

Introduction à l'Informatique Graphique

- Généralités
- Concepts d'une scène 3D
- Coordonnées généralisées
- OpenGL et notion de Shaders

Introduction à l'Informatique Graphique

- **Généralités**
- Concepts d'une scène 3D
- Coordonnées généralisées
- OpenGL et notion de Shaders

Contexte

Omniprésence des modèles 3D

Cinéma/VFX



Application de loisirs

Jeux vidéos

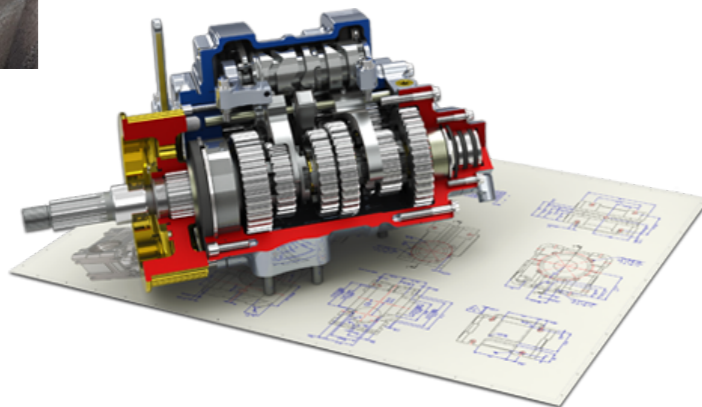


Réalité virtuelle

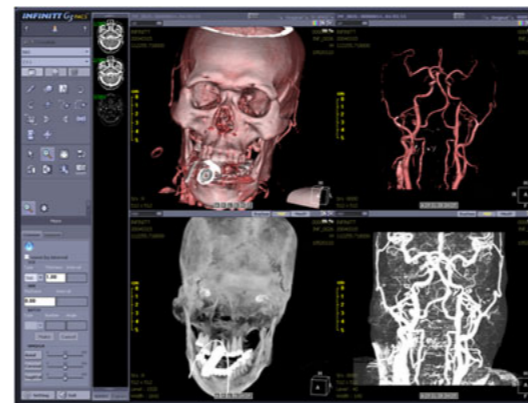


Nombreux autres domaines

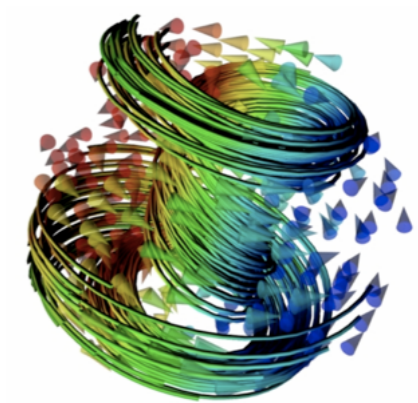
CAO



Imagerie médicale



Physique

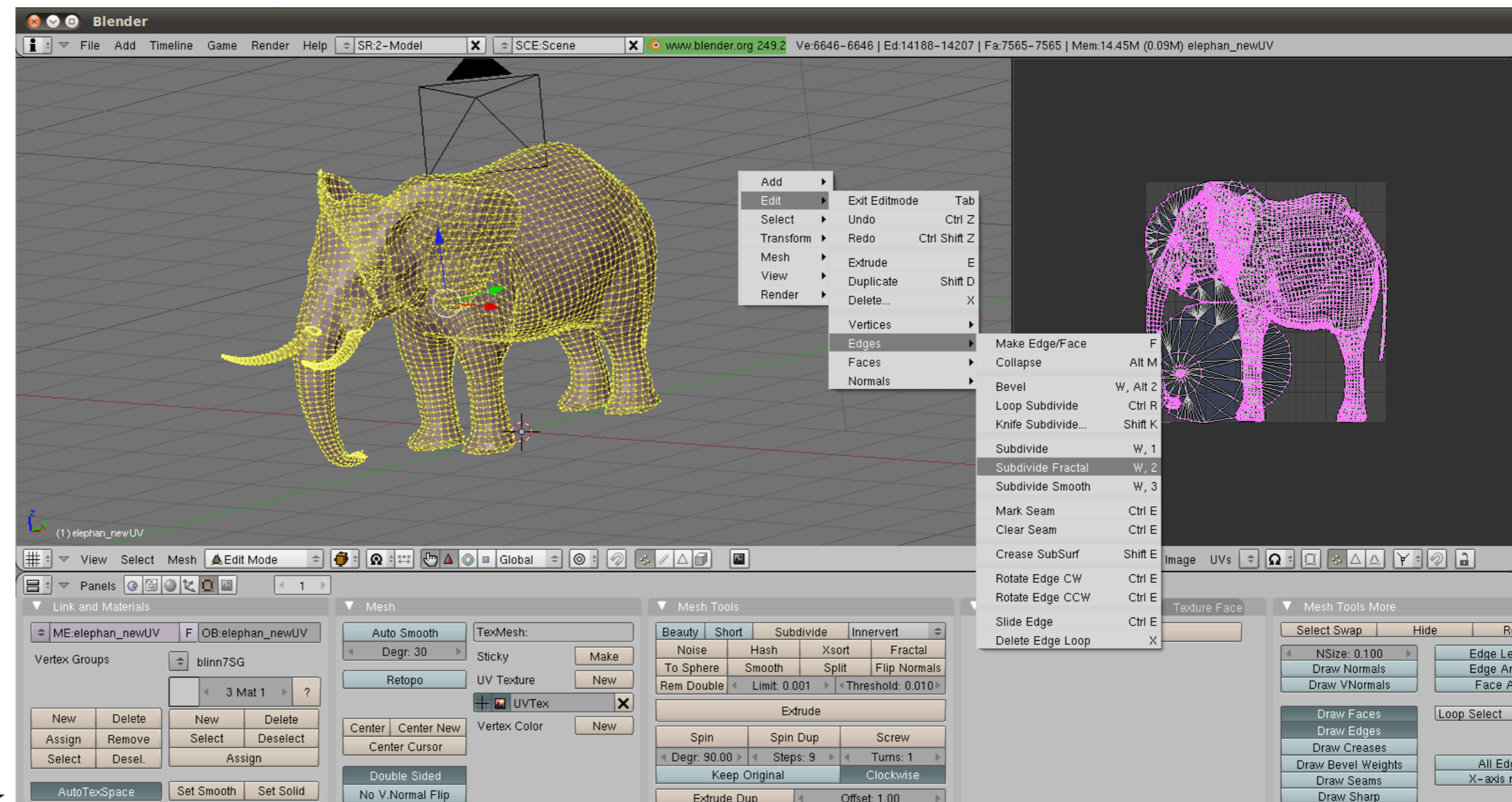
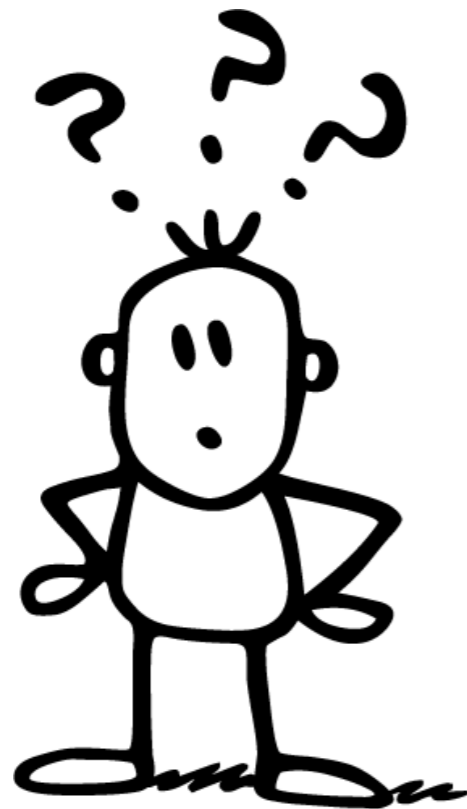
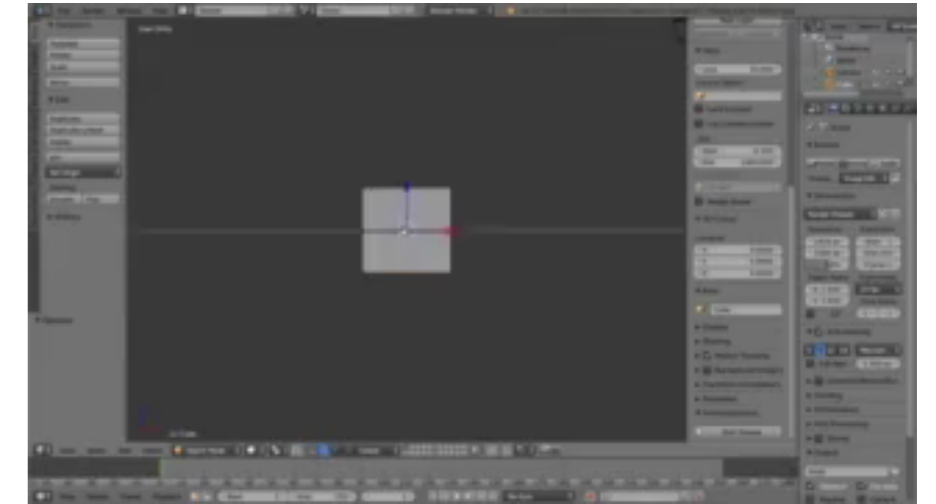


Création 3D

Conclusion: Il est aisé de concevoir et d'animer ses propres modèles 3D.

Création 3D

~~Conclusion: Il est aisé de concevoir et d'animer ses propres modèles 3D.~~



Création 3D

~~Conclusion: Il est aisé de concevoir et d'animer ses propres modèles 3D.~~

Cout/temps passé sur la 3D n'a jamais été aussi élevé

- Films animations/VFX

- *Pirates of the Caribbean* (\$300M), *Avengers* (\$350M), *Toy Story 3* (\$200M)

- *Cout moyen par séquence VFX (<10s) \$50 000*

- *Cout animation 3D > cout dessin manuelle images à images*

- Jeux vidéos AAA

\$100M, 2 à 4 ans de développements

>100 développeurs, milliers d'artistes

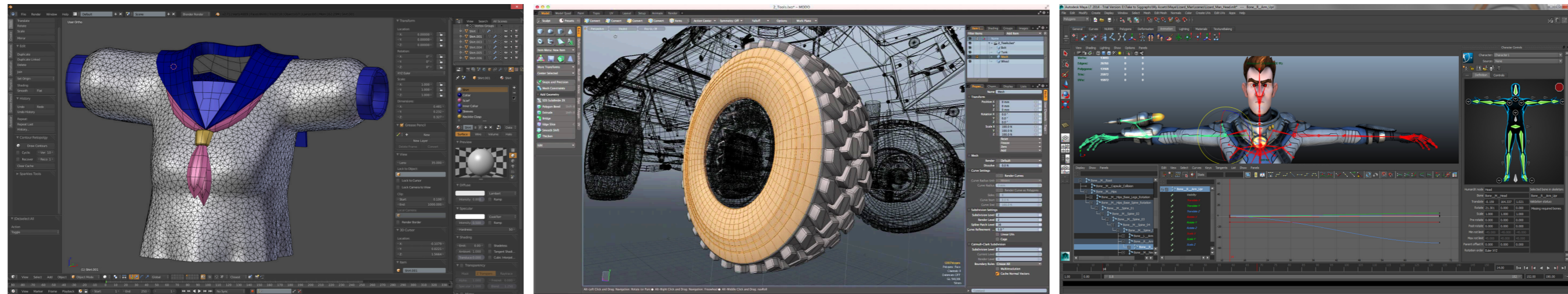


Création 3D

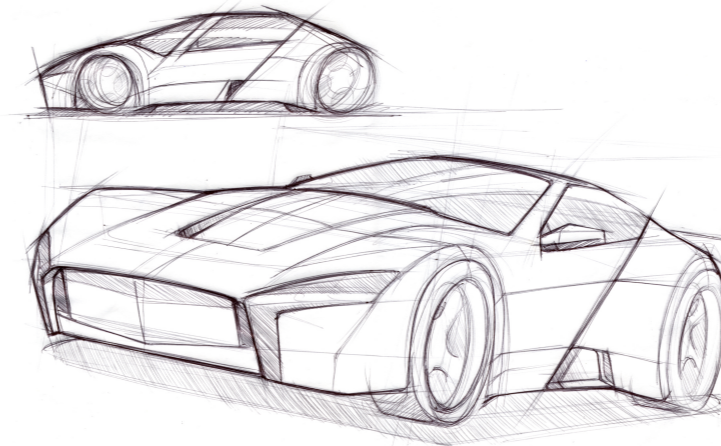
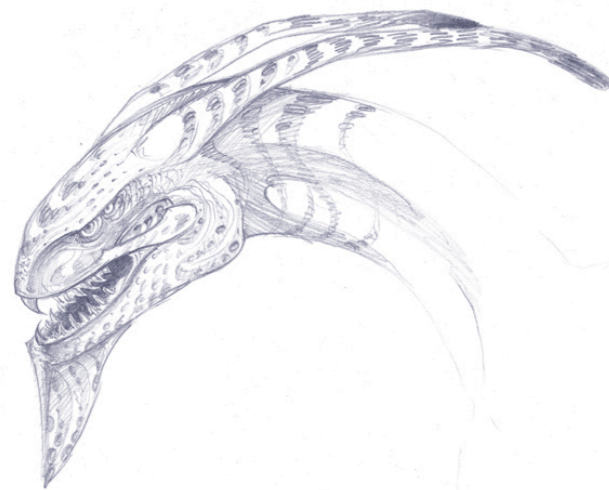
Les outils 3D se sont améliorés

... mais restent complexes et très techniques (3 ans d'études infographistes).

