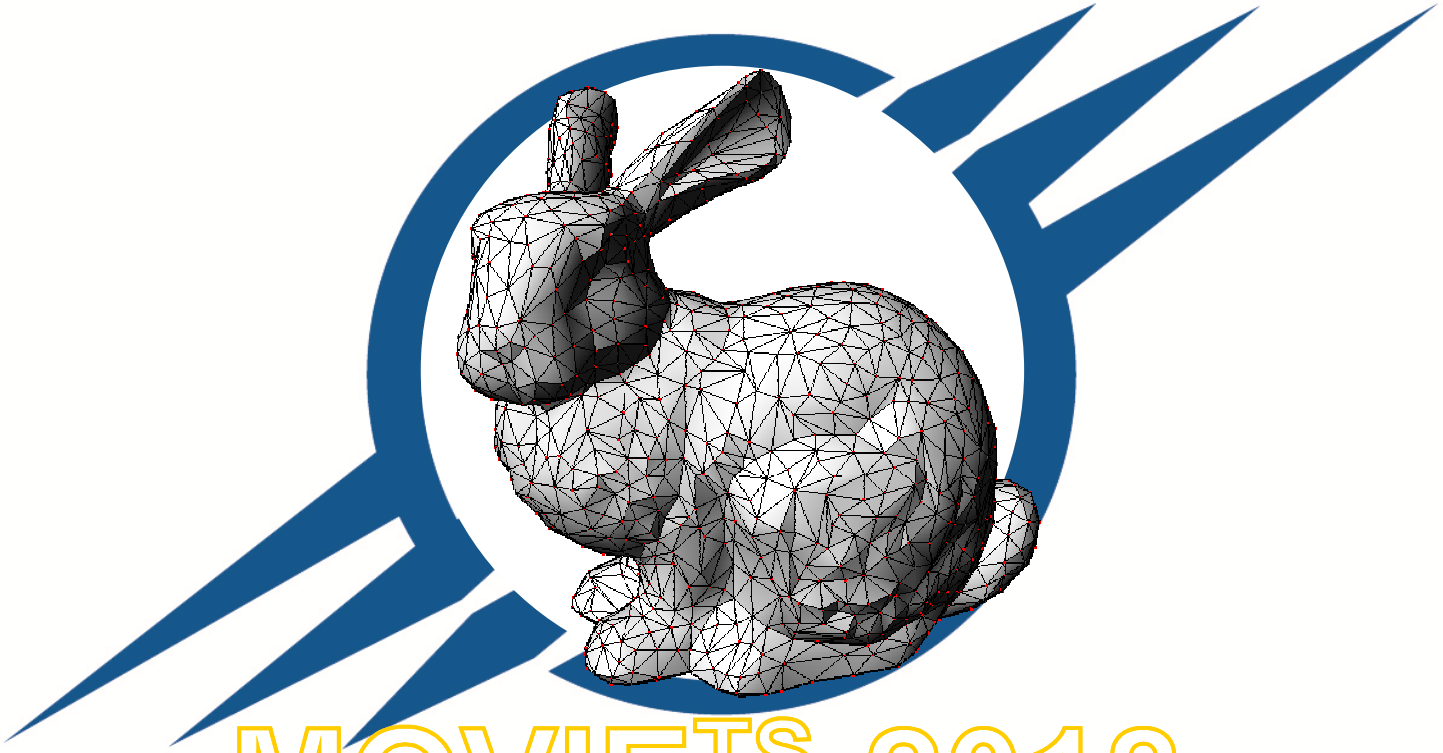
Modélisation des mondes virtuels (1)

MINES PARIS ParisTech



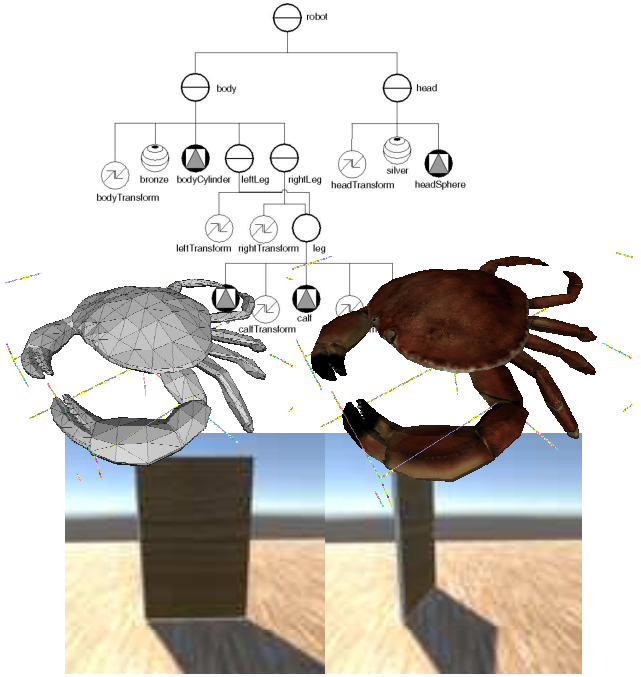
MOVIE^{TS}-2018

http://people.mines-paristech.fr/olivier.stab/geomodeling.pdf

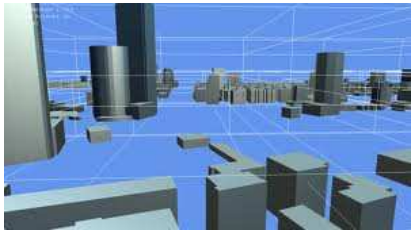
http://people.mines-paristech.fr/olivier.stab/MOVIE

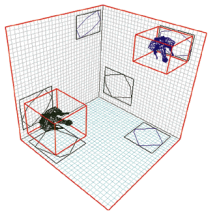
Les cours

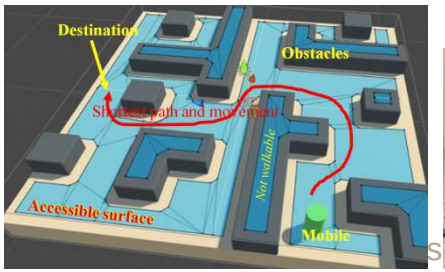
- De la scène à l'image
  - Structure d'une scène
  - Du triangle à l'image
- TP : prise en main



- Modélisation géométrique
- Gestion de l'espace
- Triangulations
- TP : navigation







De la modélisation géométrique aux mondes virtuels

1960

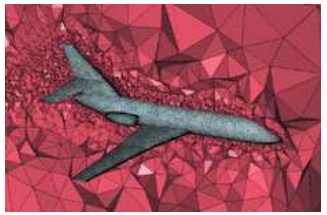
CFAO : automobile, aéronautique…

Amiga

PovRay

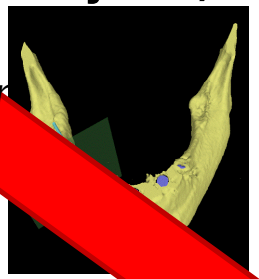
1990

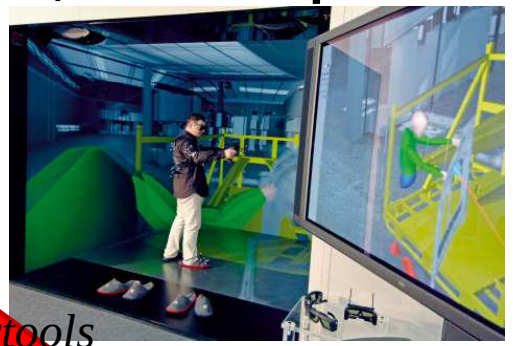
Le calcul et la visualisation scientifique



