Introduction à l'Informatique Graphique

Florence Zara

Université Claude Bernard Lyon 1, LIRIS

Plan du cours

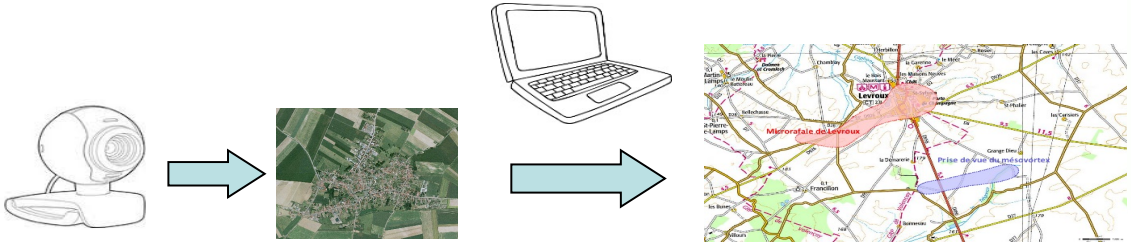
- Informatique et image : différents domaines de recherche
  - Traitement d'images, synthèse d'images, Réalité Augmentée, Réalité Virtuelle
- Informatique et Image : différents domaines d'applications
- Comment créer des images virtuelles ?
  - Modélisation, animation, visualisation, pipeline graphique
- Présentation en détails des étapes de création d'une scène 3D
- Matériel permettant d'effectuer les calculs nécessaires à la création des images

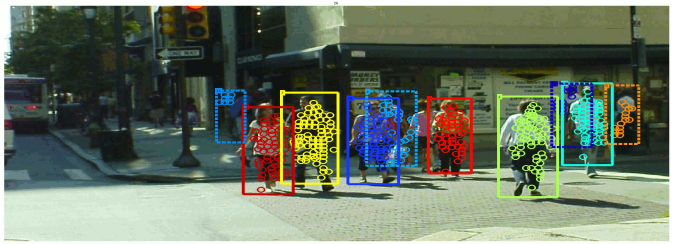
Informatique et Images :

différents domaines de recherche

Informatique et images : différents domaines de recherche

Analyse d'images (*Image Analysis*)
- Traitement d'images (*Image Processing*)
- Reconnaissance des formes (*Pattern Recognition*)
- Vision par ordinateur (*Computer Vision*)

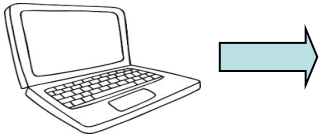


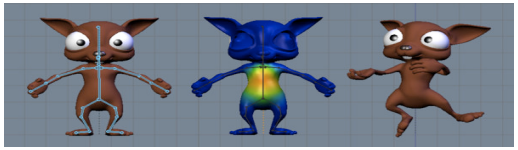


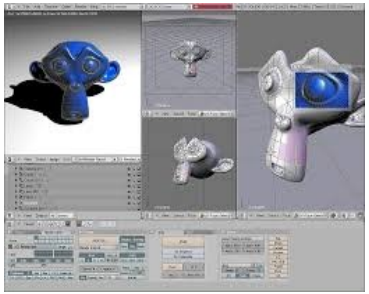
Informatique et images : différents domaines de recherche

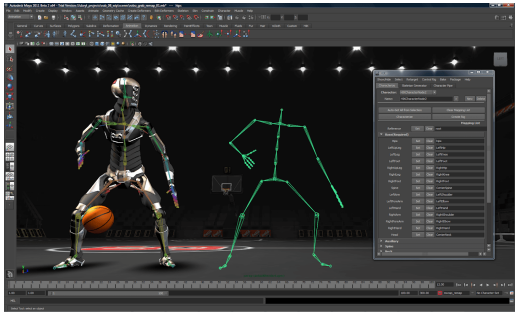
Synthèse d'images (*Computer Graphics*)

- Modélisation
- Animation
- Rendu



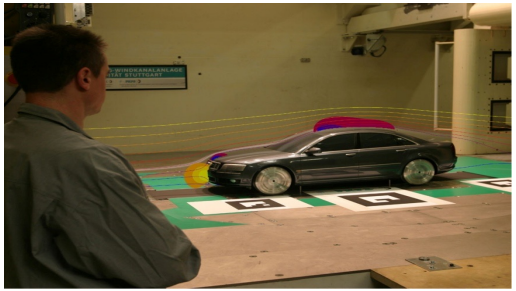






Informatique et images : différents domaines de recherche

Mélange des 2 = image réelle + image virtuelle = **Réalité Augmentée**

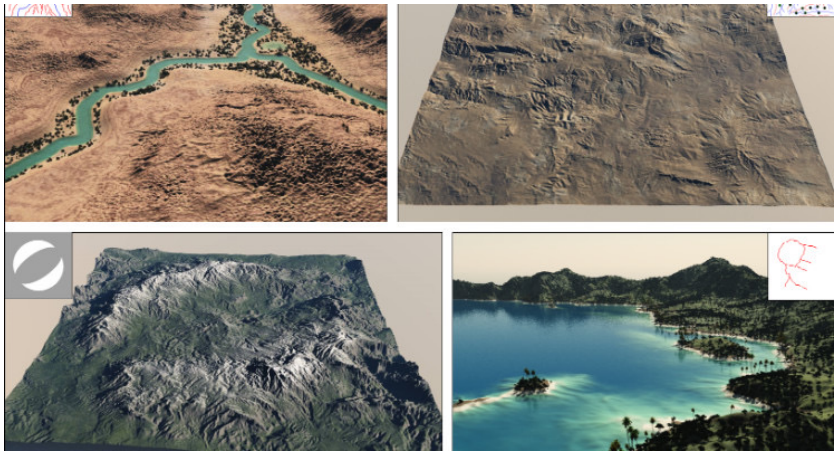


Informatique et images : différents domaines de recherche

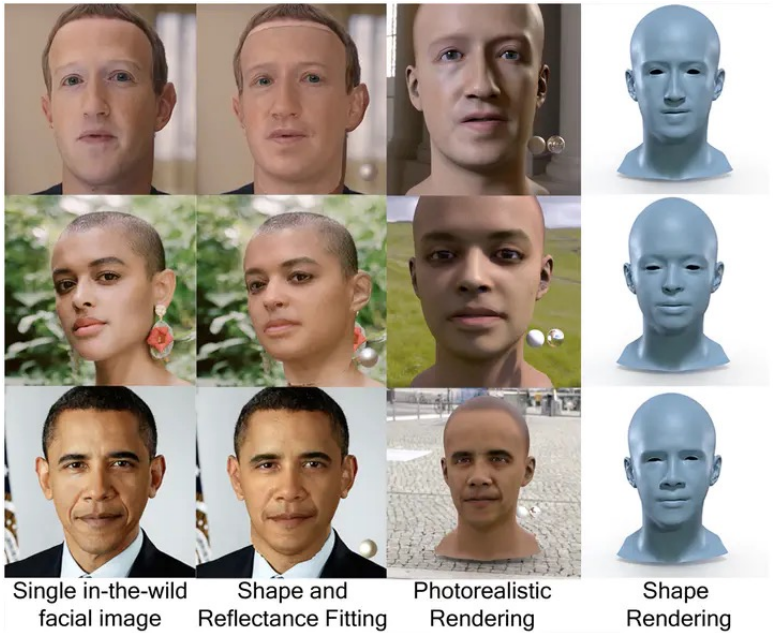
Synthèse d'image + matériel de vision + robotique : **Réalité Virtuelle**

Immersion + interaction

Informatique et image : utilisation de l'Intelligence Artificielle



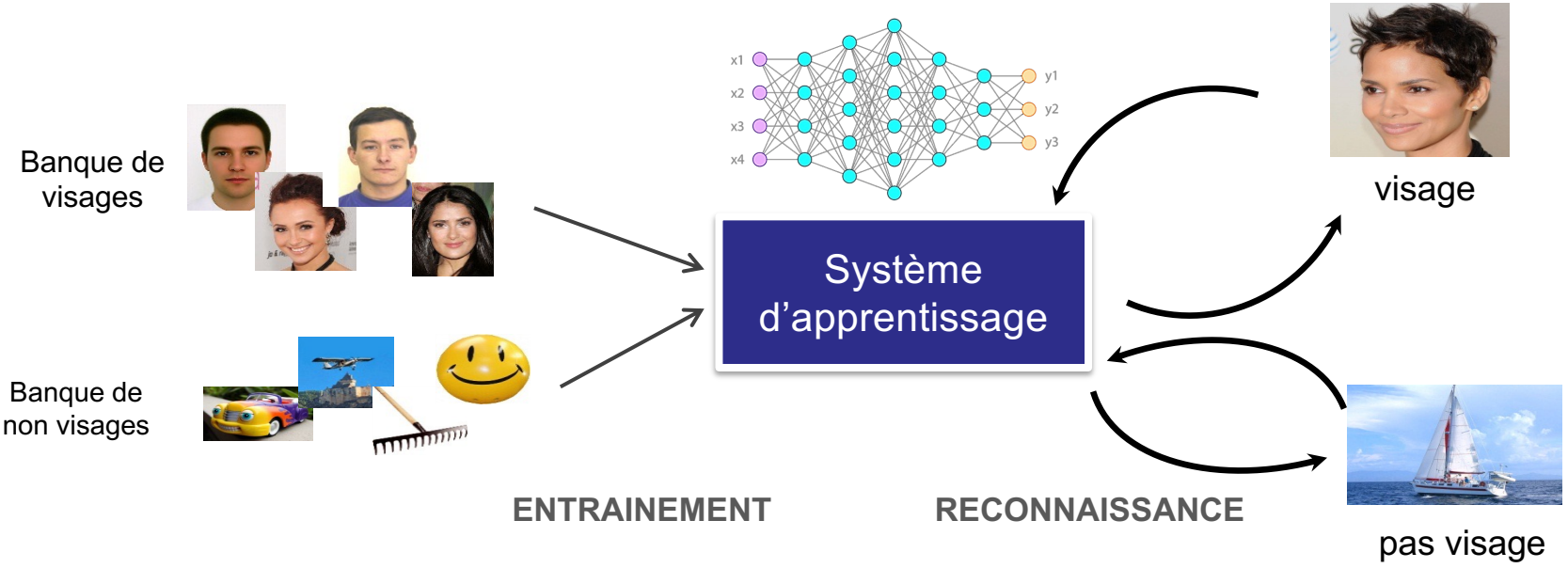
Pour générer / analyser des images



Informatique et images : utilisation de l'Intelligence Artificielle

Apprentissage automatique pour le traitement d'images

A partir d'une banque d'exemples, l'ordinateur apprend à classer les éléments



Informatique et Images :

différents domaines d'applications

Pourquoi créer des images virtuelles ?

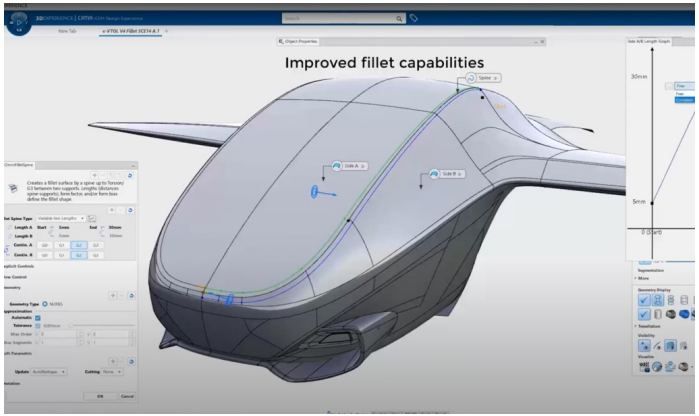
De nombreux domaines d'application pour l'informatique graphique (*Computer Graphics*)

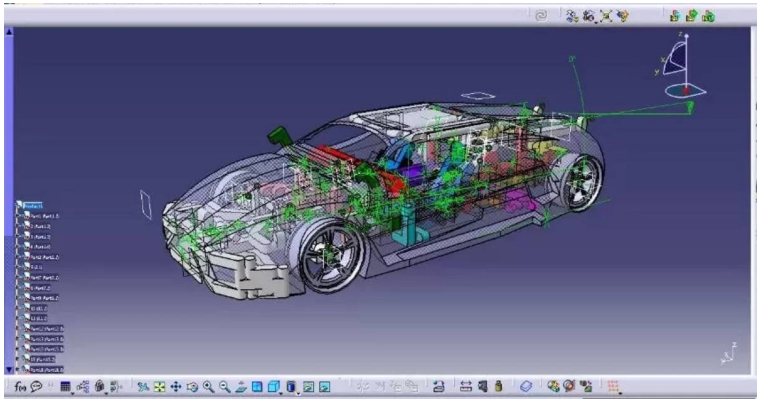
Illustration de quelques exemples :

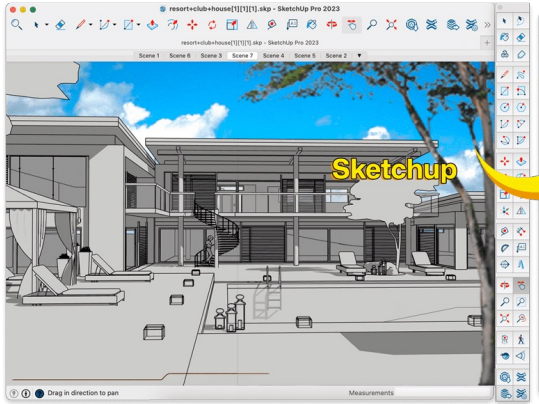
- Architecture / Conception Assistée par Ordinateur (CAO)
- Applications pour le médical
- Visualisation scientifique
- Loisirs numériques : films d'animation, jeux vidéo, effets spéciaux

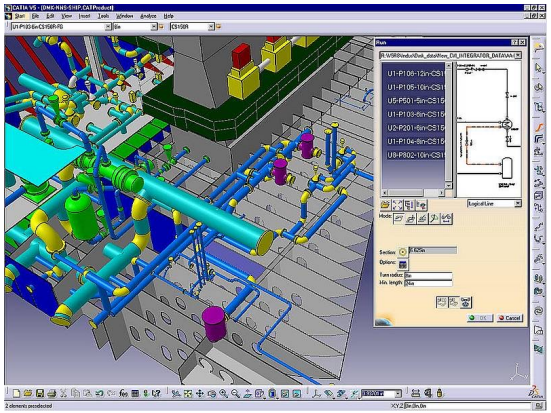
Mais beaucoup d'autres…

Domaines d’applications - Architecture / CAO

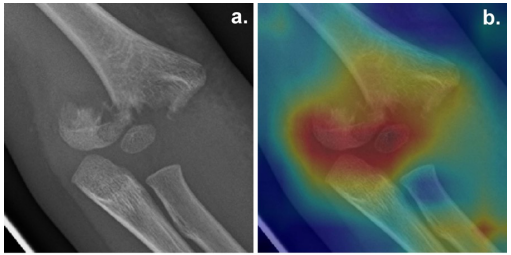


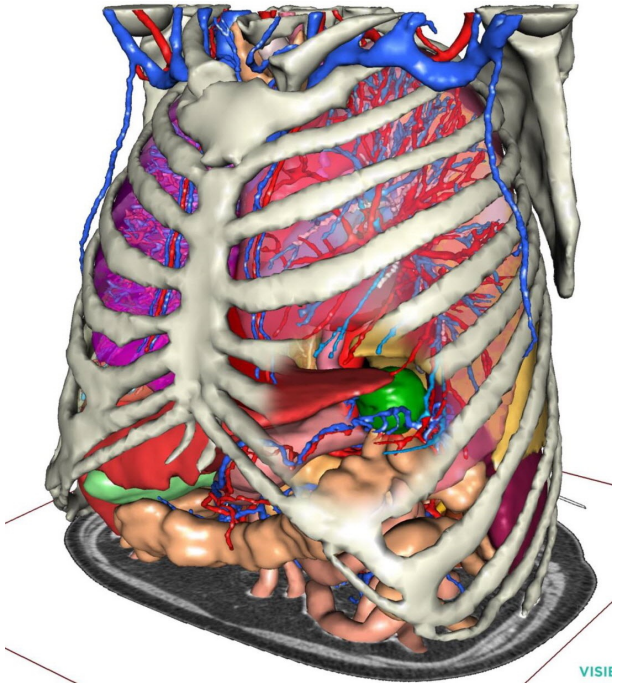




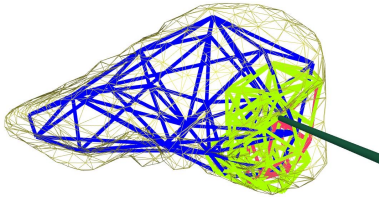
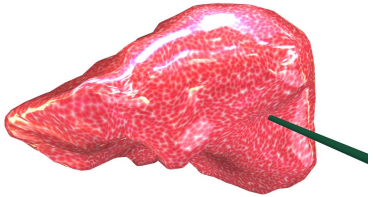
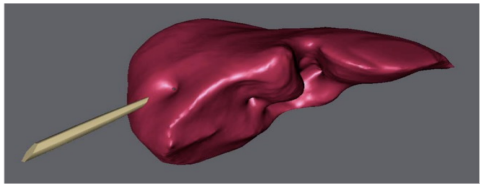


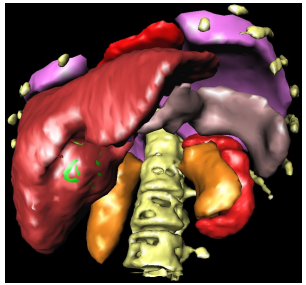
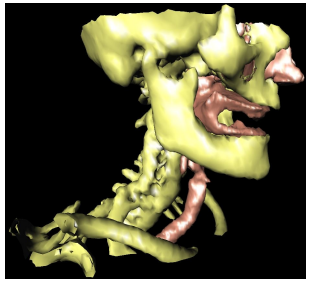
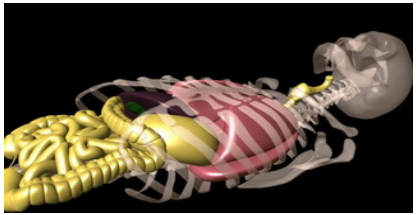
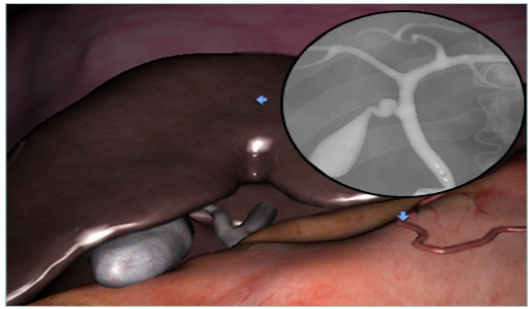
Domaines d’applications - Imagerie médicale



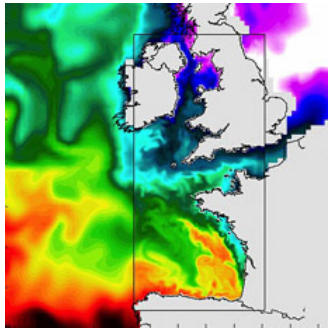


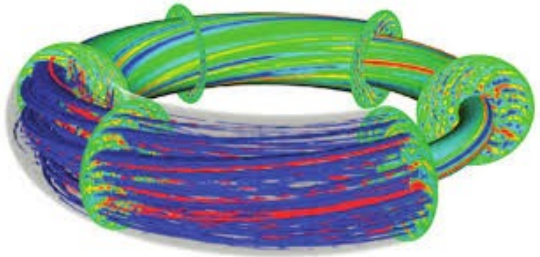
Domaines d'applications – Simulation biomécanique

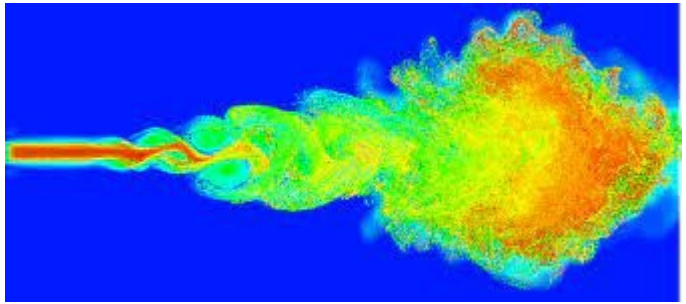




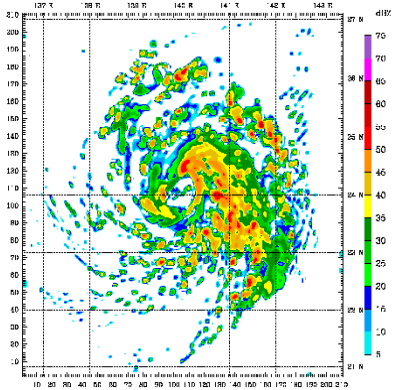





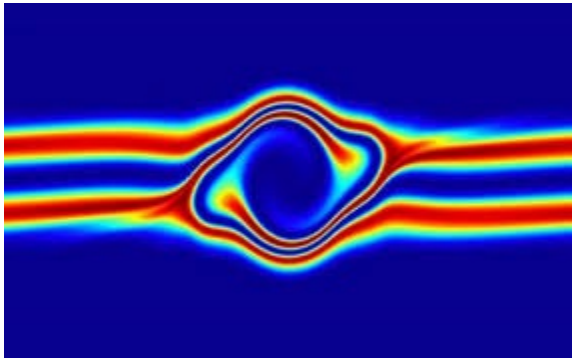
Domaines d'applications - Visualisation scientifique







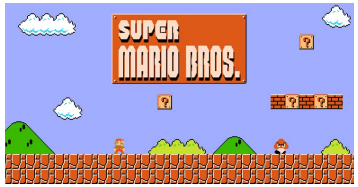




Domaines d’applications - Films d’animation

Pixar

Domaines d’applications - Jeux vidéo



Domaines d’applications – Effets spéciaux

Domaines d'applications – en bilan

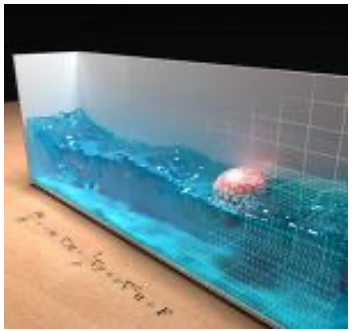
Pas les mêmes besoins selon les domaines d'application

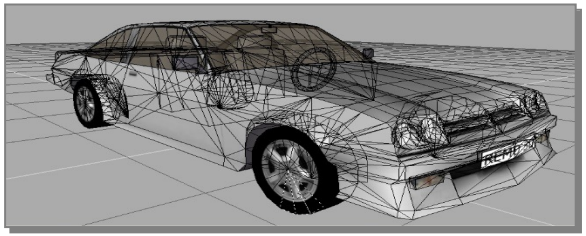
- **Réalisme** plus ou moins important
- **Temps d'exécution** plus ou moins important
- **Interactivité ou non** avec les images créées

Pas les mêmes méthodes / algorithmes employés pour créer ces images

- Modèles et algorithmes différents
- Utilisation ou non du GPU, multi-cœurs : exécution séquentielle ou parallèle
- De plus en plus d'alternatives avec l'emploi de l'Intelligence Artificielle

Comment créer des images virtuelles ?





Création d'une image – Au fait, c'est quoi une image en Informatique ?

- Rectangle (2D) : tableau 2D de pixels (= *picture element*)
  - nombre de lignes
  - nombre de colonnes
  - format des pixels (bit, niveaux de gris, niveaux de couleurs)
  - *compression éventuelle*

Continuous image

Digital image

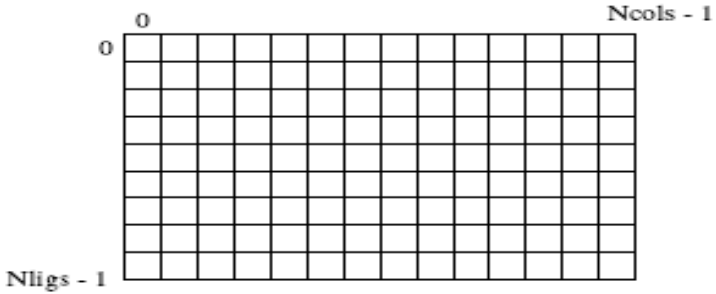
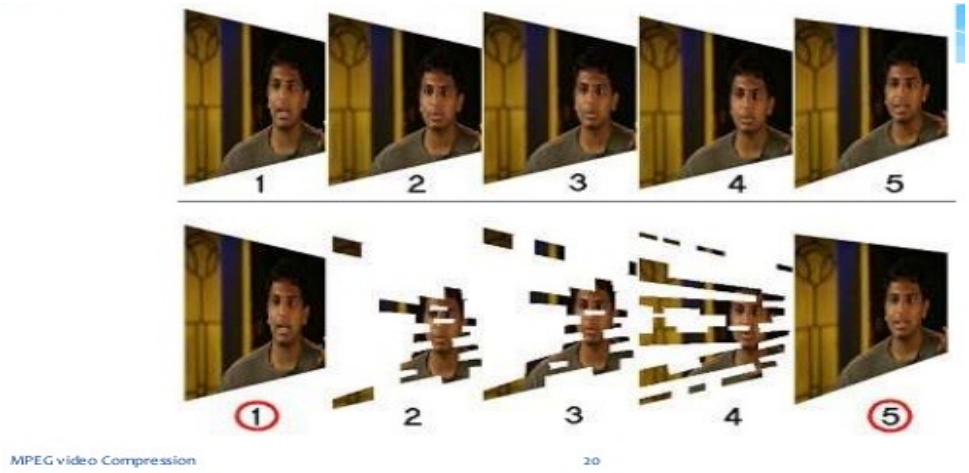


Image = ensemble de pixels

Et une video ? C'est une séquence d'images

Il y a souvent de la compression : des images, et entre les images

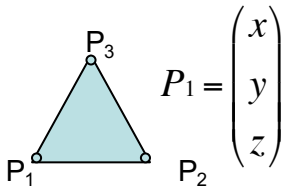


Etapes de bases pour créer des images virtuelles animées

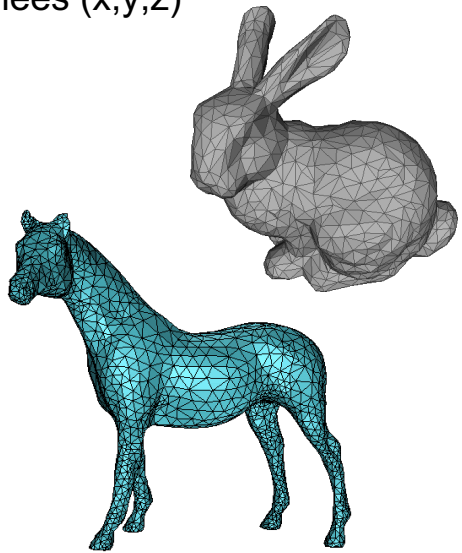
1. **Création de la scène 3D**
   - Un ou plusieurs objets à créer et positionner dans la scène
   - Objet constitué de sommets / d'arêtes / de faces / de volumes élémentaires
   - Chaque sommet est défini par sa position dans l'espace 3D : coordonnées (x,y,z)

Considérons le cas où les objets sont décrits par des **triangles**

Représentation de l'objet = maillage surfacique / triangulaire qui est défini par :
- un ensemble de sommets
- un ensemble de faces (triangles)
- faces décrites par 3 sommets
- faces reliées ensemble par des arêtes communes



$$P_1 = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$



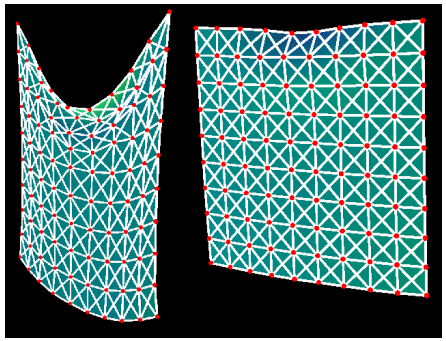
C'est **l'étape de modélisation des objets et création** de la scène