La quantité et la qualité demandé a augmenté plus rapidement que les outils

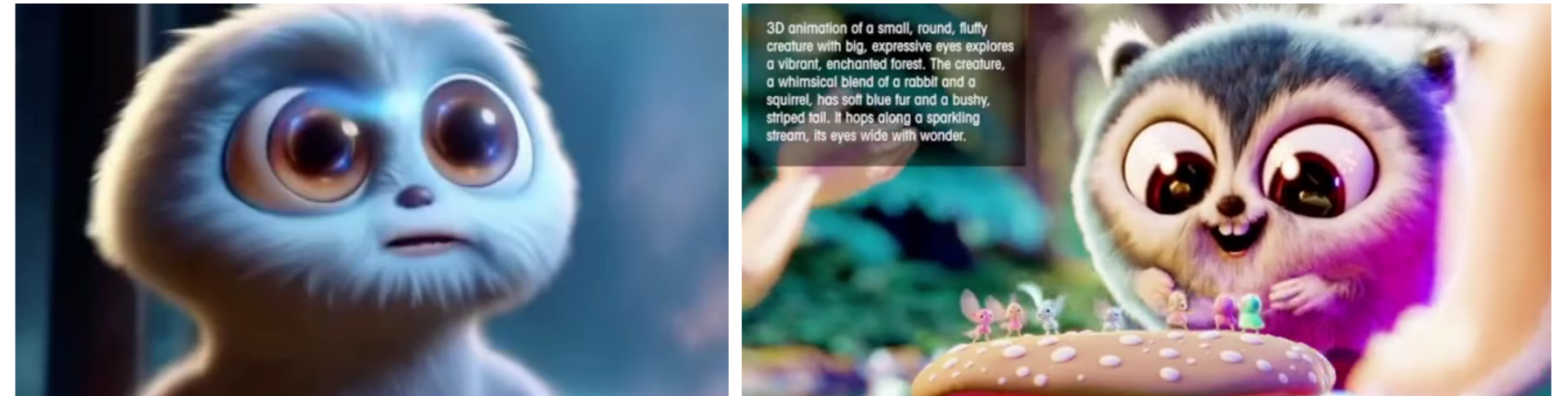


Dessins/sculpture (à la main) restent plus efficaces pour le prototypage/design



Création 3D

Les outils automatiques à base d'IA se développent
... mais restent très peu contrôlables
... avec un coût environnemental qui explose



IA générative reste très expérimentale pour de vraies productions
Contrôlabilité + qualité + efficacité reste un challenge

Error: WebGL2 not supported.

Full screen

Informatique Graphique

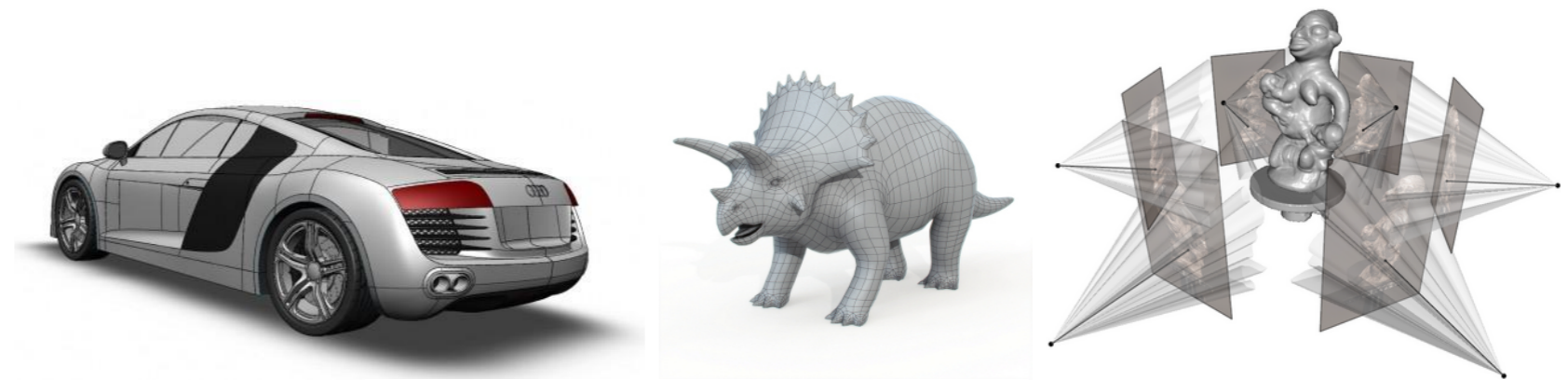
Informatique graphique: Ensemble de sciences et techniques permettant de générer et manipuler des données visuelles.

- Efficace en temps, énergie, espace mémoire
- De manière contrôlable, interactive
- Permettant de générer des images de qualité

Applications diverses: loisirs, design, simulation naturelles, médicales, biologiques, etc.

Trois grands domaines

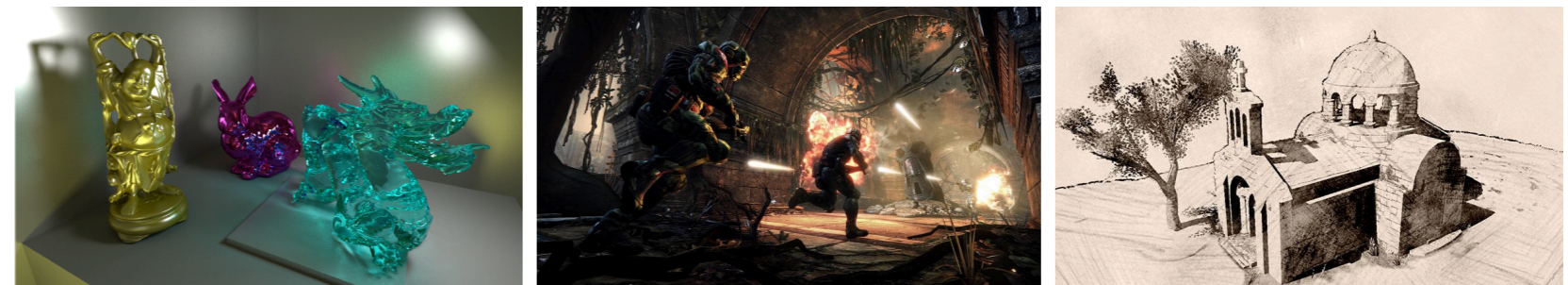
Modélisation (/Modeling)



Animation



Synthèse d'images/Rendu (/Rendering)



Thèmes reliés et vocabulaire

Sous-thèmes reliées

- **Réalité virtuelle (VR):** Interaction avec scène 3D vue avec casque de VR (+ capteurs)
- **Réalité augmentée (AR):** Superposition éléments virtuels + vue réelle
- **Réalité mixte/étendue (XR):** VR + AR
- **Visualisation (/scientifique):** Représentation visuelle dans un objectif d'explication ou de communication
- **Simulation 3D:** Méthode d'animation basée sur un modèle inspiré de la physique.
- **CAO (/CAD) (Computer Aided Design):** Conception de design à partir de modeleurs 3D

Autre vocabulaire

- **Infographie:** Utilisation de logiciels de création de données visuelles (Maya, Blender, 3DS Max, etc.)
- **Analyse/traitement d'images:** Extraction d'information à partir d'images
"Opposé" de la synthèse d'images
- **Computer Vision:** Analyse d'images / vidéos pour comprendre une scène tel qu'un humain le ferait (reconnaissance de formes, de mouvements, etc.)

Introduction à l'Informatique Graphique

- Généralités
- **Concepts d'une scène 3D**
- Coordonnées généralisées
- OpenGL et notion de Shaders