Les jeux, la RV, la robotique…

Blender





Virtools

concevoir, fabriquer, calculer, simuler, visualiser, manipuler…

Internet, TV, cinéma, art et culture, mode…

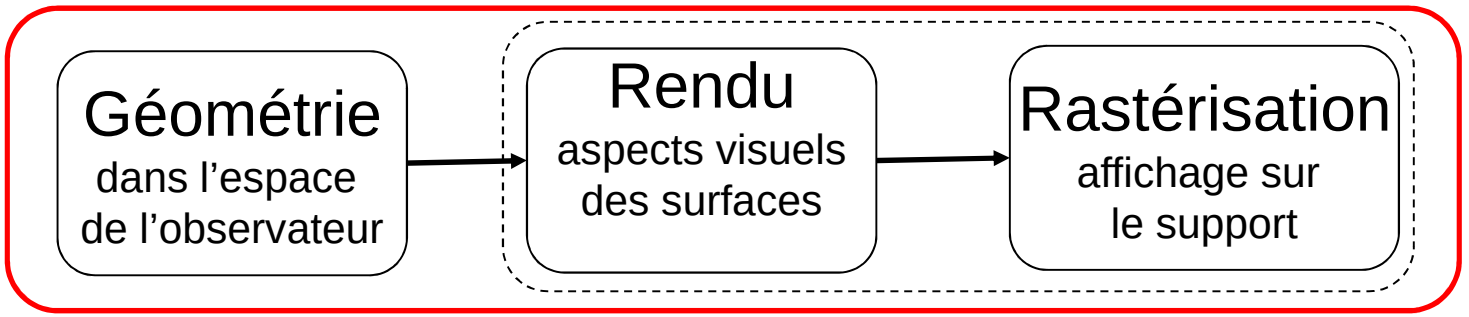
Unity *Unreal*

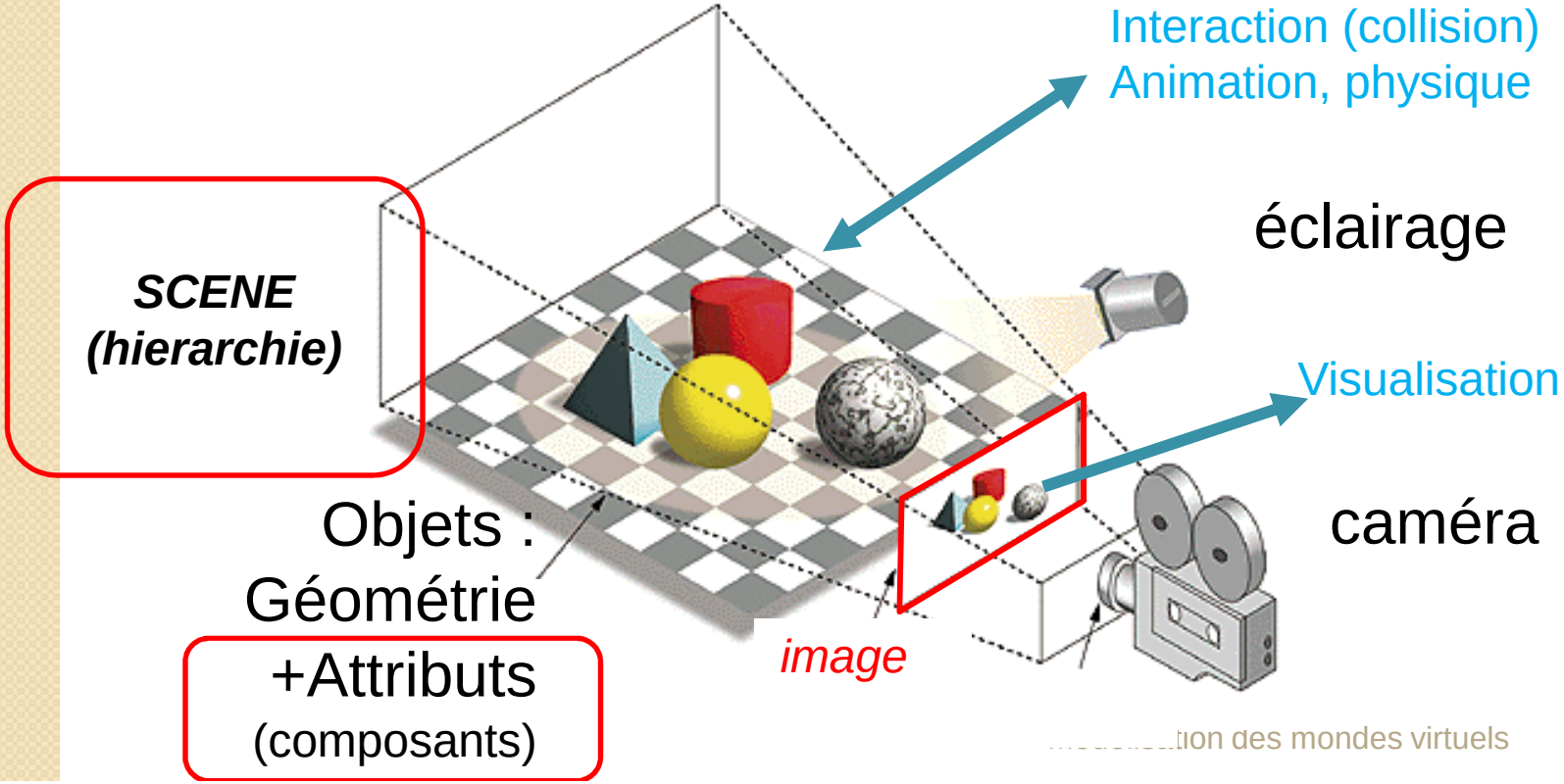
2018

Scan et impression 3D

Mondes virtuels : les besoins

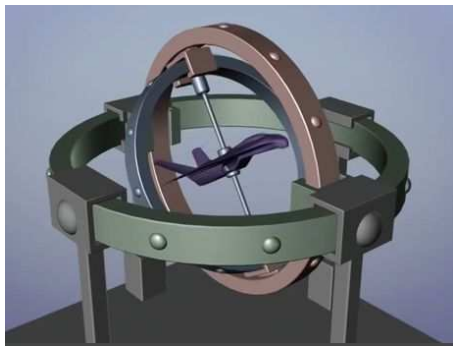
Moteur de rendu, moteur physique, moteur de jeu…