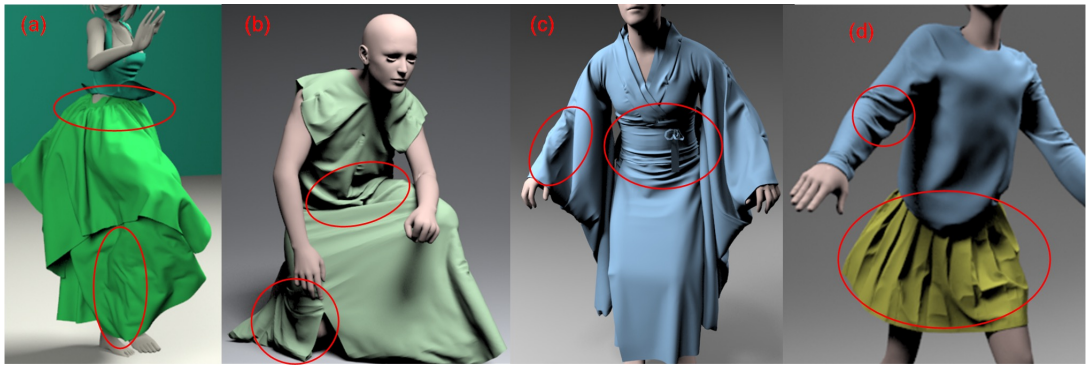
Etapes de bases pour créer des images virtuelles animées

2. Les coordonnées (x,y,z) des sommets des objets peuvent changer au cours du temps

- Les objets sont en mouvement : translation, rotation
- Les objets peuvent se déformer :
  - Changement de coordonnées des sommets induit déformation des faces / volumes de l'objet

C'est **l'étape d'animation / simulation** des objets de la scène



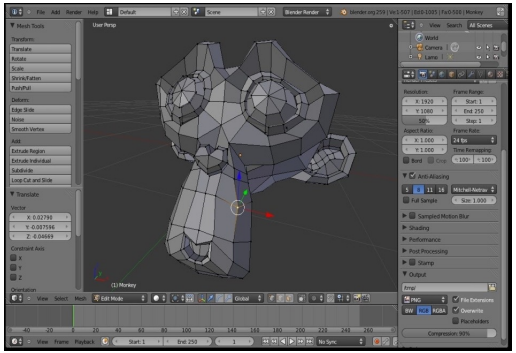


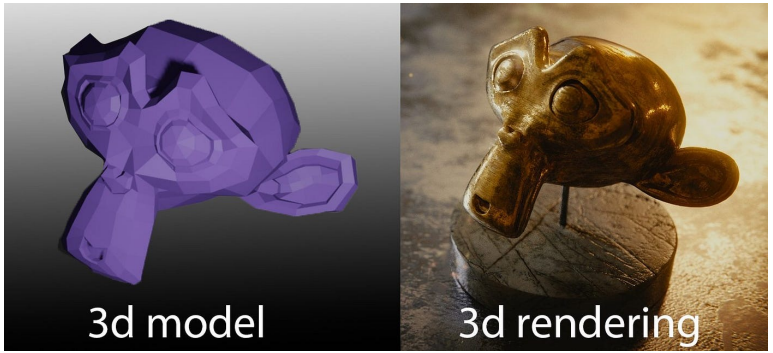
Etapes de bases pour créer des images virtuelles animées

3. Informations supplémentaires pour afficher la scène 3D sur l'écran 2D

- Des primitives graphiques (couleur, propriétés matériaux) sont attachées aux sommets
- Des lumières doivent être positionnées dans la scène : position (x,y,z) des spots ayant une couleur
- Un observateur (ou une caméra) doit être positionné : position (x,y,z) de la caméra
- *Qui s'ajoutent à la position et orientation des objets de la scène dans le repère monde (repère initial)*

Ces informations permettront de faire l'affichage à l'écran de la scène qui est **l'étape de rendu**





En résumé, les étapes de bases pour créer des images virtuelles animées

1- Modélisation : représentation mathématique des objets virtuels

2- Simulation / animation : déformation et mouvement des objets virtuels

3- Visualisation / rendu : affichage des objets virtuels - pipeline graphique

modélisation en 3D

rendu

Chaque étape correspond à un domaine d'expertise à part entière

Présentation en détails des étapes de création d'une scène 3D

1- Modélisation géométrique : représentation mathématique des objets virtuels

2- Simulation / animation : déformation et mouvement des objets virtuels

3- Visualisation / rendu : affichage des objets virtuels

Modélisation géométrique

Intérêt

Permet de représenter un objet qui sera ensuite manipulé
- objets réels ou virtuels / inventés
- carrosserie de voiture, réacteur nucléaire, etc.

Permet de rendre observables certaines caractéristiques
- visite de réacteur nucléaire, etc.

Remplace les maquettes à taille réduite
- réduction des coûts

Utilisation

Affichage, animation, simulation physique, etc.

Modélisation géométrique

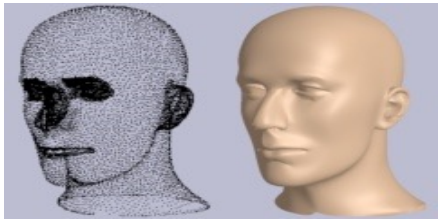
Il existe plusieurs types de modèles géométriques

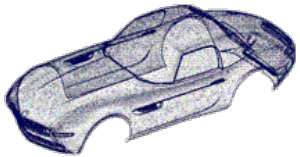
- modèles non structurés
  - points
  - soupes de polygones
- modèles surfaciques
  - maillage surfacique
  - surface paramétrique
  - surface de subdivision
  - surface implicite
- modèles volumiques
  - maillages volumiques
    - voxels, tétraèdres, etc.
- modèles procéduraux
  - fractales
  - système de particules
- modèles à base d'images
  - acquisition
  - rendu

Modèles non structurés - Nuages de points

Modèles obtenus

- par digitalisation manuelle ou par scanner
- par reconstruction à partir d'images
- par échantillonnage d'un autre modèle





simple à afficher mais demande beaucoup de ressource mémoire

Modèles non structurés - Soupes de polygones

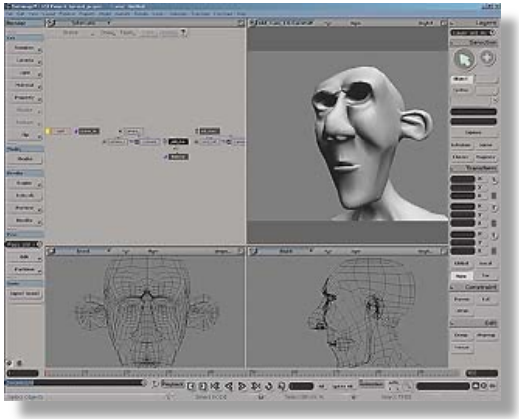
Ensemble non structuré de facettes

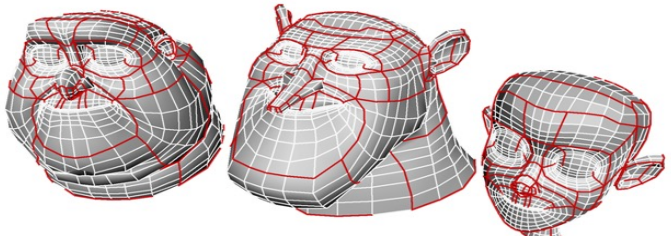
Avantages :

- représentation native de OpenGL
- nombreux logiciels d'édition

Inconvénients :

- opérations autre que l'affichage compliquées
- édition fastidieuse





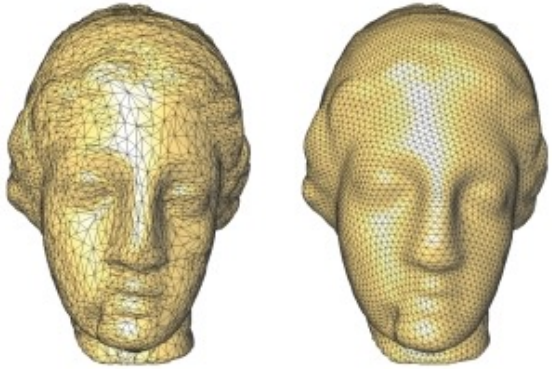
Modèles surfaciques – Maillages surfaciques

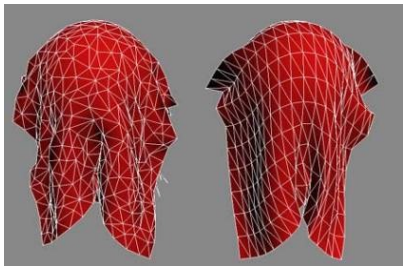
Ensemble connecté de polygones
- triangles, quads, polygones convexes

Permet de représenter
- la forme et la topologie de l'objet

Affichage facile avec librairie dédiée (OpenGL)

Permet d'effectuer facilement des calculs
- normales, courbures, simplification, etc.





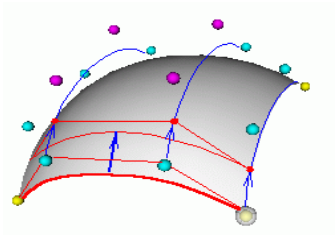
⇨ modèle souvent employé en animation (simulation de textiles par exemple)

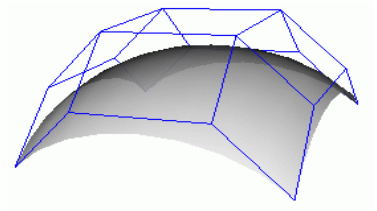
Modèles surfaciques – Surfaces paramétriques

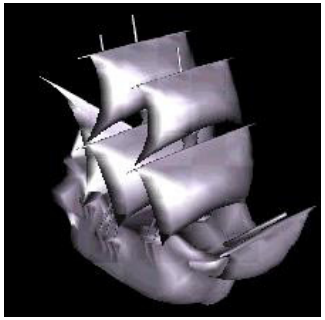
Définition par des points de contrôle (splines)

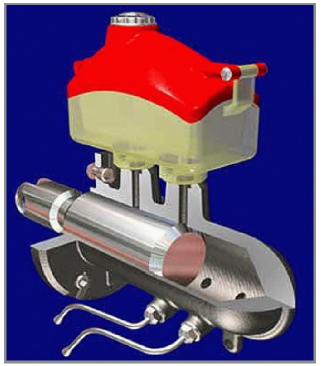
Convertible en maillage pour effectuer le rendu

Edition aisée









Modèles surfaciques – Surfaces de subdivision

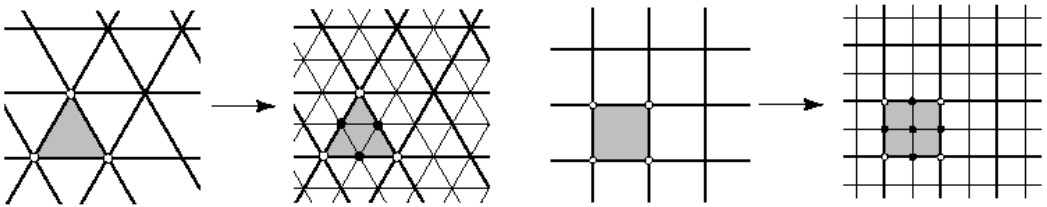
Schéma de subdivision appliqué récursivement sur un maillage

→ raffinement d'un ensemble de primitives géométriques en un ensemble plus dense

→ création de nouveaux points et de nouvelles faces

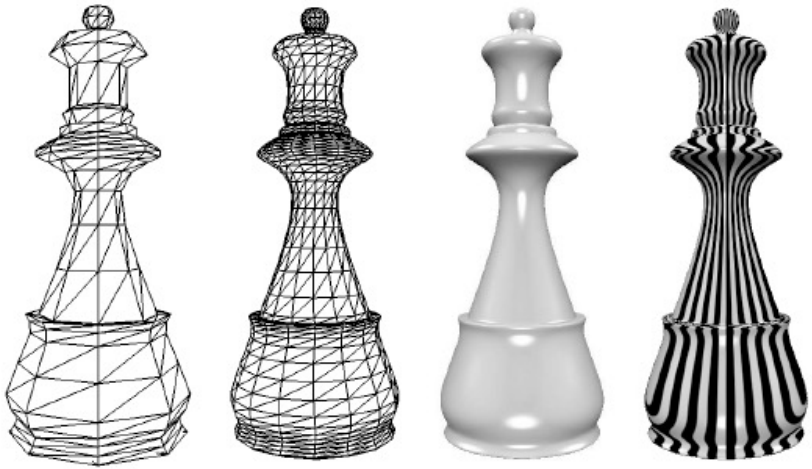
→ représentation multi-résolution naturelle d'une surface

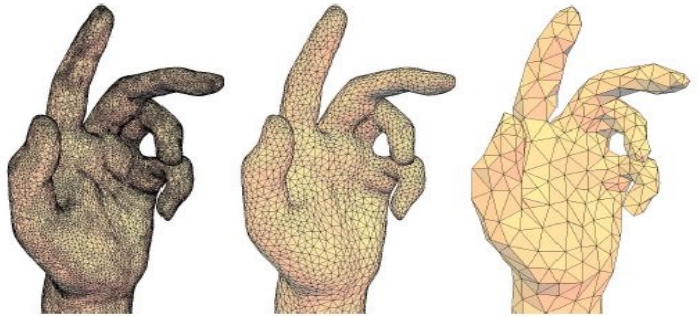
→ obtention d'une surface lisse



Modèles surfaciques – Surfaces de subdivision

Il existe de nombreux schémas de subdivision
→ les divisions de surface Catmull-Clark, Doo-Sabin, Loop, etc.

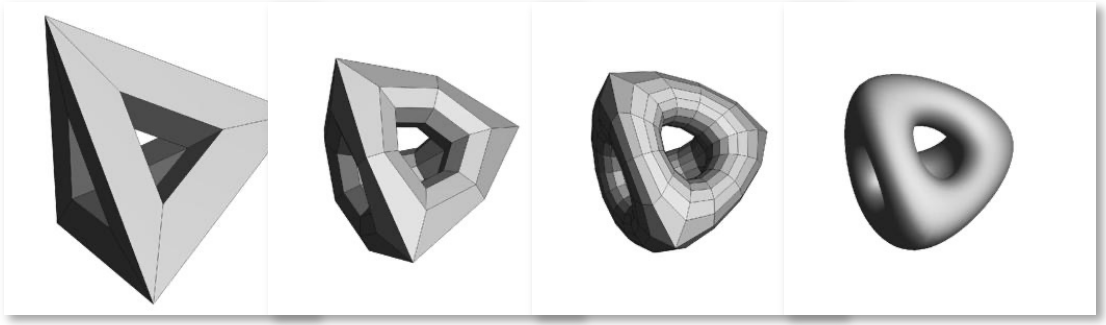


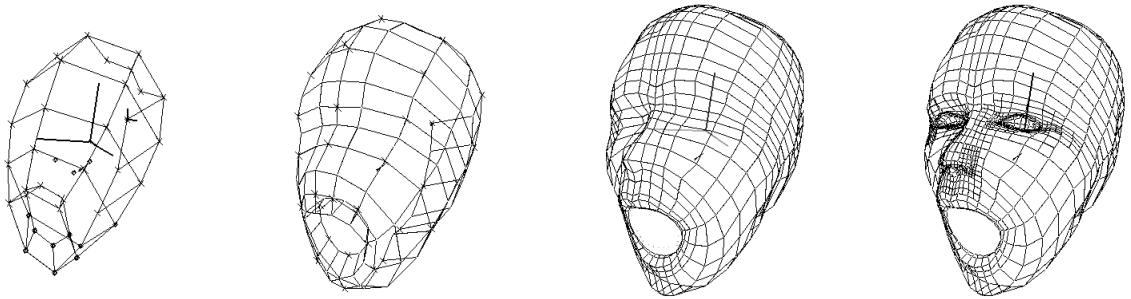


(a) 25,000 vertices. (b) 5,000 vertices. (c) 500 vertices.

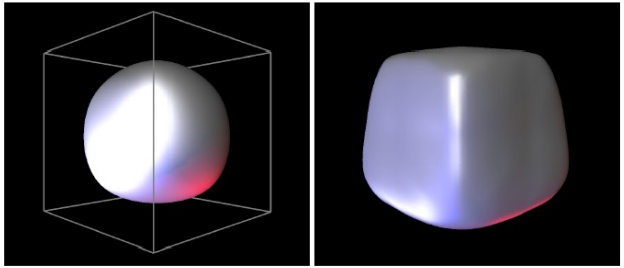
Modèles surfaciques – Surfaces de subdivision

Modèles surfaciques – Surfaces de subdivision



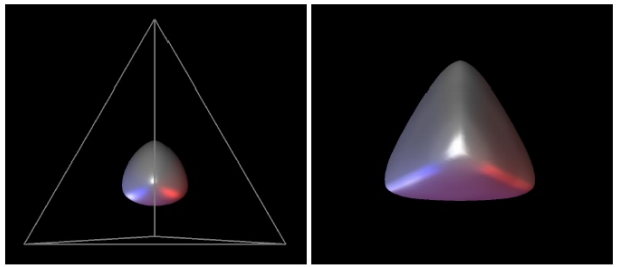


Modèles surfaciques – Surfaces de subdivision



Loop *Butterfly*

Catmull-Clark *Doo-Sabin*

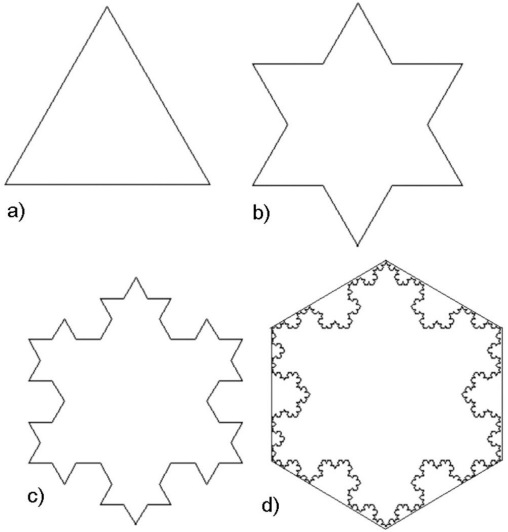


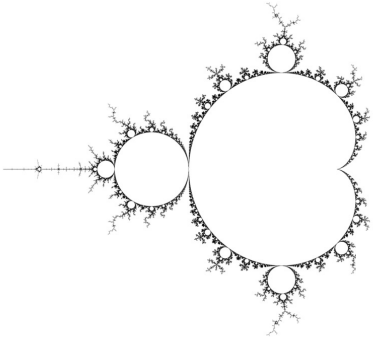
Loop *Butterfly*

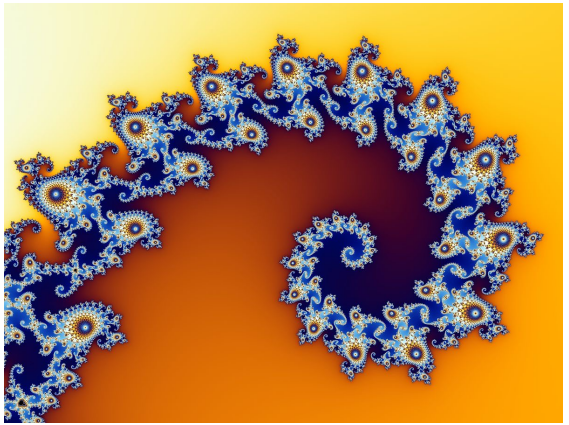
Catmull-Clark *Doo-Sabin*

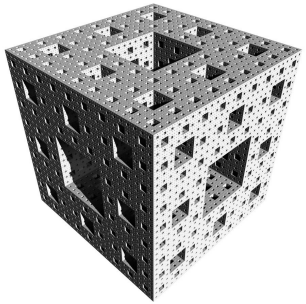
Modèles procéduraux - Fractales

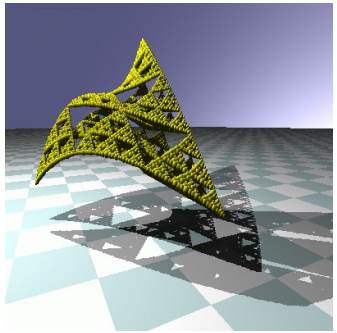
Utilisation de fractale pour élaborer des modèles



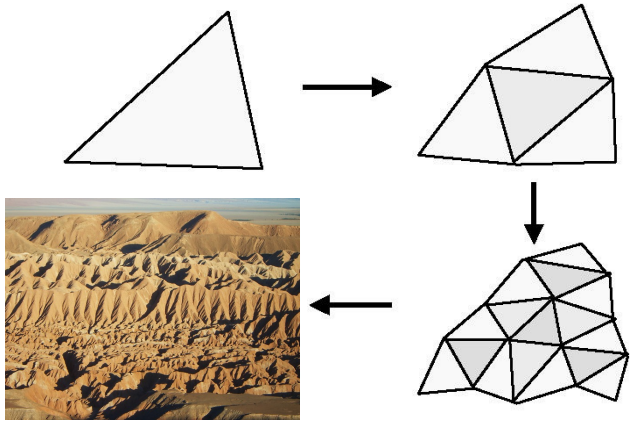




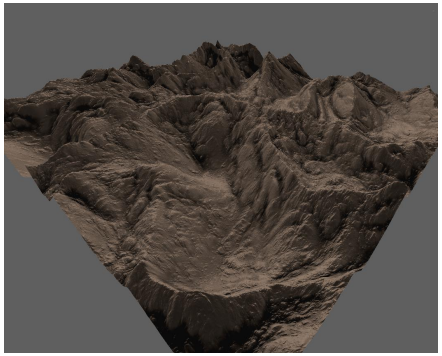


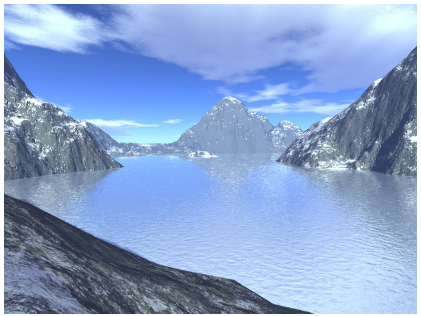


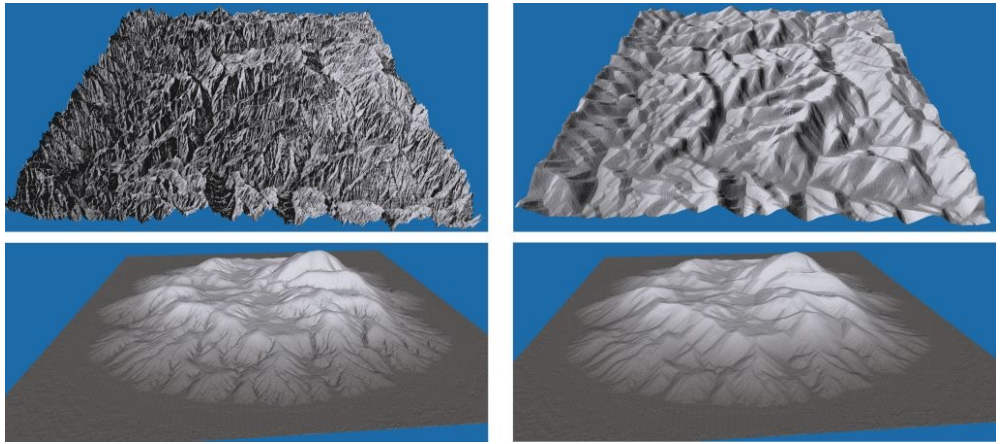
Modèles procéduraux – Fractales



Modèles souvent utilisés pour les terrains :
générés au cours de l'exécution

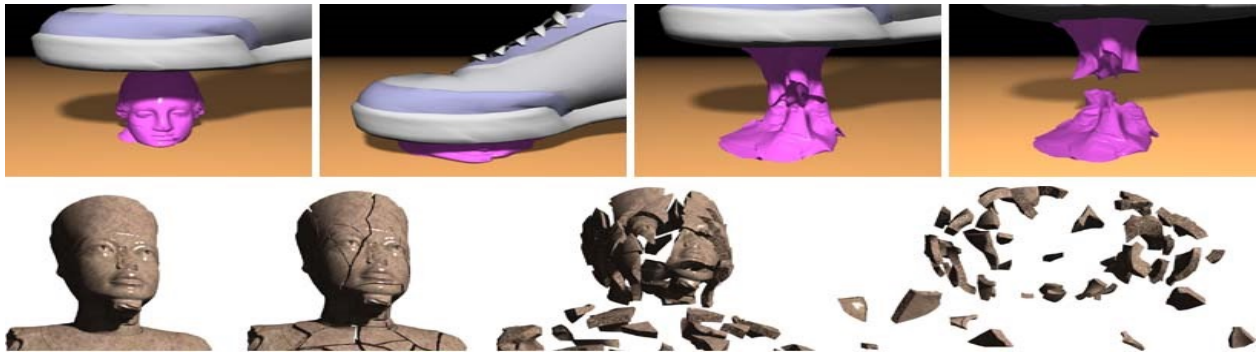






Modélisation géométrique

Modélisation procédurale



Modèles surfaciques – Surfaces implicites

Surfaces implicites donnent une représentation volumique de l'objet

La surface n'est pas directement connue,

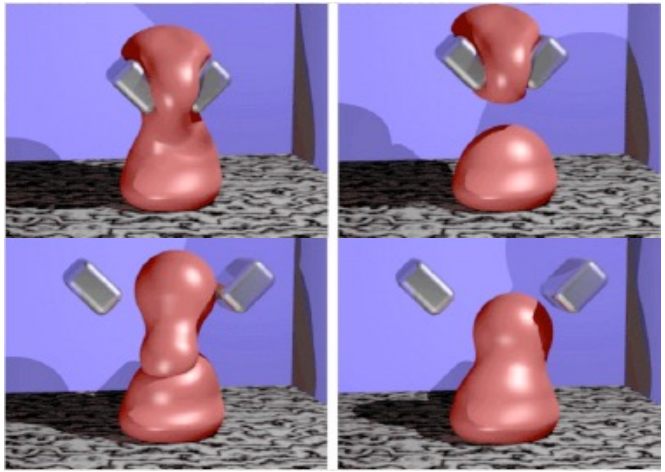
mais caractérisée comme l'ensemble des points de l'espace p solution d'une équation f(p) = 0

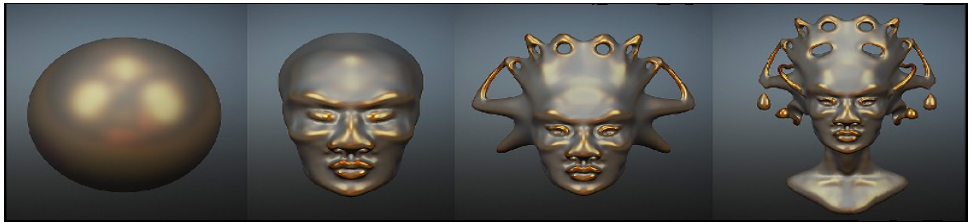
→ permet de représenter un équipotentiel

→ permet de déterminer si un point se trouve à l'intérieur ou l'extérieure du volume délimité par la surface (collisions, lancer de rayon)

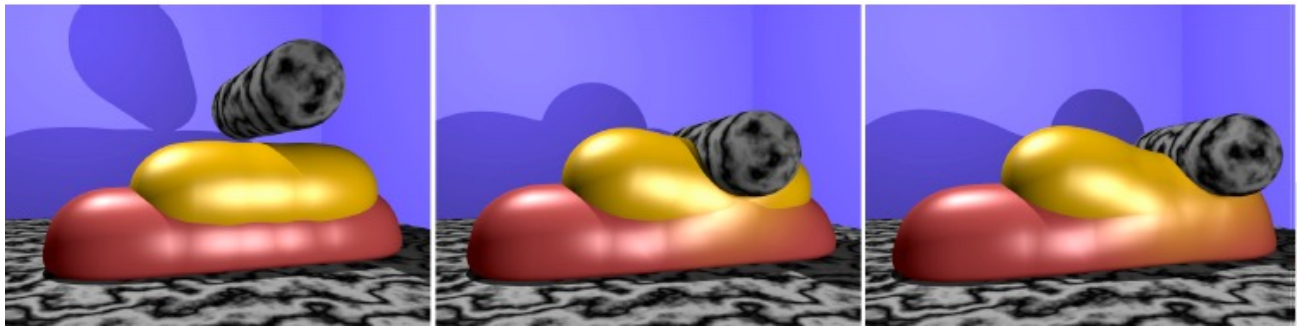
Surfaces implicites adaptées pour les objets déformables
avec changement de topologie ou de forme géométrique