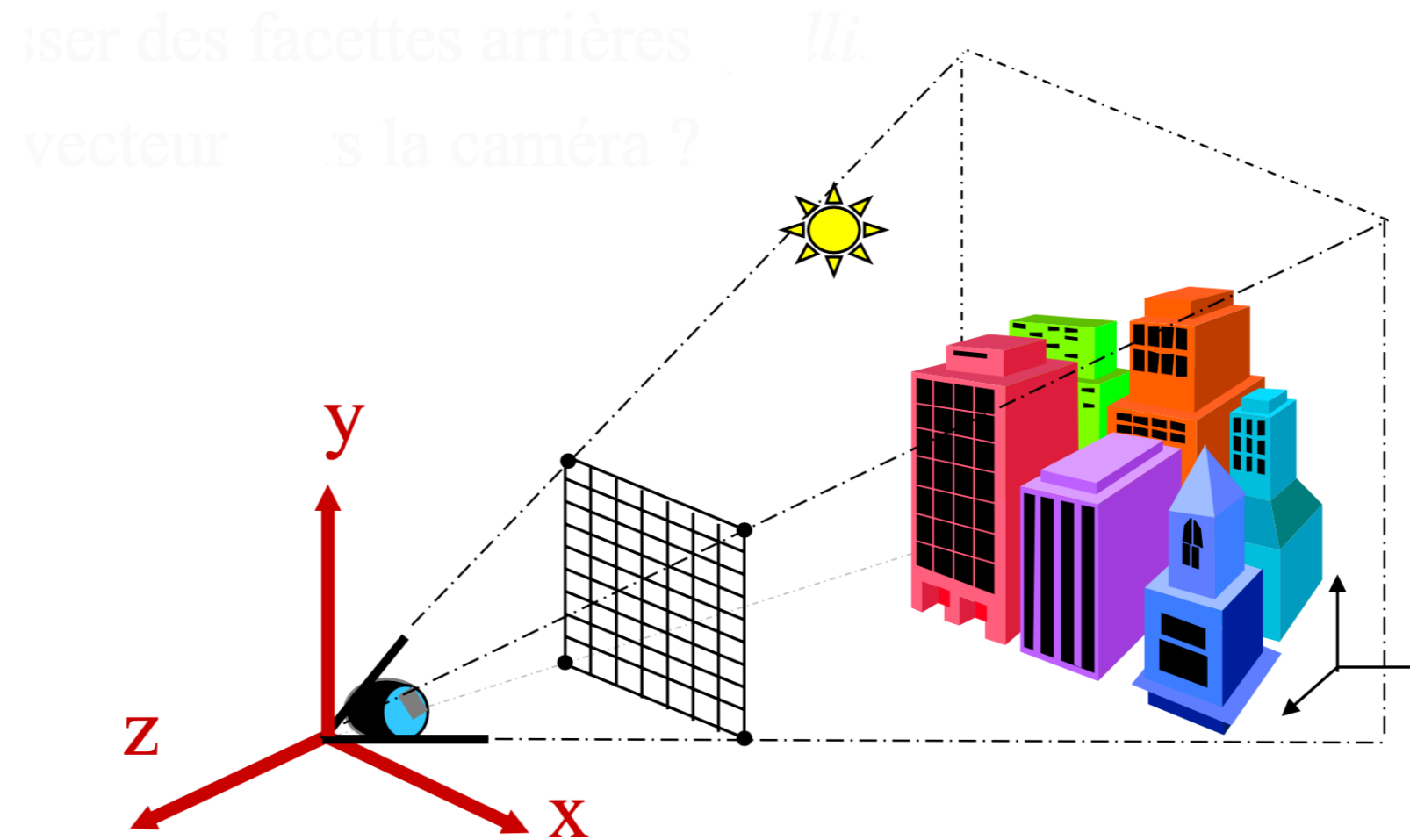
Principes d'une scène 3D - Concepts

Description

- Modèle 3D: Une surface ou un volume
- Source de lumière
- Caméra

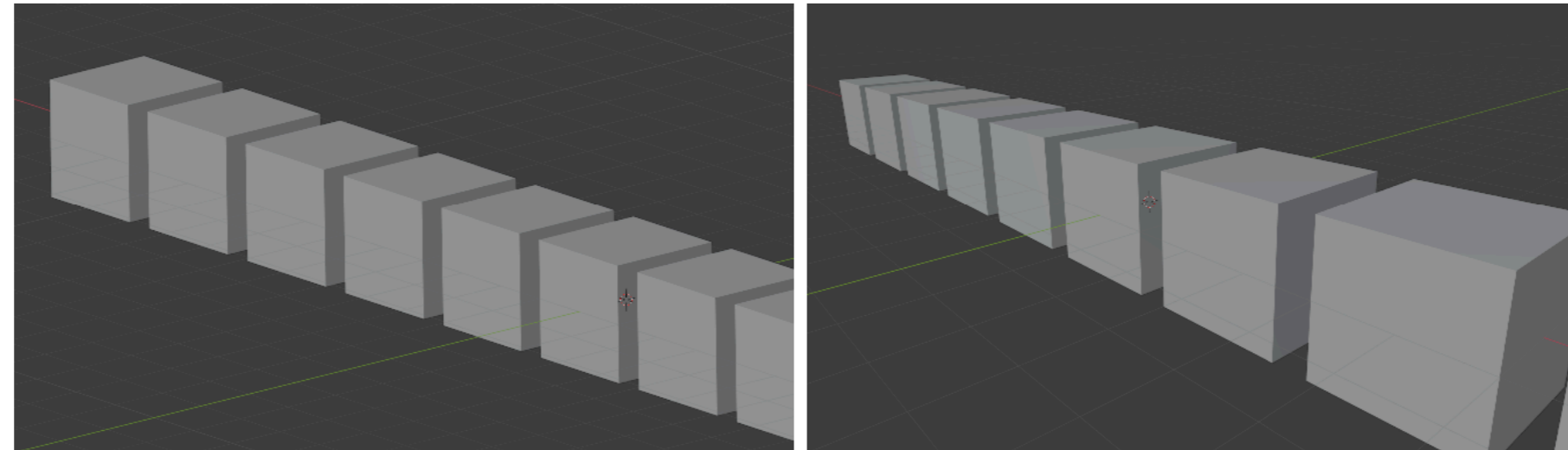
Sortie

- Image vue de la caméra



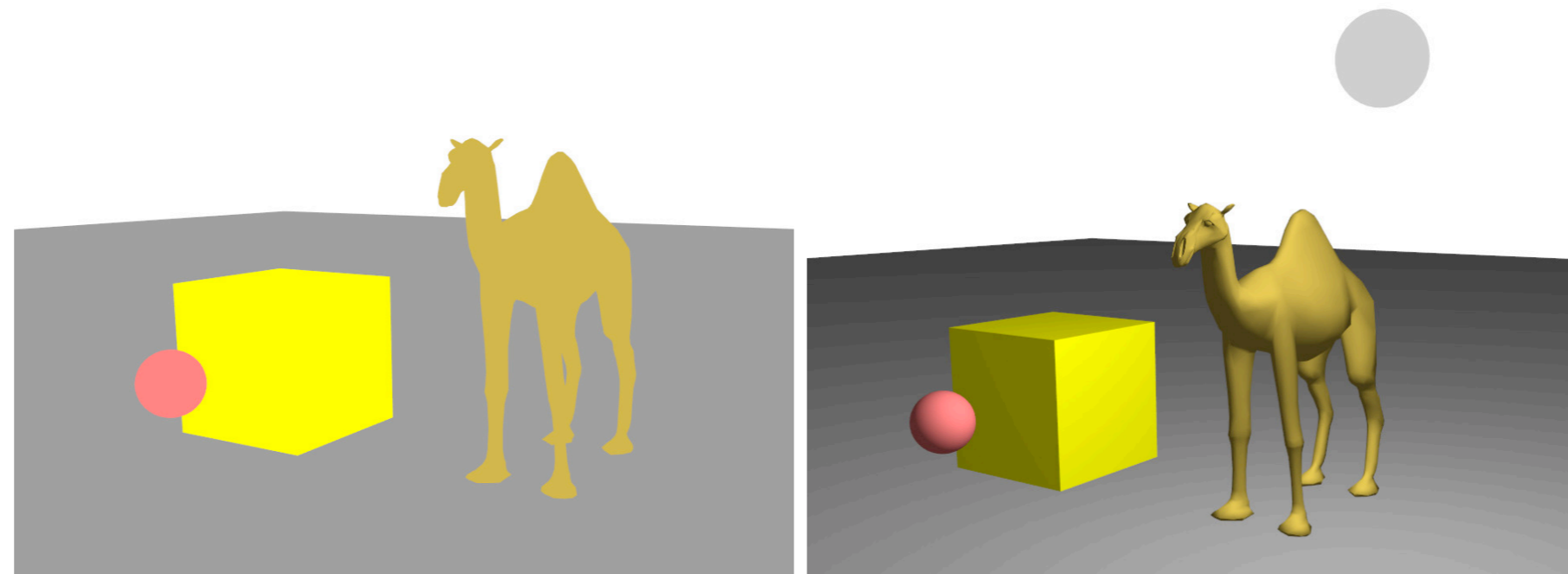
Perception d'une scène 3D sur une image

Effet de perspective



Objets éloignés: plus petits

Couleur indicatrice de la géométrie



Illumination varie en fonction de l'orientation de la surface à la source lumineuse

Definition image

Grille 2D de pixel, $N \times M$ pixels.

1 pixel = 1 couleur

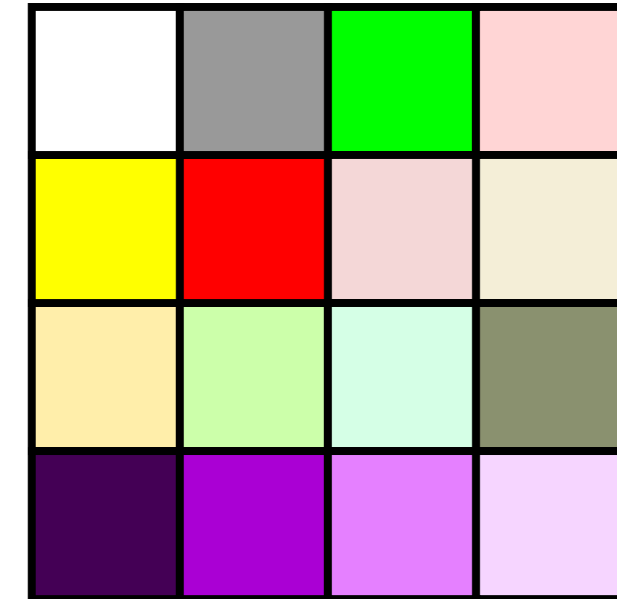
Format standard RGB (Red, Green, Blue)

ou RGBA - A: Canal alpha

En flottants: $c = (r, g, b)$ avec $r, g, b \in [0, 1]$

Exemple: $(0, 0, 0)$:noir, $(1, 0, 0)$:rouge, $(1, 1, 0)$:jaune, $(1, 1, 1)$:blanc

Format "additif" des couleurs



Autres formats:

RGB en entiers: $(r, g, b) \in [0, 255]^3$

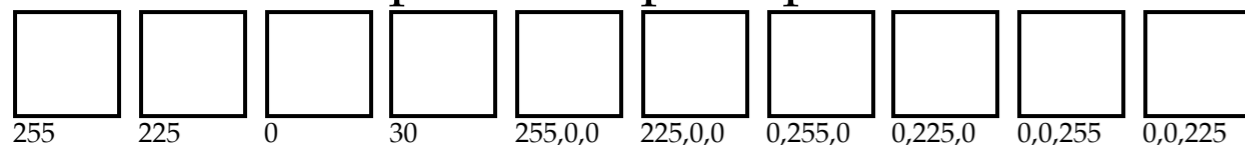
CMYK: Cyan, Magenta, Yellow, Black - impression