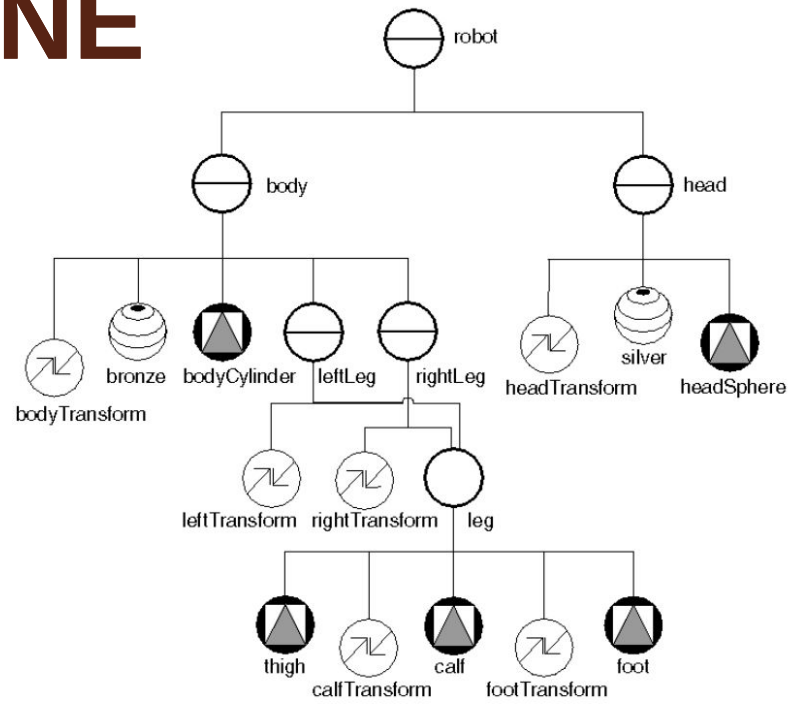




1. LA SCÈNE

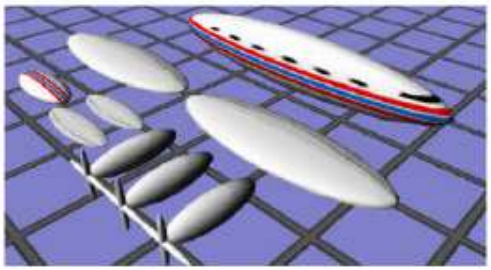
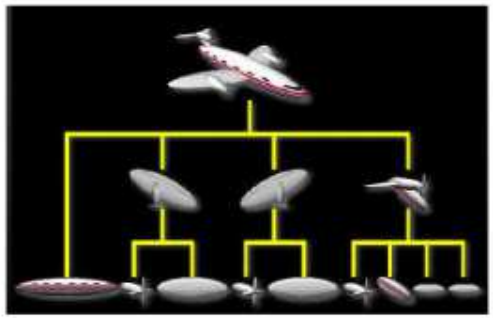
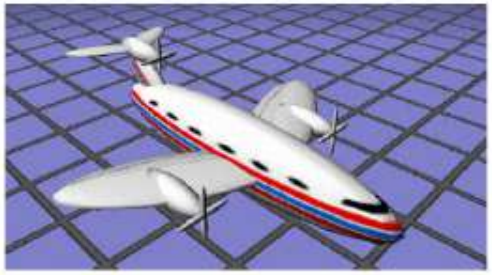
$$R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$





1.1 Organisation d’une scène

- Le « Scene Graph » : modèle hiérarchique (1988 PHIGS, 1992 Inventor, 1991 VRML.1… X3D)
- Exemples : VRML, Inventor, Java3D, OpenSG, SVG, Three.js (WebGL), …
- Les moteurs « engine » (parcours du scene graph) pour le Rendu, l’Animation, physique…
- Logiciels : Ogre3D, Cortona, Unity, CorelDraw…

