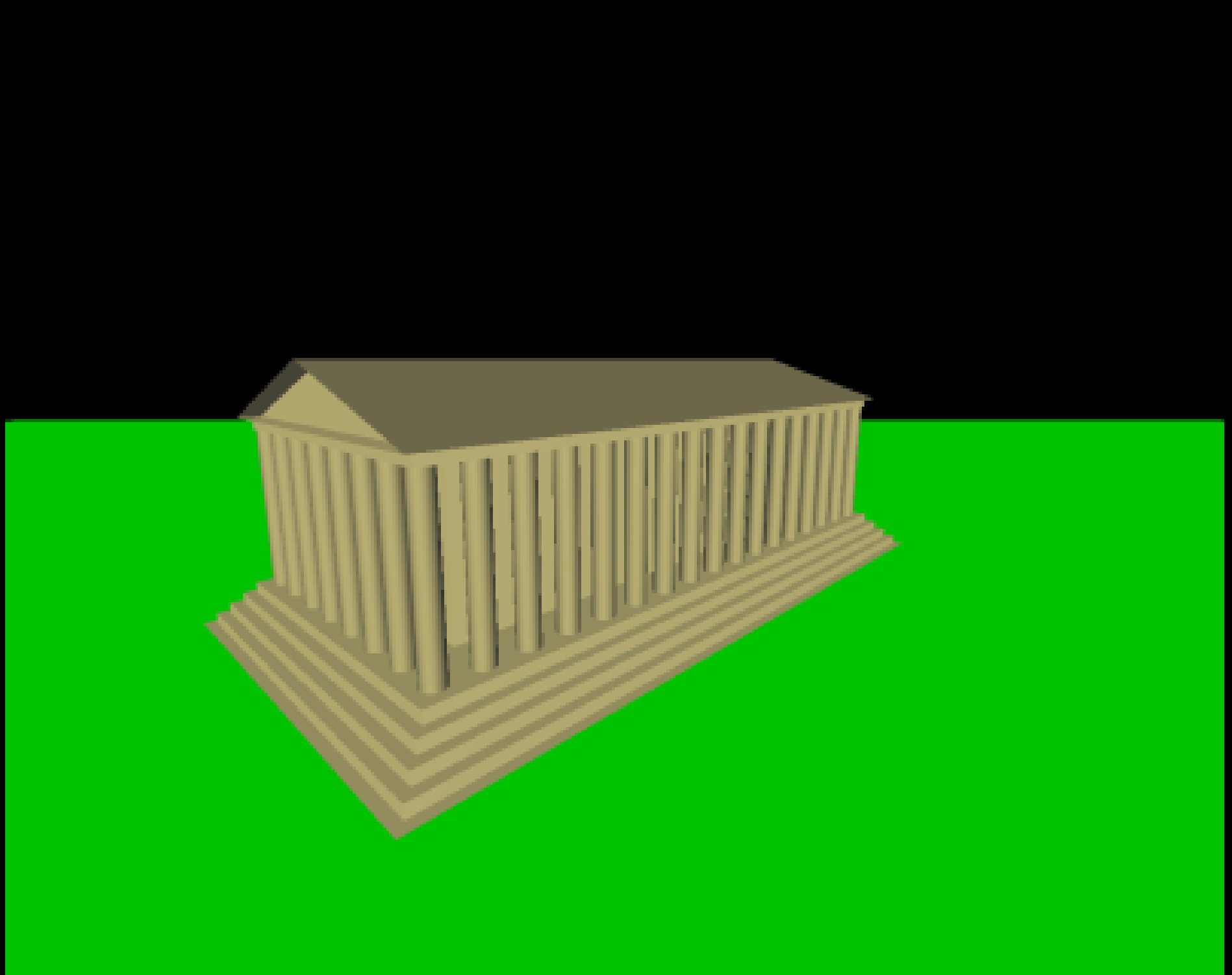
artistic

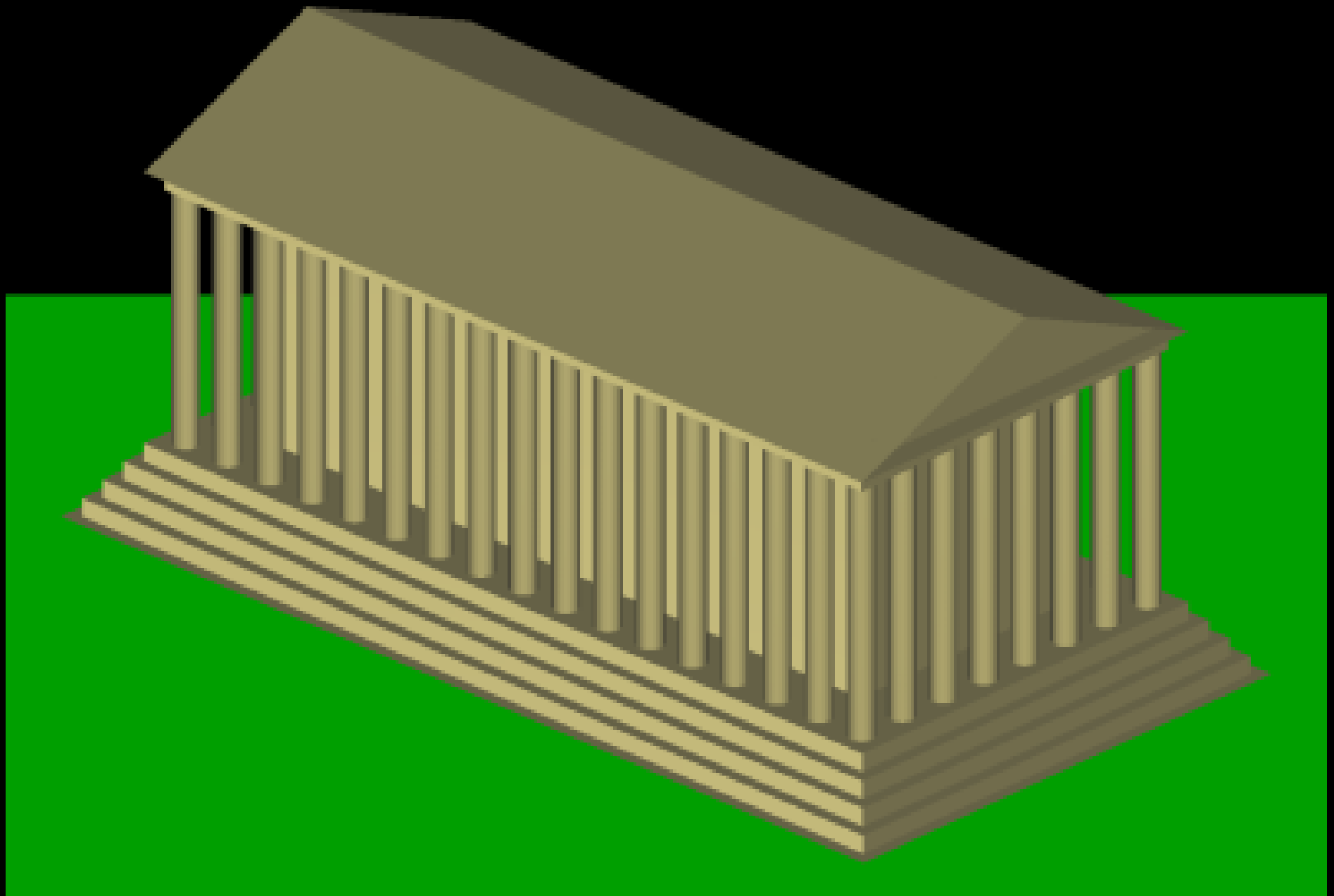
OMBRES

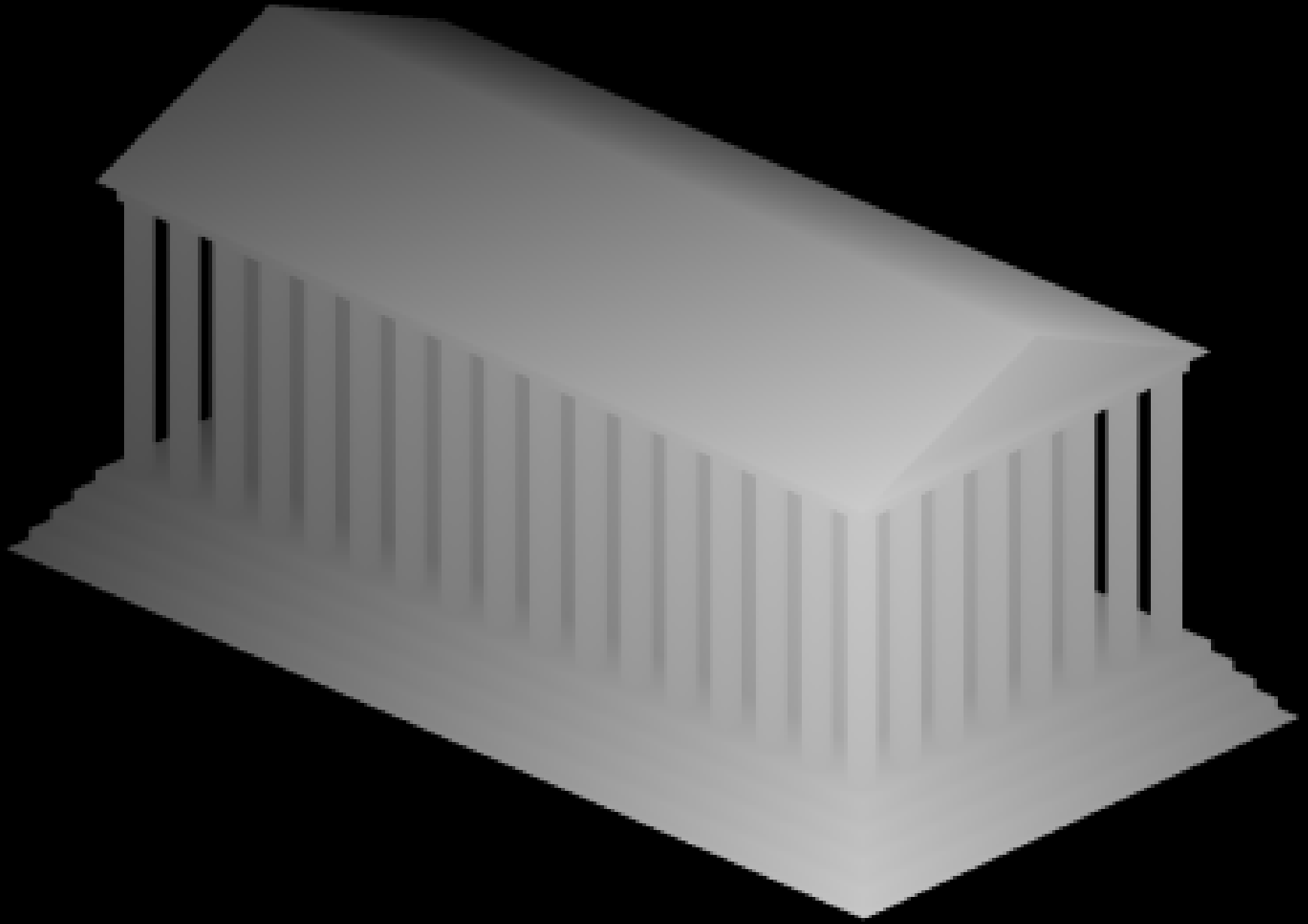