HSV: Hue, Saturation, Value

YP_bP_r: Luminance, Chrominance bleue, Chrominance rouge

CIELAB: Perceptuellement uniforme

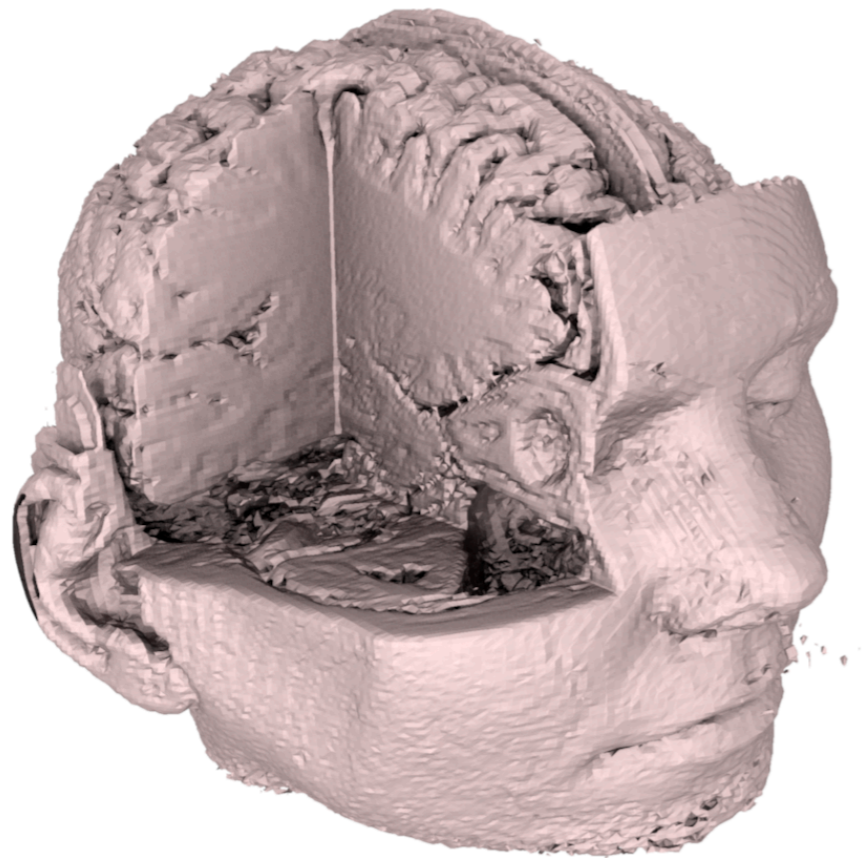
Rem. RGB - espace non perceptuel



255 225 0 30 255,0,0 225,0,0 0,255,0 0,225,0 0,0,255 0,0,225

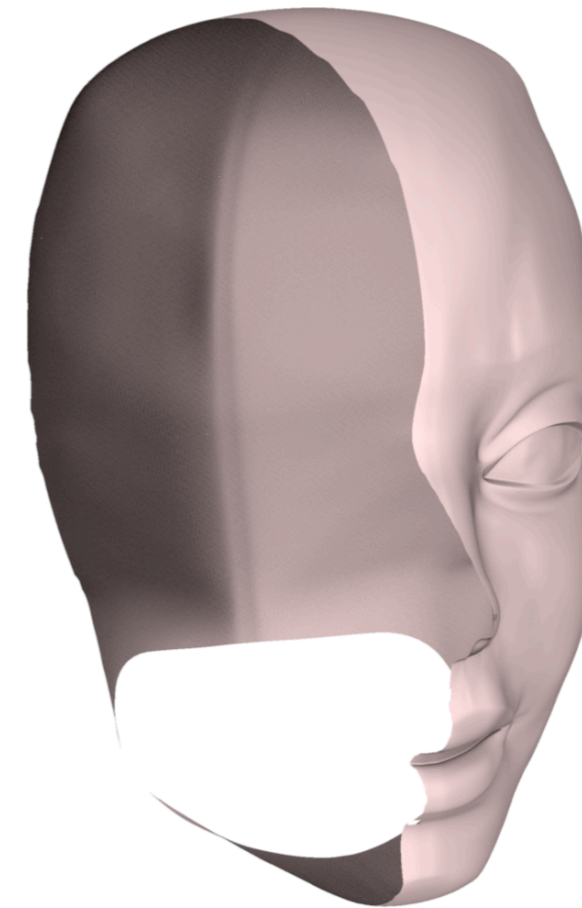
Représentation de Modèle 3D

Volume



Description complète
+ Densité

Surface



Uniquement partie visible
+ Econome en mémoire
+ Rendu efficace sur GPU

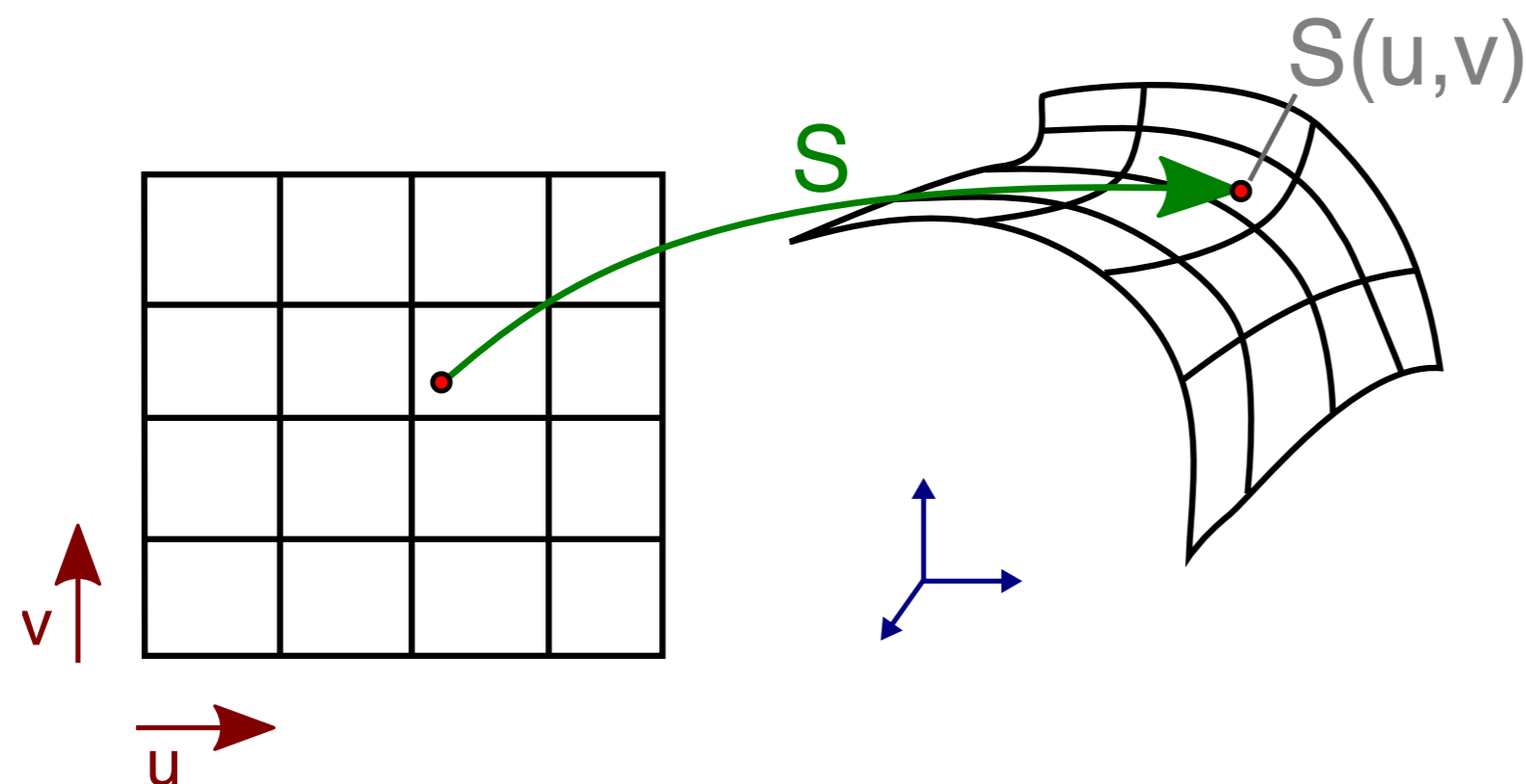
⇒ Informatique Graphique temps réel: **Surface** en majorité

Description d'une surface

Représentation Explicite

Fonction paramétrique

$$S(u, v) = (x(u, v), y(u, v), z(u, v))$$

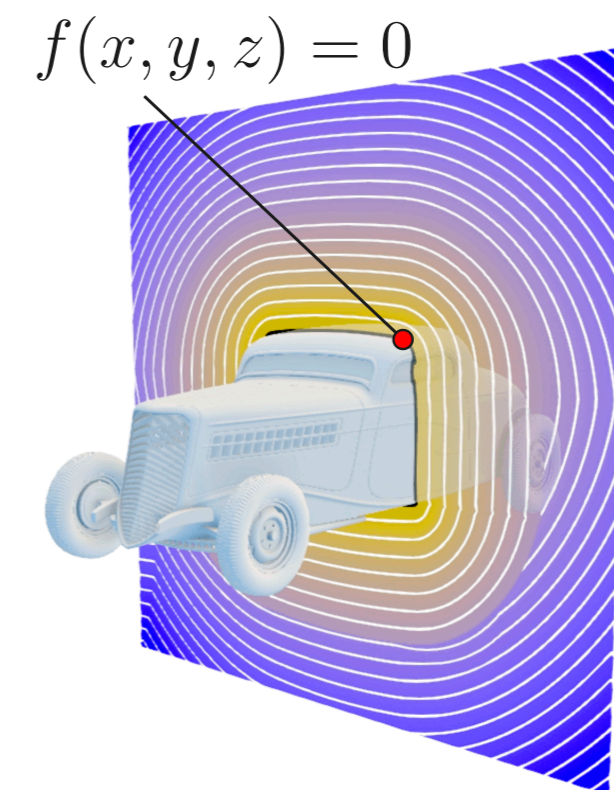


+ Notion de voisinage naturel, calcul différentiel

Représentation Implicite

Isosurface d'un champ scalaire

$$S = \{(x, y, z) \in \mathbb{R}^3 \mid f(x, y, z) = 0\}$$



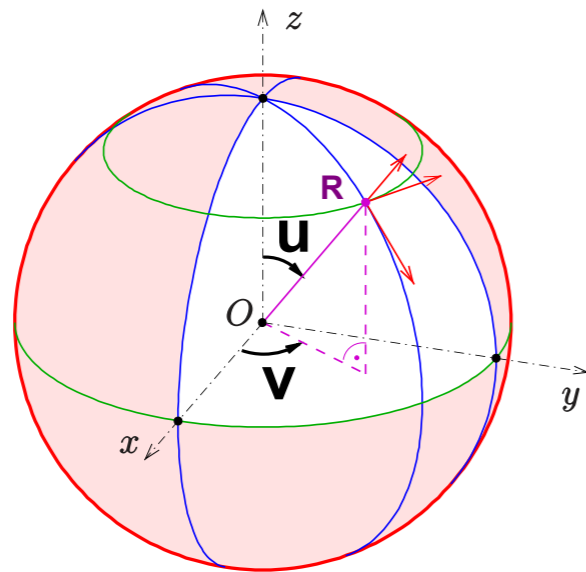
+ Adaptation topologique

Description d'une surface - Exemple d'une sphère

Représentation Explicite

Fonction paramétrique

$$S(u, v) = (x(u, v), y(u, v), z(u, v))$$



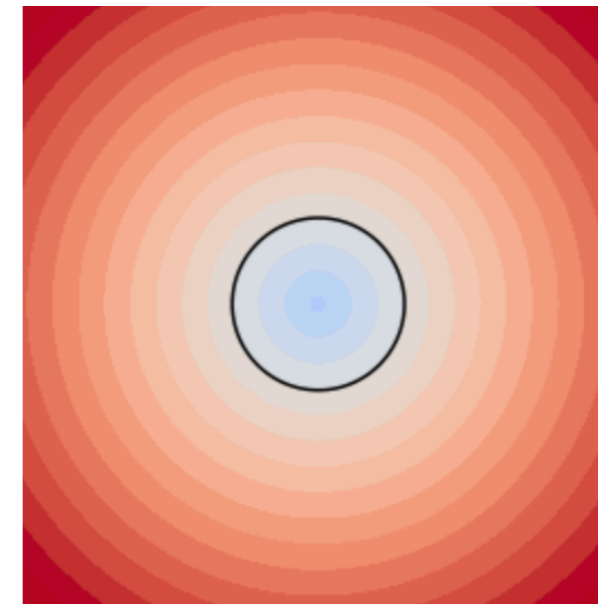
$$S(u, v) = \begin{cases} x(u, v) = R \sin(u) \cos(v) \\ y(u, v) = R \sin(u) \sin(v) \\ z(u, v) = R \cos(u) \end{cases}$$

$$u \in [0, \pi], v \in [0, 2\pi]$$

Représentation Implicite

Isosurface d'un champ scalaire

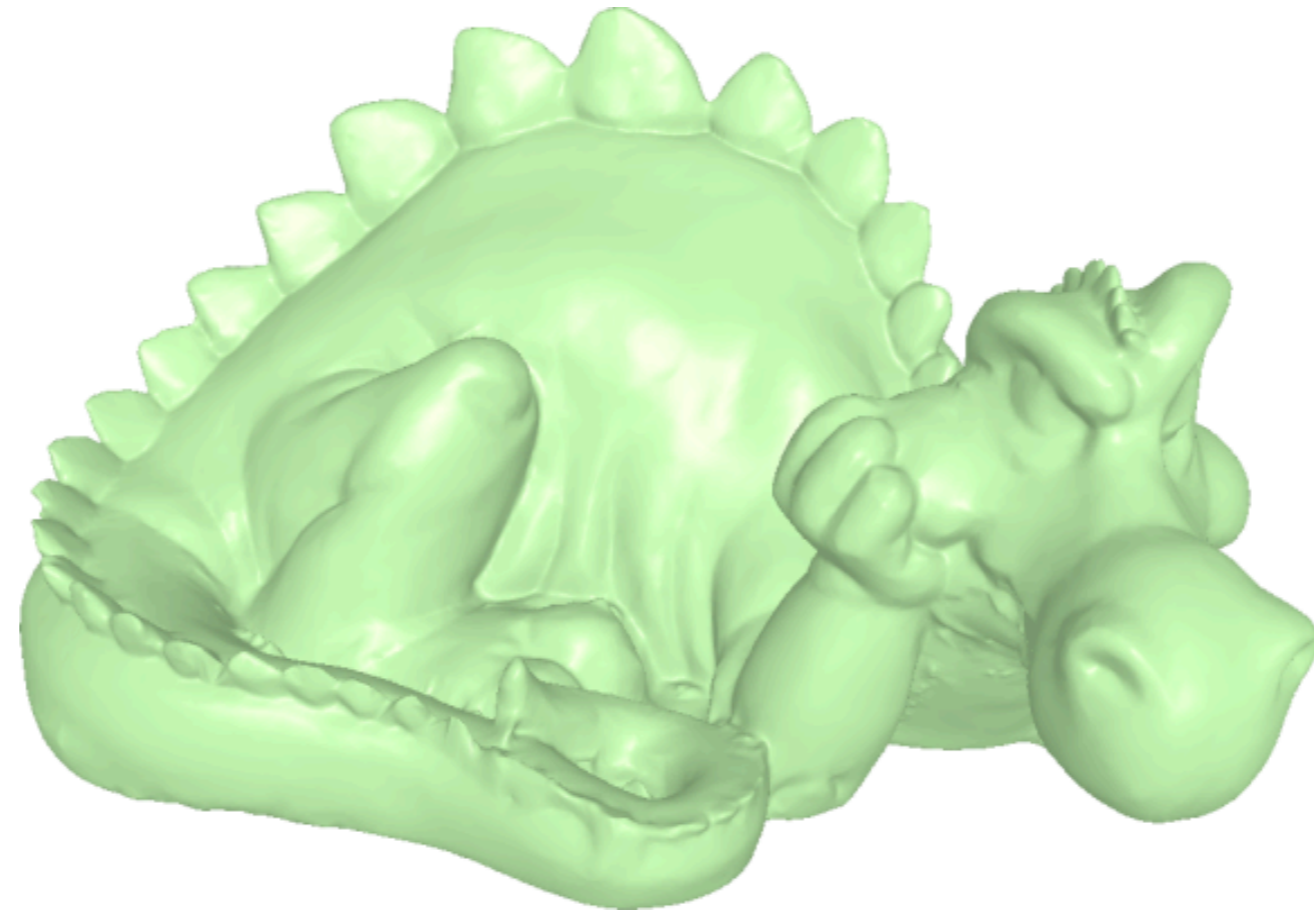
$$S = \{(x, y, z) \in \mathbb{R}^3 \mid f(x, y, z) = 0\}$$



$$f(x, y, z) = x^2 + y^2 + z^2 - R^2$$

Description d'une surface

Quelle fonction pour représenter cette surface ?



$$S(u,v) = ?$$

$$f(x,y,z) = ?$$

Description d'une surface: Discrétisation et primitives

Utilisation approximation par morceaux, primitives et éléments discrets

Description idéale:

- **Approximation** précise de surface arbitraire
- Utilisation d'un **faible nombre de primitives**
- **Rendu efficace** par le GPU
- Manipulation aisée pour la **modélisation**