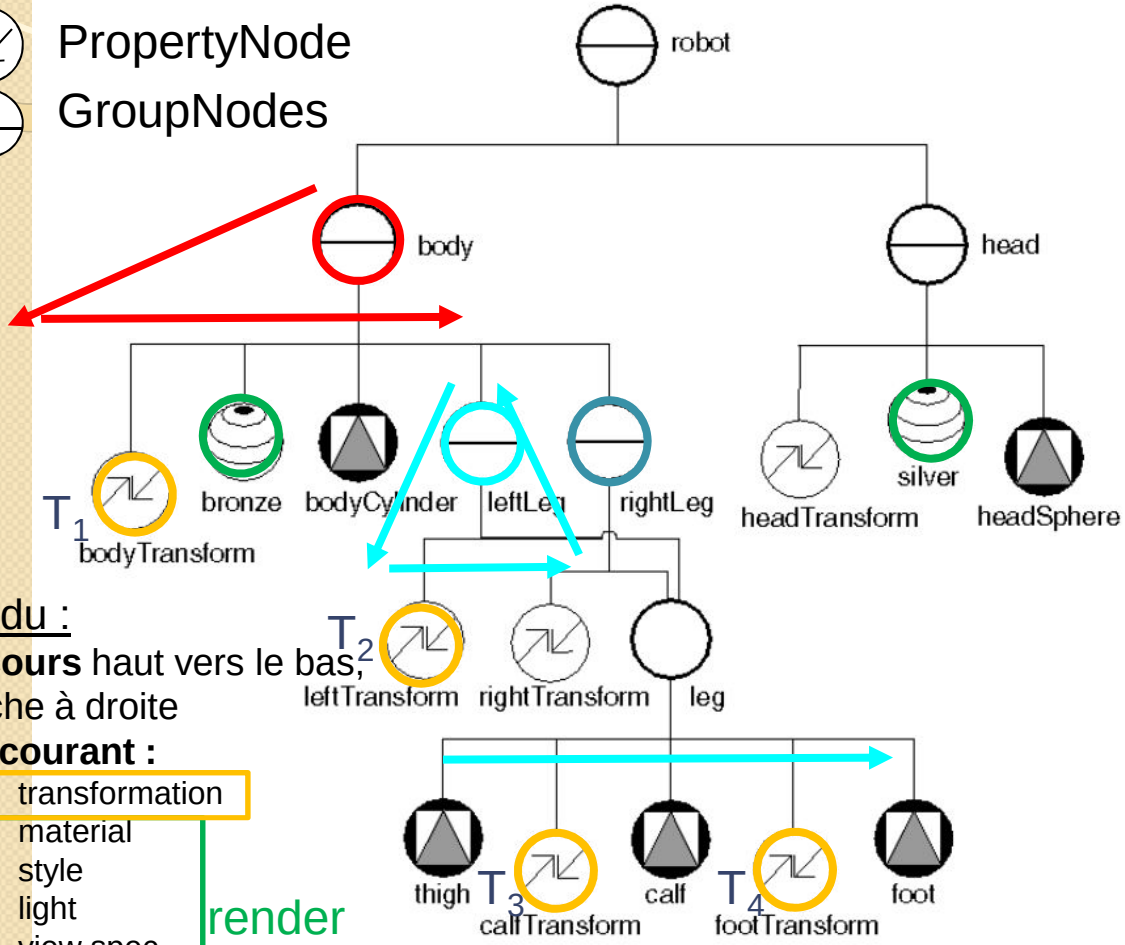




SG : le parcours des engines

ShapeNode

PropertyNode

GroupNodes



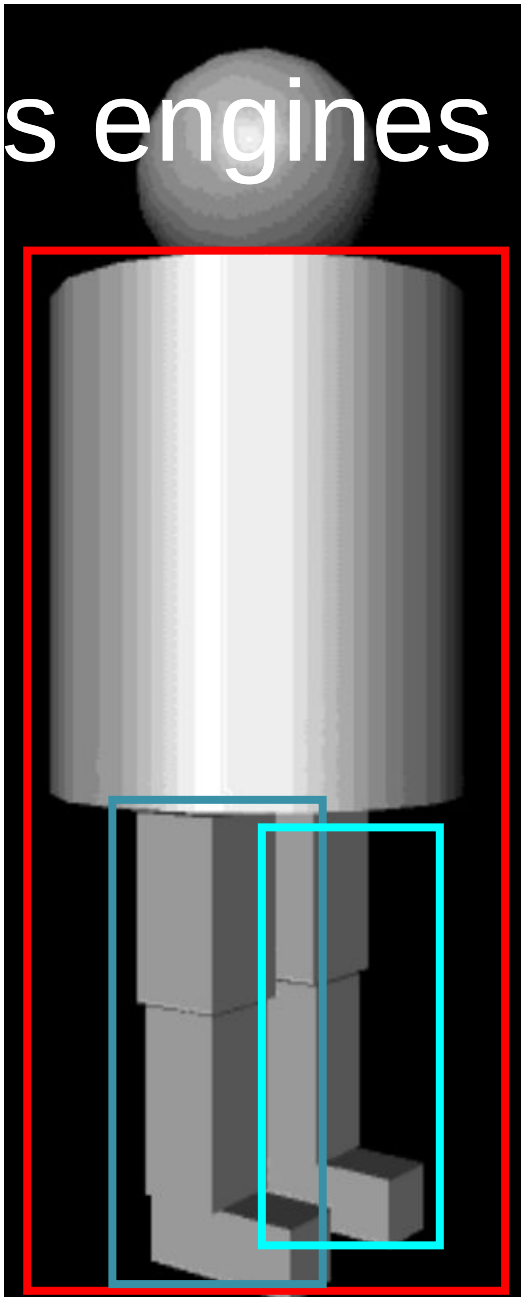
Rendu :

parcours haut vers le bas, gauche à droite

état courant :

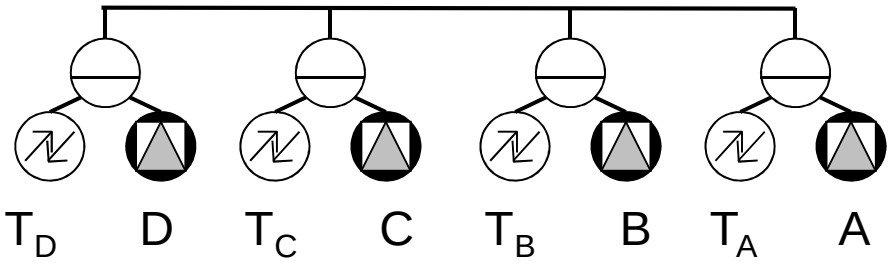
- transformation
- material
- style
- light
- view spec.
- …