Technique de base : **shadow mapping**

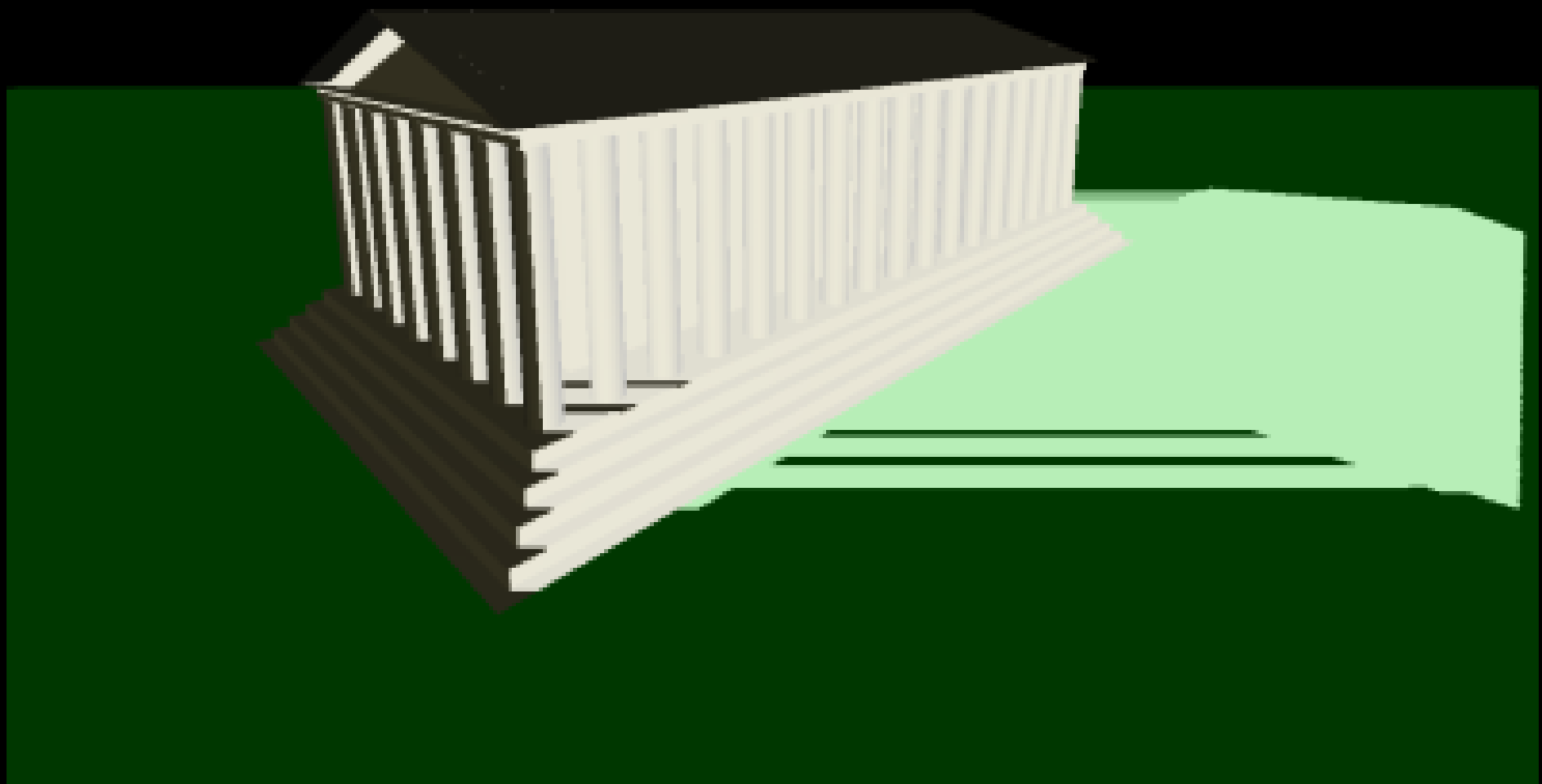
Idée clé: voir le monde à travers la source de lumière.