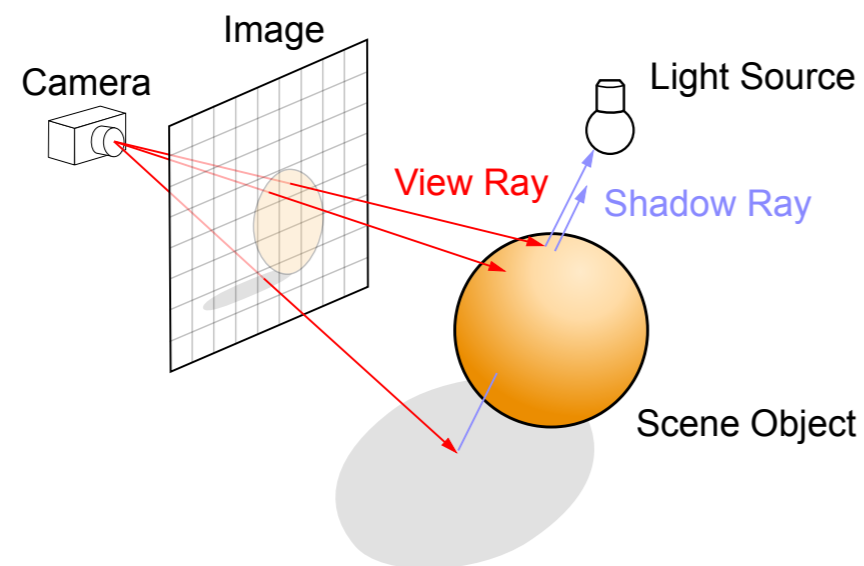
Exemples de modèles:

- *Maillages (/Mesh): maillages triangulaires, polygonales*
- *Polynomes: Béziers, Splines, NURBS*
- *Implicites: Voxels, Skelettes, RBS, MLS*
- *Point sets, Gaussian Splats*

Cartes graphiques \Rightarrow spécialisées pour le rendu de maillages triangulaires

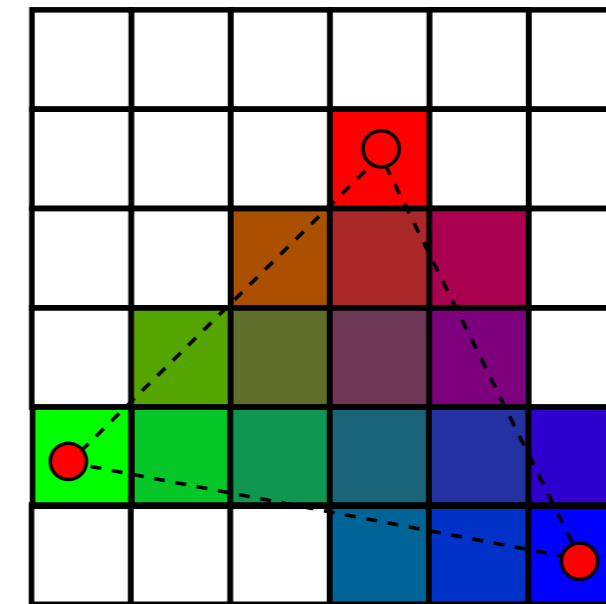
Rendu 3D

Lancé de rayons / Ray tracing



- Lance rayon entre source de lumière et caméra
- Calcul des interesections des rayons avec formes 3D
- + Rendu photo-réaliste
(Ombres douces, reflection, refraction, etc.)
- + Surfaces arbitraires
- Cout de calcul

Projection/Rasterization



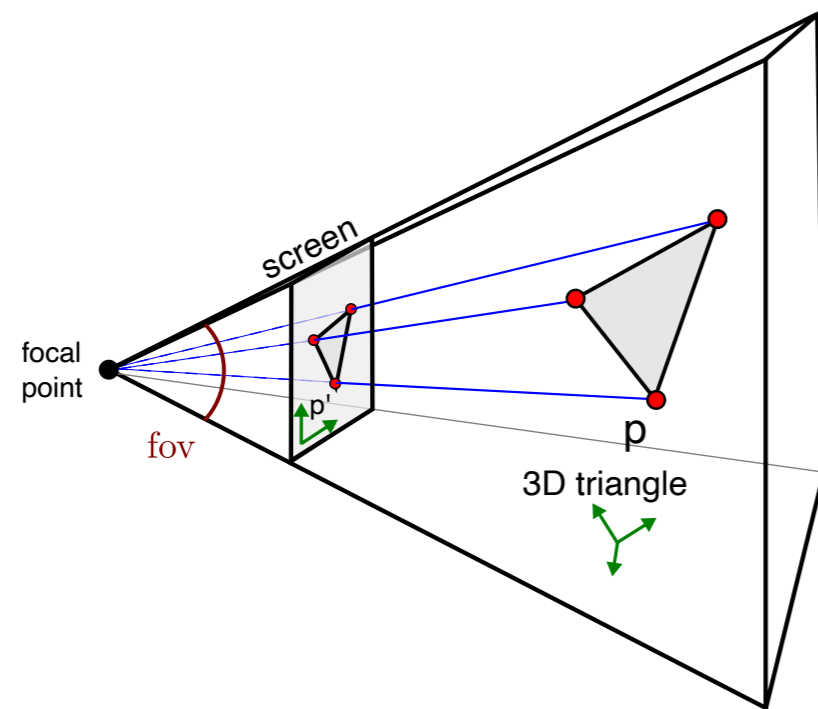
- Suppose une forme composée de triangles
 1. Projette chaque triangle sur le plan caméra
 2. Rasterize les triangles en pixels de couleurs
- + Approche dédiée des GPUs
- Limité aux triangles
- Pas d'effets natifs (ombres, transparence, etc.)

⇒ Le standard du rendu temps réel sur GPU

Projection / Rasterization

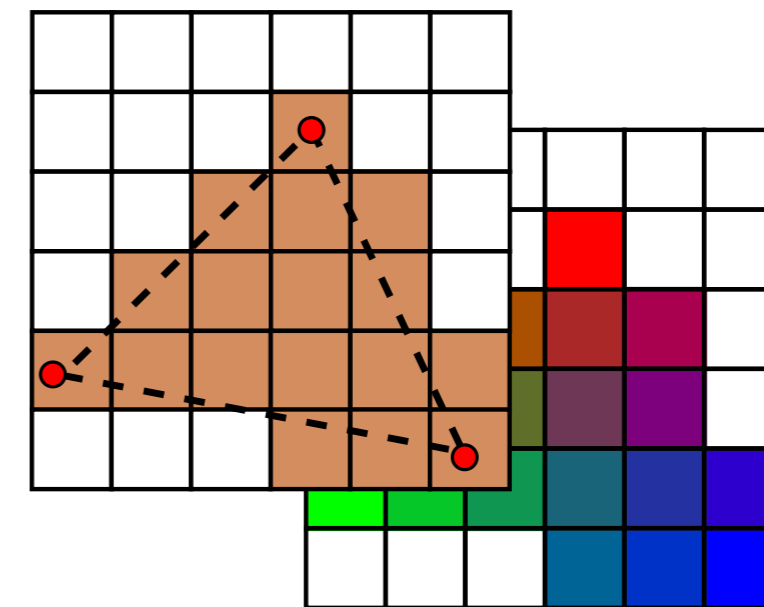
Surfaces composées de **triangles uniquement**

1. Projette sommets des triangles sur plan caméra



- Projection = opération matricielle
Espace projectif $p' = M p$

2. Rasterization



- Discrétisation en pixels (/ fragments)
- Interpolation des attributs (couleurs, etc) sur chaque fragment

=> A vitesse interactive ($\geq 25-60$ fps)

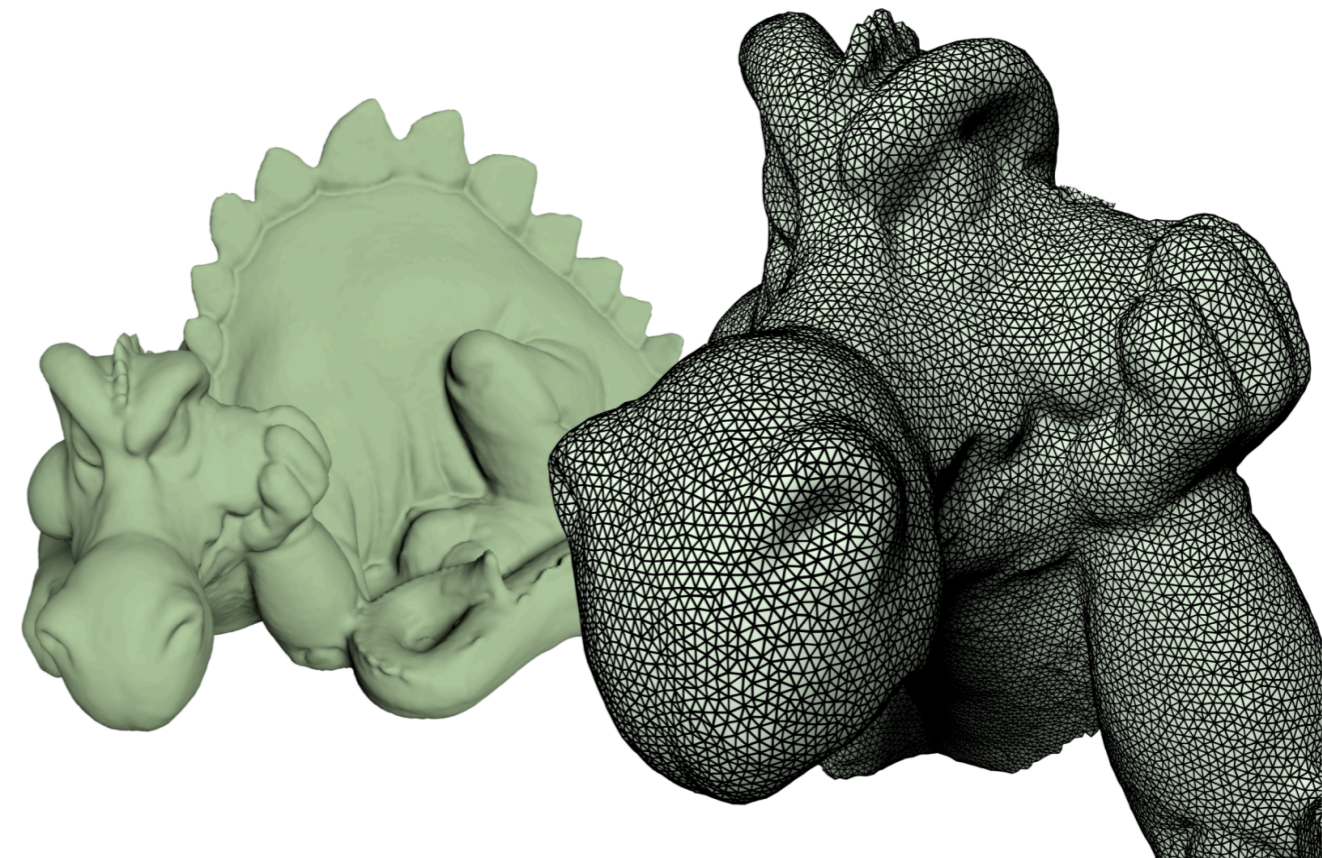
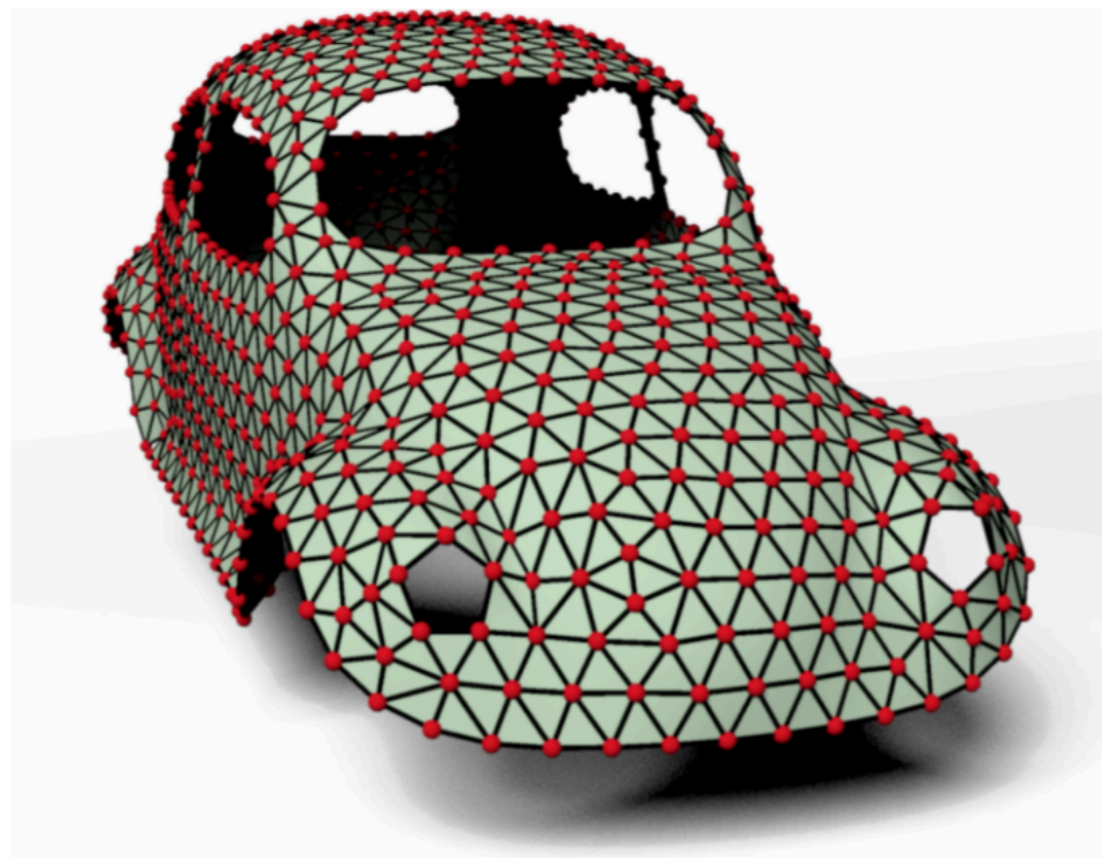
- Projeter l'ensemble des triangles
- Remplir tous les pixels parcouru par les couleurs

Maillage triangulaires

Représentation la plus simple pour une surface continue: ensemble de triangles

Décrite comme un ensemble de

- **Sommets (vertex/vertices)** $\mathcal{V} = (p_1, \dots, p_{N_v}), p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$
- ▲ **Faces**, triplets de Sommets $\mathcal{F} = (f_1, \dots, f_{N_f}), f_i = (p_{i_1}, p_{i_2}, p_{i_3})$
- / **Arêtes (Edges)** optionnels $\mathcal{E} = (e_1, \dots, e_{N_e}), e_i = (p_{i_1}, p_{i_2})$



Propriété des triangles

Triangle T défini par (p_1, p_2, p_3) .