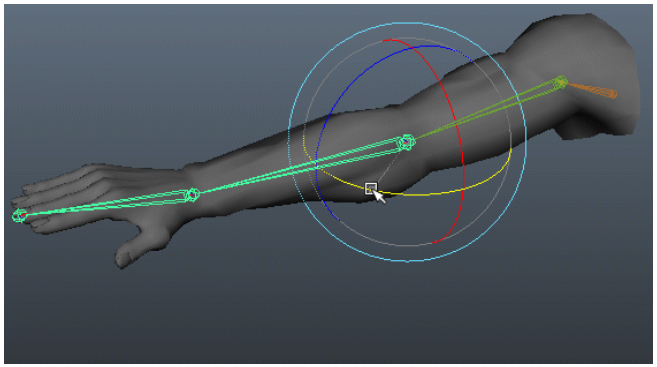
render

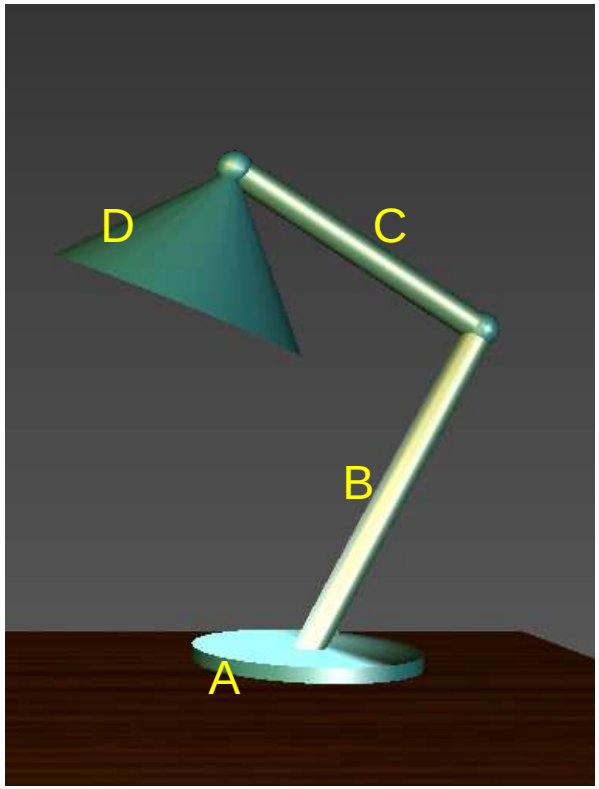


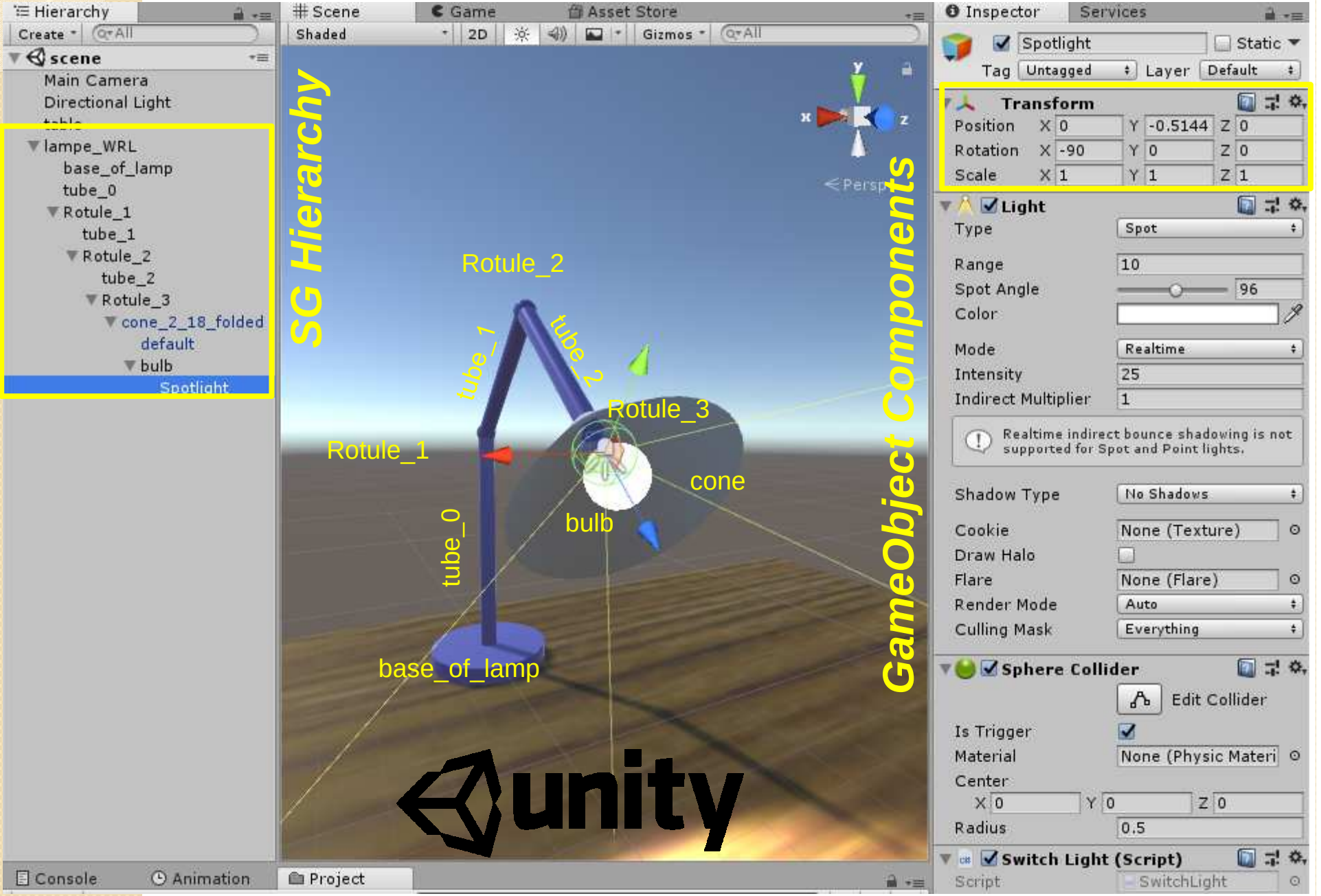
Positionnement et orientation absolus

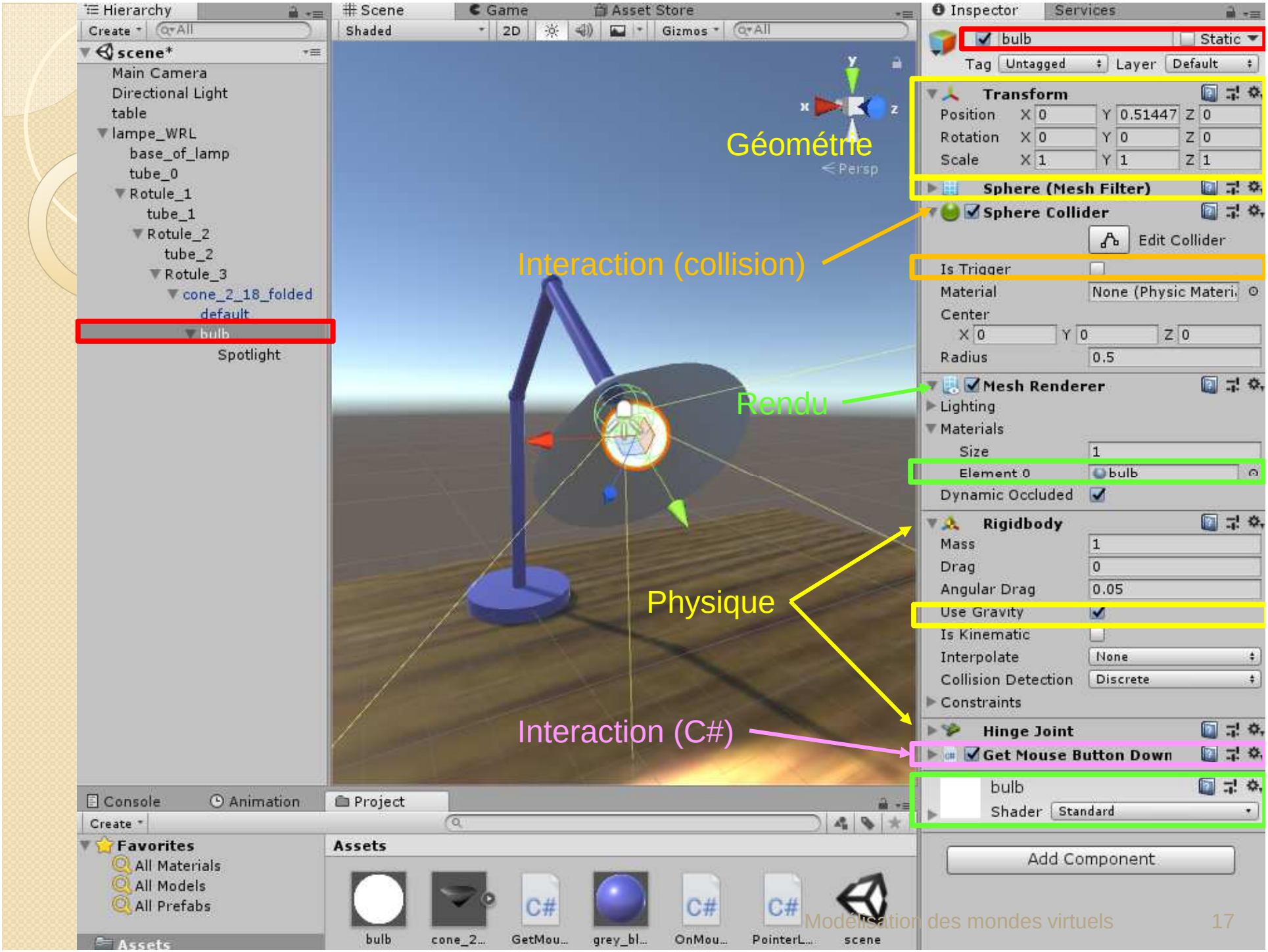
Corriger l’arbre pour pouvoir animer le bras