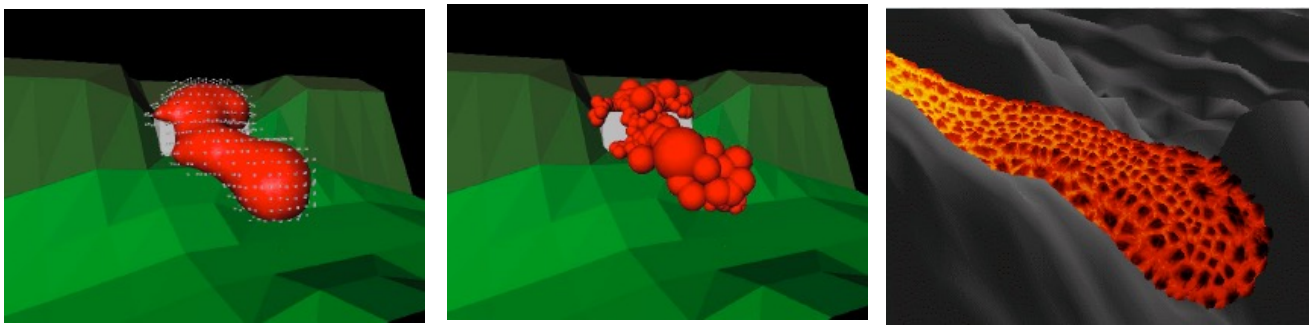
Modèles surfaciques – Surfaces implicites





Modèles surfaciques – Surfaces implicites

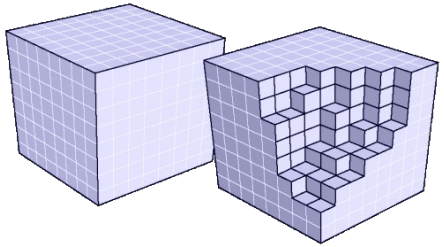




Modèles volumiques – Voxels

Discrétisation régulière de l'espace

- 2D → pixels
- 3D → voxels (élément de volume)

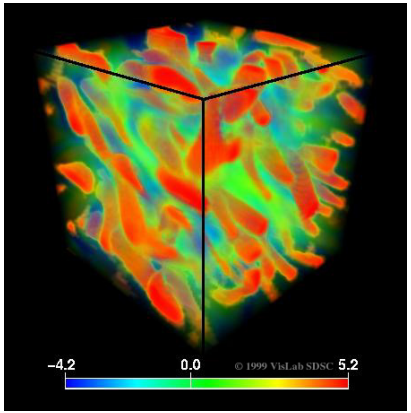


Une valeur est associée à chacun des voxels

mais pas de connexion entre les éléments de volumes

Modèles volumiques – Voxels

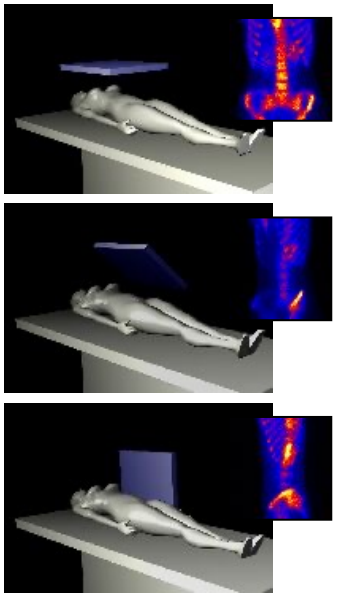
Valeurs obtenues par l'évaluation discrète d'une fonction issue d'une simulation numérique → valeurs de f(x, y, z)

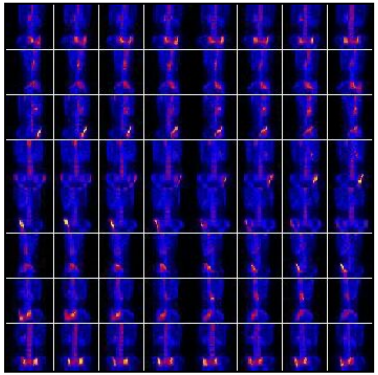


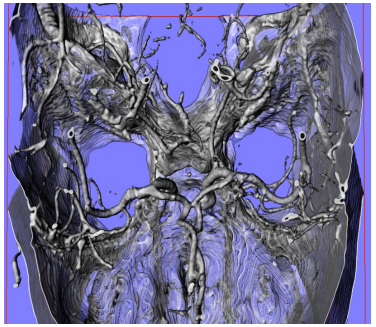
⇨ Rendu Volumique employé pour l'affichage des voxels

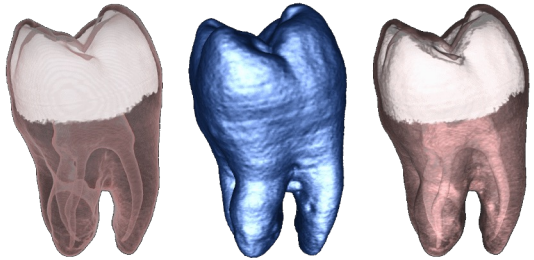
Modèles volumiques – Voxels

Valeurs obtenues à partir de coupes









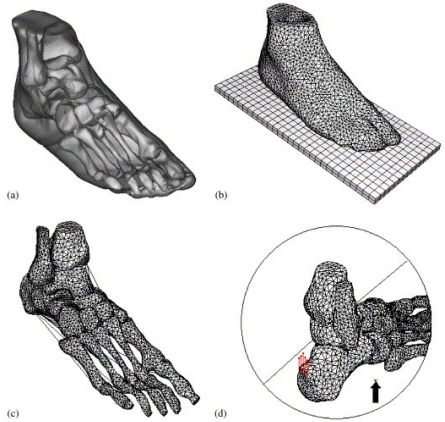
Modèles volumiques – Maillages volumiques

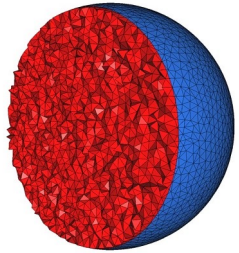
Ensemble connecté de polyèdres :
tétraèdres, hexaèdres, prismes, etc.

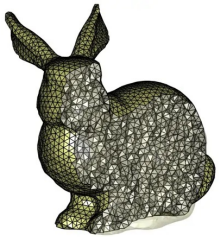
Permet de représenter à la fois :
la forme et la topologie de l'objet

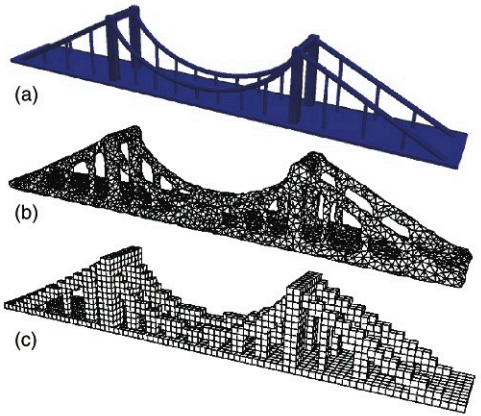
Affichage facile avec OpenGL

Obtention possible à partir d'un maillage surfacique









Bilan sur les modèles géométriques

Il existe de nombreux modèles qui sont adaptés à différents besoins

Dans le cadre de la réalisation **de simulations**, modèles usuellement employés :

- **Maillages surfaciques** et **maillages volumiques**

Le modèle géométrique permet de décrire les caractéristiques des objets présents dans la scène 3D :

- de base : forme, topologie des objets
- et plus encore : comportement physique, lumineux, propriétés du matériau, etc.

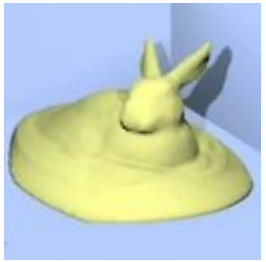
Présentation en détails des étapes de création d'une scène 3D

1- Modélisation : représentation mathématique des objets virtuels

2- Simulation / animation : déformation et mouvement des objets virtuels

3- Visualisation / rendu : affichage des objets virtuels




Simulation / Animation d'objets 3D

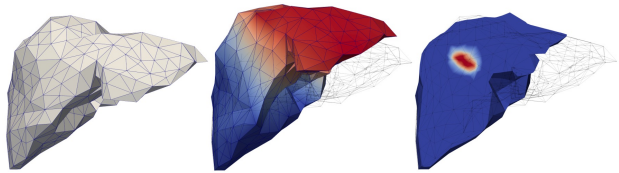
Il existe différentes techniques d'animation en Informatique Graphique :

- Interpolation à partir de positions clés
  - cinématique directe et inverse

On va voir quelques exemples

- Capture de mouvements
  - grande qualité mais spécifique

- Utilisation de modèles physiques
  - complexe mais permet <u>simulation automatique et réaliste</u>



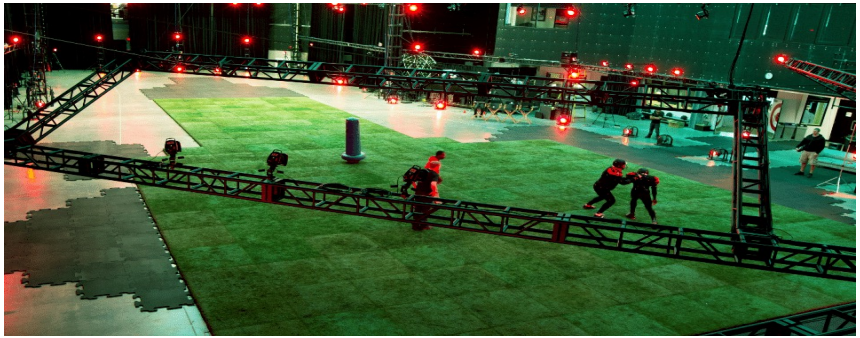
Réalisme nécessaire dans le cadre d'applications médicales :

Besoin de simuler au mieux le comportement des organes

Organes / tissus mous modélisés en employant des modèles physiques

Animation de personnage

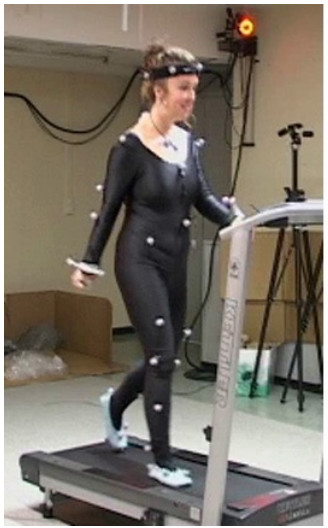
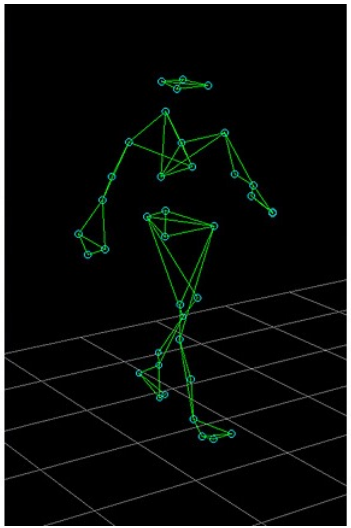
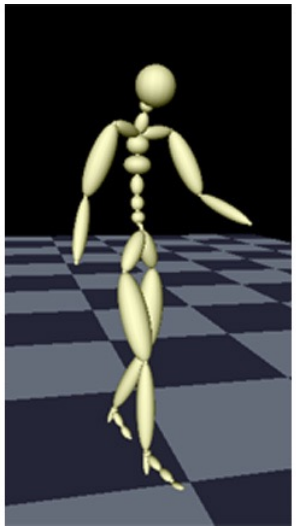
- *Motion Capture* (capture de mouvement) pour produire une animation



Animation de personnage

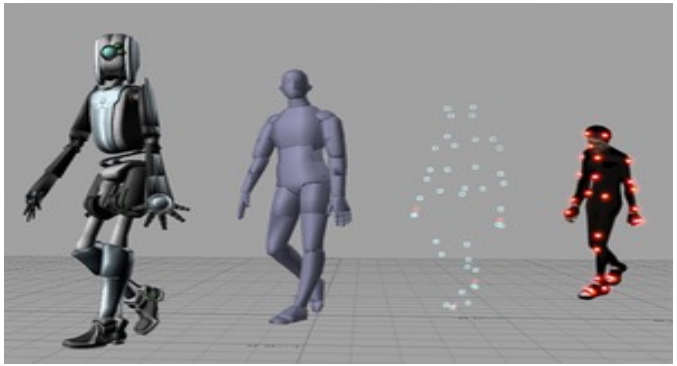
Les étapes de la capture de mouvement :

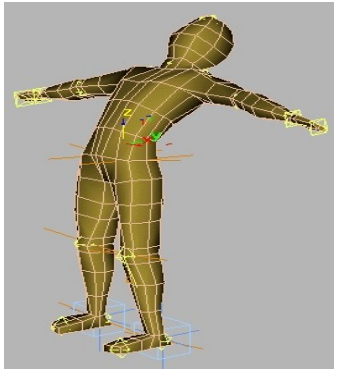
- Segmentation des marqueurs sur les images de chaque camera
- Reconstruction 3D de chaque marqueur
- Correspondance avec un squelette

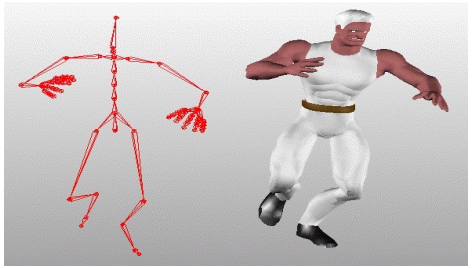




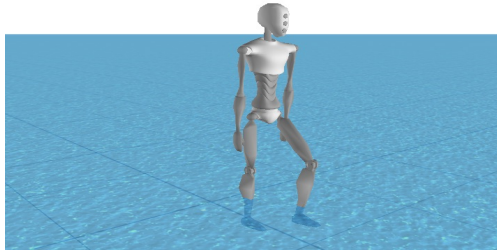

Animation de personnage

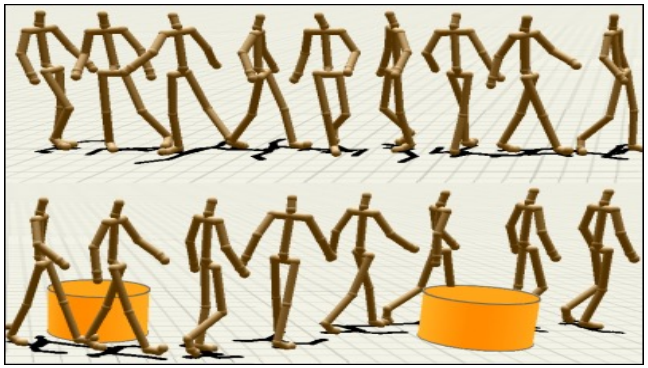
- *Skinning*
- Contrôle de mouvement



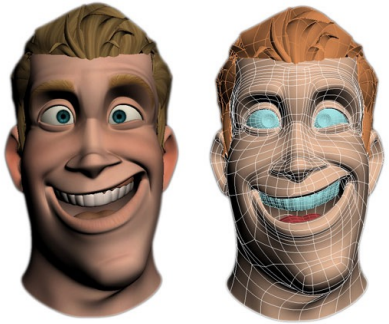


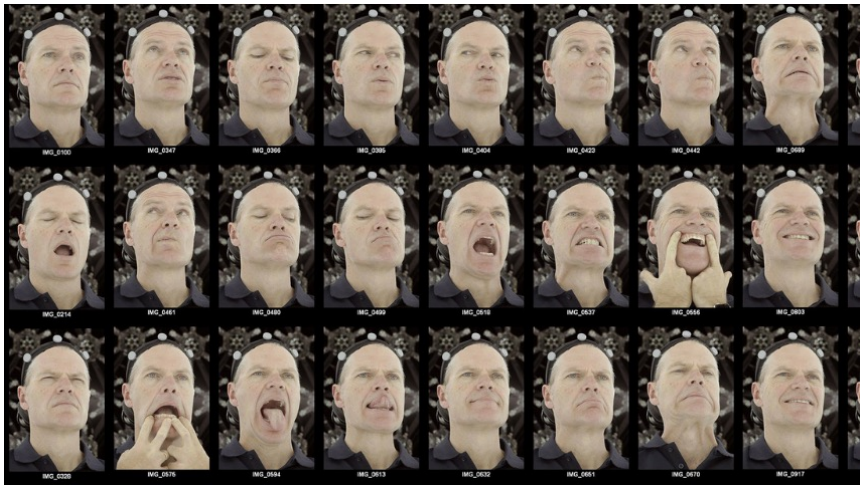




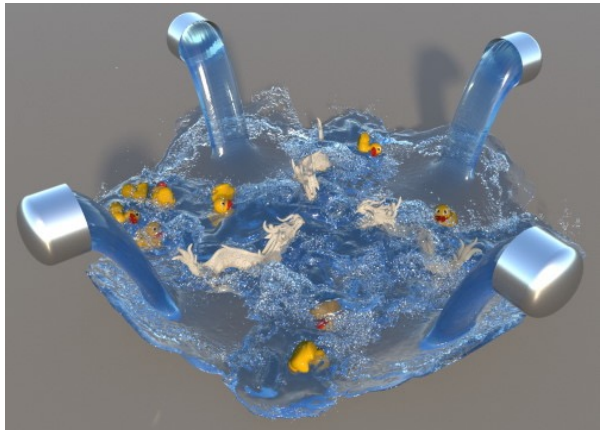


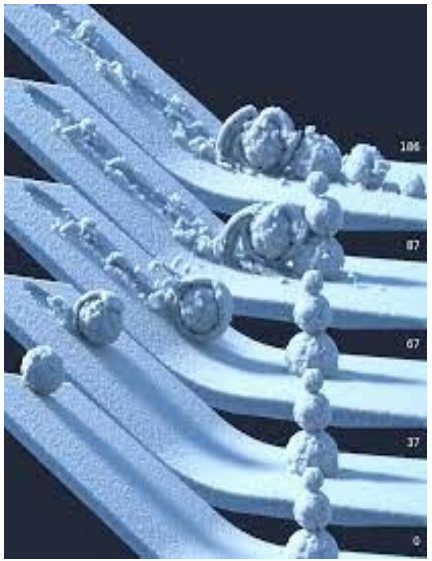
Animation de visage



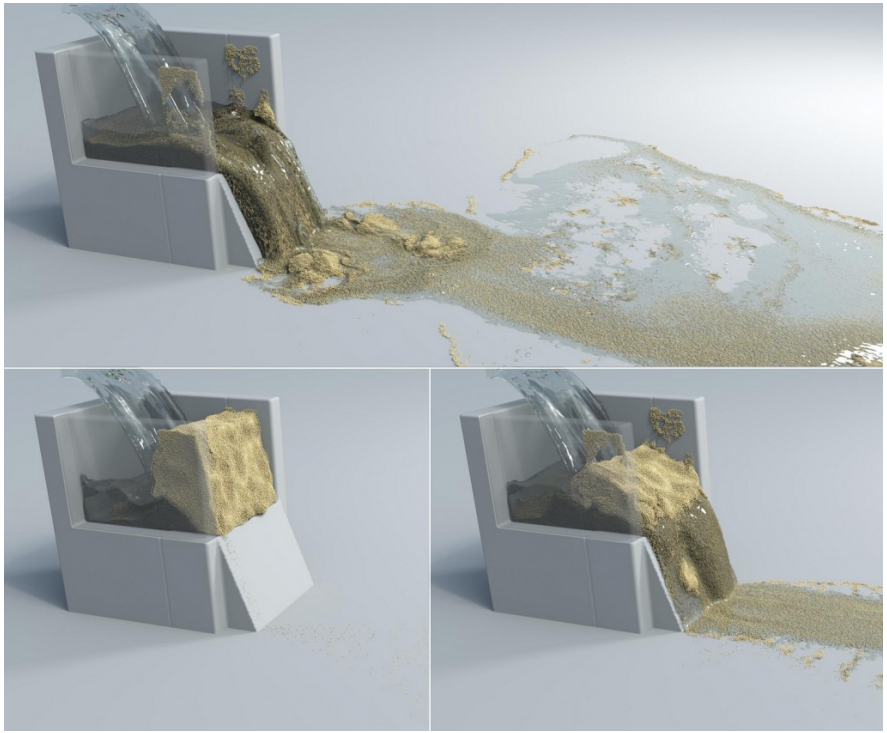


Simulation de fluide / neige / feu : particules en mouvement

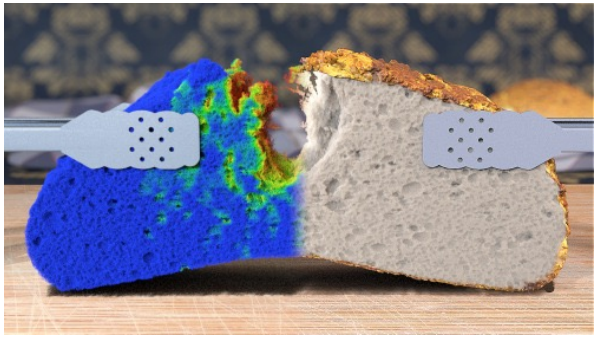




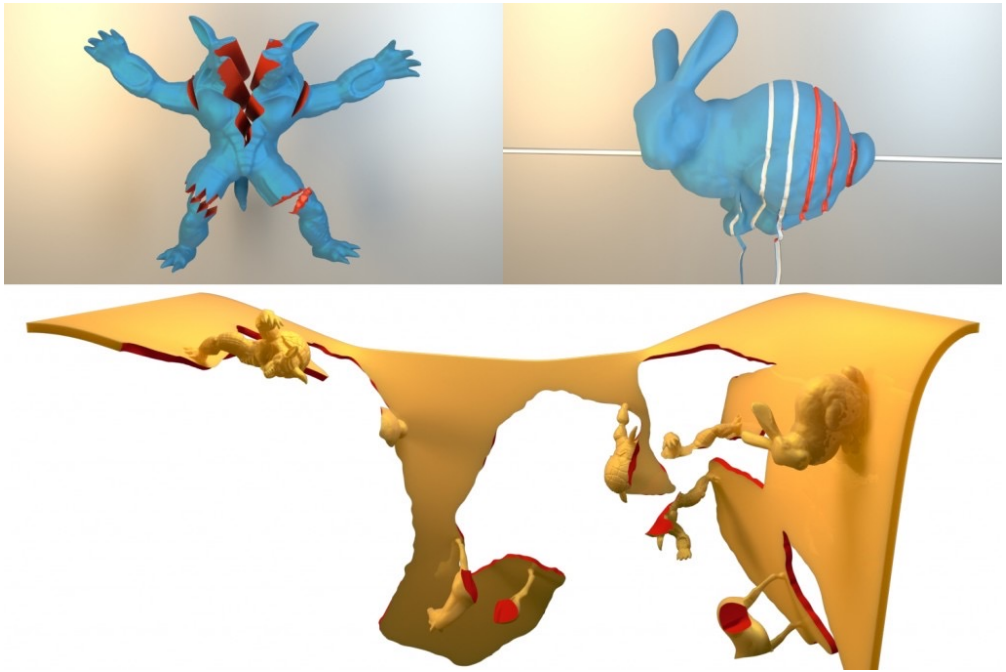
Simulation de sable : particules en mouvement



Simulation d'objets surfaciques / d'objets 3D déformables

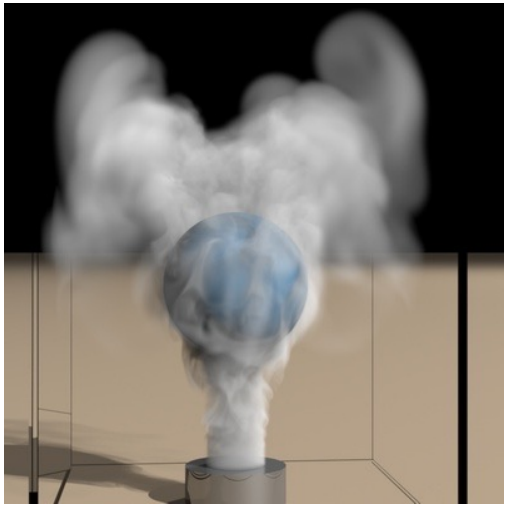


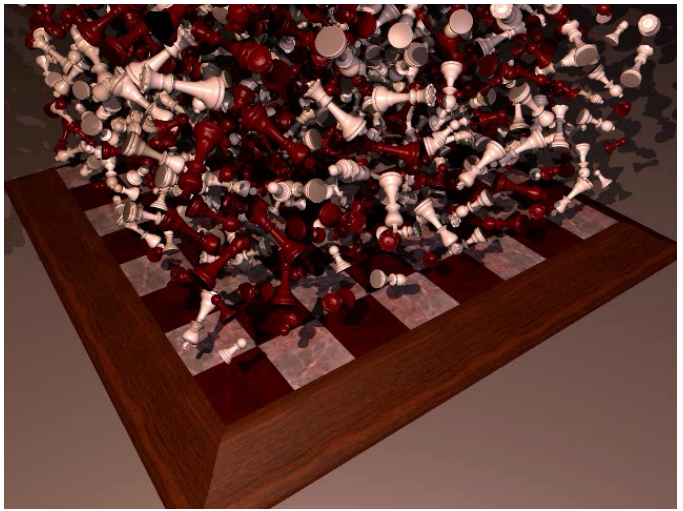
Simulation de la découpe d'objets 3D déformables



Medical VR Surgical Simulator

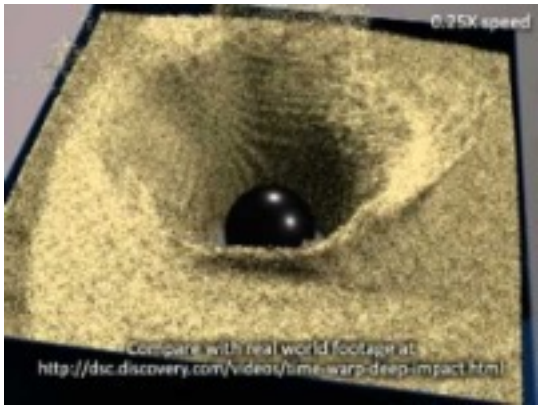
Animation par modèles physiques



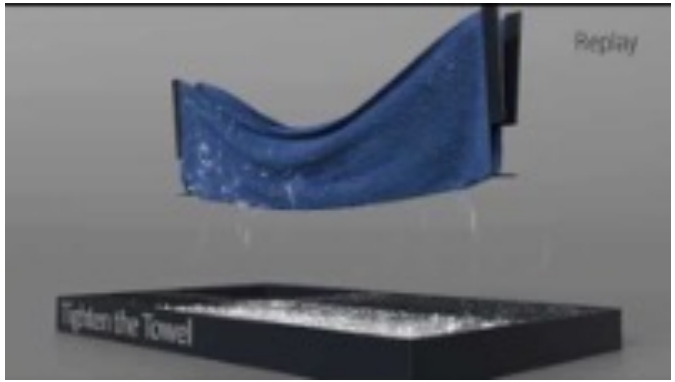


Quelques vidéos

Animation par modèles physiques

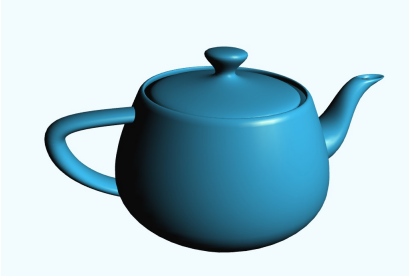


Quelques vidéos



La modélisation physique est adaptée aux types d'objets à animer

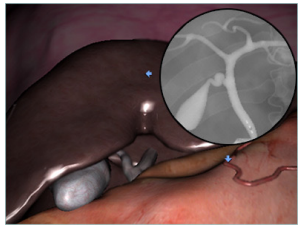
Objets rigides

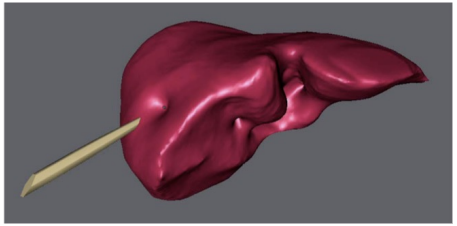


Tower destruction

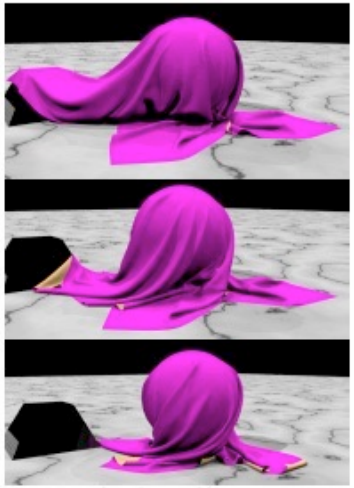
Objets déformables

Dans le cadre d'applications médicales :
simulation d'objets déformables
en interaction
avec objets rigides