Paramétrisation (u, v) :
$$\begin{cases} p \in T = S(u, v) = p_1 + u(p_2 - p_1) + v(p_3 - p_1) \\ 0 \leq u + v \leq 1 \end{cases}$$

Coordonnées barycentriques:
$$\begin{cases} p \in T = \alpha p_1 + \beta p_2 + \gamma p_3 \\ (\alpha, \beta, \gamma) \in [0, 1], \alpha + \beta + \gamma = 1 \end{cases}$$

Calcul des coordonnées barycentriques pour $p = (x, y, z) \in T$:

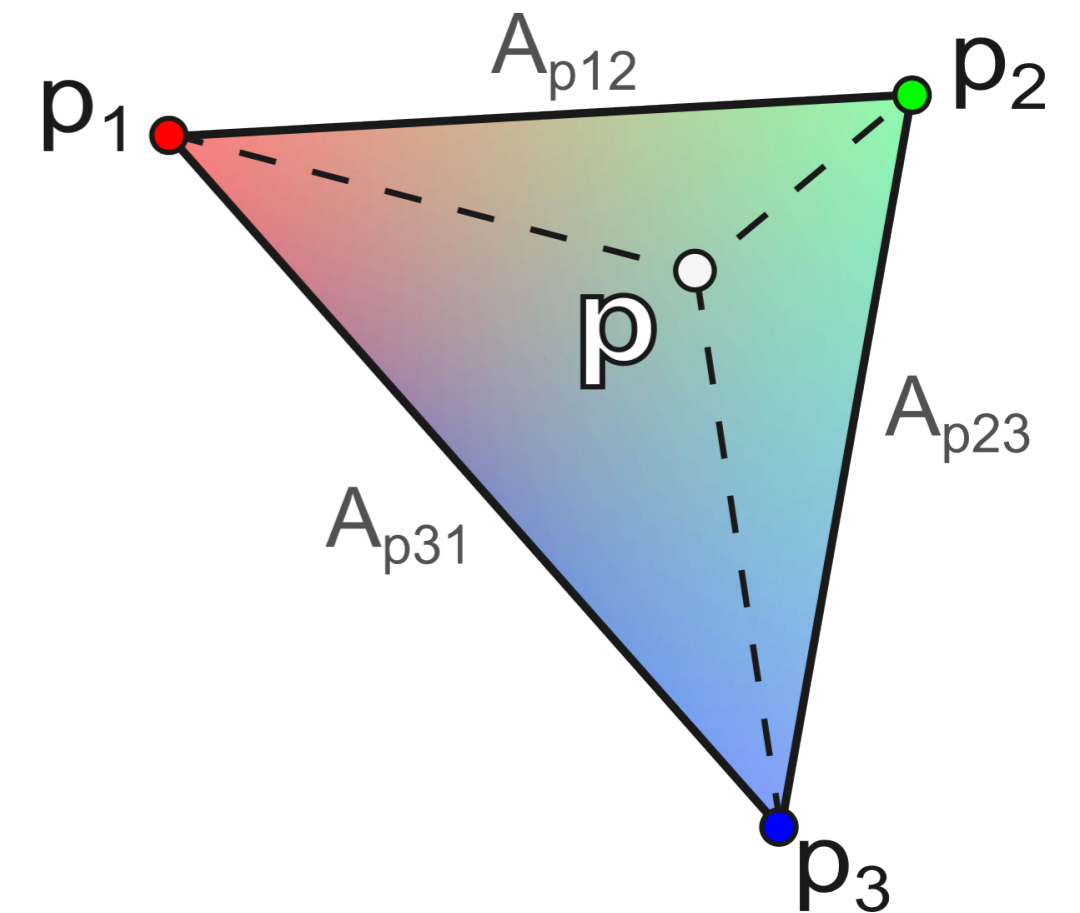
$$\alpha = \frac{A_{p23}}{A_{123}}, \beta = \frac{A_{p31}}{A_{123}}, \gamma = \frac{A_{p12}}{A_{123}}$$

$$A_{123} = \frac{1}{2} \|(p_2 - p_1) \times (p_3 - p_1)\|$$

$$A_{p23} = \frac{1}{2} \|(p_2 - p) \times (p_3 - p)\|$$

$$A_{p31} = \frac{1}{2} \|(p_3 - p) \times (p_1 - p)\|$$

$$A_{p12} = \frac{1}{2} \|(p_1 - p) \times (p_2 - p)\|$$



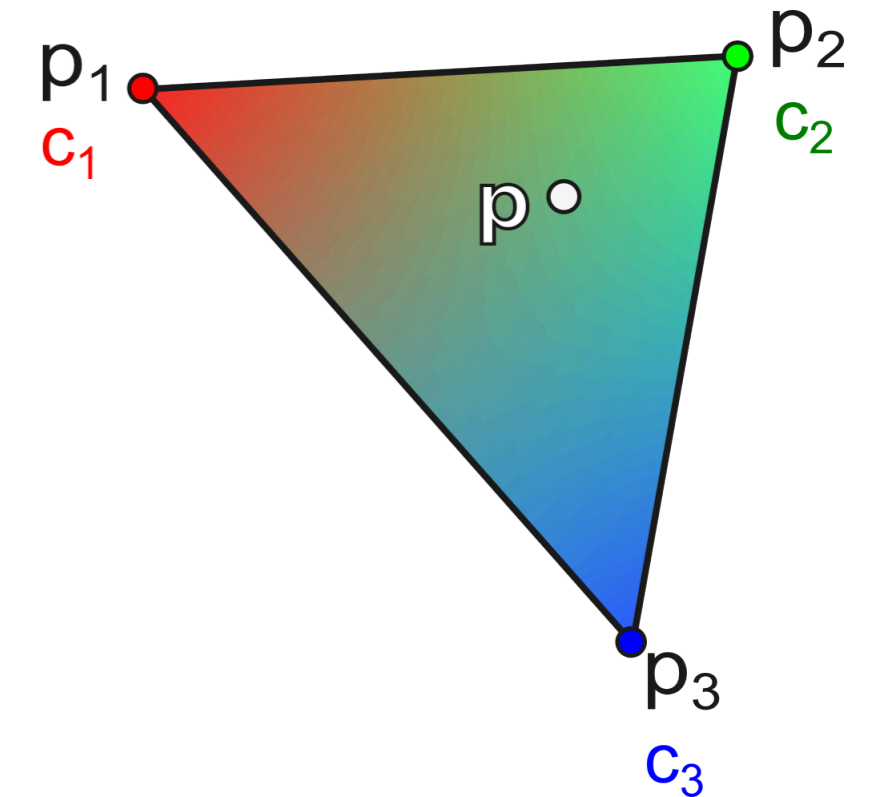
Propriété des triangles - Applications

Interpolation barycentrique/linéaire

Soit un triangle T associées aux couleurs (c_1, c_2, c_3) aux sommets p_1, p_2, p_3

Quelle est la couleur $c(p)$ associée au point p à l'intérieur du triangle?

- 1) Calcul des coordonnées barycentriques (α, β, γ) de p
- 2) Couleur interpolée: $c(p) = \alpha c_1 + \beta c_2 + \gamma c_3$

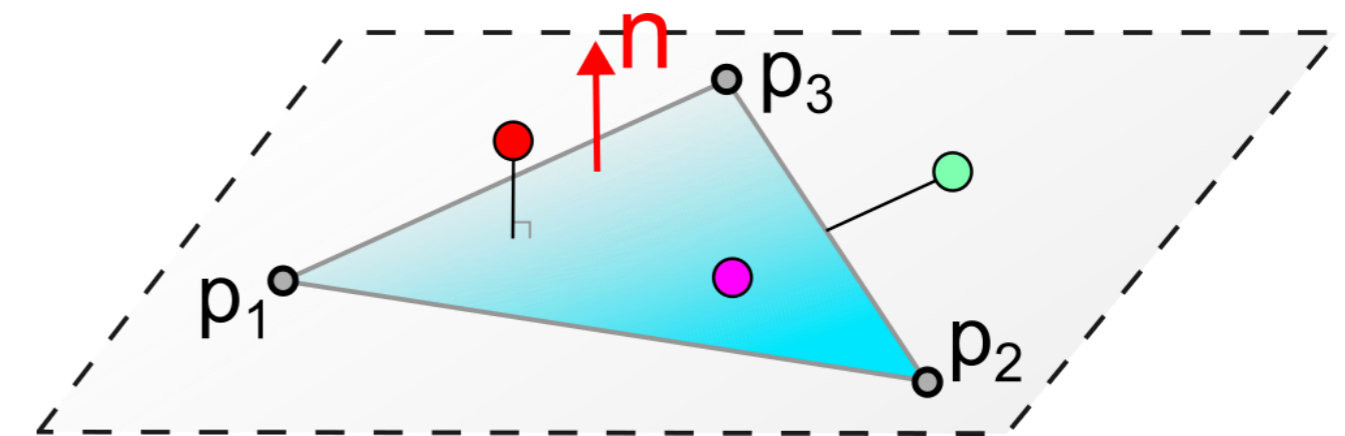


Test d'intersection

Soit un triangle T défini par (p_1, p_2, p_3) .

Testez si le point p est à l'intérieur du triangle T .

- 1) Condition nécessaire: Test si p est dans le plan du triangle
 \Rightarrow Test $(p - p_1) \cdot n = 0$
avec $n = (p_2 - p_1) \times (p_3 - p_1)$, normale à T
- 2) Condition suffisante: Test si p est à l'intérieur du triangle
Calcul des coordonnées barycentriques α, β, γ
 \Rightarrow Test $0 \leq (\alpha, \beta, \gamma) \leq 1$ et $\alpha + \beta + \gamma = 1$



Triangulation description

Exemple d'un tétraèdre (p_0, p_1, p_2, p_3) .