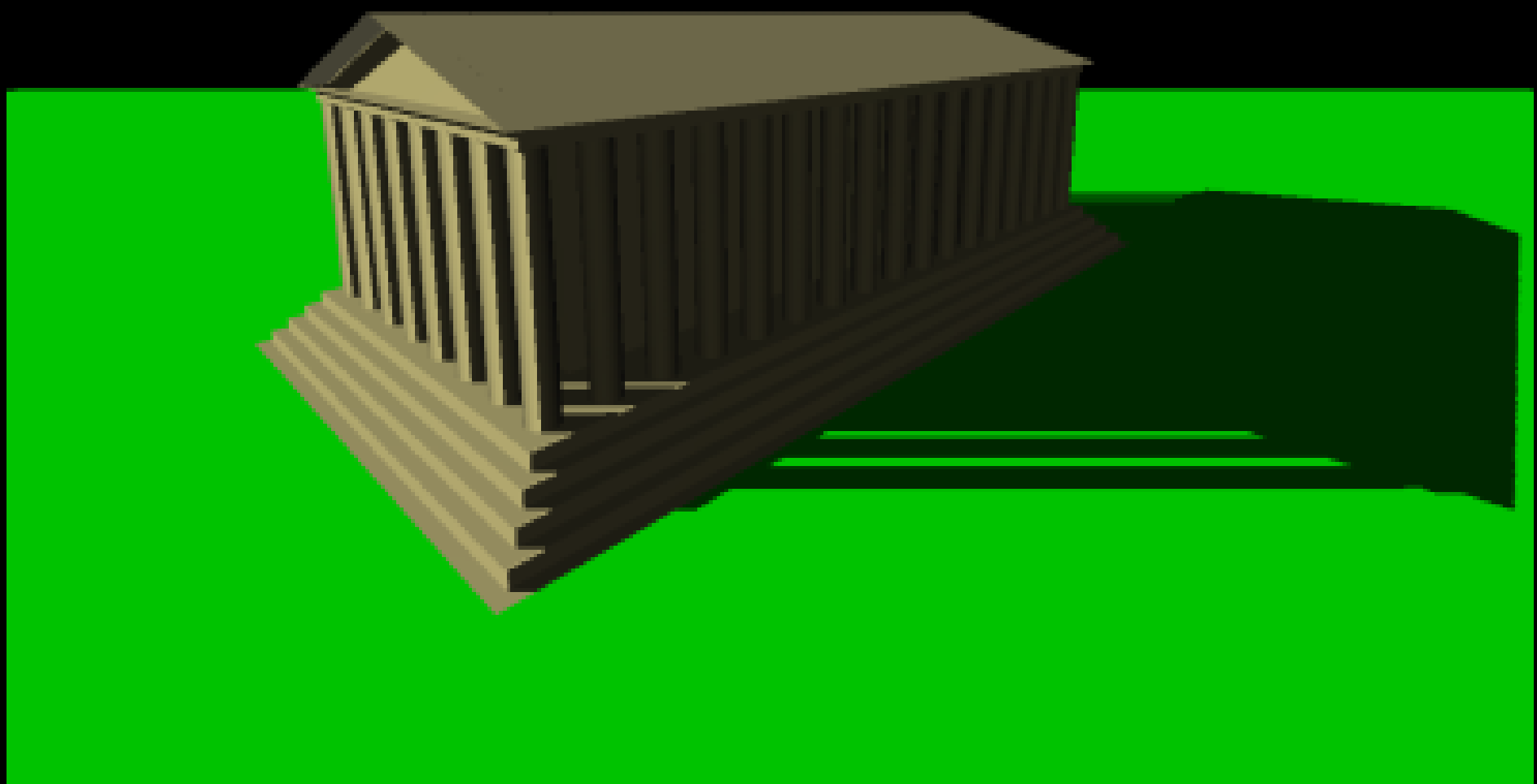




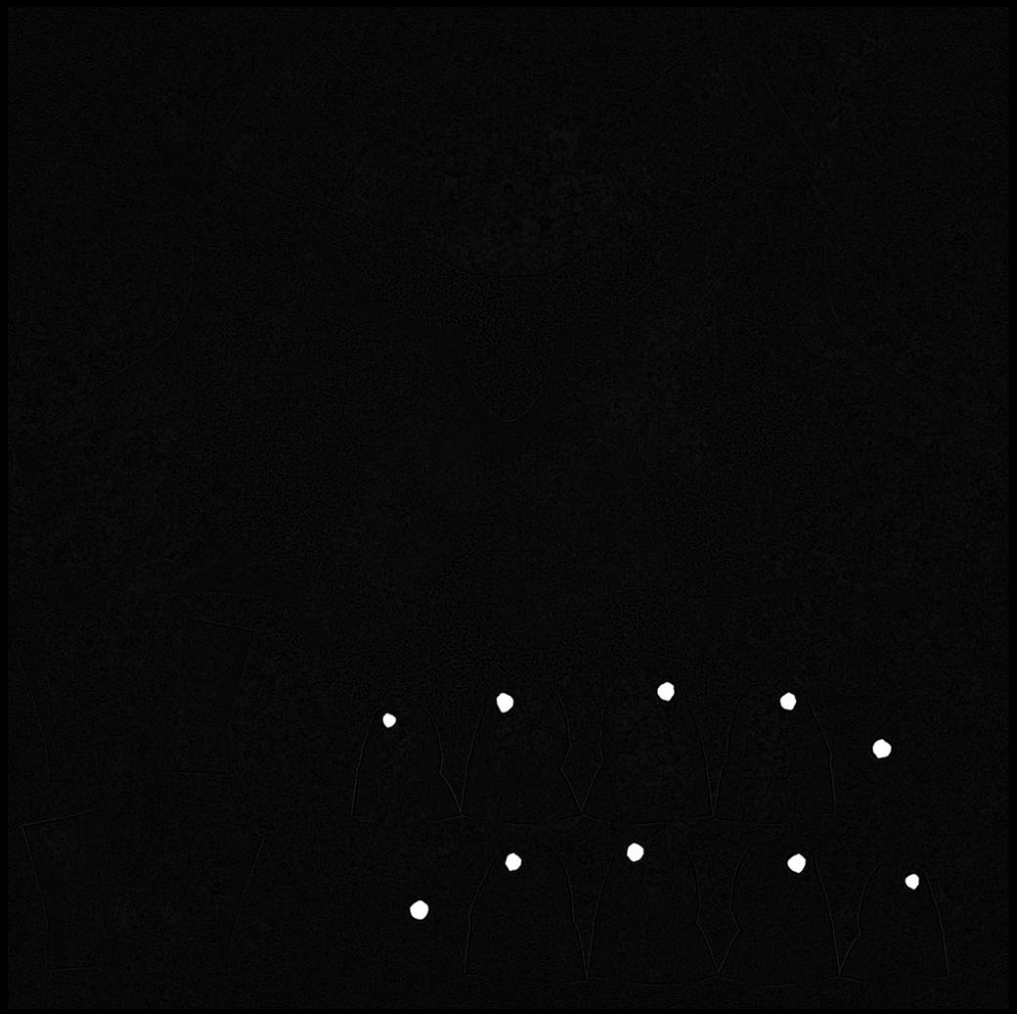


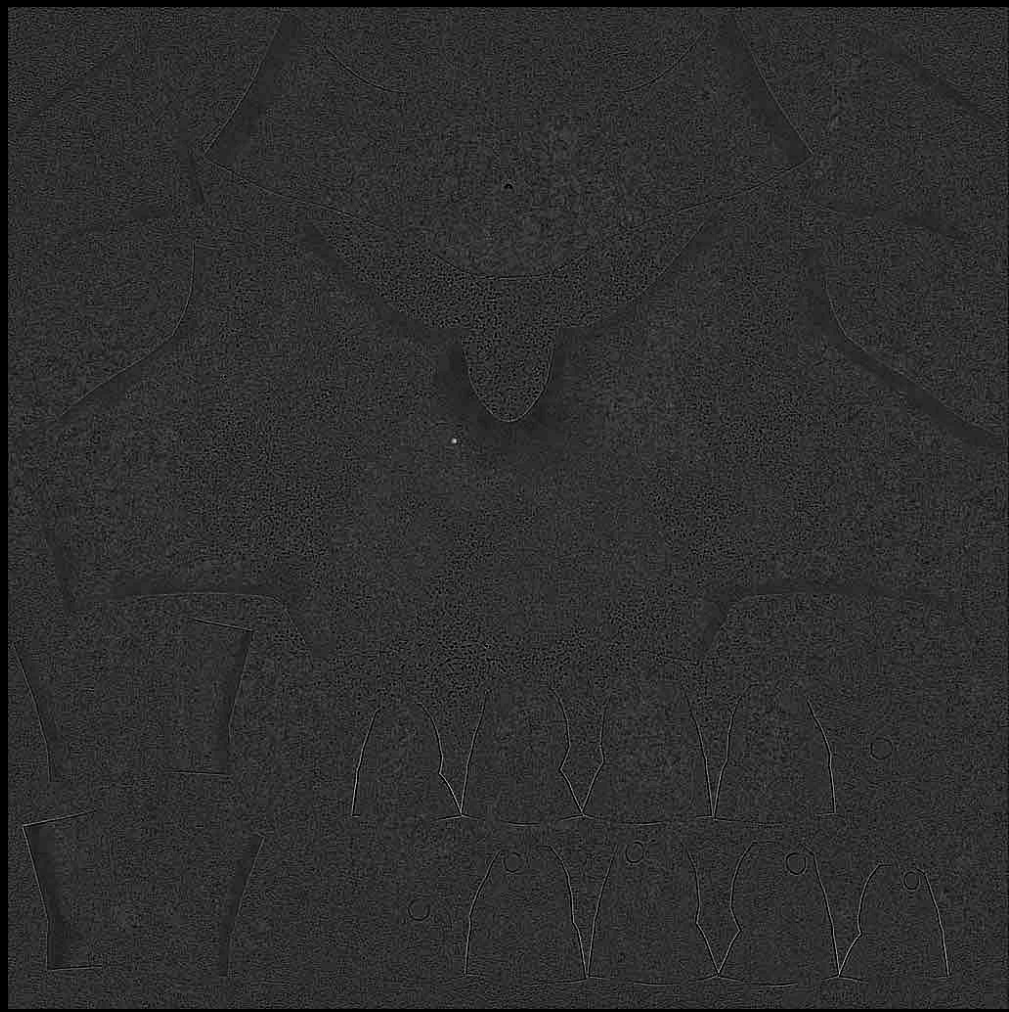




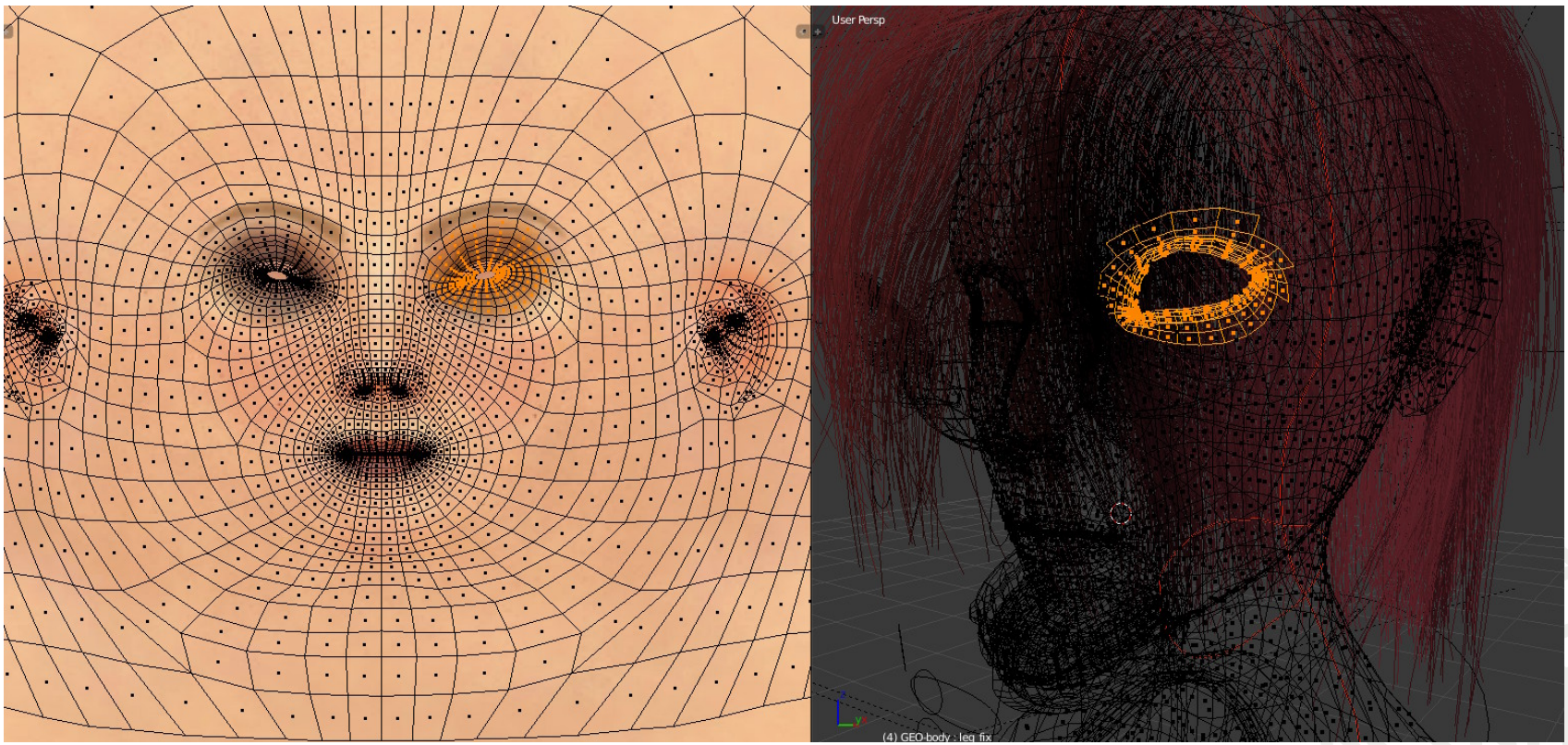


TEXTURES





COORDONNÉES DE TEXTURES

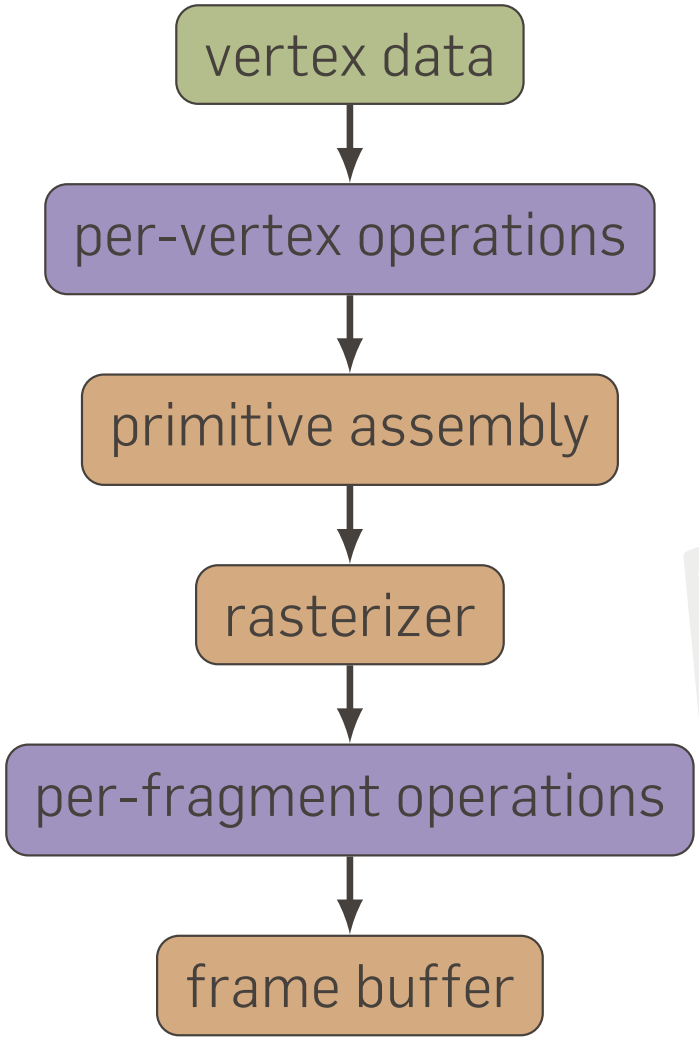