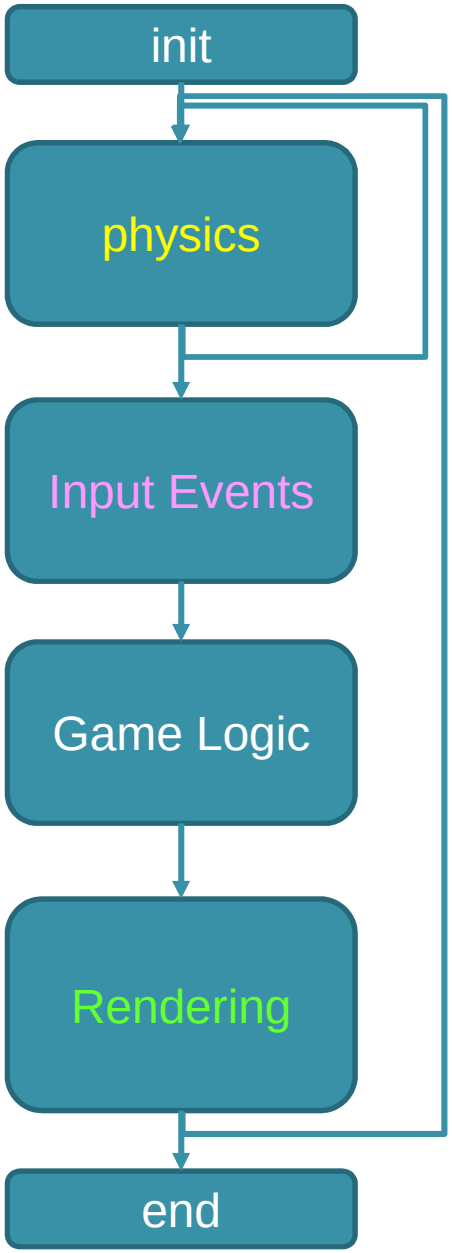







Hierarchy
Create
scene
Main Camera
Directional Light
lampe_WRL
base_of_lamp
tube_0
Rotule_1
tube_1
Rotule_2
tube_2
Rotule_3
cone_2_18_folded
default
bulb
Spotlight
SG Hierarchy
Scene
Game
Asset Store
Shaded
2D
Gizmos
Rotule_2
tube_1
tube_2
Rotule_3
Rotule_1
cone
bulb
tube_0
base_of_lamp
unity
GameObject Components
Inspector
Services
Spotlight
Static
Tag
Untagged
Layer
Default
Transform
Position X 0 Y -0.5144 Z 0
Rotation X -90 Y 0 Z 0
Scale X 1 Y 1 Z 1
Light
Type
Spot
Range
10
Spot Angle
96
Color
Mode
Realtime
Intensity
25
Indirect Multiplier
1
Realtime indirect bounce shadowing is not supported for Spot and Point lights.
Shadow Type
No Shadows
Cookie
None (Texture)
Draw Halo
Flare
None (Flare)
Render Mode
Auto
Culling Mask
Everything
Sphere Collider
Edit Collider
Is Trigger
Material
None (Physic Materi
Center
X 0 Y 0 Z 0
Radius
0.5
Switch Light (Script)
Script
SwitchLight
Console
Animation
Project


Géométrie
Interaction (collision)
Rendu
Physique
Interaction (C#)



Start()

FixedUpdate()

OnTrigger()
OnCollision()

OnMouse()

UpDate()

LateUpdate()

WaitForEndOfFrame()
OnApplicationPause()

OnApplicationQuit()

unity

C#

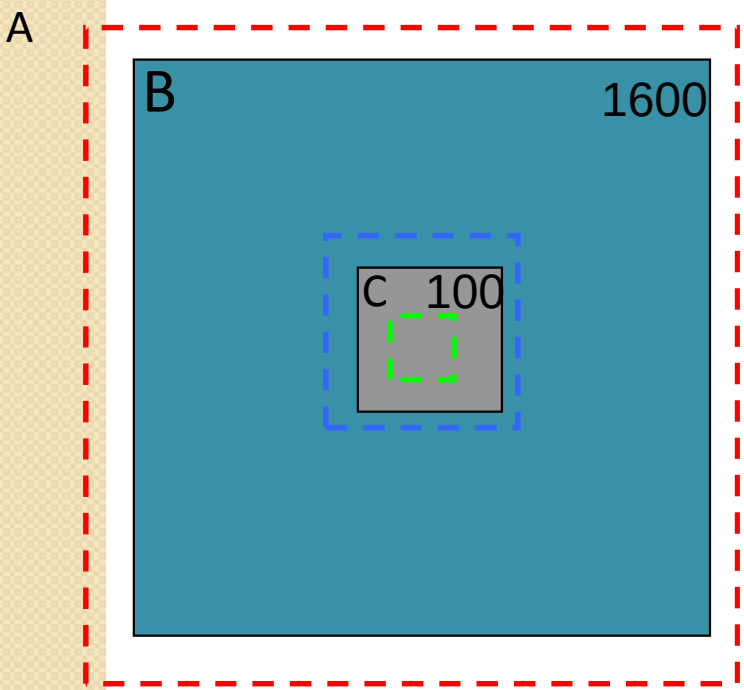
```
…using UnityEngine;

public class OnMouseOverColor : MonoBehaviour
//=========================================
{
public Color m_MouseOverColor;
private Color m_OriginalColor
private MeshRenderer m_Renderer;

void Start()
{
  m_Renderer = GetComponent<MeshRenderer>();
   m_OriginalColor = m_Renderer.material.color;
}

void OnMouseOver()
{   m_Renderer.material.color = m_MouseOverColor;}

void OnMouseExit()
{  m_Renderer.material.color = m_OriginalColor;}

void Update()
{ …
}

…
```