1ère Solution: *Soupe de triangles*

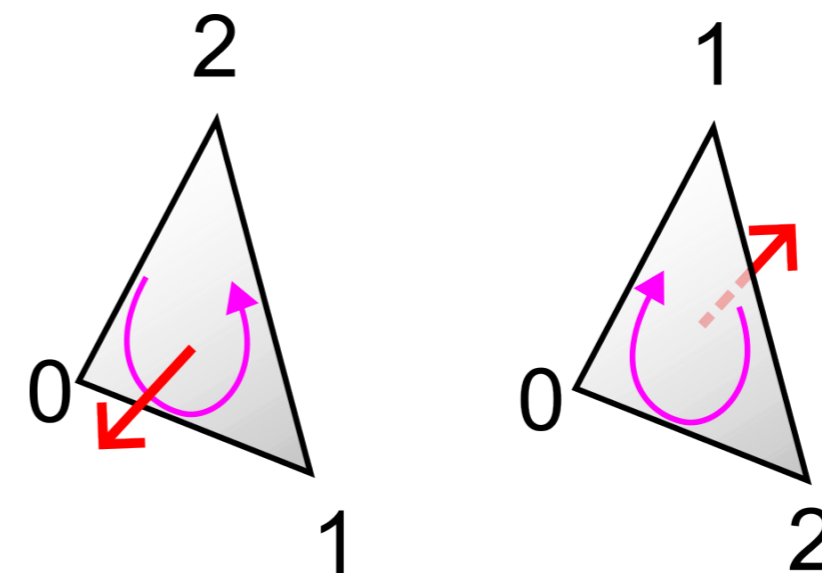
```
triangles = [(0.0, 0.0, 0.0), (1.0, 0.0, 0.0), (0.0, 0.0, 1.0),  
            (0.0, 0.0, 0.0), (0.0, 0.0, 1.0), (0.0, 1.0, 0.0),  
            (0.0, 0.0, 0.0), (0.0, 1.0, 0.0), (1.0, 0.0, 0.0),  
            (1.0, 0.0, 0.0), (0.0, 1.0, 0.0), (0.0, 0.0, 1.0)]
```

2ème solution: *Géométrie, Connectivité (/topologie)*

```
geometry = [(0.0, 0.0, 0.0), (1.0, 0.0, 0.0), (0.0, 1.0, 0.0), (0.0, 0.0, 1.0)]  
connectivity = [(0,1,3), (0,3,2), (0,2,1), (1,2,3)]
```

Rem: L'ordre des indices peut être utilisé pour définir l'orientation des faces.

ex: Main droite / counterclockwise



Exemple de format de fichier de maillage: OBJ

```
v -0.000000 0.620276 0.108446  
v -0.000000 0.685780 0.104094  
v 0.011128 0.685780 0.102245  
v 0.014793 0.620276 0.106125  
v 0.034724 0.684975 0.079817  
v 0.040413 0.620278 0.086828  
v 0.029160 0.619800 0.099405  
v 0.024110 0.685194 0.093530  
v 0.046714 0.554312 0.085764  
v 0.033793 0.547222 0.100284  
v 0.015067 0.542780 0.113608  
v -0.000000 0.541146 0.117759  
v 0.051177 0.430214 -0.047903  
v 0.049948 0.435812 -0.035967  
v 0.028863 0.449897 -0.050037  
v 0.028839 0.444346 -0.059194  
v 0.017691 0.251925 0.023686  
v 0.034131 0.252216 0.014535  
v 0.036689 0.275442 0.012672  
v 0.015166 0.271140 0.025837  
v 0.014869 0.285441 0.024957
```

Introduction à l'Informatique Graphique

- Généralités
- Concepts d'une scène 3D
- **Coordonnées généralisées**
- OpenGL et notion de Shaders

Transformations affines

Translation

$$t(p) = (x + t_x, y + t_y, z + t_z)$$

Homothétie/Scaling

$$s(p) = (s_x x, s_y y, s_z z)$$

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

Rotation

Several possible representations (see later)

$$R = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & j \end{pmatrix}, R R^T = 1 \text{ et } \det(R) = 1$$

Isometrie

$$(a, b, c) \perp (d, e, f) \perp (g, h, j)$$

$$\|(a, b, c)\| = \|(d, e, f)\| = \|(g, h, j)\| = 1$$

Transvection, Cisaillement/Shearing

$$sh_{xy}(p) = (x + \lambda y, y, z)$$

$$sh_{xz}(p) = (x + \lambda z, y, z)$$

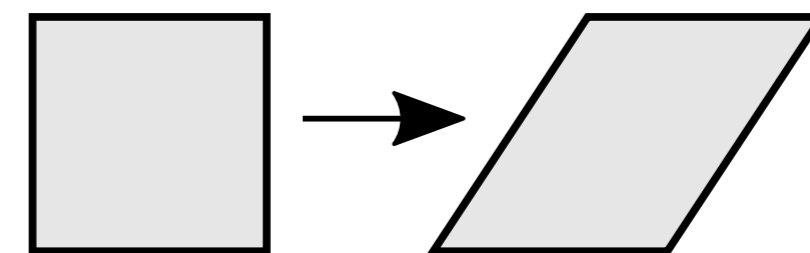
$$sh_{yx}(p) = (x, y + \lambda x, z)$$

...

$$Sh_{xy} = \begin{pmatrix} 1 & \lambda & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} Sh_{xz} = \begin{pmatrix} 1 & \lambda & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} Sh_{yx} = \begin{pmatrix} 1 & 0 & 0 \\ \lambda & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \dots$$

Usually avoided in Graphics

Note: $\det(Sh) = 1 \rightarrow$ constant volume (but no isometry)



Transformations affines et vecteurs / matrices 4D

Translation, Rotation, et Scaling très utilisées pour placer des formes dans l'espace 3D

Rotation et scaling \rightarrow transformation linéaire \rightarrow matrices 3×3 .

Translation \rightarrow transformation non linéaire, non représentable par une matrice 3×3 .

Problème d'encodage d'une chaîne d'opérations

[+] Si uniquement linéaire: $r_1 \circ s \circ r_2 \rightarrow R_1 S R_2 = M \in M_3(\mathbb{R}^3)$

[-] Avec translation: $r_1 \circ tr \circ r_2$ contient deux composantes

- une linéaire $R_1 R_2 \in M_3(\mathbb{R}^3)$

- une translation $R_1 tr \in \mathbb{R}^3$.

Possibilité d'avoir une représentation unifiée en utilisant une représentation 4D

$$p = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad M = \begin{pmatrix} m_{00} & m_{01} & m_{02} & t_x \\ m_{10} & m_{11} & m_{12} & t_y \\ m_{20} & m_{21} & m_{22} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformations affine en 2D

Principe général en 2D (vecteurs unifiés à 3 composantes)

Pour un point $p = (x, y)$

$$\text{Rotation } \mathbf{R} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}, \quad \text{Scaling } \mathbf{S} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}, \quad \text{Translation } (x + t_x, y + t_y)$$

Idée - Ajouter une coordonnée supplémentaire $p = (x, y, 1)$ (coordonnées homogènes / généralisées).

$$\text{Expression linéaire de la translation: } p' = \mathbf{T} p, \text{ avec } p' = \underbrace{\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{T}} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

$$\text{Idem avec les rotations } \mathbf{R} = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{et scaling } \mathbf{S} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Coordonnées généralisées

Coordonnées généralisées en 2D:

Translation T , rotation R , scaling S peuvent être exprimées en produit de matrices

$$\text{ex. } M = T_0 R_0 S_0 T_1 R_1 S_1 \dots \quad M = \left(\begin{array}{cc|c} m_{00} & m_{01} & t_x \\ m_{10} & m_{11} & t_y \\ \hline 0 & 0 & 1 \end{array} \right).$$

m_{ij} : partie linéaire (rotation and scaling); $t_{x/y}$: translation

Coordonnées généralisées en 3D

Vecteurs à 4-composantes, et matrices 4×4 .

$p = (x, y, z, 1)$ - position 3D

$$M = \left(\begin{array}{ccc|c} m_{00} & m_{01} & m_{02} & t_x \\ m_{10} & m_{11} & m_{12} & t_y \\ m_{20} & m_{21} & m_{22} & t_z \\ \hline 0 & 0 & 0 & 1 \end{array} \right) - \text{transformation affine générale (rotation, scaling, translation)}$$

$$\text{On pourra écrire par bloc: } M \begin{pmatrix} p \\ 1 \end{pmatrix} = \left(\begin{array}{c|c} \mathbf{L} & t \\ \hline 0 & 1 \end{array} \right) \begin{pmatrix} p \\ 1 \end{pmatrix} = \mathbf{L}p + t$$

Points et vecteurs

Intérêt coordonnées généralisées: différentiation points / vecteurs

Point $(x, y, z, \mathbf{1})$

$$\mathbf{M} \begin{pmatrix} p \\ 1 \end{pmatrix} = \left(\begin{array}{c|c} \mathbf{L} & t \\ \hline 0 & 1 \end{array} \right) \begin{pmatrix} p \\ 1 \end{pmatrix} = \mathbf{L}p + t$$

La translation s'applique

Vecteur $(x, y, z, \mathbf{0})$

$$\mathbf{M} \begin{pmatrix} p \\ 0 \end{pmatrix} = \left(\begin{array}{c|c} \mathbf{L} & t \\ \hline 0 & 1 \end{array} \right) \begin{pmatrix} p \\ 0 \end{pmatrix} = \mathbf{L}p$$

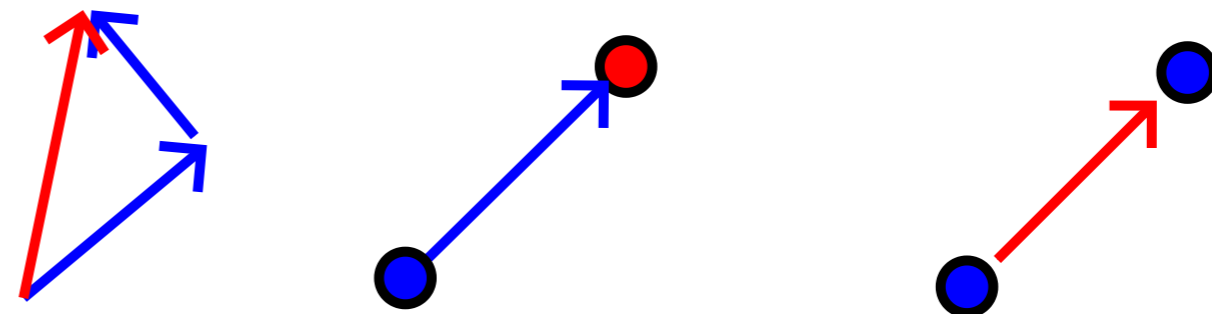
La translation ne s'applique pas

Cohérence avec opérations géométriques:

vecteur \pm vecteur \rightarrow vecteur

position \pm vecteur \rightarrow position

position - position \rightarrow vecteur



Coordonnées homogènes, espace projectif

Cas d'un point généralisé avec coordonnées $w \neq 1$

$$p_{4D} = (x, y, z, w) \Rightarrow \text{"renormalization/projection"} \text{ sur l'espace des points 3D: } p_{3D} = (x/w, y/w, z/w, 1)$$

Ex.

$$\text{- Somme de deux points: } (x_1, y_1, z_1, 1) + (x_2, y_2, z_2, 1) = (x_1 + x_2, y_1 + y_2, z_1 + z_2, 2)$$

$$\Rightarrow \text{homogénéisation: } p_{3D} = \left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}, \frac{z_1+z_2}{2}, 1 \right) \Rightarrow \text{Barycentre}$$

Intérêt: Encoder opérations rationnelles par une matrice

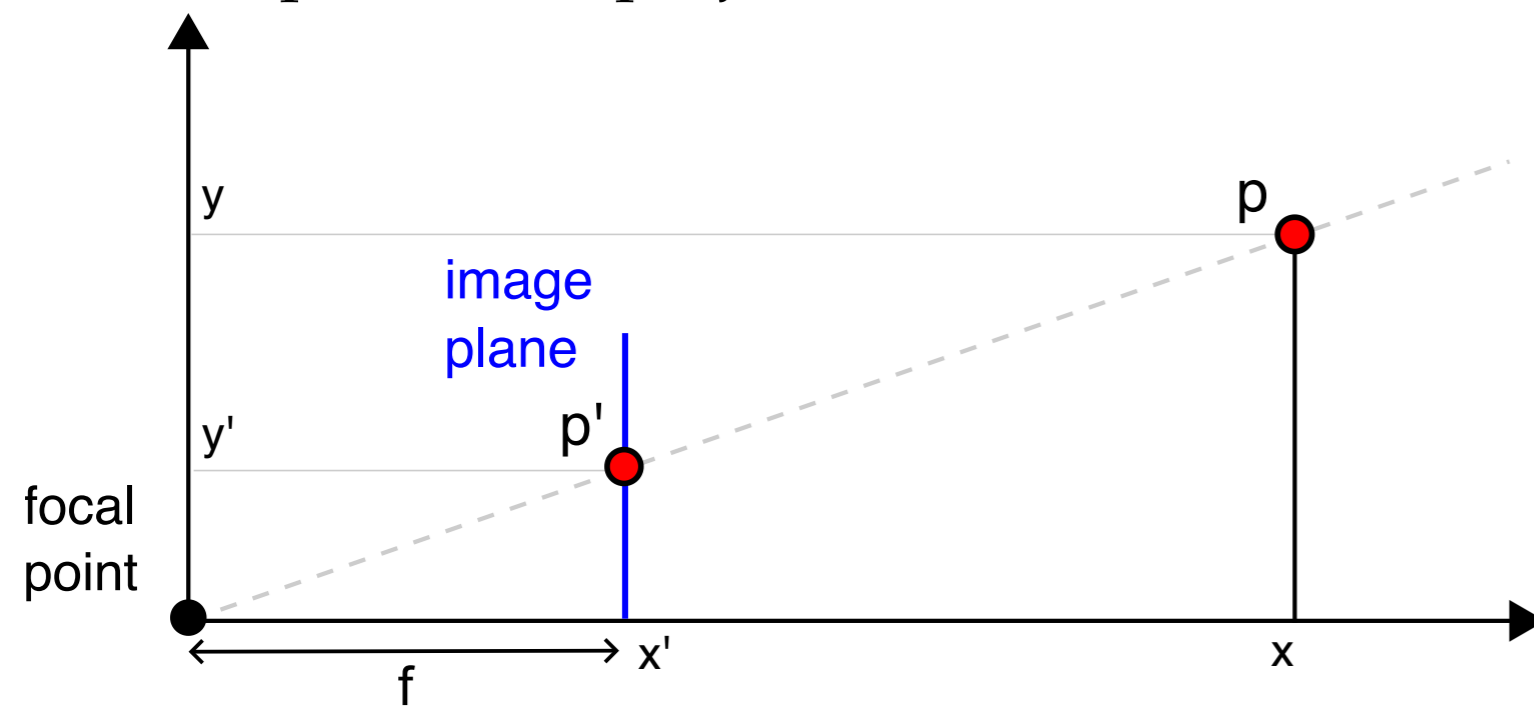
$$p_{4D} = \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} m_{00}x + m_{01}y + m_{02}z + m_{03}w \\ m_{10}x + m_{11}y + m_{12}z + m_{13}w \\ m_{20}x + m_{21}y + m_{22}z + m_{23}w \\ m_{30}x + m_{31}y + m_{32}z + m_{33}w \end{pmatrix}$$

$$\Rightarrow p_{3D} = \begin{pmatrix} \frac{m_{00}x + m_{01}y + m_{02}z + m_{03}w}{m_{30}x + m_{31}y + m_{32}z + m_{33}w} \\ \frac{m_{10}x + m_{11}y + m_{12}z + m_{13}w}{m_{30}x + m_{31}y + m_{32}z + m_{33}w} \\ \frac{m_{20}x + m_{21}y + m_{22}z + m_{23}w}{m_{30}x + m_{31}y + m_{32}z + m_{33}w} \\ 1 \end{pmatrix}$$