TECHNIQUES DE RENDU

Deux grandes techniques :

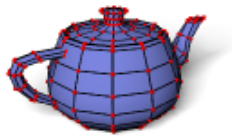
- le **raytracing** (rendu de la lumière physiquement réaliste, lent)
- la **rasterization** (essentiellement, une projection de la géométrie)

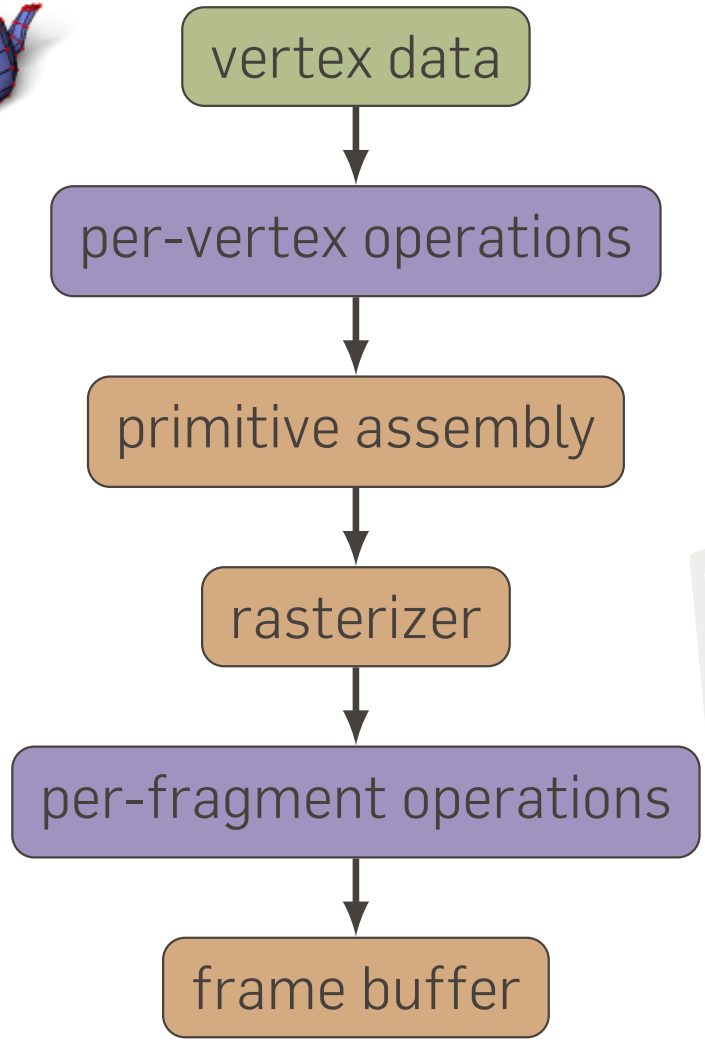
LE PIPELINE OPENGL



[Source: Basé sur Thorsten Thormählen, univ. Magdeburg]

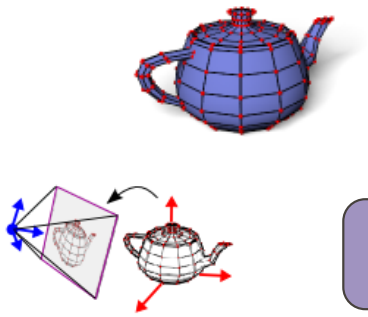
LE PIPELINE OPENGL





[Source: Basé sur Thorsten Thormählen, univ. Magdeburg]