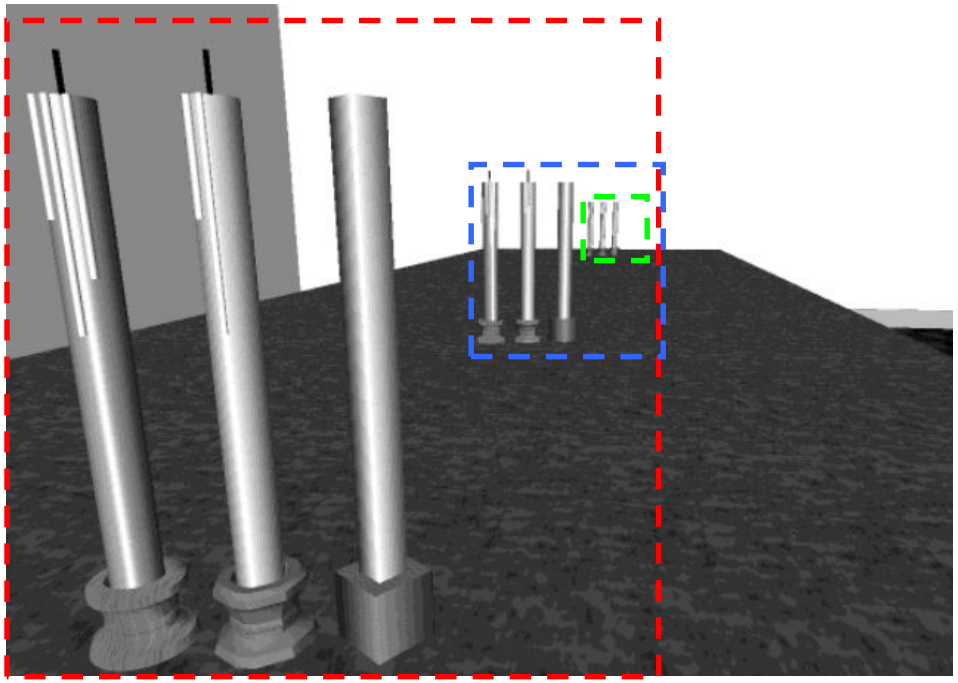
Acces via l'inspector

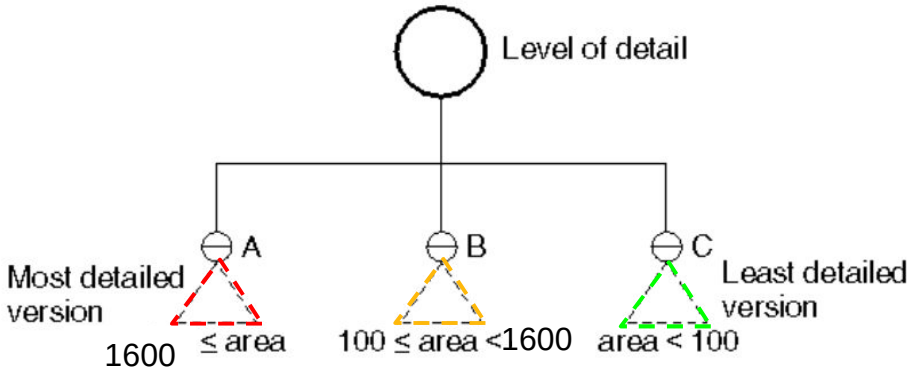
Du GameObject qui contient le script

Switch Node : Level Of Detail (LOD)

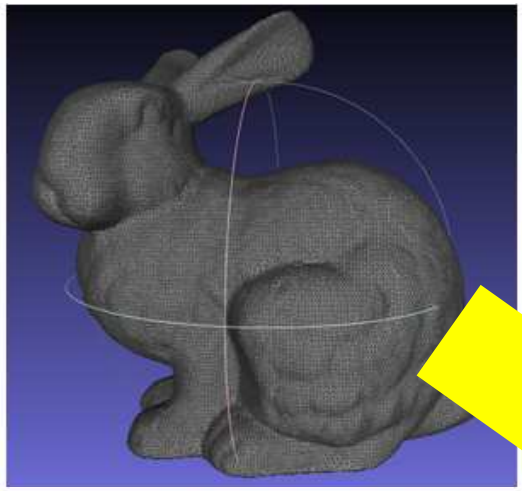
Mécanisme automatique de rendu multi-résolution







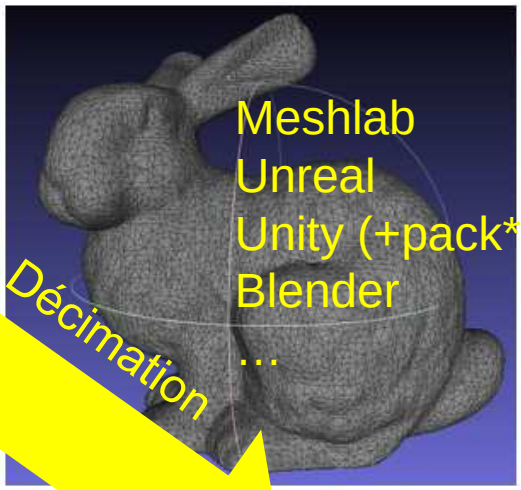
LOD dans la pratique



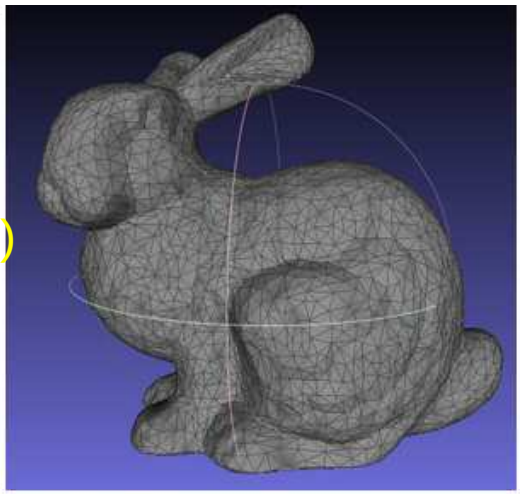
V : 34834, E: 104288, F : 69451, BE : 223, H : 5

Décimation

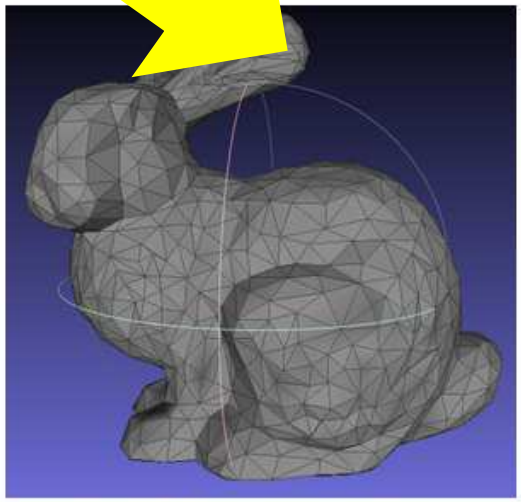
Meshlab
Unreal
Unity (+pack*)
Blender
…



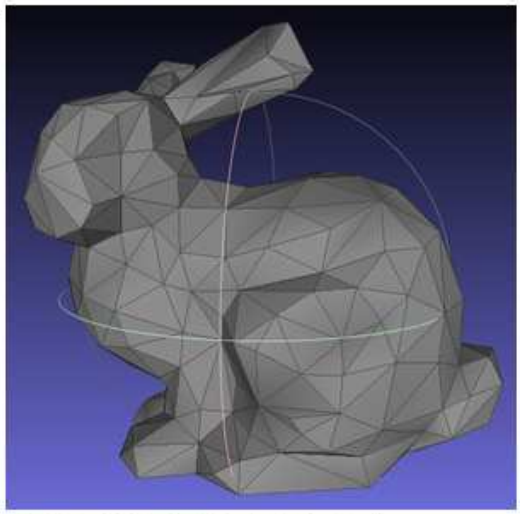
V : 1049… …34, BE : 154, H : 5



V : 3155, E: 9407, F : 6249, BE : 67, H : 5



V : 952, E: 2829, F : 1874, BE : 36, H : 5



V : 289, E: 854, F : 562, BE : 22, H : 5

MAKE YOUR GAME RUN SMOOTH - Unity LOD Tutorial

Brackeys

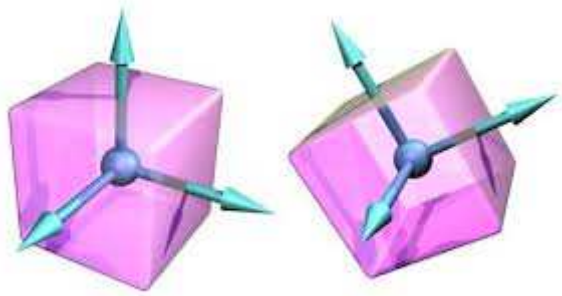
95 k vues

Brackeys : Game DEV Tutorial
https://www.patreon.com/brackeys
LOD in Blender and transfert to Unity3D :
https://www.youtube.com/watch?v=ifNyVS2_6f8

*UltimateGameTools pack

1.2 Position & orientation

- Position et orientation des objets
  - Rotations Angles d'Euler & Gimbal Lock
  - Rotations & quaternion
- Transformations (projections)
  - Coordonnées homogènes
  - Projections //, perspective…



Rotations : cas 3D

- Rotations :

Translation ?