Exemples de simulation d'objets déformables en Informatique Graphique



https://youtu.be/p5uhnSw8_Xw
https://youtu.be/xvyGpBKevLM

Animation par modèles physiques – Quelques vidéos

Simulation de fluide :

https://www.youtube.com/watch?v=8jD1bz4N3_0
https://youtu.be/uRoH3HcftX8
https://youtu.be/chnS24QfgNY
https://youtu.be/Qve54Z71VYU
https://youtu.be/3yyLIG935zc
https://youtu.be/Jl54WZtm0QE

Simulation de sable :

https://youtu.be/8iyvhGF9f7o
https://youtu.be/ZoZ0ZAzr6eg

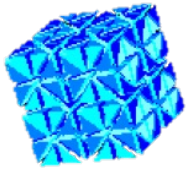
Simulation de fracture :

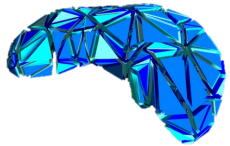
https://youtu.be/lNri-x2nK7o

Simulation de neige :

https://youtu.be/7Lbk9xJJRNU
https://youtu.be/1ES2Cmbvw5o
https://youtu.be/YQ7e06-MZec
https://youtu.be/V9tEXiQavNo

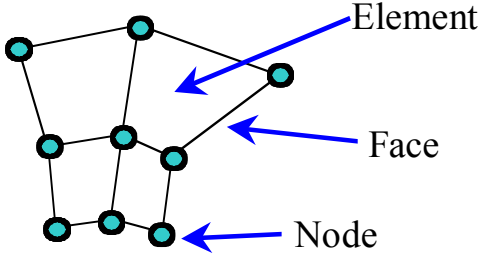
Animation par modèles physiques – Principe général

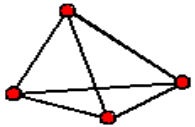




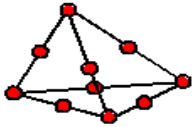
Modélisation de l'objet :

Représentation de l'objet en un ensemble de sommets
positionnés dans l'espace 3D
et connectés ensemble pour former un maillage

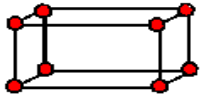




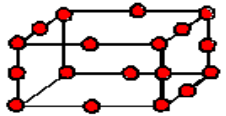
4 noded tetrahedron



10 noded tetrahedron



8 noded brick



20 noded brick

Objectif de la simulation :

donner du mouvement aux objets modélisés
en modifiant la position des sommets dans l'espace 3D au cours du temps

→ Calcul de la position **x(t)** des sommets des objets de la scène 3D **à chaque instant t**

Animation par modèles physiques

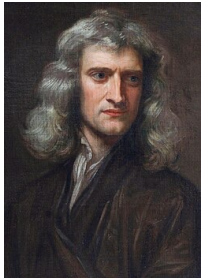
Il existe un grand nombre de modèles physiques :

- Méthode de résolution des Eléments Finis : basée sur la Mécanique des Milieux Continus
  - Simulation précise mais couteuse en temps de calcul
- Modèle discret des masses-ressorts : issue de l'informatique graphique
  - Simulation interactive mais approximation de la réalité
- Approche *Position Based Dynamics* : issue de l'informatique graphique
  - Basée sur la résolution de contraintes
- *Et bien d'autres...*

Compromis précision / réalisme et temps de calculs à faire

Animation par modèles physiques – Principe de la dynamique Newtonienne

Boucle d'animation est gouvernée par la **dynamique Newtonienne,**
c'est-à-dire qu'elle est basée sur les **lois de Newton**



Objet caractérisé par sa **position**, **vitesse** et **accélération**

Vitesse = dérivée par rapport au temps de la position : v = x'
Accélération = dérivée par rapport au temps de la vitesse : a = v' = x''

Position = intégration de la vitesse par rapport au temps
Vitesse = intégration de l'accélération par rapport au temps

Animation par modèles physiques – Principe de la dynamique Newtonienne

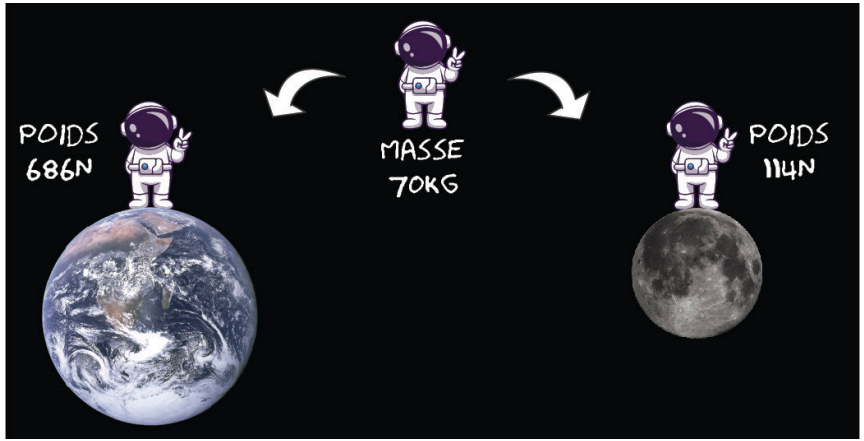
Objet également caractérisé par sa **masse** (en Kg) et par sa **force** (en N)

Accélération de l'objet proportionnelle à l'intensité de la force

Force d'un Newton = intensité de la force requise pour donner une accélération d'un mètre par seconde au carré à une masse d'un kilogramme

$$P = m \times g$$

en Newton (N) — en kg — en N / kg



Animation par modèles physiques – Principe de la dynamique Newtonienne

Première loi de Newton

En l'absence de toute force, un objet au repos reste au repos

Si l'objet est en mouvement,
et qu'aucune force extérieure ne lui est appliquée,
sa vitesse reste constante

⇨ Mouvement d'un objet ne peut être modifié que par l'intervention d'une force

Animation par modèles physiques – Principe de la dynamique Newtonienne

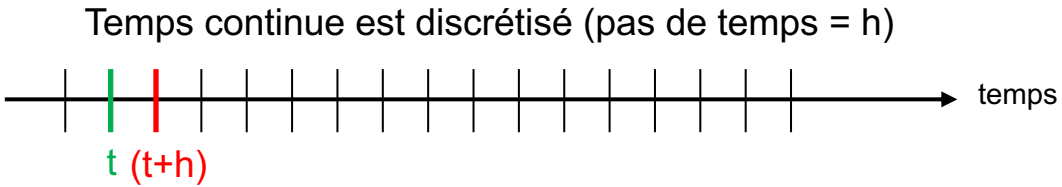
Seconde loi de Newton : **principe fondamentale de la dynamique**

Soit un objet de masse constante m
accélération au temps t $\vec{a}(t)$
et force au temps t $\vec{F}(t)$

Equation du mouvement :

$$\vec{F}(t) = m\vec{a}(t) \Rightarrow \vec{a}(t) = \vec{F}(t)/m$$

Animation par modèles physiques – Principe de la dynamique Newtonienne

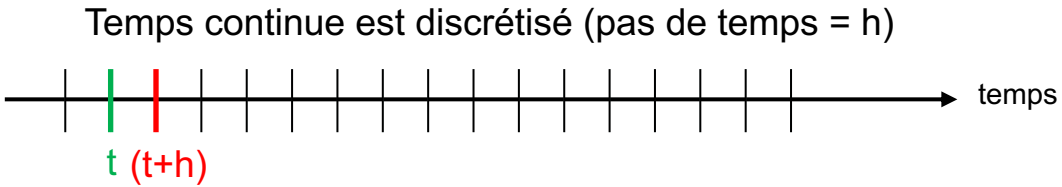


Etapes de calculs à effectuer à chaque pas de temps de la simulation = **boucle de simulation**

1. Calcul des **forces** exercées sur l'objets au temps t
   - forces internes dues au modèle physique employé
   - forces extérieures (gravité, etc.)
2. Calcul des **accélérations** (PFD) au temps t : somme des forces = masse x accélération
3. Intégration des accélérations pour obtenir les **vitesses** au temps (t+h)
4. Intégration des vitesses pour obtenir les **positions** au temps (t+h)
5. *Affichage de l'objet* au temps (t+h)

Et on recommence pour le temps suivant (t+2h)...

Animation par modèles physiques – Principe de la dynamique Newtonienne



Etapes de calculs à effectuer à chaque pas de temps de la simulation = **boucle de simulation**

1. Calcul des **forces** exercées sur l'objets

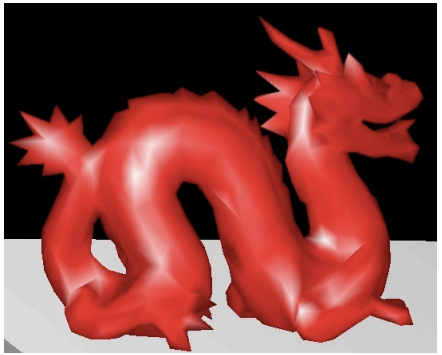
L'enjeu de la simulation réside dans le calcul des forces internes à l'objet

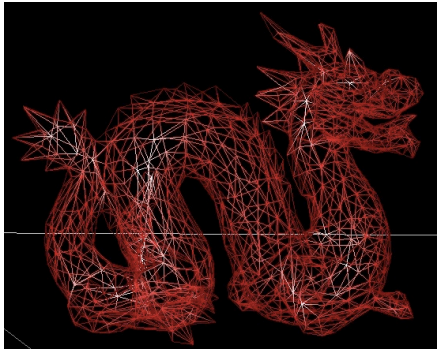
La formulation des forces internes dépend de la représentation choisie pour l'objet

Elle est définie par le modèle physique de l'objet

Animation par modèles physiques – Modèle physique

Systèmes de particules





Matériel est discrétisé en un ensemble de particules

Masse associée à chacune des particules

Vecteurs position, vitesse et accélération associée à chacune des particules

Equation va permettre de calcul les forces appliquées sur chacune des particules

Animation par modèles physiques – Modèle physique

Système masses-ressorts

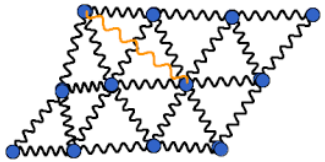
Matériel discrétisé en un ensemble de particules/masses **connectées entre elles par des ressorts**

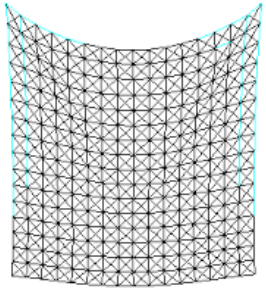
Masse associée à chacune des particules

Vecteurs position, vitesse et accélération associée à chacune des particules

Les objets modélisés peuvent être :

- des courbes (cheveux, cordes, etc.) → 1D
- des surfaces (textiles, surface de l'eau, etc.) → 2D
- des volumes (objet volumique visqueux) → 3D





Formulation d'un système masses-ressorts

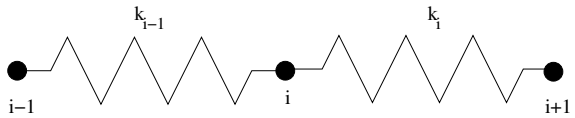
Système contient :

- p particules de **masses m_i** et de **position x_i** pour $1 \leq i \leq p$
- (p-1) **ressorts**

Particule i est connectée par deux ressorts aux points i-1 et i+1

$Ressort_{ij}$

- relie les masses m_i et m_j
- avec une raideur $k_{ij} > 0$
- et une longueur au repos l_{ij}



Forces internes appliquées sur les masses = forces exercées par les ressorts

Equation du mouvement pour un système masses-ressorts

Soit $\mathcal{A}_i$ = ensemble des indices j tels que m_i connectée à m_j

Soit F_i les forces externes appliquées à la masse i

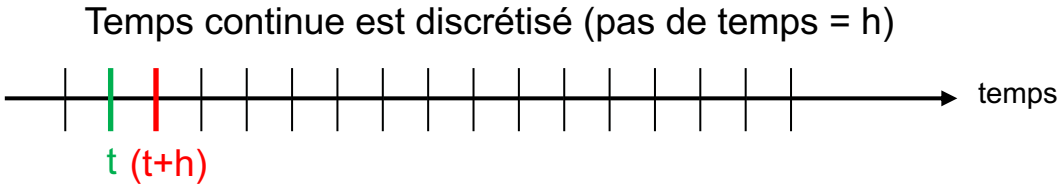
Equation du mouvement de la particule i (PFD) :

$$m_i\ddot{x}_i = \text{somme des forces exercées sur i}$$

$$\Longrightarrow m_i\ddot{x}_i = \sum_{j \in Ai} k_{ij}(\| x_i - x_j \| - l_{ij}) + F_i$$

Calcul des forces permet l'obtention des **accélérations** des points au temps t

Animation par modèles physiques – Principe de la dynamique Newtonienne



Etapes de calculs à effectuer à chaque pas de temps de la simulation = **boucle de simulation**

3. **Intégration** des accélérations pour obtenir les nouvelles **vitesses** au temps (t+h)
4. **Intégration** des vitesses pour obtenir les nouvelles **positions** au temps (t+h)

Emploi d'une **méthode d'intégration numérique**
pour obtenir approximation de cette intégrale
car il n'y a pas de solution exacte

Méthodes d'intégration numériques

Pour calculer les nouvelles vitesse et position à partir de l'accélération

Il existe de nombreuses méthodes d'intégration

Compromis entre le temps de calcul et la stabilité numérique / précision

Deux grandes classes :

- méthodes explicites : Euler explicite, Leapfrog, etc.
- méthodes implicites : Euler implicite, etc.

Méthodes d'intégration numériques

Exemple : Utilisation de la méthode d'intégration d'Euler semi-implicite

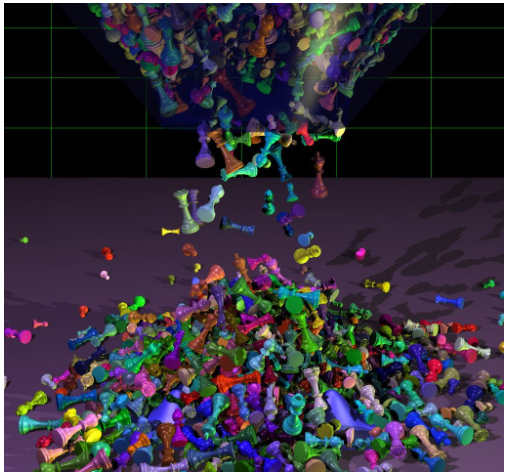
Facile à mettre en œuvre
Discrétisation du temps avec un **pas de temps h qui doit être petit**

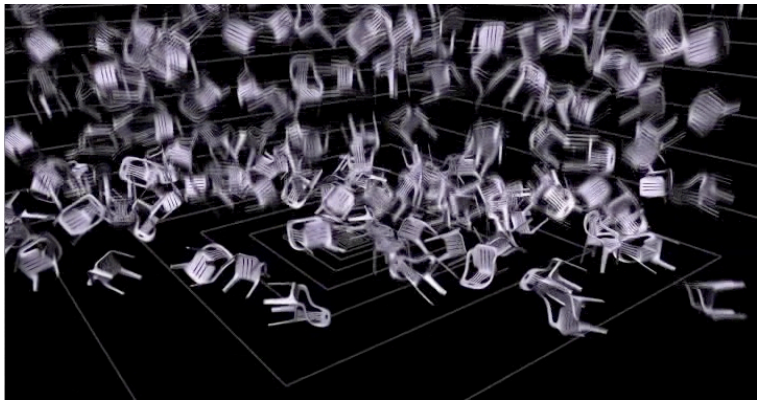
Boucle de simulation :

Calcul des forces	$F(t)$	$= \ldots$
Calcul de l'accélération	$a(t)$	$= M^{-1} F(t)$
Calcul de la vitesse au temps t+h	$v(t+h)$	$= v(t) + h\, a(t)$
Calcul de la position au temps t+h	$x(t+h)$	$= x(t) + h\, v(t+h)$

Traitement des collisions

Gestion de plusieurs objets dans la scène



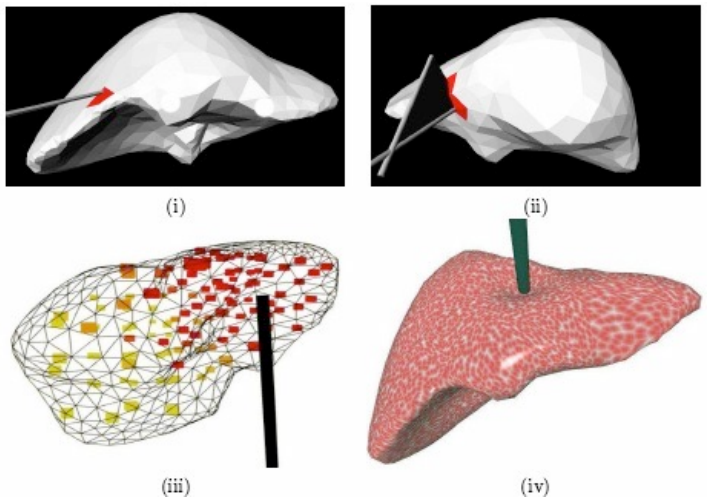


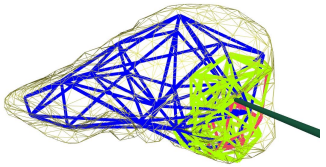
Souvent plusieurs objets en interaction dans la scène
→ collisions à traiter entre les objets en interaction

Animation par modèles physiques Traitement des collisions

Exemple application pour le médical

Collisions à traiter entre différents objets : entre organes ou entre organe et objet chirurgical





Deux problèmes à résoudre :

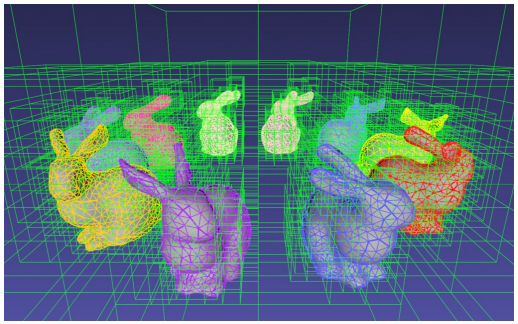
- Détection des collisions dans la scène
- Traitement de ces collisions : réponse à donner

Détection des collisions

Plusieurs méthodes :

- Regarde s'il existe un plan partitionnant l'espace en 2 demi espaces
  - l'un contient l'objet A et l'autre l'objet B : pas de collision

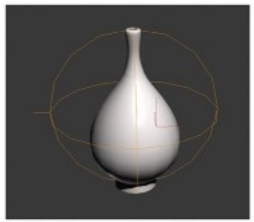
- Calcul de la distance entre les objets
  - si elle est inférieure ou égale à 0, il y a collision



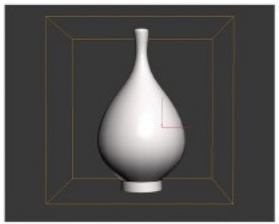
Utilisation de structure de données permettant d'optimiser la détection

→ Objets sont encapsulés dans des boîtes

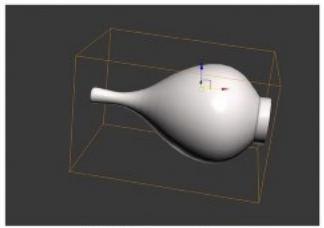
→ Détection des collisions entre ces volumes englobants



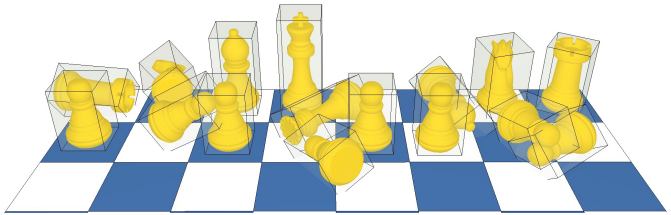
a Sphere bounding box



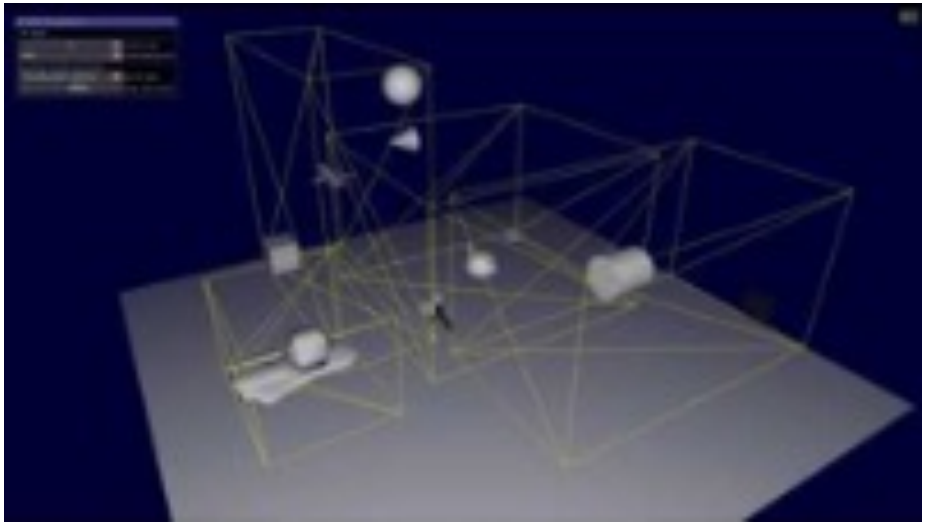
b AABB bounding box

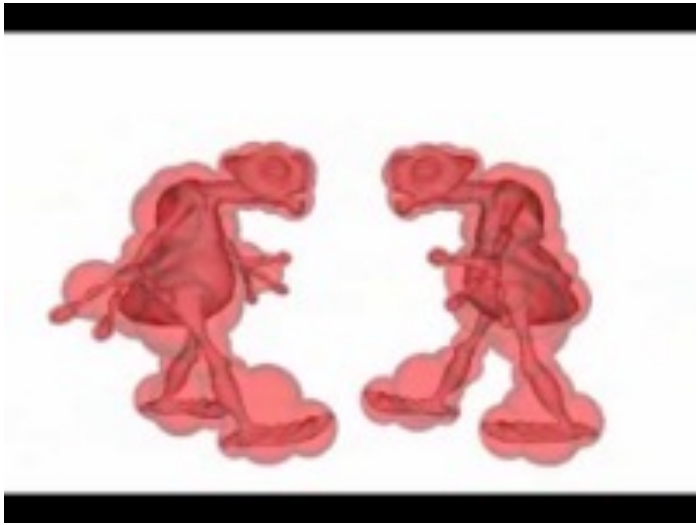


c OBB bounding box



Détection des collisions

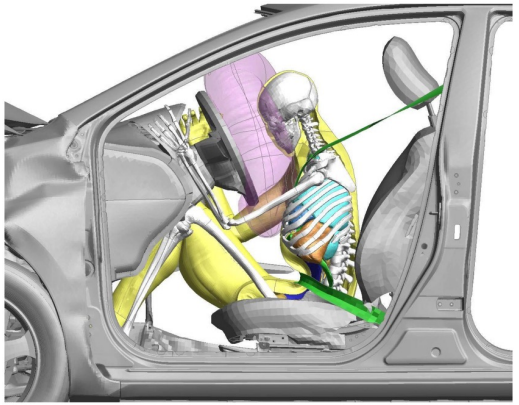




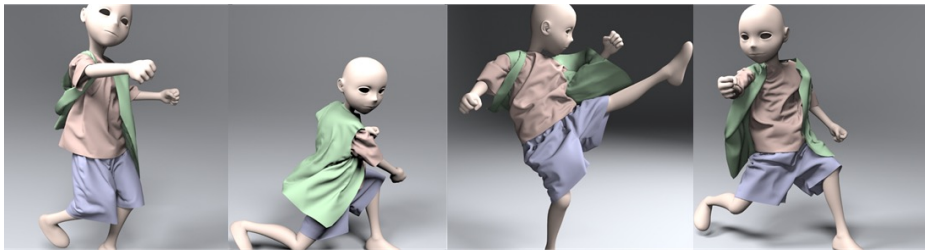
Réponses à la collision

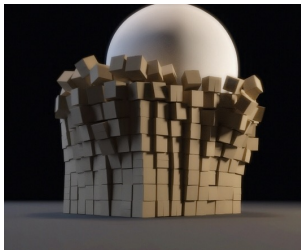
Différents types de collisions possibles :

- Contact en collision :
  - objets qui **rebondissent** l'un sur l'autre (force d'impulsion)
- Contact établi :
  - **glissement** ou **frottement** entre les objets (force de contact)



⇨ **Modification de la vitesse** des objets pour répondre aux collisions
Puis intégration de cette nouvelle vitesse dans la boucle d'animatior





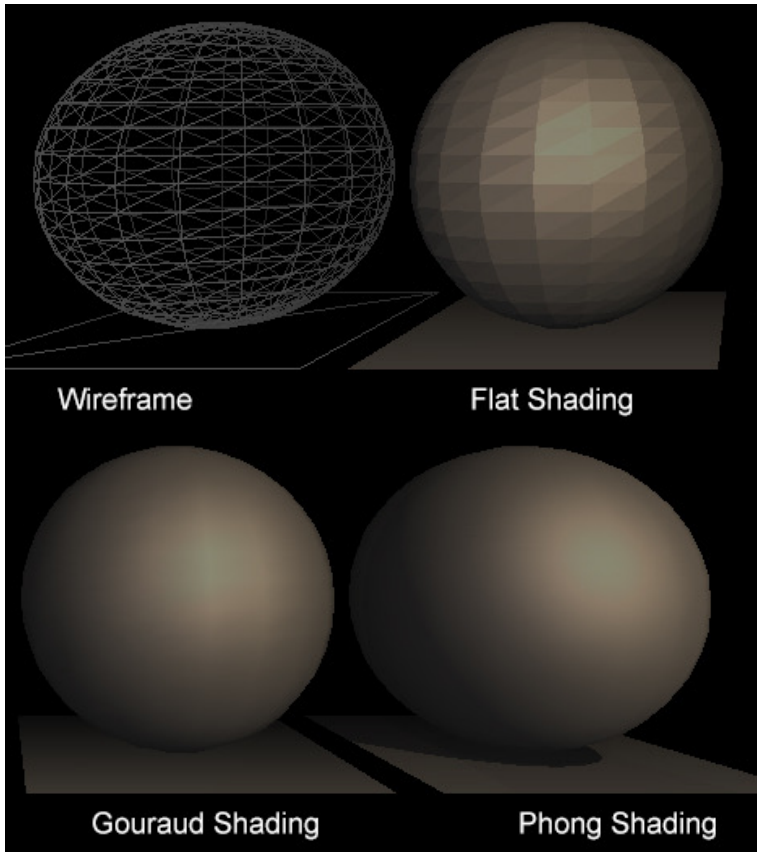
Présentation en détails des étapes de création d'une scène 3D

1- Modélisation géométrique : représentation mathématique des objets virtuels

2- Simulation / animation : déformation et mouvement des objets virtuels

3- Visualisation / rendu : affichage des objets virtuels

Il existe différentes techniques pour effectuer le rendu de la scène 3D



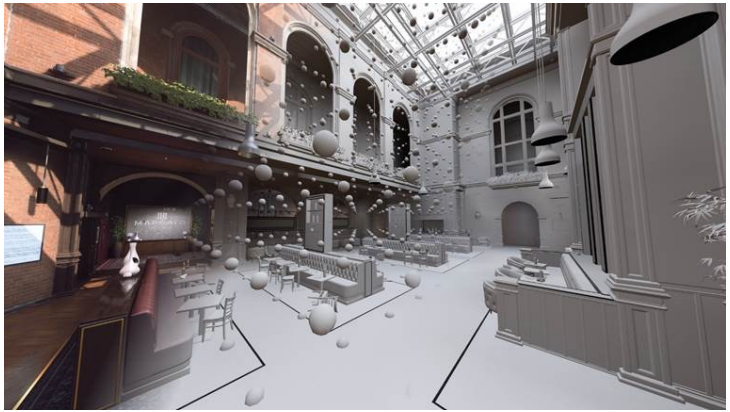
Affichage de base du modèle géométrique de l'objet

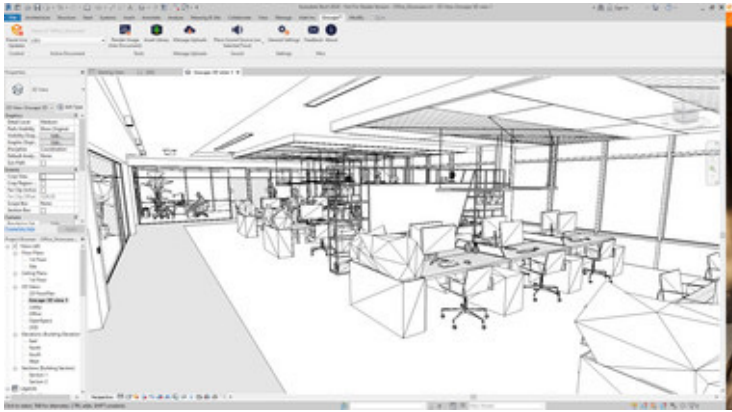
Textures

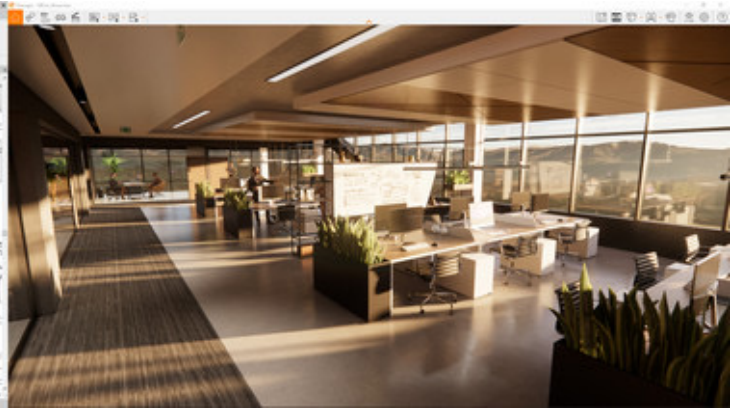
Technique choisie dépend du besoin...