LE PIPELINE OPENGL



vertex data

per-vertex operations

primitive assembly

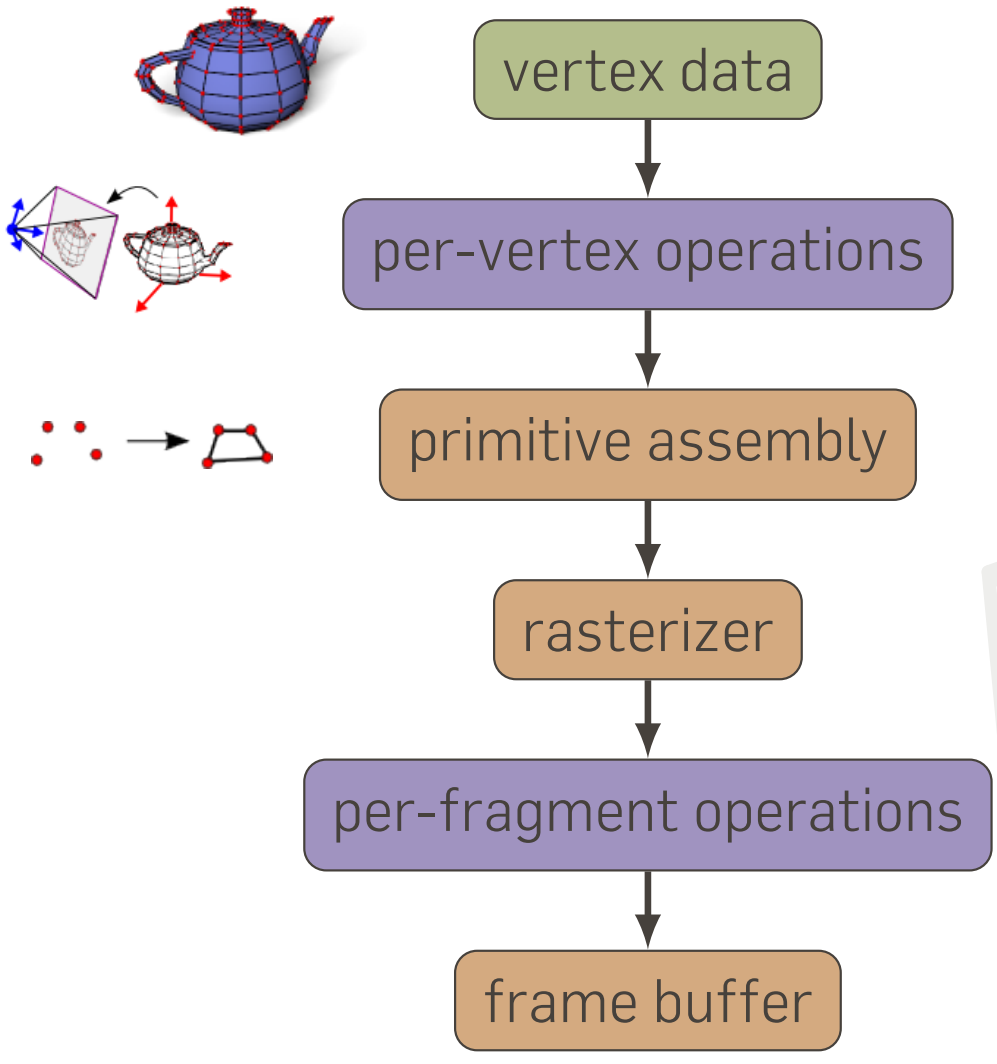
rasterizer

per-fragment operations

frame buffer

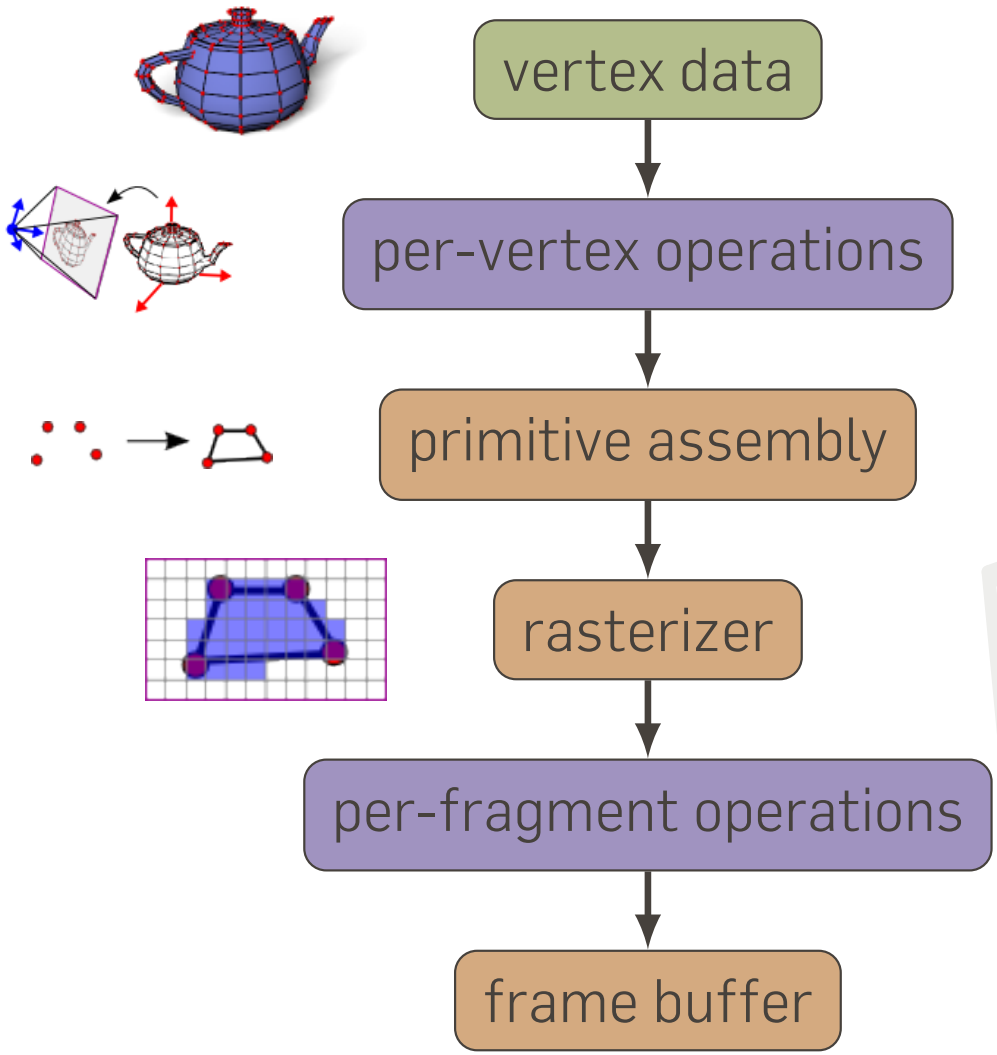
[Source: Basé sur Thorsten Thormählen, univ. Magdeburg]

LE PIPELINE OPENGL



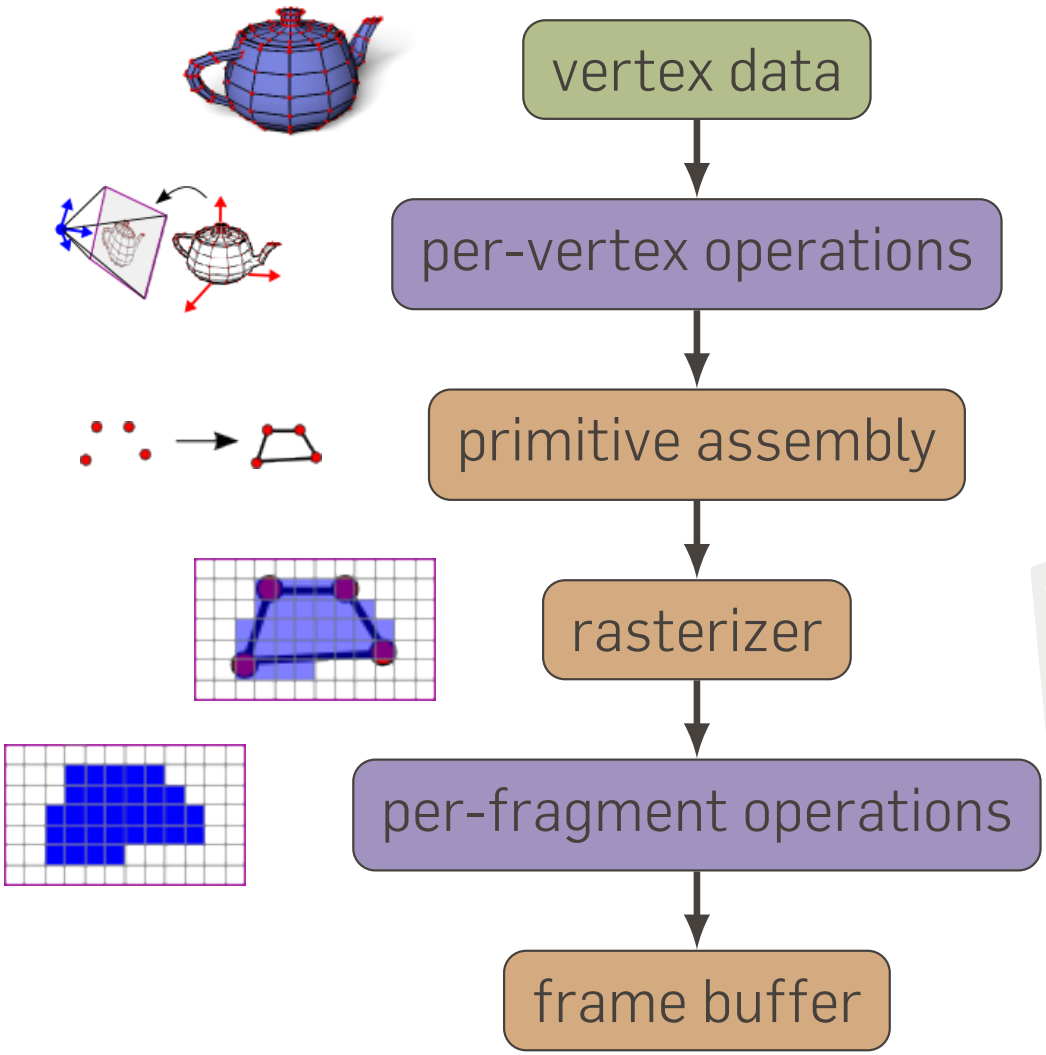
[Source: Basé sur Thorsten Thormählen, univ. Magdeburg]