Application à la perspective

Perspective modélisé par une division par la profondeur

Exemple en 2D (projection 1D)



$$y' = x' \frac{y}{x} = f \frac{y}{x} \quad (f: \text{ focale})$$

Modèle *linéaire*:

$$p' = \begin{pmatrix} f \\ f \frac{y}{x} \\ 1 \end{pmatrix} \Rightarrow p' = \begin{pmatrix} fx \\ fy \\ x \end{pmatrix} = \underbrace{\begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 1 & 0 & 0 \end{pmatrix}}_{P_p} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Espace projectif

- Points *réels* (2D | 3D) sont sur $z|w = 1$

Coordonnées réelles des points sont obtenues après homogénéisation (division by dernière composante).

- Vecteurs sont sur $z|w = 0$

Introduction à l'Informatique Graphique

- Généralités
- Concepts d'une scène 3D
- Coordonnées généralisées
- **OpenGL et notion de Shaders**

CPU et GPU



CPU: Central Processing Unit

Mémoire associée: RAM

⇒ Tâches arbitraires séquentielles:

Calcul, branching, accès mémoires, I/O, etc.

Coeurs: 2 à 64, indépendants

RAM: 4 à 64 Go



GPU: Graphics Processing Unit

Mémoire associée: VRAM

⇒ Tâches parallèles avec opérations similaires:

Calculs vectoriels, matriciels sur tableaux de données.

Architecture: SIMD

Single Instruction, Multiple Data

Coeurs: 1000 à 16000, partagés

VRAM: 4 à 16 Go

GPU: Supercalculateur pour réaliser la même opération sur un grand nombre de données.

OpenGL

OpenGL: Open Graphics Library

Une API pour communiquer avec le GPU, orienté graphique 3D

API: Application Programming Interface = Ensemble standardisé de variables et en-tête de fonctions.

Bas niveau, Haute performance, Multi plateformes

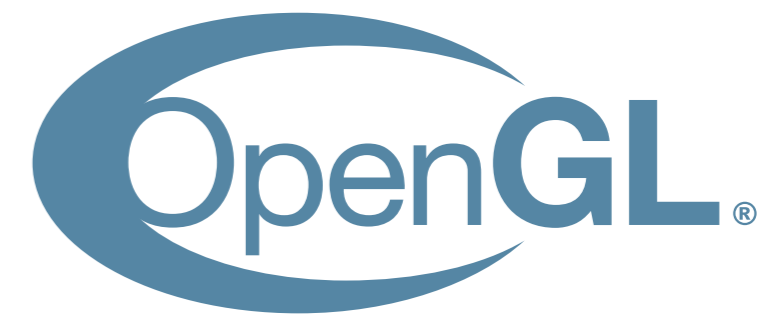
Rem.

API \neq logiciel, librairie

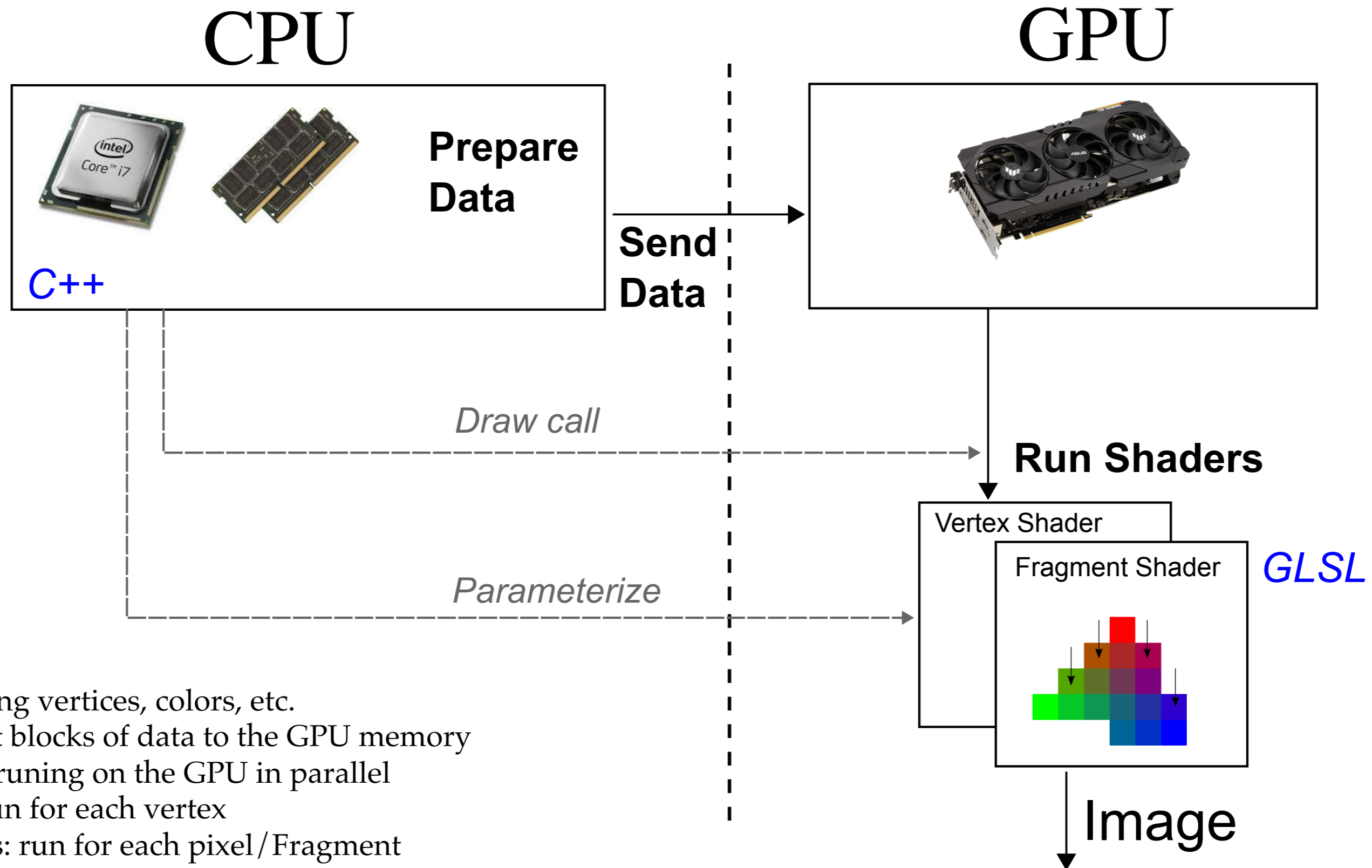
Différents systèmes / GPU ont différentes implémentations d'OpenGL

Installation dépend du driver

Autres APIS: Vulkan, WebGL, WebGPU, DirectX (Windows), Metal (Mac).



Communication CPU / GPU with OpenGL



1- Preparing the data:

Loading / Computing vertices, colors, etc.

2- Send data: Transfert blocks of data to the GPU memory

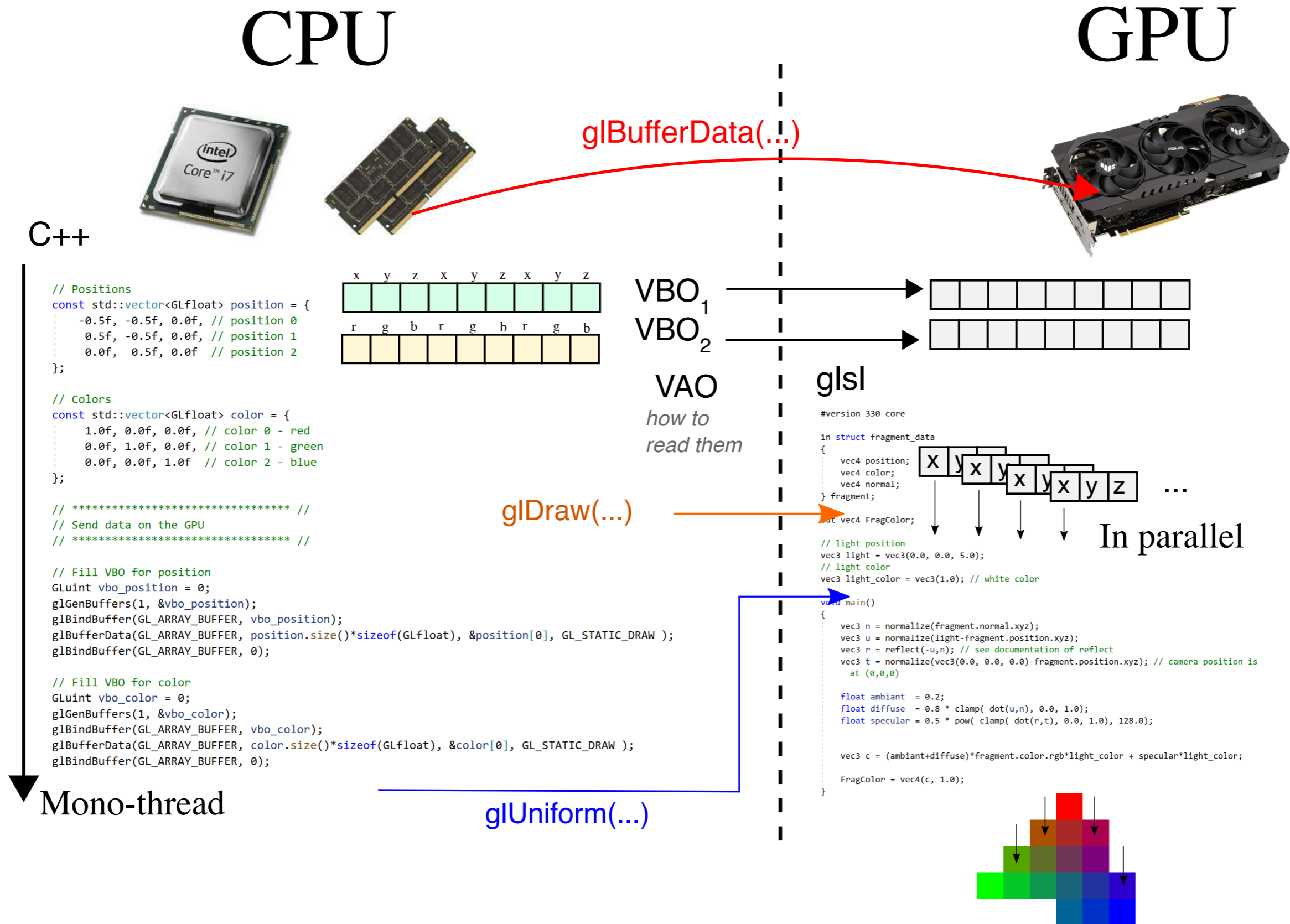
3- Shaders: Programs runing on the GPU in parallel

- Vertex shaders: run for each vertex

- Fragment shaders: run for each pixel / Fragment

Output: The final image

Communication CPU / GPU with OpenGL - details



Shader

Vertex Shader

```
#version 330 code
layout(location = 0) in vec4 position;
layout(location = 1) in vec4 color;

out vec4 vertexColor;

void main()
{
    gl_Position = position;
    vertexColor = color;
}
```

Fragment Shader

```
#version 330 code
in vec4 vertexColor;
out vec4 fragColor;

void main()
{
    fragColor = vertexColor;
}
```