Rendu réaliste

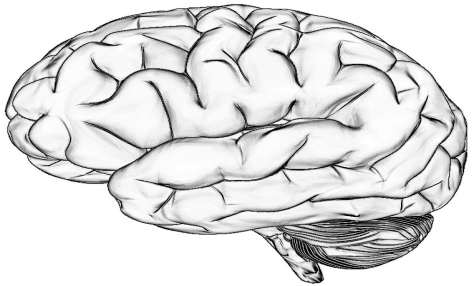
Rendu temps réel

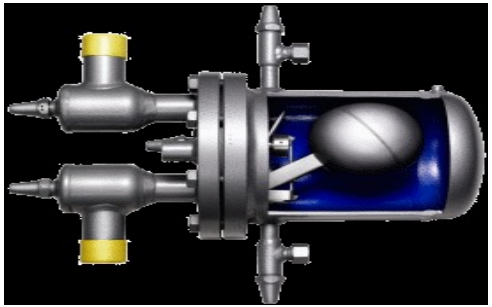




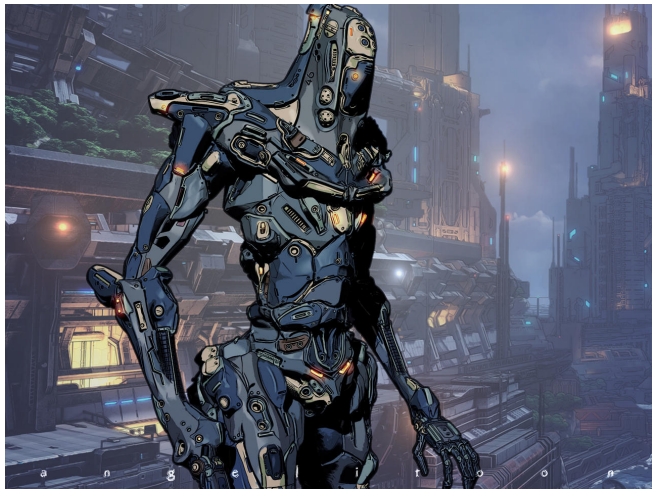


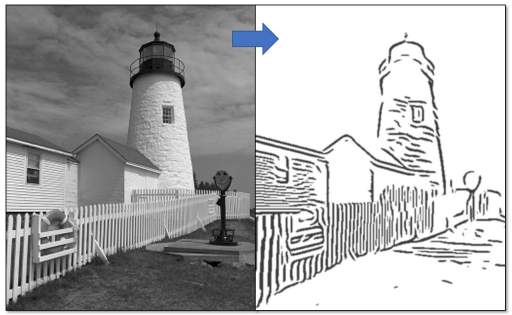
Rendu non photo réaliste (NPR)



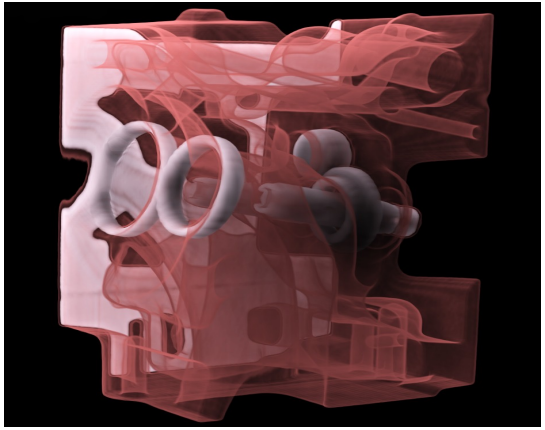


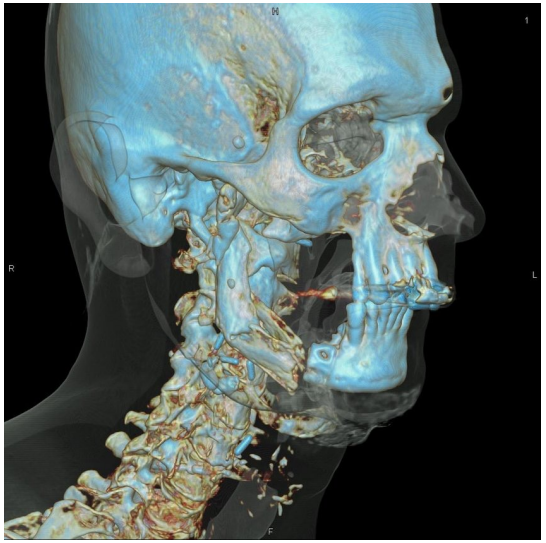
Rendu non photo réaliste (NPR)

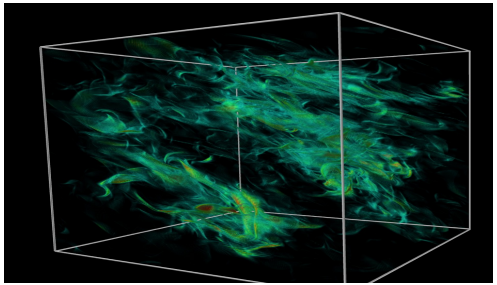


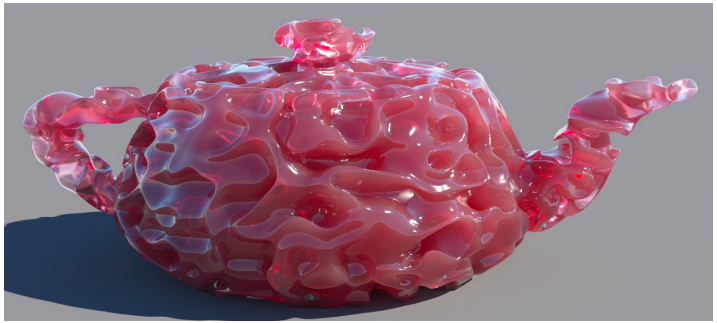


Rendu volumique

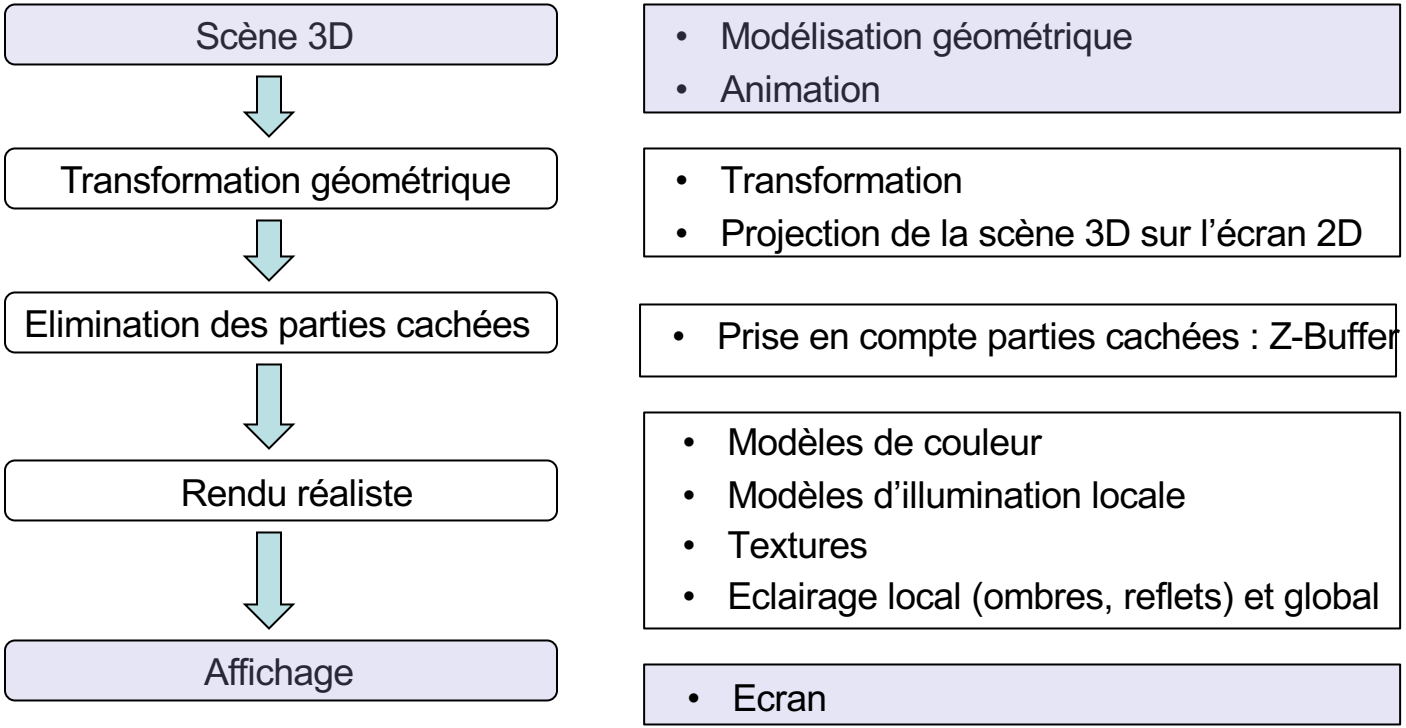




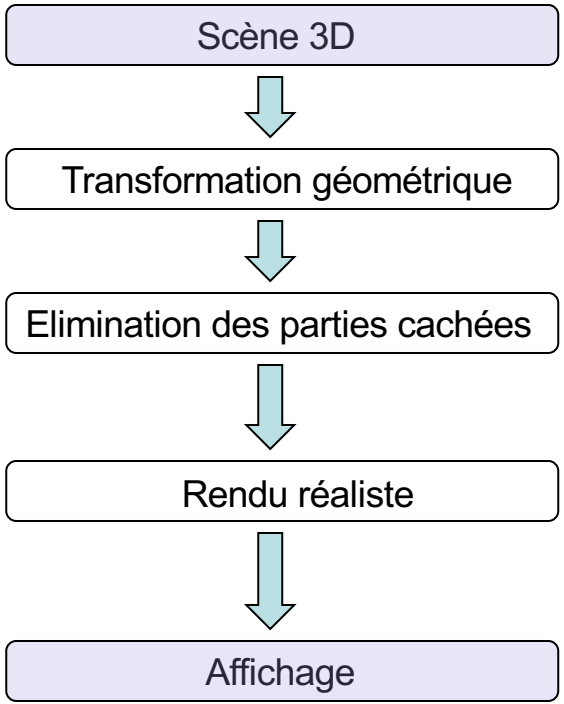




Pipeline graphique pour effectuer le rendu / affichage de la scène



Pipeline graphique pour effectuer le rendu / affichage de la scène

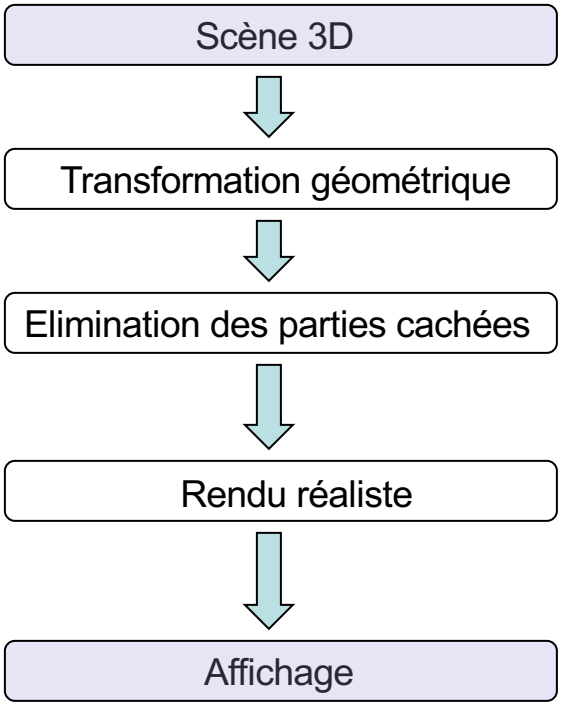


- Modélisation géométrique
- Animation

- Utilisation de la librairie **OpenGL** :
  - Rendu (éclairage, etc.)
  - Affichage à l'écran (projection, éliminations des parties cachées, ...)

- Rendu volumique

Pipeline graphique pour effectuer le rendu / affichage de la scène



- Modélisation géométrique
- Animation

- Utilisation de la librairie OpenGL :
  - Rendu (éclairage, etc.)
  - Affichage à l'écran (projection, éliminations des parties cachées, ...)

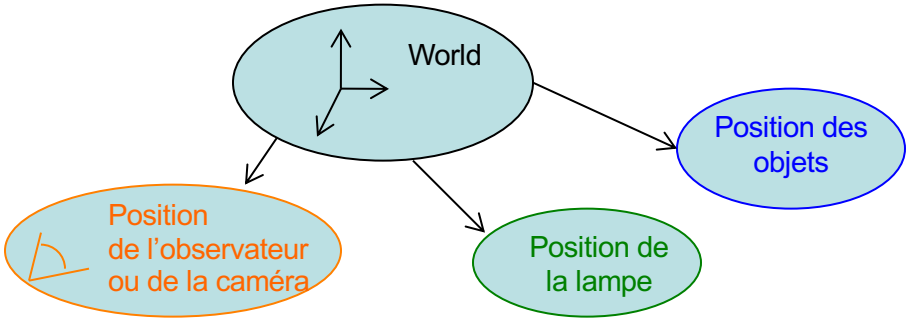
- Rendu volumique

Passage de la scène 3D à l'image 2D = calcul de la couleur de chacun des pixels

Pipeline graphique pour effectuer le rendu / affichage de la scène

La scène 3D est constituée :

- d'objets positionnés / orientés dans le repère monde (repère initial)
- de lumières ayant une position (x,y,z)
- d'une caméra ayant une position (x,y,z)



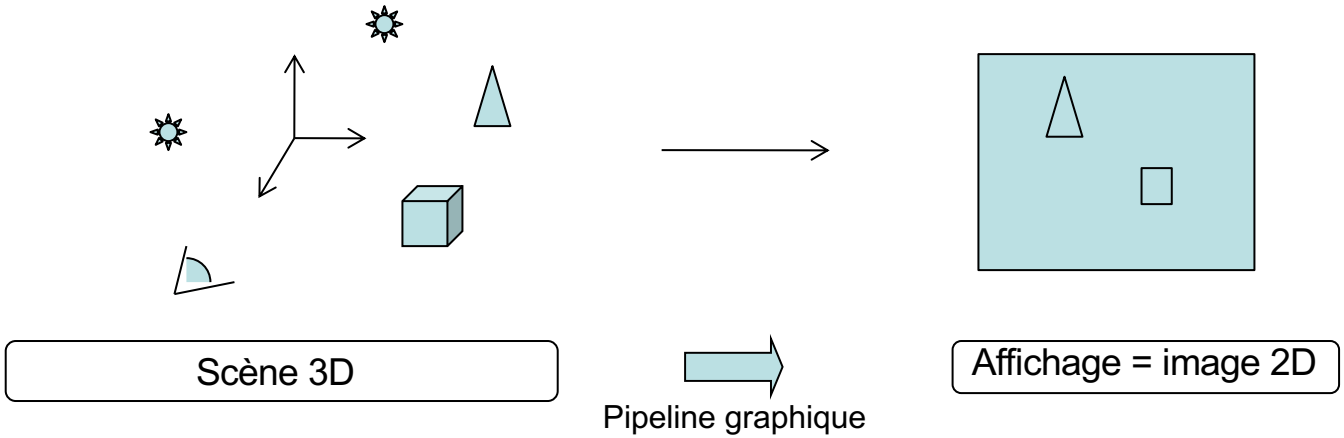
La description de cet ensemble va permettre de générer l'image 2D à un instant donné
Pour cela les objets vont remplir une partie de l'image 2D à un instant donné

Pipeline graphique pour effectuer le rendu / affichage de la scène

Utilisation d'une **librairie graphique = librairie OpenGL**

La librairie OpenGL ne fournit pas de management de la scène et du rendu mais fournit une API au dessus de la carte graphique qui permet de lui donner des ordres

Permet d'effectuer le passage de la scène 3D à l'image 2D de l'écran
pour cela les objets vont être affichés un par un



Nous allons survoler les différents problèmes
et voir les étapes à réaliser afin de créer l'image

Positionnement des objets dans la scène

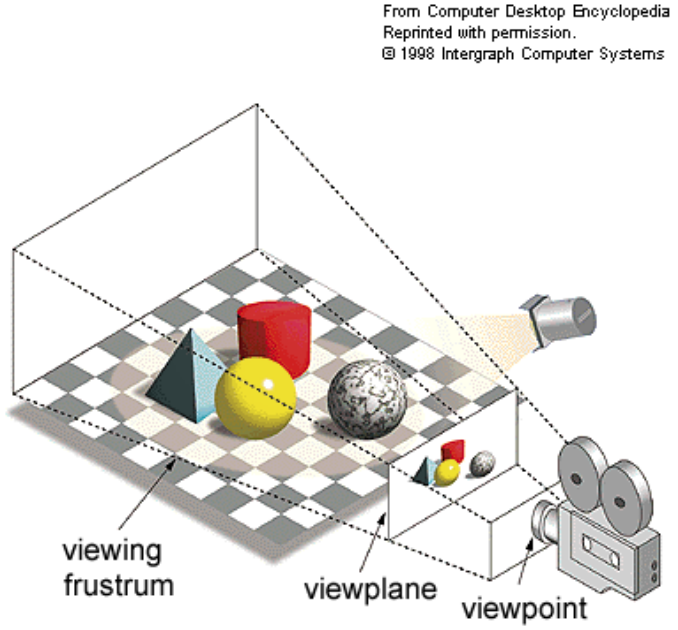
Affichage de la scène

Problème de l'occlusion

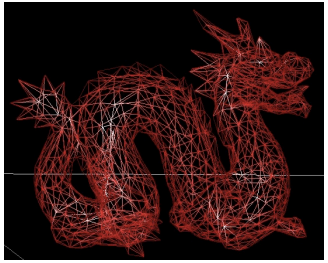
Pipeline graphique

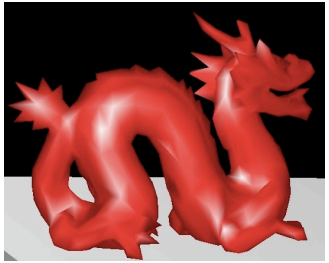
Description hiérarchique des objets

Texture



Rendu / affichage de la scène





Positionnement des objets dans la scène 3D

Exemple d'un objet modélisé par un **maillage triangulaire**

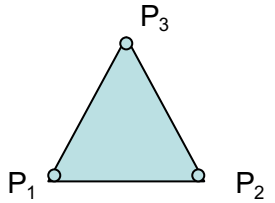
Les triangles permettent :

- d'approximer n'importe quel type de surface
- la projection d'un triangle est un triangle
- les triangles sont plans

Le maillage est ainsi défini par :

- un ensemble de sommets représentés par des vecteurs représentant leurs coordonnées
- un ensemble de facettes (triangles) décrites par 3 sommets (reliés par des arêtes)

$$P_1 = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$



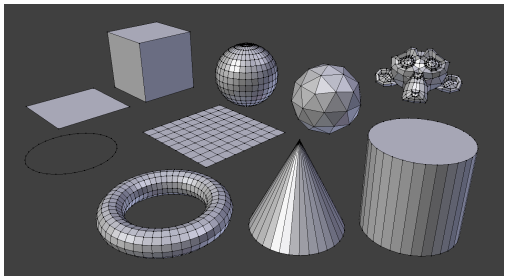
Les triangles sont localisés dans la scène par leurs 3 points dans l'espace

Positionnement des objets dans la scène 3D

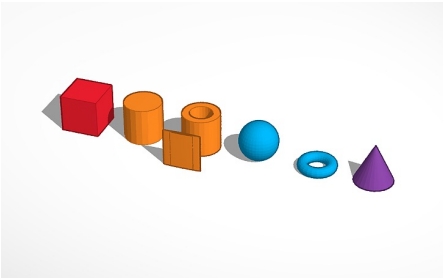
Il faut réaliser sur ces données plusieurs transformations géométriques :

- translation
- rotation
- changement d'échelle
- projections
- etc.

Cela permet de **positionner les objets dans la scène**



Objets 3D créés



On les positionne / transforme pour créer la scène 3D

Rappel sur les transformations géométriques

Rappel de transformations géométriques en 2D

Un point est défini par deux coordonnées : x et y

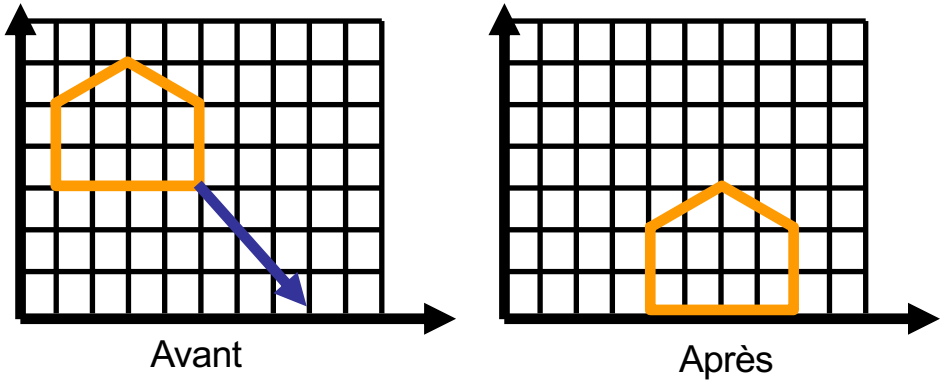
Différentes transformations possibles pour déplacer ce point :

- Translation
- Rotation
- Changement d'échelle

Rappel sur les transformations géométriques

Translation d'un point 2D

$x' = x + T_x$ et $y' = y + T_y$



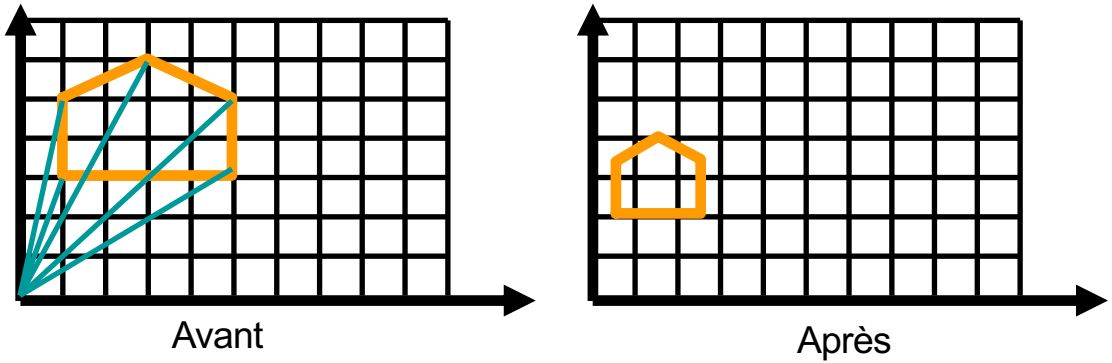
Notation : P' = P + T (somme vectorielle)

avec $P = \begin{pmatrix} x \\ y \end{pmatrix}$ et $T = \begin{pmatrix} T_x \\ T_y \end{pmatrix}$

Rappel sur les transformations géométriques

Changement d'échelle d'un point 2D

$x' = S_x x$ et $y' = S_y y$



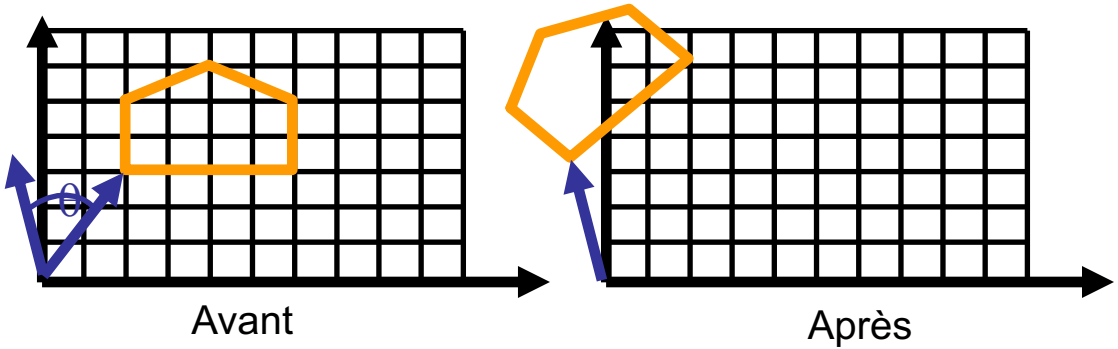
Notation : P' = S P (multiplication matricielle)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Rappel sur les transformations géométriques

Rotation d'un point 2D d'un angle θ

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



Notation : P' = R P (multiplication matricielle)

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

Rappel sur les transformations géométriques

La notation pour ces transformations est simple et concise

Mais elle n'est pas unifiée :

- addition ou multiplication
- comment faire pour concaténer plusieurs transformations ?