LE PIPELINE OPENGL



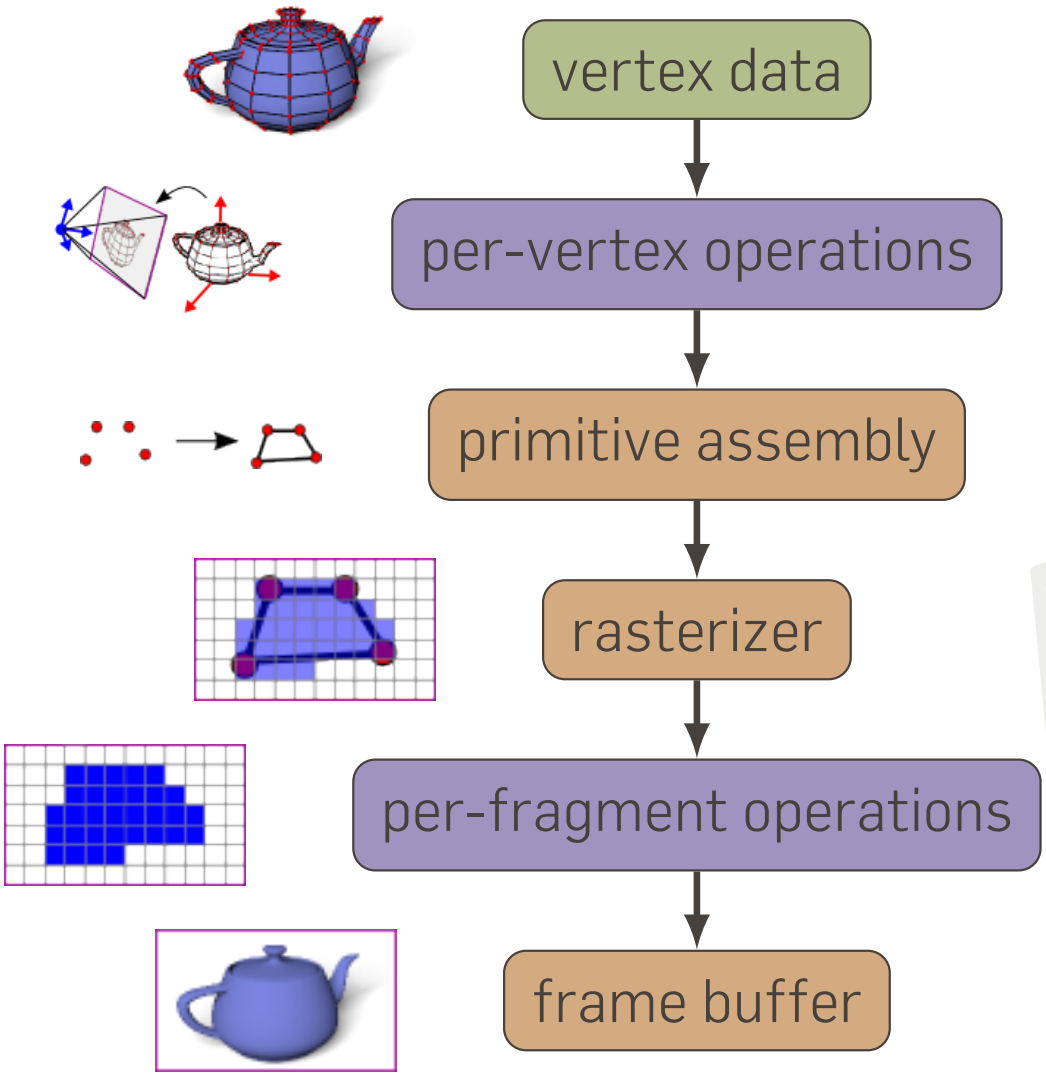
[Source: Basé sur Thorsten Thormählen, univ. Magdeburg]