On souhaite obtenir une notation unique

- qui permette de noter facilement les combinaisons de transformations

⇨ **Utilisation de coordonnées dites homogènes**

Coordonnées homogènes en 2D

Coordonnées homogènes employées en Image, Vision et Robotique

On ajoute une troisième coordonnée notée w

Un point 2D devient un vecteur à 3 coordonnées : $\begin{pmatrix} x \\ y \\ w \end{pmatrix}$

Deux points sont alors égaux ssi :

x' / w' = x / w et y' / w' = y / w

Si w=0, on a un point à l'infini (utile pour les projections)

Transformations géométriques avec les coordonnées homogènes – En 2D

Notation de la translation en coordonnées homogènes en 2D :

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} \Rightarrow \begin{cases} x' = x + wT_x \\ y' = y + wT_y \\ w' = w \end{cases}$$

$$\Rightarrow \begin{cases} \frac{x'}{w'} = \frac{x}{w} + T_x \\ \frac{y'}{w'} = \frac{y}{w} + T_y \end{cases}$$

Transformations géométriques avec les coordonnées homogènes – En 2D

Notation du changement d'échelle en coordonnées homogènes en 2D :

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} \Rightarrow \begin{cases} x' = xS_x \\ y' = yS_y \\ w' = w \end{cases}$$

$$\Rightarrow \begin{cases} \frac{x'}{w'} = \frac{x}{w} S_x \\ \frac{y'}{w'} = \frac{y}{w} S_y \end{cases}$$

Transformations géométriques avec les coordonnées homogènes – En 2D

Notation de la rotation en coordonnées homogènes en 2D :

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} \Rightarrow \begin{cases} x' = \cos\theta x - \sin\theta y \\ y' = \sin\theta x + \cos\theta y \\ w' = w \end{cases}$$

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

Transformations géométriques avec les coordonnées homogènes – En 2D

Notation de composition des transformations :

Il suffit de multiplier les matrices entre-elles :

composition d'une rotation et d'une translation :

$$M = R\,T$$

Exemple d'une rotation autour d'un point Q

- Translater Q à l'origine : T_Q
- Rotation autour de l'origine : R_Θ
- Translation en retour vers Q = $-T_Q$

$$P' = (-T_Q)\,R_\Theta T_Q\,P$$

Regardons maintenant pour les points définis en 3D

Transformations géométriques avec les coordonnées homogènes – En 3D

Notation en coordonnées homogènes en 3D :

Introduction d'une **quatrième coordonnée notée w**

Un point 3D devient un vecteur à 4 coordonnées :

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

Passage en coordonnées homogènes :

Coordonnées cartésiennes → Coordonnées homogènes

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \rightarrow \begin{pmatrix} x'/w \\ y'/w \\ z'/w \\ w \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

w est le facteur d'échelle

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \rightarrow \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

cas avec w=1

Transformations géométriques avec les coordonnées homogènes – En 3D

Deux points sont alors égaux ssi :

$x' / w' = x / w$, $y' / w' = y / w$ et $z' / w' = z / w$

Toutes les matrices de transformations sont de taille 4 X 4 : M_{4x4}

Transformation inverse obtenue en utilisant matrice M^{-1}_{4x4}

Transformations géométriques avec les coordonnées homogènes – En 3D

Notation de la translation en coordonnées homogènes en 3D :

Matrice de la translation $T(T_x, T_y, T_z)$:

$$\begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\Rightarrow \begin{cases} x' = x + wT_x \\ y' = y + wT_y \\ z' = z + wT_z \\ w' = w \end{cases}$$

Transformations géométriques avec les coordonnées homogènes – En 3D

Notation du changement d'échelle en 3D :

Matrice du changement d'échelle $S(S_x, S_y, S_z)$:

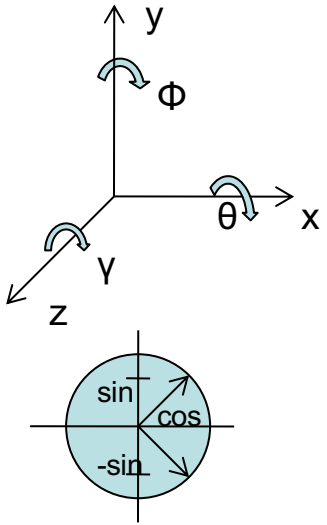
$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\Rightarrow \begin{cases} x' = xS_x \\ y' = yS_y \\ z' = zS_z \\ w' = w \end{cases}$$

Transformations géométriques avec les coordonnées homogènes – En 3D

Rotation en coordonnées homogènes en 3D :

Rotation dépend d'un axe et d'un angle



Rotation autour de l'axe X :

(coordonnée en x non modifiée)

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

Rotation autour de l'axe Y :

(coordonnée en y non modifiée)

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

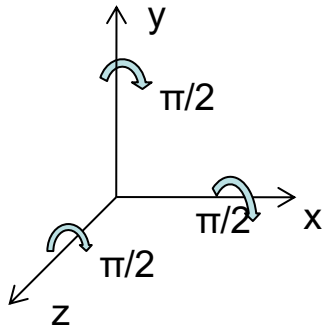
Rotation autour de l'axe Z :

(coordonnée en z non modifiée)

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

Transformations géométriques avec les coordonnées homogènes – En 3D

Rotation en coordonnées homogènes en 3D de l'angle π /2 :



Rotation autour de l'axe X :

(coordonnée en x non modifiée
coordonnée en y changée en z
coordonnée en z changée en -y)

$$R_x(\frac{\pi}{2}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation autour de l'axe Y :

(coordonnée en y non modifiée
coordonnée en z changée en x
coordonnée en x changée en –z)

$$R_y(\frac{\pi}{2}) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation autour de l'axe Z :

(coordonnée en z non modifiée
coordonnée en x changée en y
coordonnée en y changée en -x)

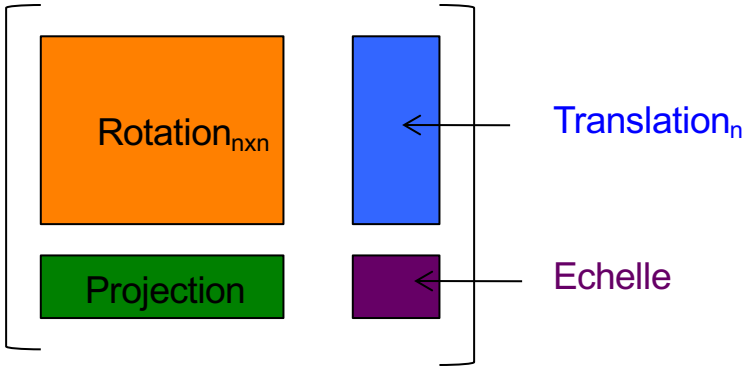
$$R_z(\frac{\pi}{2}) = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformations géométriques avec les coordonnées homogènes – En 3D

Matrice de transformation générale en nD :

Utilisation des coordonnées homogènes

Matrice de taille (n+1) x (n+1)



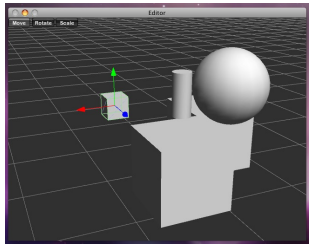
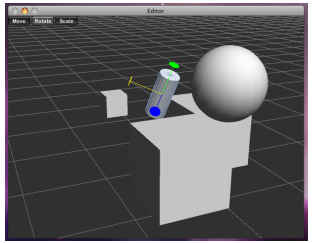
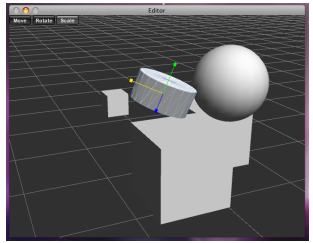
Attention : La multiplication de matrices n'est pas commutative

L'ordre des transformations est donc important :

rotation puis translation ≠ translation puis rotation

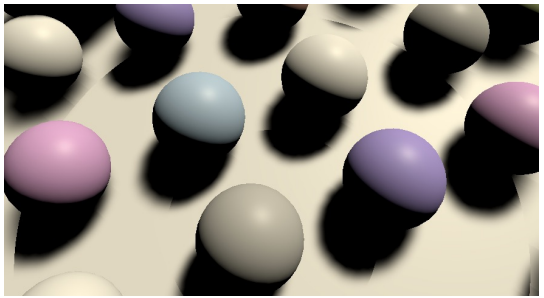
Positionnement des objets dans la scène 3D

Ces transformations géométriques (translation, rotation, changement d'échelles, etc.) sont appliquées aux **objets pour créer la scène**

On crée une scène OpenGL en explicitant les transformations à appliquer aux objets

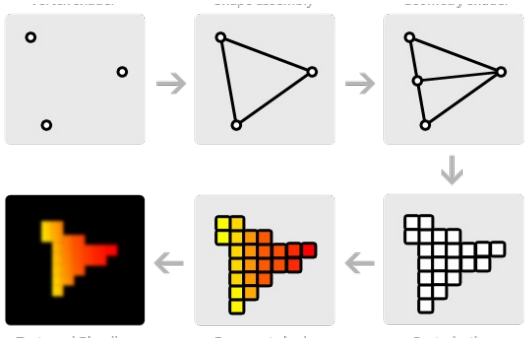
Ce n'est que de l'affichage :
maillage d'une sphère créé une fois,
affichage x fois,
en appliquant des transformations géométriques



Les sommets sont attachés aux **primitives graphiques** (couleur, etc.)

Rendu / affichage de la scène

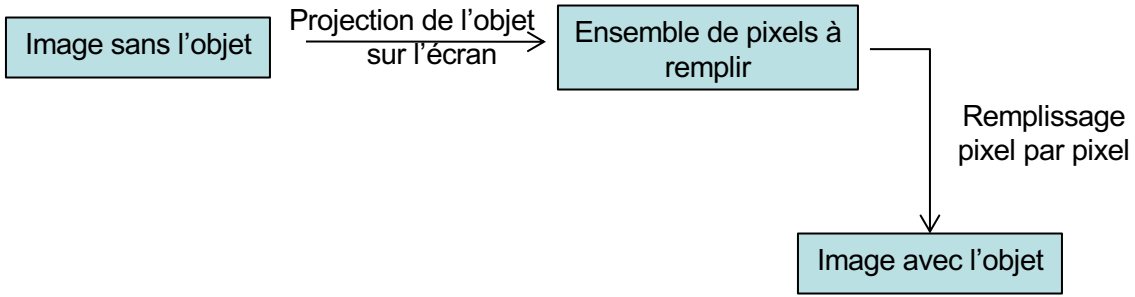
Affichage de la scène



Il faut déterminer quels seront les pixels impactés par les objets

L'affichage s'effectue en deux étapes :

1. Placement des objets dans l'écran
2. Remplissage de l'objet pixel par pixel

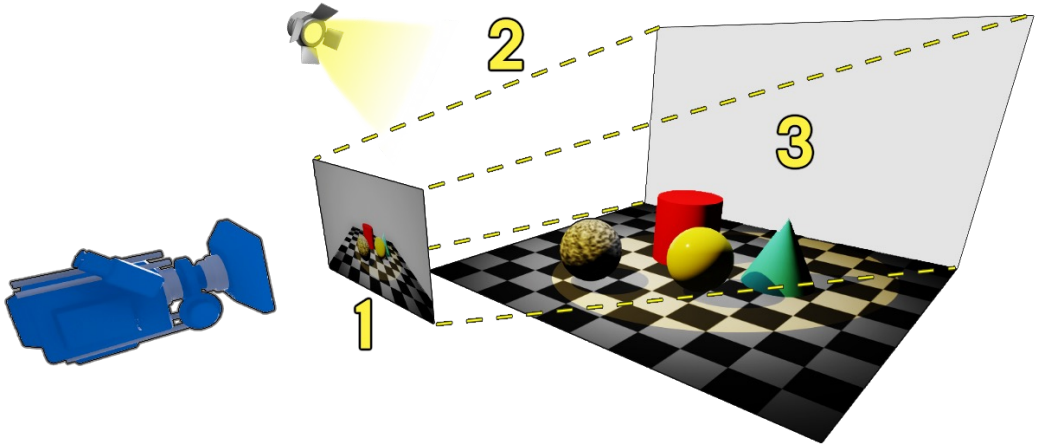


Rendu / affichage de la scène — Problème de l'occlusion

Le problème apparaît quand un objet se trouve devant un autre

Prendre en compte qu'un objet n'est pas toujours visible selon le point de vue de l'observateur

Mais ceci n'est pas pré-calculable car dépend du point de vue



Algorithme de profondeur pour savoir si c'est l'objet considéré qui remplit pixel ou un autre

Rendu / affichage de la scène

Pipeline graphique (fait par la carte graphique)

1. *Clipping* des objets 3D selon la pyramide de vue

2. Projection des points sur le plan image

3. Remplissage des triangles dans l'image (*rasterisation*)
   a. Suppression des parties cachées (*Z-Buffer*)
   b. Calcul de la couleur (illumination)

Rendu / affichage de la scène

Pipeline graphique

1. *Clipping* des objets 3D selon la pyramide de vue

2. Projection des points sur le plan image

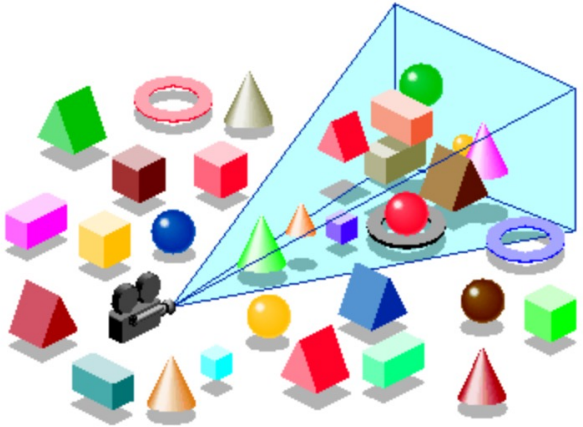
3. Remplissage des triangles dans l'image (*rasterisation*)
   a. Résolution de l'occlusion avec la suppression des parties cachées = *Z-Buffer*
   b. Calcul de la couleur des pixels = illumination

Pipeline graphique pour afficher la scène *Clipping*

viewport ou cône de vision : définit la partie de la fenêtre dans laquelle les tracés vont être réalisés

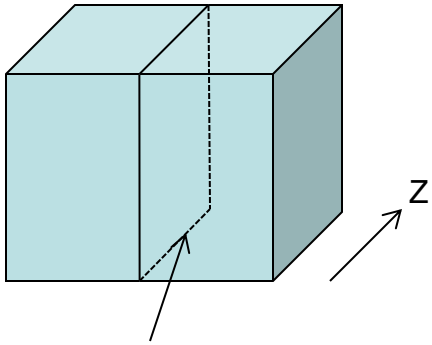
clipping : supprime la partie de la scène qui se trouve en dehors du *viewport* ou du cône de vision

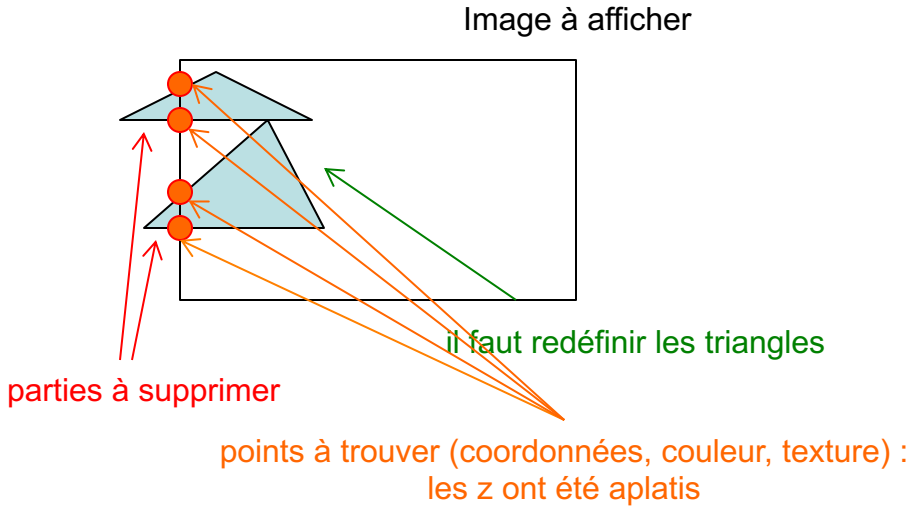
⇨ réduit le champ de vision de la scène
de la même façon que notre champ de vision réel ne permet pas de voir tout en même temps



Pipeline graphique pour afficher la scène

Illustration du *clipping*





Rendu / affichage de la scène

Pipeline graphique

1. *Clipping* des objets 3D selon la pyramide de vue

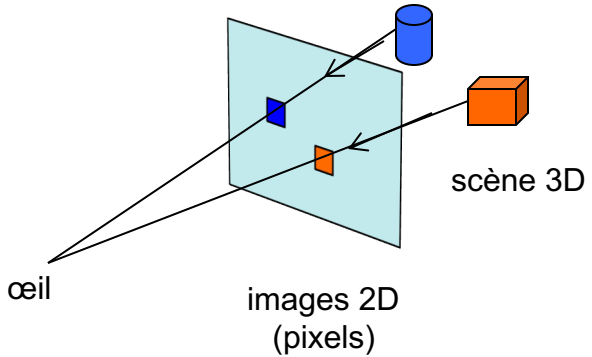
2. Projection des points sur le plan image

3. Remplissage des triangles dans l'image (*rasterisation*)
   a. Suppression des parties cachées (*Z-Buffer*)
   b. Calcul de la couleur (illumination)

Projection des triangles

La scène 3D doit être affichée sur un écran 2D

Transformation à effectuer pour passer d'un espace 3D à un espace 2D



Les objets sont projetés sur l'écran dans la direction de l'œil / de la caméra

Transformation des coordonnées 3D des modèles vers les coordonnées 2D des pixels

Pipeline graphique pour afficher la scène

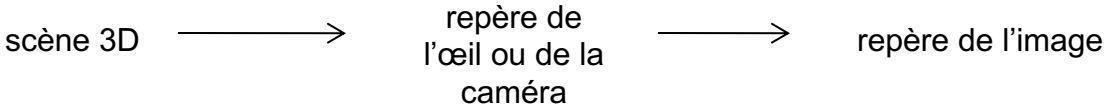
Projection de la scène 3D

La scène est représentée par rapport à un observateur virtuel

- Point de vue : position de l'observateur
- Direction de visée : direction vers laquelle est tournée l'observateur
- Direction en haut : verticale pour l'observateur

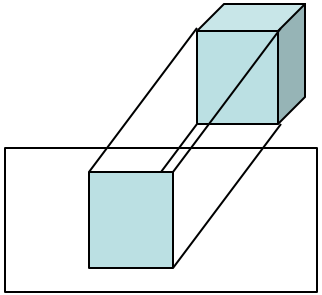
La projection de la scène 3D sur l'image se fait en deux étapes :

1. On doit ramener tous les points de la scène dans le repère de l'œil (translation et changement d'échelle)
2. Puis, projection du repère de l'œil vers le repère de l'image

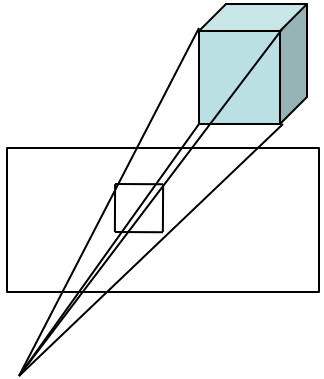


Projection de la scène 3D sur l'écran 2D

Différents types de projection possibles :



Projection parallèle



Projection perspective

Projection de la scène 3D sur l'écran 2D

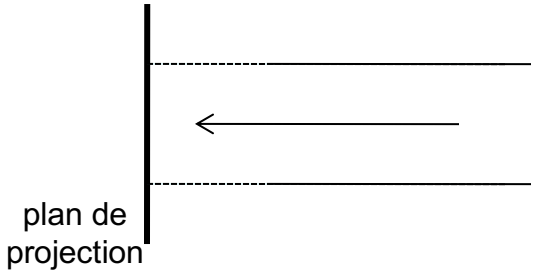
Projection parallèle

Projection parallèle sur le plan z=0

Direction de projection est (0, 0, 1)

⇨ x' = x, y' = y, z'=0 et w'=w

$$P = \begin{pmatrix} x \\ y \\ z \\ w = 1 \end{pmatrix} \Rightarrow P' = \begin{pmatrix} x' = x \\ y' = y \\ z' = 0 \\ w' = 1 \end{pmatrix} \Rightarrow \text{matrice de projection :} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Projection de la scène 3D sur l'écran 2D

Problème de la projection parallèle

Pas de rétrécissement des objets dans le lointain

⇨ pas possible de rendre ce type d'images

⇨ **manque de réalisme**

Projection de la scène 3D sur l'écran 2D

Projection perspective

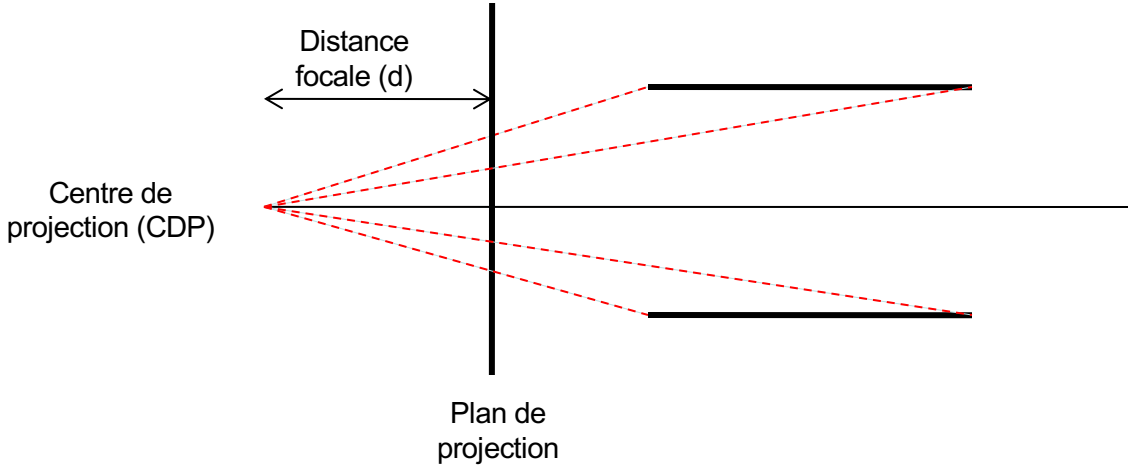
Intérêt : les objets lointains sont plus petits

Deux droites parallèles se rejoignent en un point appelé *point de fuite*

Plusieurs types de projection : un, deux ou trois points de fuite

Projection de la scène 3D sur l'écran 2D

Projection perspective – Notion de distance focale

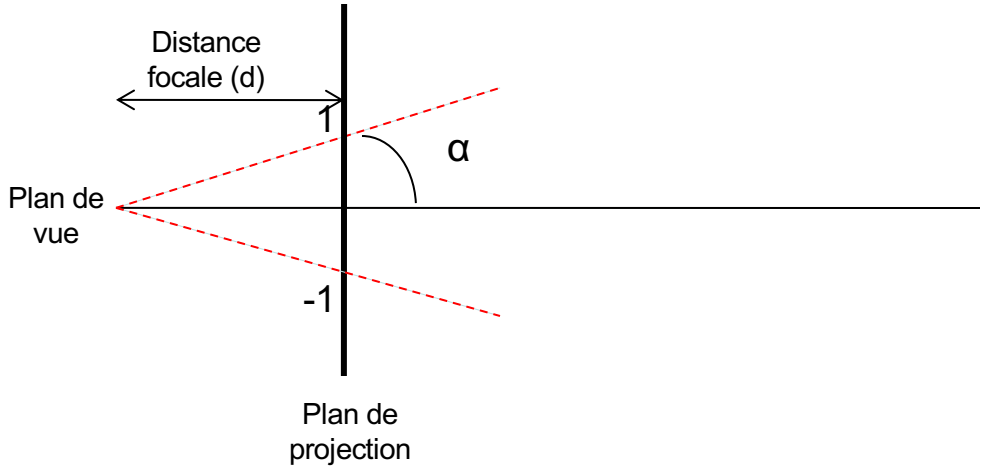


Projection de la scène 3D sur l'écran 2D

Projection perspective - Notion d'ouverture de vue

Angle qui exprime la largeur du champ visuel

Relation avec la distance focale : $\tan\alpha = \frac{1}{d}$

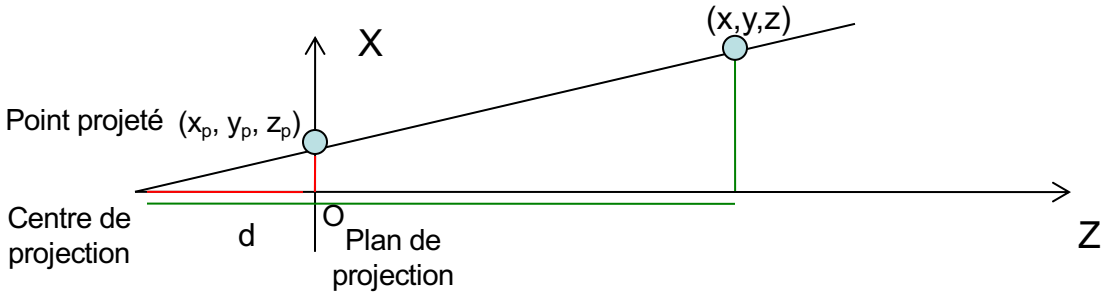


Projection de la scène 3D sur l'écran 2D

Equations de perspectives :

$$\begin{cases} \dfrac{x_p}{d} = \dfrac{x}{d+z} \Rightarrow x_p = \dfrac{xd}{d+z} \\ \dfrac{y_p}{d} = \dfrac{y}{d+z} \Rightarrow y_p = \dfrac{yd}{d+z} \\ z_p = 0 \end{cases}$$

(même raisonnement)

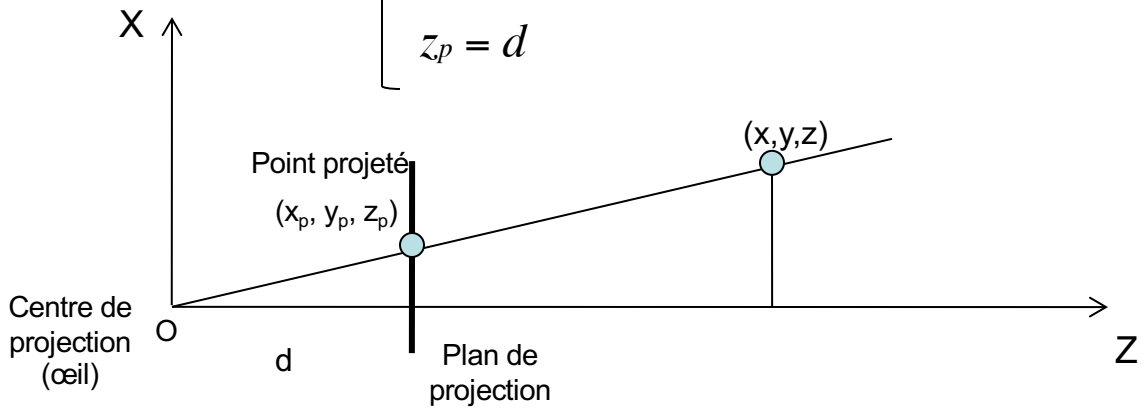


Projection de la scène 3D sur l'écran 2D

Equations de perspectives simplifiées
en plaçant l'origine / l'œil au point de projection :

$$\begin{cases} \dfrac{x_p}{d} = \dfrac{x}{z} \Rightarrow x_p = \dfrac{xd}{z} \\ \dfrac{y_p}{d} = \dfrac{y}{z} \Rightarrow y_p = \dfrac{yd}{z} \\ z_p = d \end{cases}$$

(même raisonnement)



Projection de la scène 3D sur l'écran 2D

Equations de perspectives en coordonnées homogènes (cas simplifié) :

$$\begin{cases} x_p = \dfrac{xd}{z} = \dfrac{x}{w} \\ y_p = \dfrac{yd}{z} = \dfrac{y}{w} \\ z_p = d = \dfrac{z}{w} \end{cases} \Rightarrow \begin{cases} wx_p = x \\ wy_p = y \\ wz_p = z \\ w = \dfrac{z}{d} \end{cases} \Rightarrow \begin{pmatrix} wx_p \\ wy_p \\ wz_p \\ w \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