LE PIPELINE OPENGL

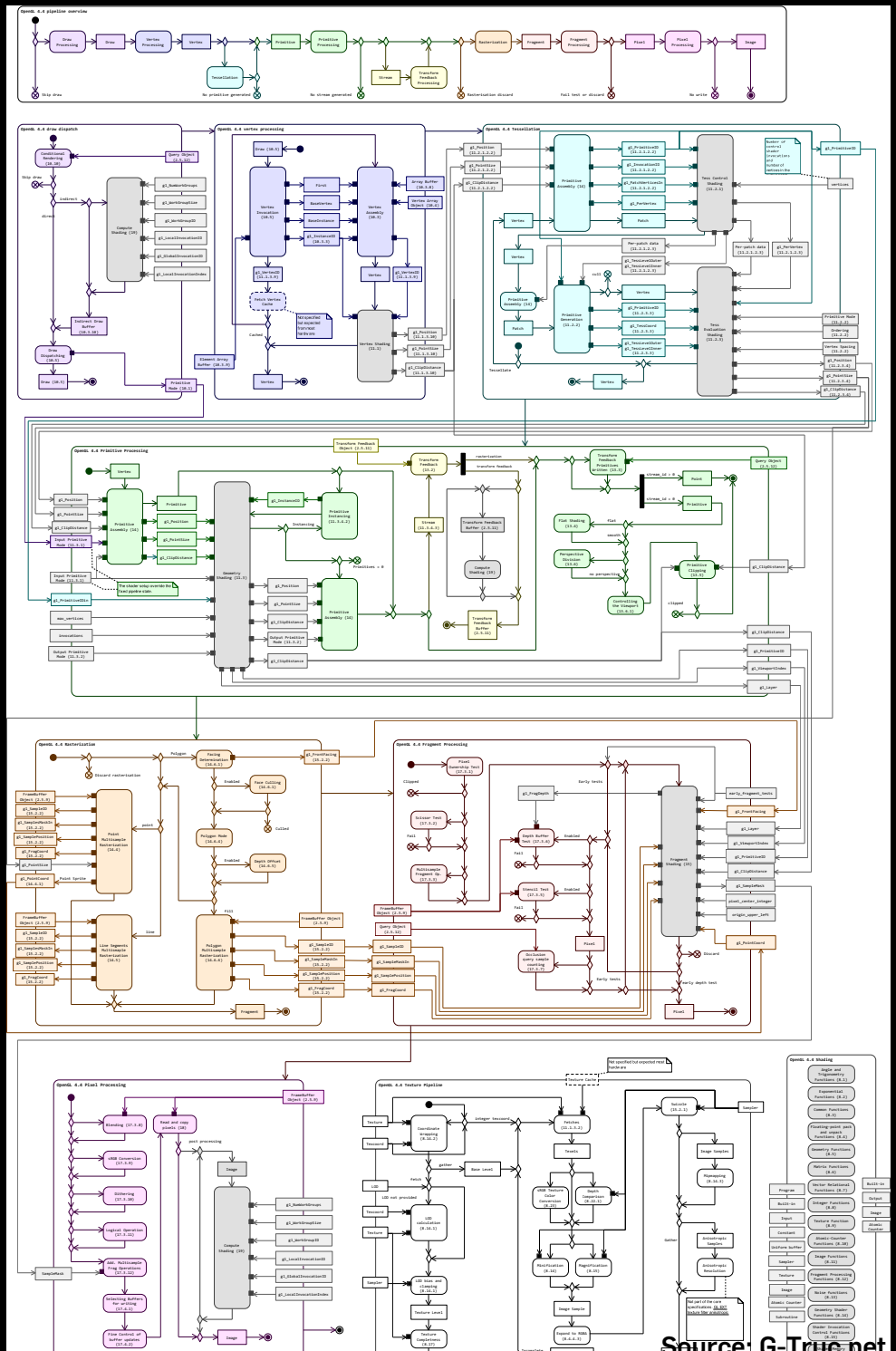


[Source: Basé sur Thorsten Thormählen, univ. Magdeburg]

LE PIPELINE OPENGL



[Source: Basé sur Thorsten Thormählen, univ. Magdeburg]


OpenGL 4.4 pipeline overview
OpenGL 4.4 Draw dispatch
OpenGL 4.4 vertex processing
OpenGL 4.4 Tessellation
OpenGL 4.4 Primitive Processing
OpenGL 4.4 Rasterization
OpenGL 4.4 Fragment Processing
OpenGL 4.4 Pixel Processing
OpenGL 4.4 Texture Pipeline
OpenGL 4.4 Shading
Source: G-Truc.net

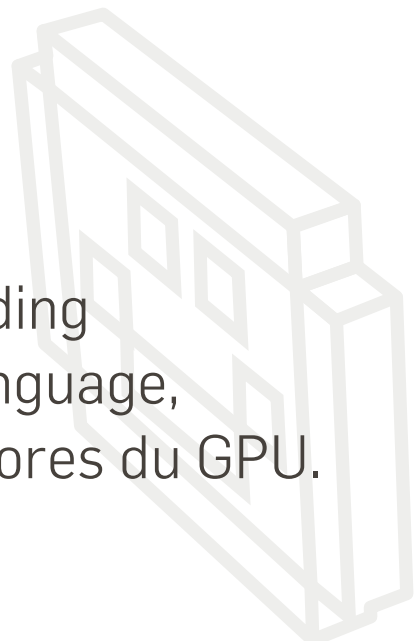
SHADERS

GPU récents: plus de 1000 cores $\Rightarrow$ parallélisation massive

Pipeline programmable:

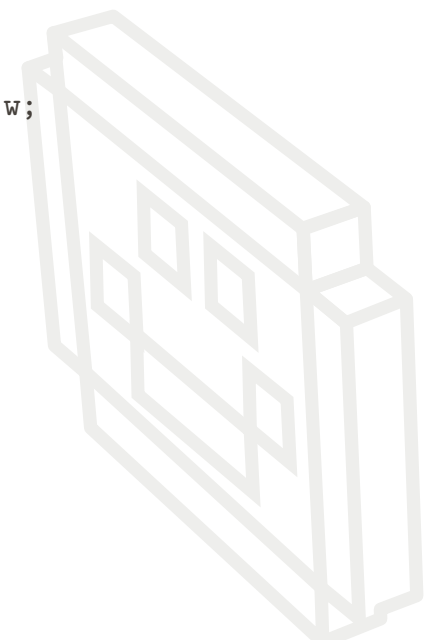
- **Vertex shader**
- **Fragment (ou Pixel) shader**
- Geometry shader
- Tessellation shader

Un **shader** est un programme écrit en GLSL (GL Shading Language, OpenGL) ou HLSL (High-Level Shading Language, Direct3D) et exécuté en parallèle sur les différents cores du GPU.



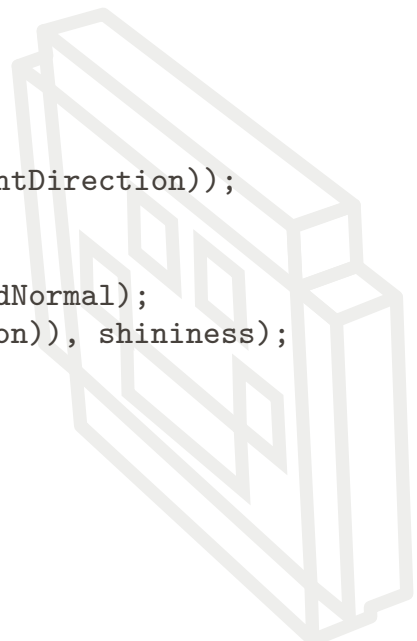
VERTEX SHADER POUR L'ILLUMINATION PHONG

```
uniform mat4 transformationMatrix;
uniform mat4 projectionMatrix;
uniform mat3 normalMatrix;
uniform vec3 light;

in vec4 position;
in vec3 normal;

out vec3 transformedNormal;
out vec3 lightDirection;
out vec3 cameraDirection;

void main() {
    vec4 transformedPosition4 = transformationMatrix*position;
    vec3 transformedPosition = transformedPosition4.xyz/transformedPosition4.w;

    transformedNormal = normalMatrix*normal;

    lightDirection = normalize(light - transformedPosition);

    cameraDirection = -transformedPosition;

    gl_Position = projectionMatrix*transformedPosition4;
}
```



FRAGMENT SHADER POUR L'ILLUMINATION PHONG

```
uniform vec3 lightColor;
uniform float shininess;
uniform vec3 ambientColor;
uniform vec3 diffuseColor;
uniform vec3 specularColor;

in vec3 transformedNormal;
in vec3 lightDirection;
in vec3 cameraDirection;

void main() {
    color.rgb = ambientColor;

    vec3 normalizedTransformedNormal = normalize(transformedNormal);
    vec3 normalizedLightDirection = normalize(lightDirection);

    float intensity = max(0.0, dot(normalizedTransformedNormal, normalizedLightDirection));
    color.rgb += diffuseColor*lightColor*intensity;

    vec3 reflection = reflect(-normalizedLightDirection, normalizedTransformedNormal);
    float specularity = pow(max(0.0, dot(normalize(cameraDirection), reflection)), shininess);

    color.rgb += specularColor*specularity;
}
```



EXEMPLE DE QUESTION

Laquelle des matrices suivantes représente la transformation 2D “une translation d’un vecteur $\vec{t} = (2, 1)$ suivie d’une rotation de $\theta = 90°$”.

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & -2 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

cette transformation ne peut pas être représenter sous forme matricielle

That's all, folk!

Questions : **severin.lemaignan@epfl.ch**

Slides :

github.com/severin-lemaignan/intro-3d-computer-graphics