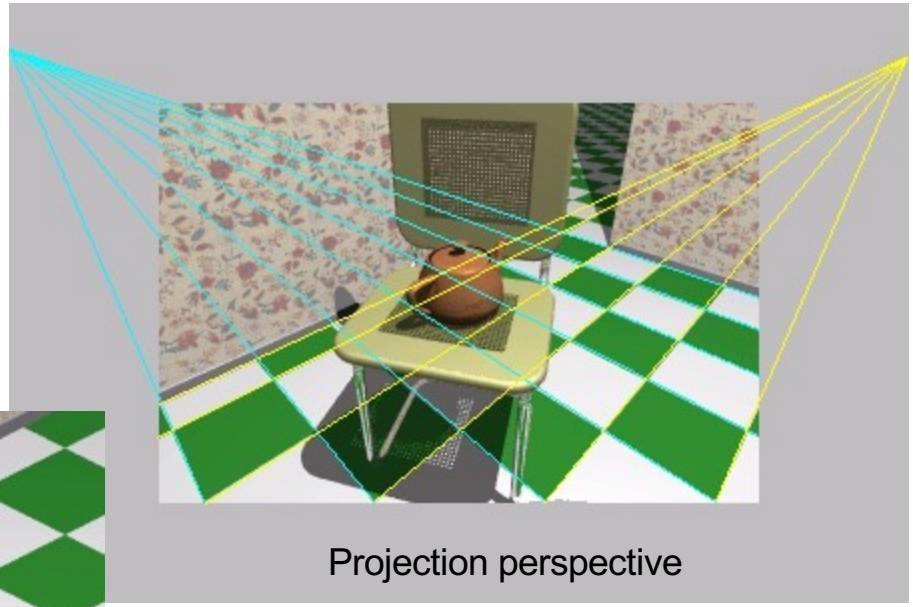
⇨ Matrice de passage $M_{I \leftarrow C}$:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix}$$

Cette matrice permet de passer du repère de l'œil / de la caméra au repère de l'image

Projection de la scène 3D sur l'écran 2D

Comparaison des deux types de projection



Projection perspective

Projection parallèle

Projection de la scène 3D sur l'écran 2D

Récapitulatif pour la projection perspective

Projection est définie par une position de l'œil (ou de la caméra)
notée E = (E_x, E_y, E_z)

Coordonnées définies dans le repère absolu (repère du monde virtuel) dans lequel sont définis les objets 3D de la scène

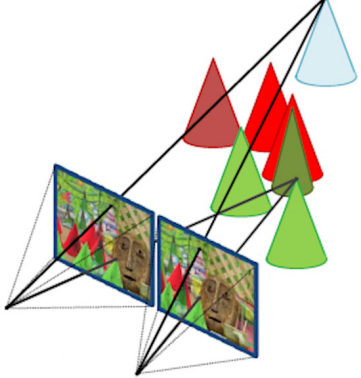
Repère local placé sur cette position définissant la direction de l'observateur

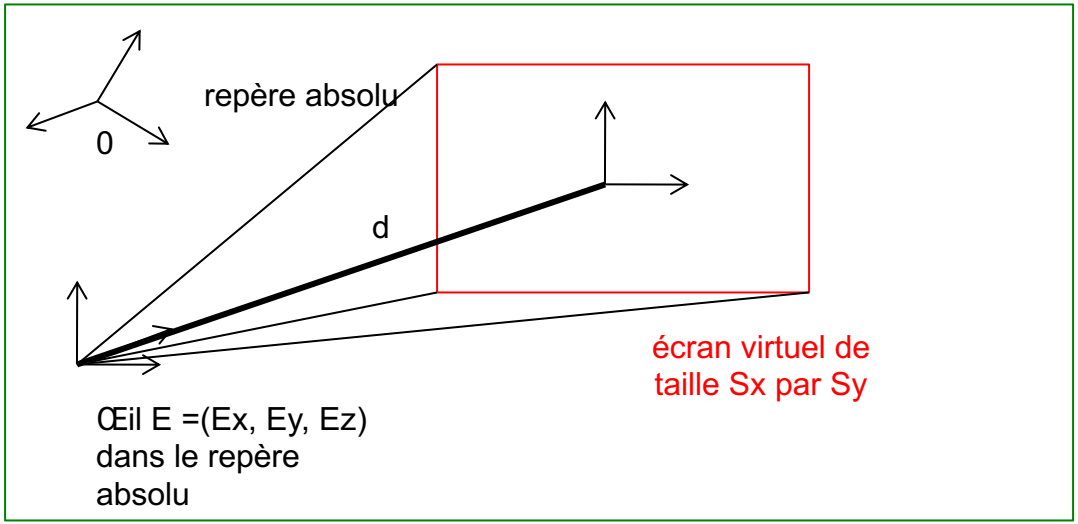
Définition de l'écran virtuel sur lequel on projettera les points 3D de taille Sx et Sy

Cet écran se trouve à une distance d de l'œil / de la caméra

Projection de la scène 3D sur l'écran 2D

Récapitulatif pour la projection perspective





monde virtuel

Projection de la scène 3D sur l'écran 2D

Récapitulatif pour la projection perspective

Pour pouvoir appliquer la projection perspective

1. Il faut tout d'abord ramener le repère vers l'œil (translation et changement de repère)
2. Puis il faut appliquer la perspective
3. Et enfin, il faut remettre à l'échelle de l'écran (Sx, Sy)

⇨ Définition des matrices de transformation de ces différentes étapes
(attention à l'ordre de ces transformations)

Projection de la scène 3D sur l'écran 2D

Matrices de transformation pour la projection perspective

$$S = \begin{pmatrix} Sx/2 & 0 & 0 & 0 \\ 0 & Sy/2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Mise à l'échelle de l'écran

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix}$$

Perspective

$$M = \begin{pmatrix} (M_{oeil})^{-1} & & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Changement de base

$$D = \begin{pmatrix} 1 & 0 & 0 & -Ex \\ 0 & 1 & 0 & -Ey \\ 0 & 0 & 1 & -Ez \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Changement de repère : translation vers l'œil qui est l'origine

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = SPMD \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

Point dans l'image — Point dans le monde virtuel

Rendu / affichage de la scène

Pipeline graphique

1. *Clipping* des objets 3D selon la pyramide de vue

2. Projection des points sur le plan image

3. Remplissage des triangles dans l'image (*rasterisation*)
   a. Suppression des parties cachées (*Z-Buffer*)
   b. Calcul de la couleur (illumination)

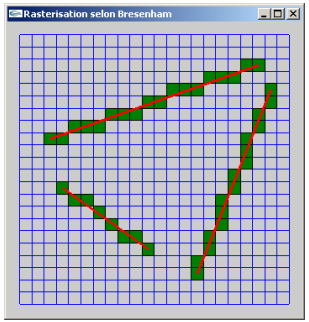
Pipeline graphique pour afficher la scène

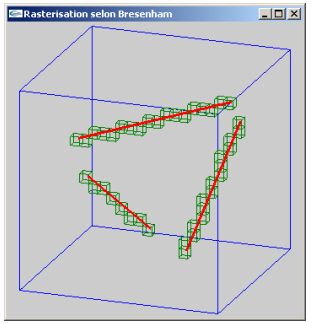
Rasterisation

Pour projeter un polygone sur l'écran, on projette successivement tous ses sommets et on les relie entre eux par des segments pour obtenir les arêtes

Problème : écran est défini par un ensemble de pixels

Algorithme de Bresenham utiliser pour tracer les segments (remplir les pixels adéquates)



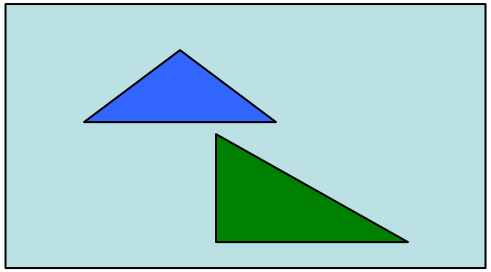


Pipeline graphique pour afficher la scène

Rasterisation

On peut ensuite déterminer tous les pixels couverts par le triangle

Il faut ensuite remplir correctement l'ensemble des pixels

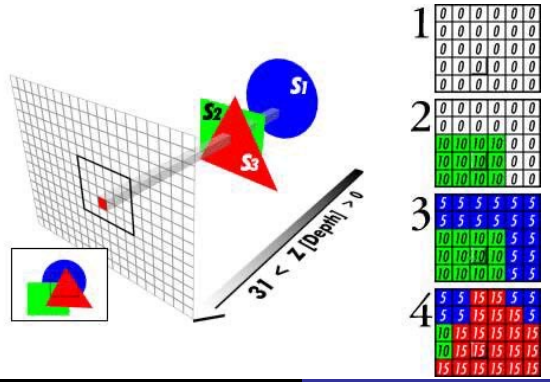


Pipeline graphique pour afficher la scène

Savoir quel objet est visible sur l'image = *test de profondeur*

Le Z-Buffer permet de gérer le problème de la visibilité
→ déterminer quels éléments de la scène doivent être rendus,
lesquels sont cachés par d'autres
et dans quel ordre l'affichage des primitives doit se faire.

Z-Buffer = tableau à 2 dimensions (X et Y) de la même taille que l'image
= chaque élément est un pixel de l'écran
= stocke les valeurs de profondeur des pixels (coordonnée Z)



Pipeline graphique pour afficher la scène

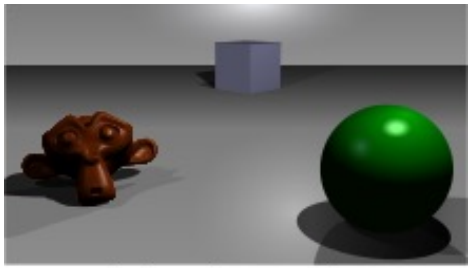
Test de profondeur

Si autre élément de la scène doit être affiché aux mêmes coordonnées (X,Y),
- la carte compare les deux profondeurs (Z),
- et n'affiche que le pixel le plus proche de la caméra

La valeur Z de ce pixel est placée dans le tampon de profondeur, remplaçant l'ancienne

Au final, l'image dessinée reproduit la perception de la profondeur habituelle et logique,
l'objet le plus proche cachant les plus lointains.

Une scène 3d simple



A simple three-dimensional scene

Le Tampon de profondeur

Z-buffer representation

Pipeline graphique pour afficher la scène

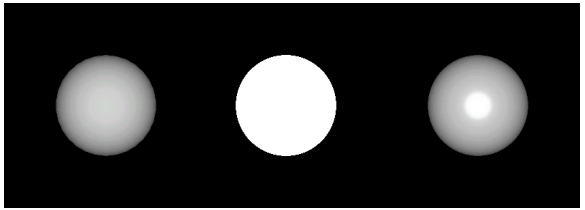
Calcul de la couleur de chacun des pixels de l'image 2D = illumination

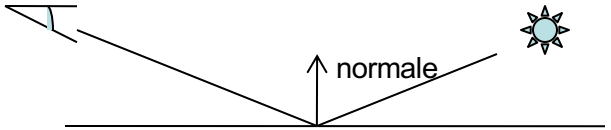
La **couleur du pixel** est définie soit :

- de manière explicite
- ou selon un modèle de fond établi aux sommets des triangles,
- puis interpolation pour remplir les pixels à l'intérieur des triangles

Tient compte de la **lumière** pour donner un aspect plus réel
Tient compte du **placement de lampes** qui ont une **couleur**
Tient compte des **couleurs des différents objets**

Nécessité de calculer la normale aux différents sommets de la surface





Pipeline graphique pour afficher la scène

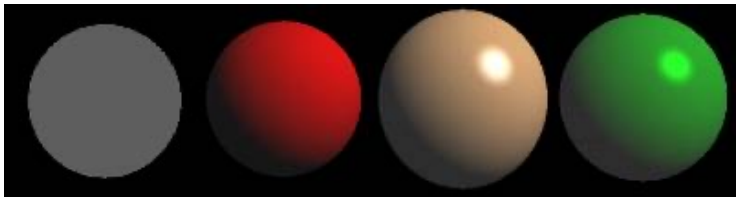
Il existe différents types de lumière :

Composante ambiante : constante qui colore les pixels d'un objet par la même couleur quelque soit l'environnement lumineux.

Composante diffuse (différente pour chaque sommet) : permet de donner un effet 3D et lissé aux objets.

Composante spéculaire : correspond au léger reflet de la lumière sur les bords des objets.

Composante émissive : simulation de la lumière émise par un objet.



Composantes ambiante, diffuse, spéculaire et émissive

Modèle de Phong

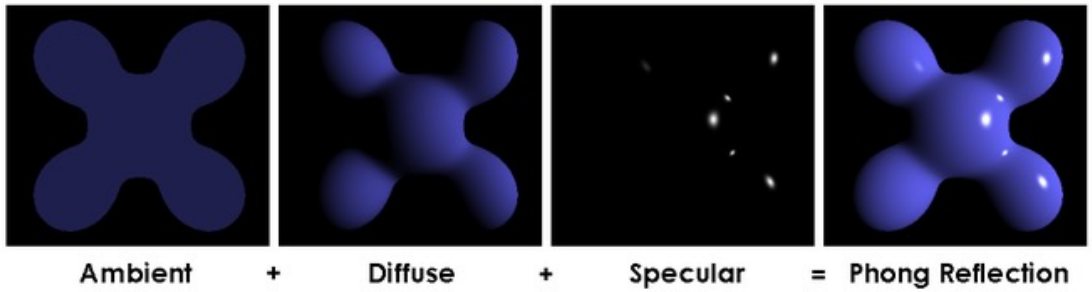
L'illumination de Phong est un modèle local pour calculer la couleur des pixels

Il calcule l'intensité en chaque point

Il combine trois éléments :

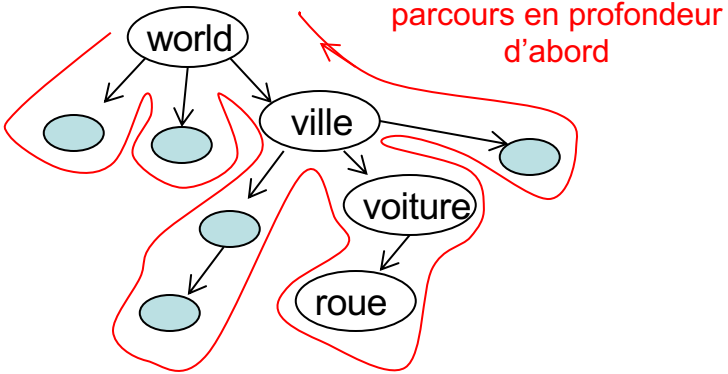
- la lumière diffuse (modèle Lambertien)
- la lumière spéculaire
- et la lumière ambiante

Formule de couleur pour les différents sommets : $K_a L_a + K_d L_d \cos\theta + K_s L_s \cos\theta$
avec θ l'angle formé par la direction de l'œil et la source lumineuse



Pipeline graphique pour afficher la scène

Description hiérarchique des objets

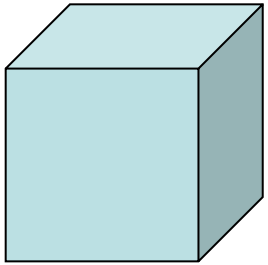


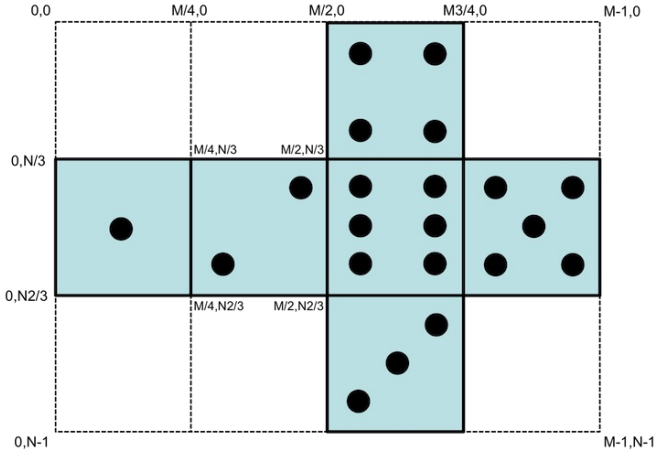
⇨ Roue connue par rapport à sa position relative vis à vis de la voiture
description hiérarchique de la scène

Dessin de la scène = traversée de la hiérarchie

Plaquage de texture = appliquer une image 2D sur la surface d'un objet 3D
→ Associer à chaque point P(x,y,z) de S un pixel (i,j) de l'image

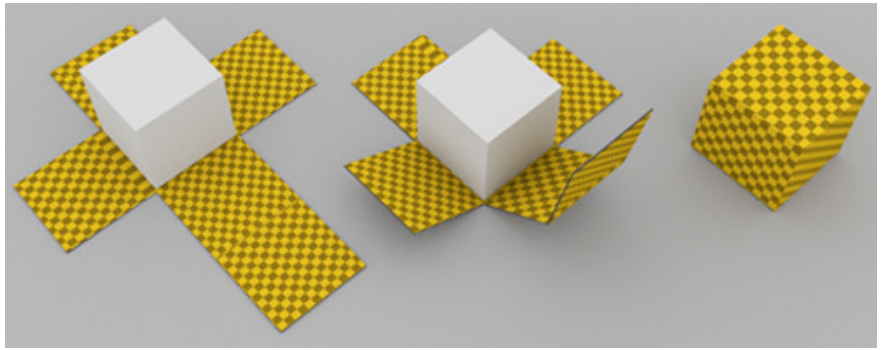
Pour chaque point P(x,y,z), on associe un couple de valeurs de texture normalisées (u,v) comprises entre 0 et 1 qui est associé au couple (i,j)

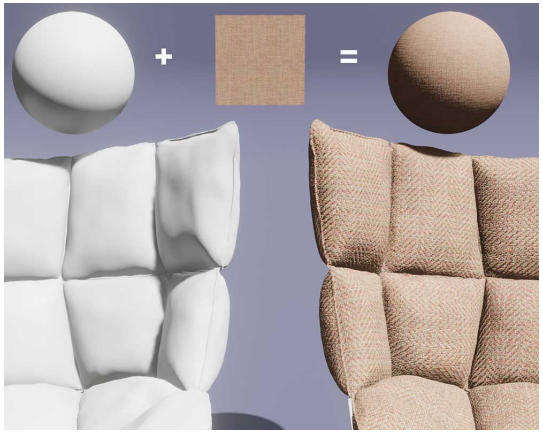




Pipeline graphique pour afficher la scène

Texture





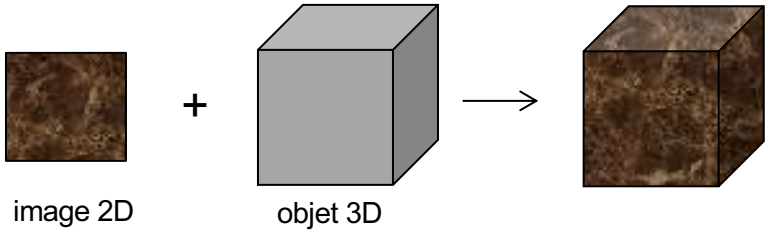
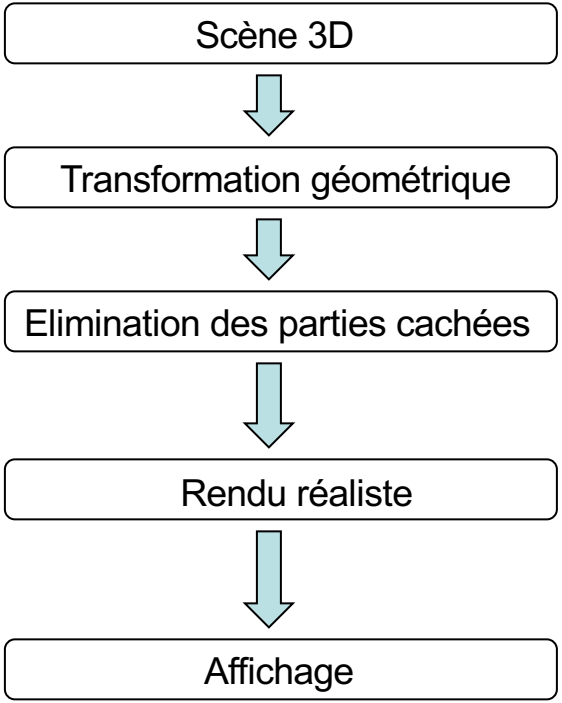


Tableau de valeurs établies à partir d'une image fournie à la carte graphique

Pipeline graphique pour effectuer le rendu / affichage de la scène



- Modélisation géométrique
- Animation

- Utilisation de la librairie OpenGL :
  - Rendu (éclairage, etc.)
  - Affichage à l'écran (projection, éliminations des parties cachées, ...)

- Rendu volumique

Autre type de rendu = Visualisation volumique

Intérêt de la visualisation volumique

Nous avons souvent des **données volumiques à visualiser** issues de simulations numériques, de données médicales, de scanners, etc.

⇒ f(x, y, z) ou f(x, y, z, t) à visualiser

La projection de ces données 3D dans un espace 2D fait perdre de l'information

La visualisation volumique permet de visualiser ces données 3D

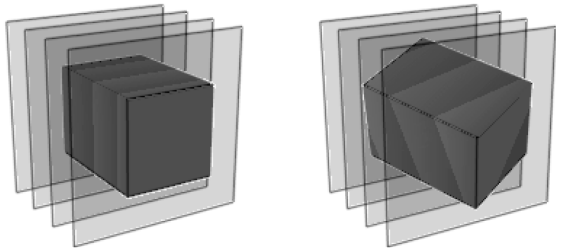
Méthode basée sur l'emploi de Texture 3D

Une texture 3D est réalisée en échantillonnant les données scalaires
→ interpolation linéaire pour trouver les valeurs des voxels

Utilisation d'une fonction de transfert permettant de souligner les détails
f(x, y, z) → (R, G, B) + opacité

Volume d'intensité intersectable par un plan quelconque pour en visualiser une tranche

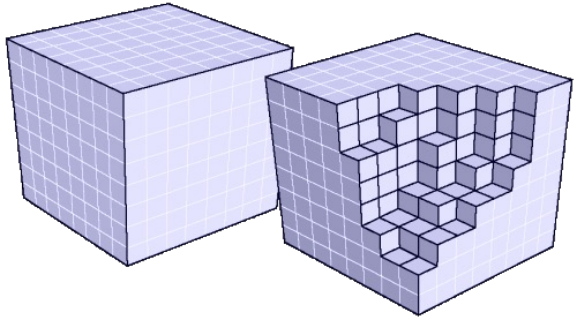
Taille limitée à la capacité de la mémoire dédiée

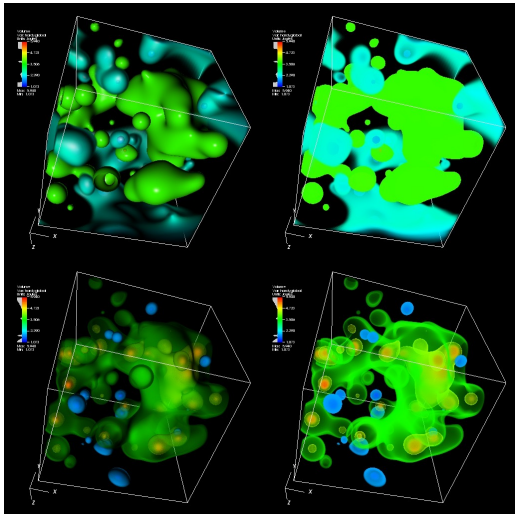


Autre méthode de rendu volumique

Le volume est vu en transparence grâce à l'emploi de *voxels* plus ou moins translucides

Juxtaposition des voxels transparents permet de voir l'intégralité du volume
représentée par une image variant selon l'angle de visualisation





Méthode du lancer de rayon (*raytracing*)

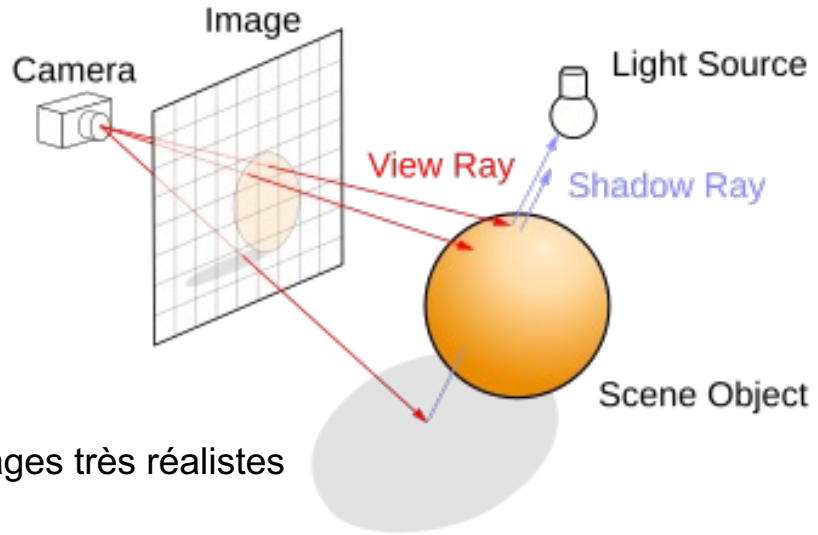
Des rayons sont lancés depuis le point focal de l'œil virtuel vers tous les *voxels* de l'image

L'absorption / réflexion des rayons est calculée en fonction de la transparence / opacité des voxels

calcul de l'intégrale du trajet d'un rayon à l'intérieur de l'objet détermine la couleur de chaque pixel de l'image

- C : fonction de transfert de la couleur
- τ : fonction de transfert de l'opacité
- S : valeur du scalaire au point considéré

$$\int_0^D C(S)\exp(-\int_0^S \tau(S')dS')ds$$

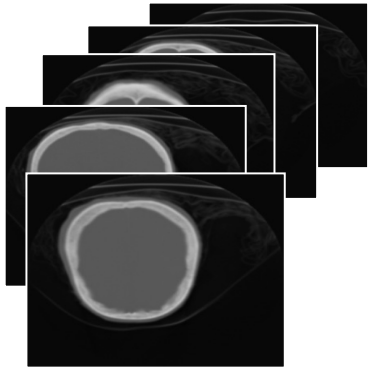


Cette méthode est coûteuse en calculs mais donne des images très réalistes

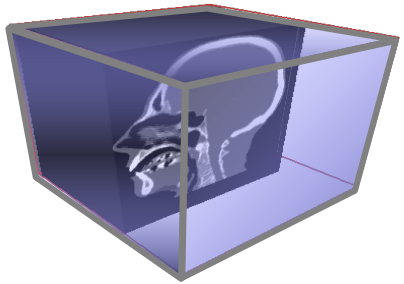
Rendu Volumique *raytracing*

Rendu volumique pour le médical

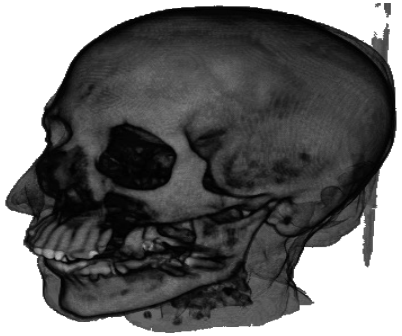
Données de départ

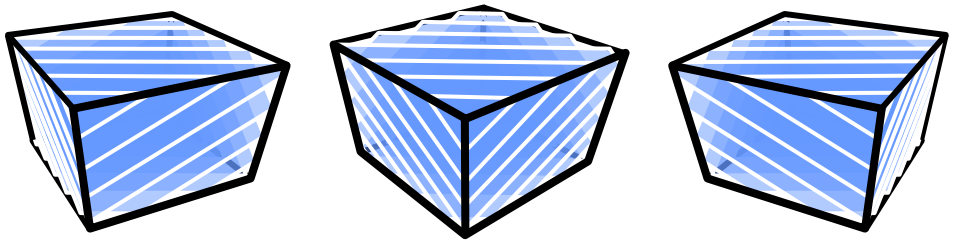


Assemblage



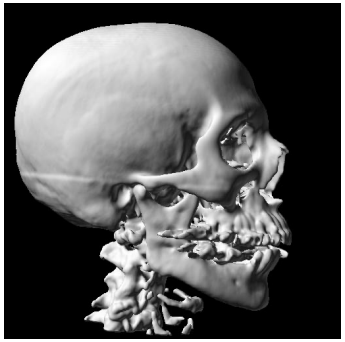
Rendu

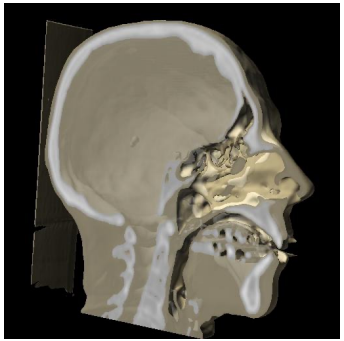




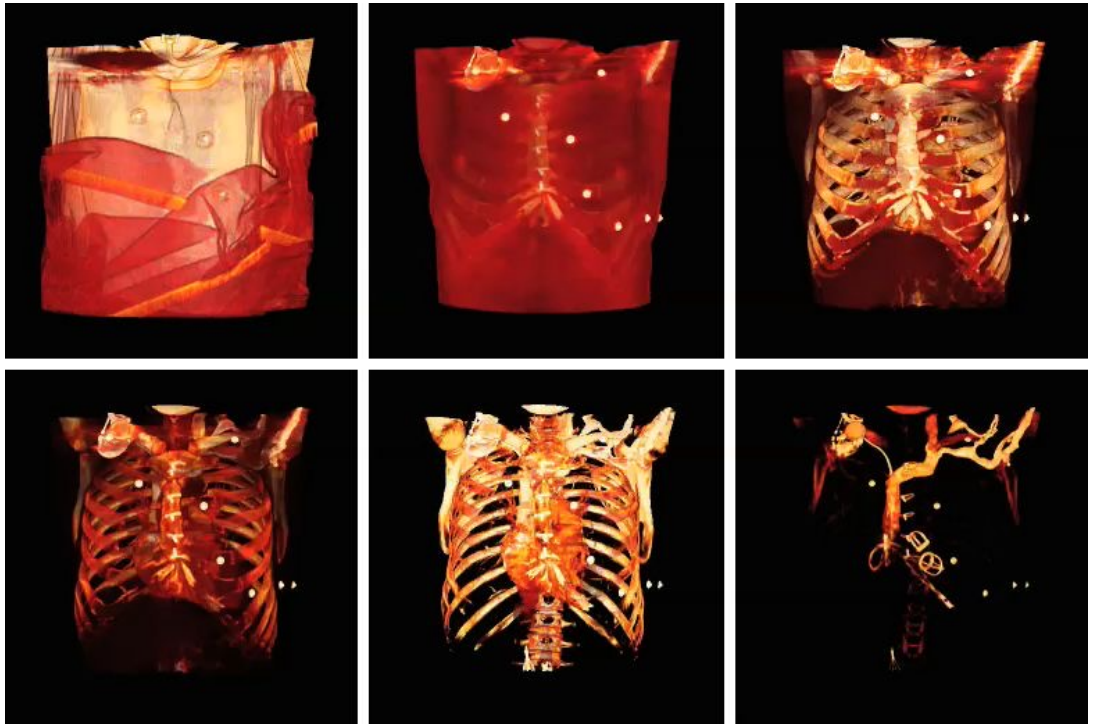
Tranches texturées
faisant face
à l'observateur

Rendu volumique pour le médical

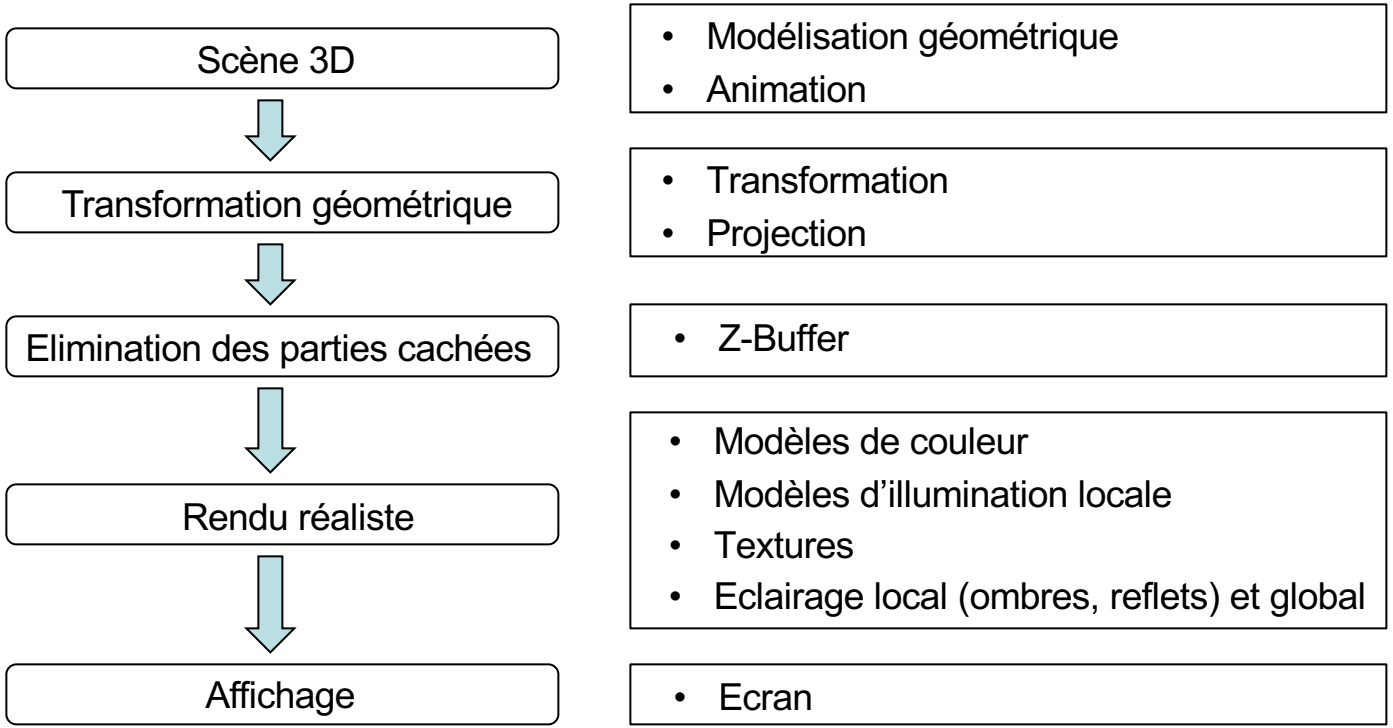




Rendu volumique pour le médical



Récapitulatif



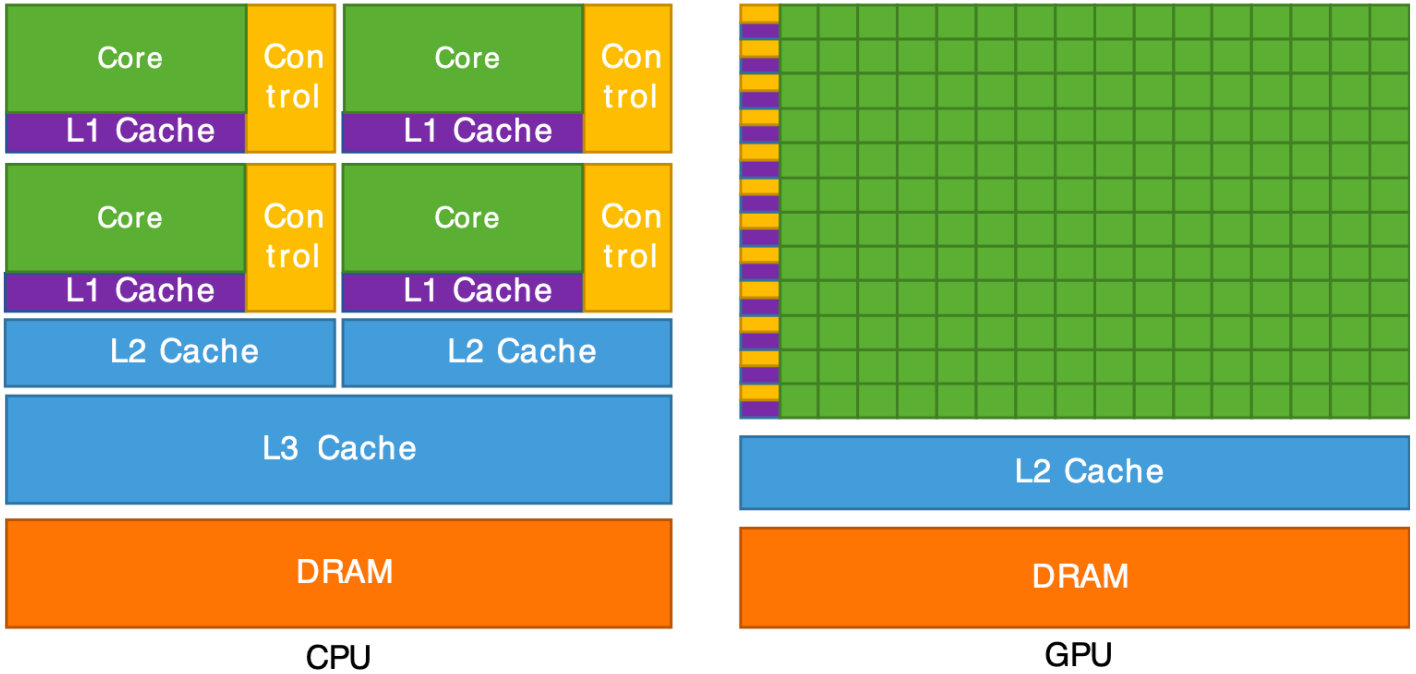
Tout ceci a été rendu possible grâce au matériel graphique (GPU) de plus en plus performant

NVIDIA

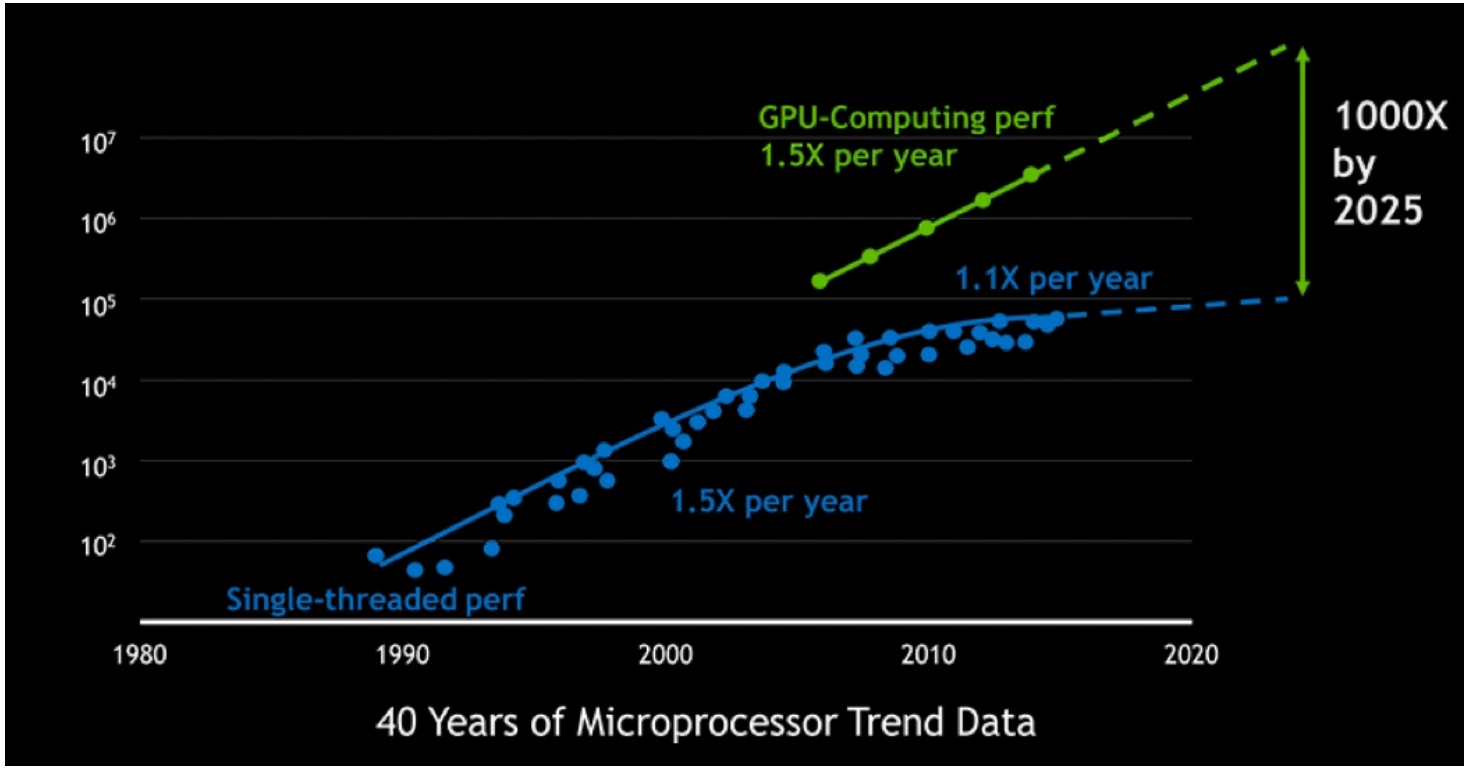
ATI

INTEL

GPU : Graphics Processing Unit

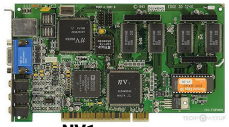


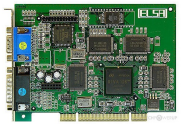
Evolution des GPU



(image NVIDIA)

Evolution des GPU


NV1


RIVA 128


RIVA TNT


RIVA TNT2 Ultra


GeForce 256 DDR


GeForce 2 Ultra


GeForce 3 Ti500


GeForce4 Ti 4800


GeForce FX 5950 Ultra


GeForce 6800 Ultra


GeForce 7950 GX2


GeForce 8800 GTS


GeForce 9800 GX2


GeForce GTX 295


GeForce GTX 480 Core 512


GeForce GTX 590

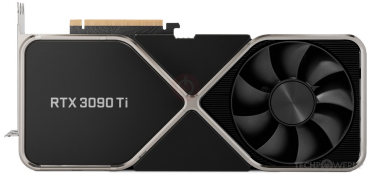

GeForce GTX 690


GeForce GTX TITAN Z

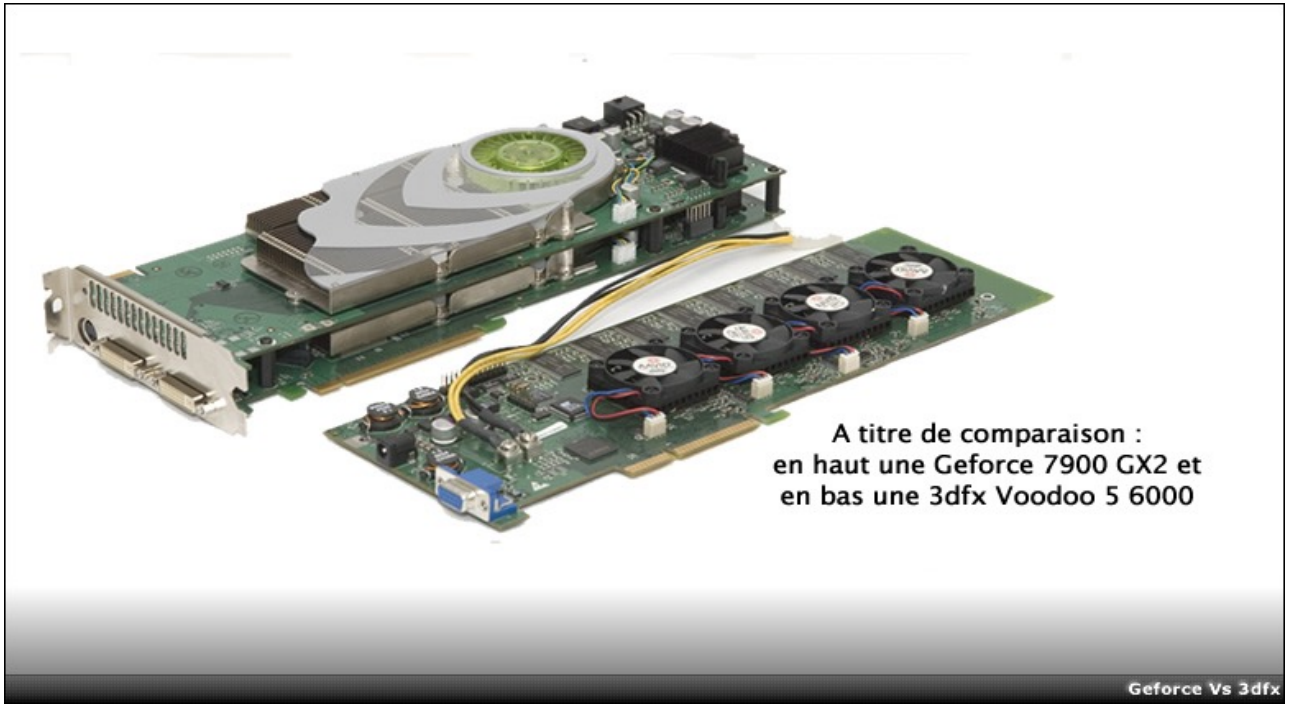

GeForce GTX TITAN X


TITAN Xp

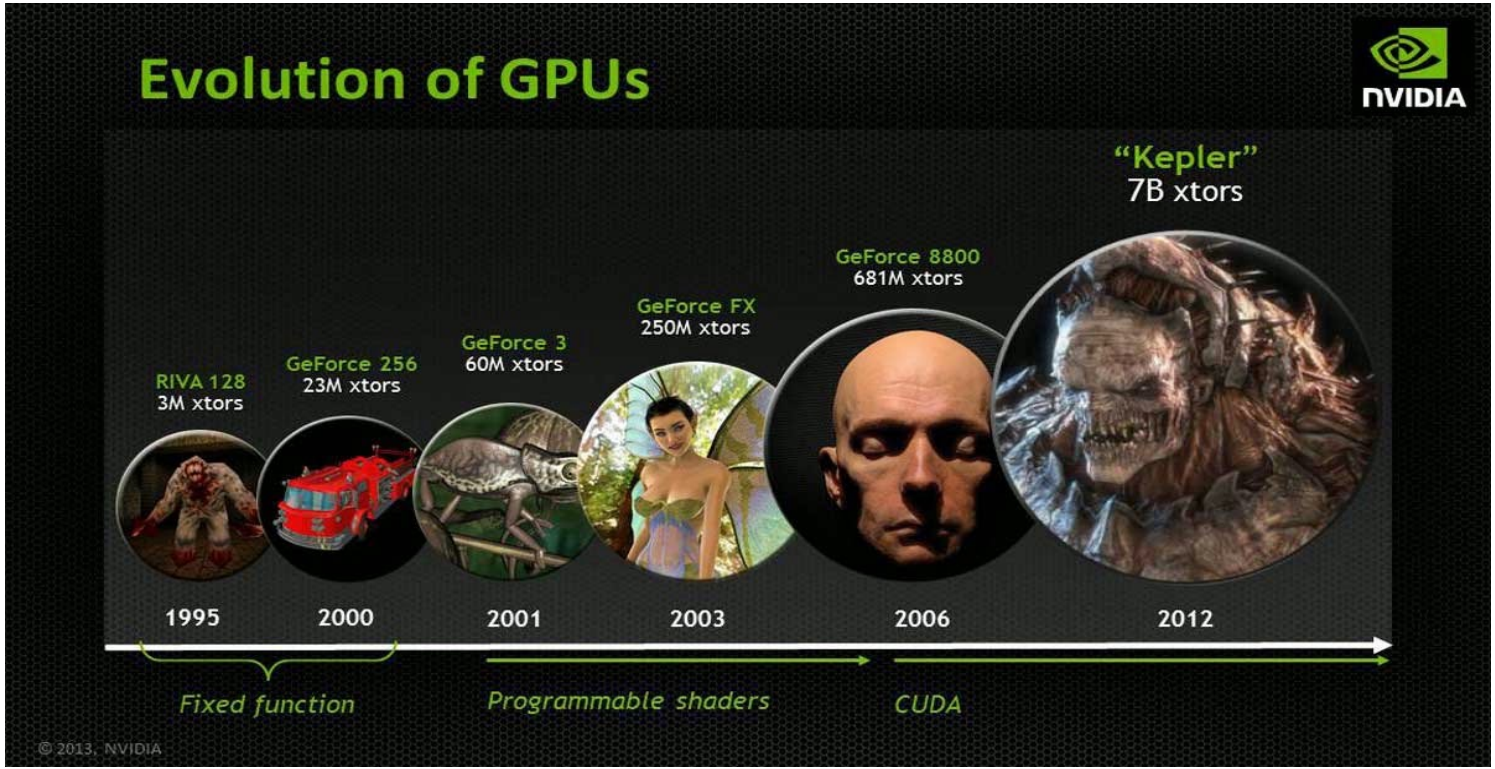

Titan RTX


RTX 3090ti

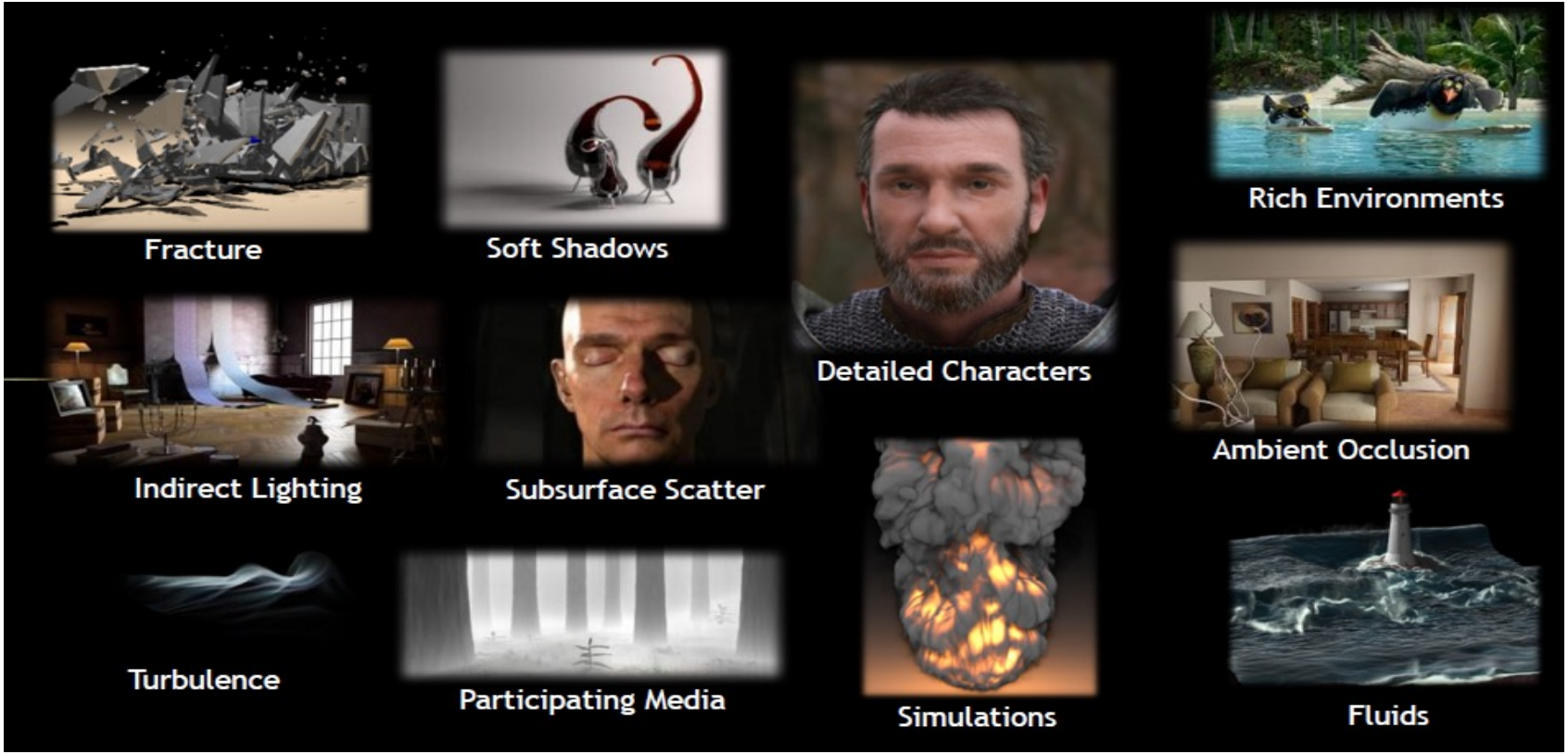
Evolution des GPU



Evolution des GPU



Evolution des GPU



Pour conclure sur les cartes graphiques

Jeux vidéo et films d'animation ont démocratisé la synthèse au grand publique (depuis 2000) : Ubisoft, Pixar, DreamWorks, etc.

GPU (NVIDIA, ATI) ont fait avancer les capacités de calculs : CG puis maintenant pour l'IA

Les grands industriels font avancer la recherche : Google et Facebook s'affrontent sur la RV à coup de rachat de startups du domaine

Algorithmes de Deep Learning modifient la vision par ordinateur, la synthèse d'images

La Robotique arrive … avec tous les problèmes de vision et d'animation que cela comporte

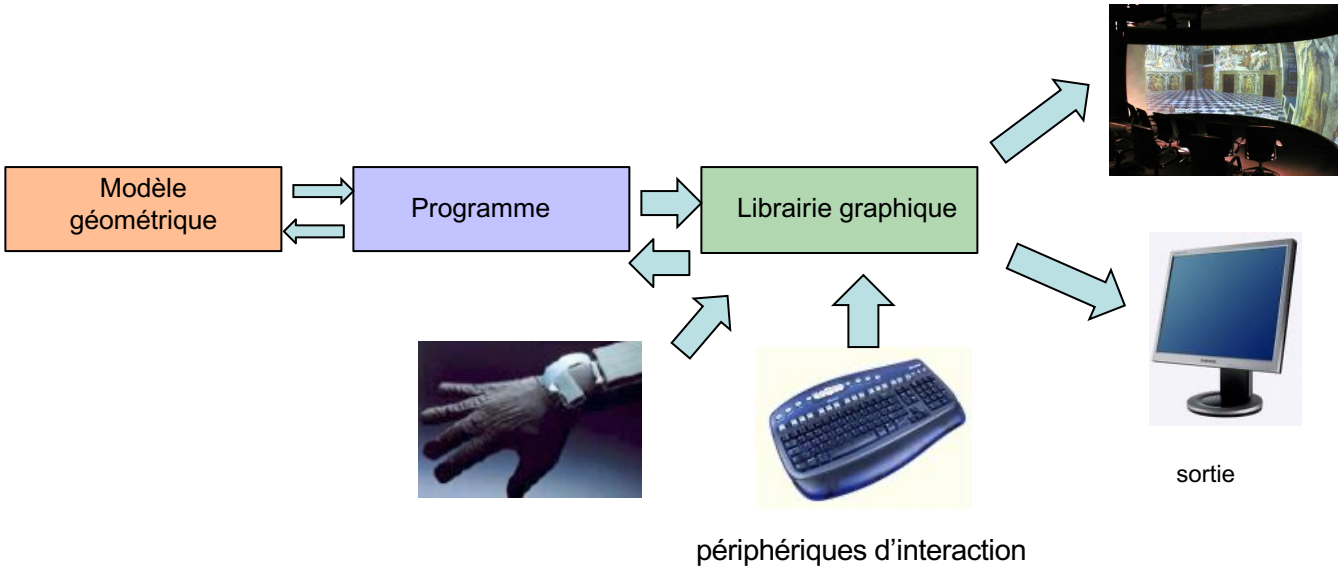
Récapitulatif des notions abordées durant ce cours

- **Informatique et image** : différents domaines de recherche et d'applications
  - Traitement d'images, synthèse d'images, Réalité Augmentée, Réalité Virtuelle
  - CAO, effets spéciaux, jeux vidéo, applications médicales, etc.

- **Création des images virtuelles s'effectue en plusieurs étapes**
  - Modélisation, animation, visualisation, rendu
  - Différentes étapes du pipeline graphique
  - Chacune des étapes est complexe, une expertise à part entière
  - Beaucoup de domaines de recherche

A noter que de plus en plus d'IA dans ces étapes...

Et ensuite, comment interagir avec l'image créée ?



C'est la Réalité Virtuelle – cela sera l'objet du prochain CM