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

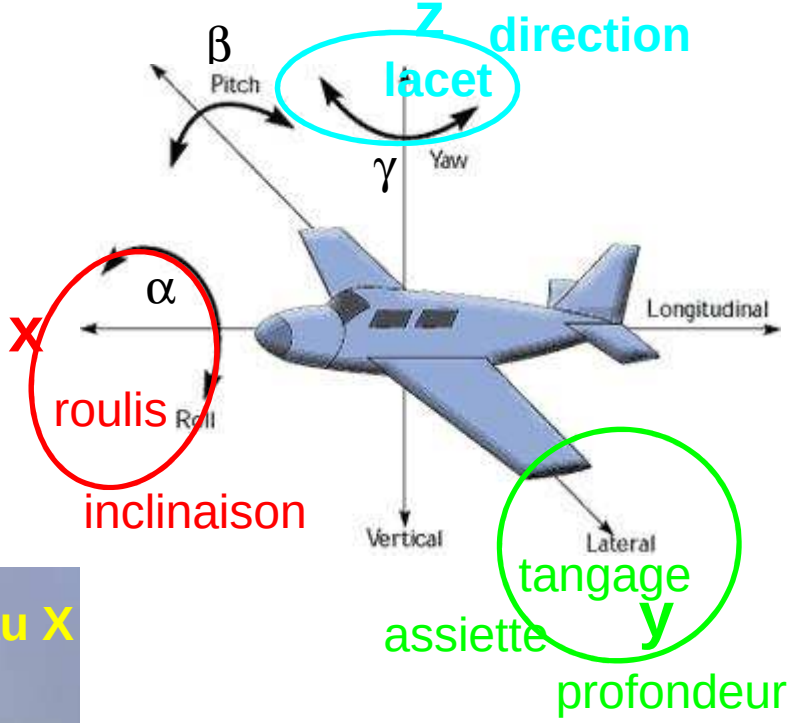
$$R_y(\psi)=\begin{pmatrix}\cos(\psi) & 0 & \sin(\psi)\\ 0 & 1 & 0\\ -\sin(\psi) & 0 & \cos(\psi)\end{pmatrix} \qquad R_x(\theta)=\begin{pmatrix}1 & 0 & 0\\ 0 & \cos(\theta) & -\sin(\theta)\\ 0 & \sin(\theta) & \cos(\theta)\end{pmatrix}$$

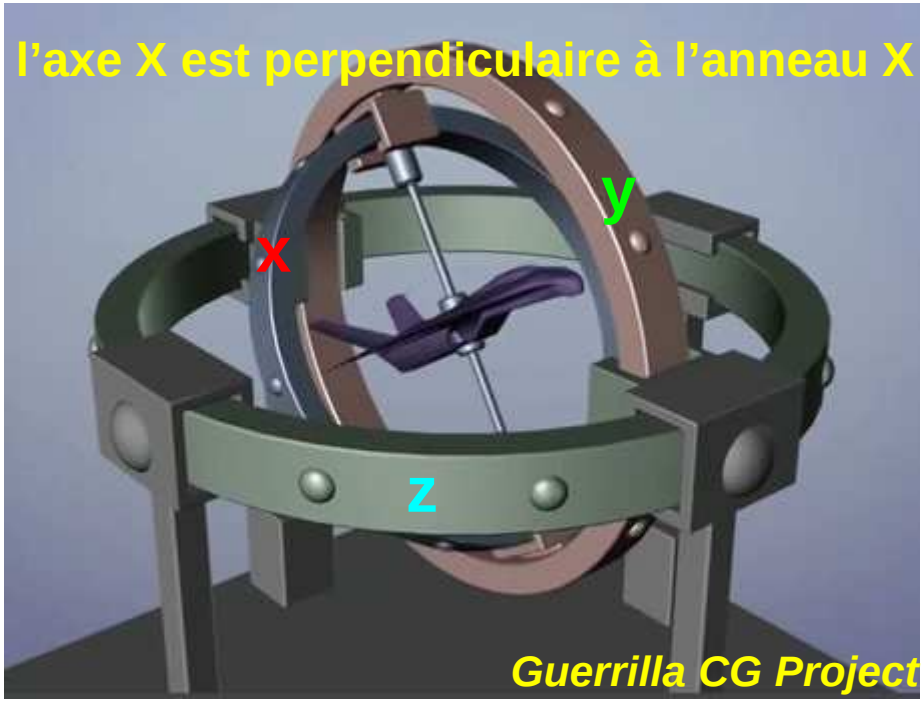
- Composition : $R_z(\alpha)R_y(\psi)R_x(\theta)$ est la rotation d'angles d'Euler.

- Multiplication des matrices (associatif mais pas commutatif)
  ➔ il faut choisir un ordre (arbitraire)

Orientation : angles d'Euler et gyroscope

La rotation autour d'un axe fait tourner les 2 autres
➔ ordre

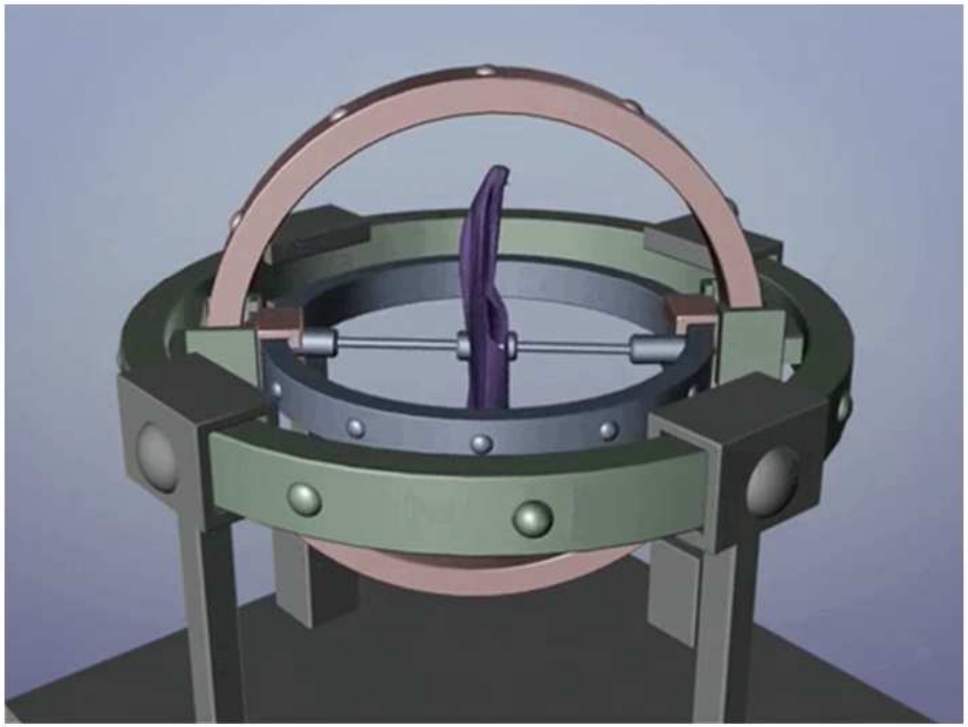




A quel ordre de rotations correspond ce gyroscope ?

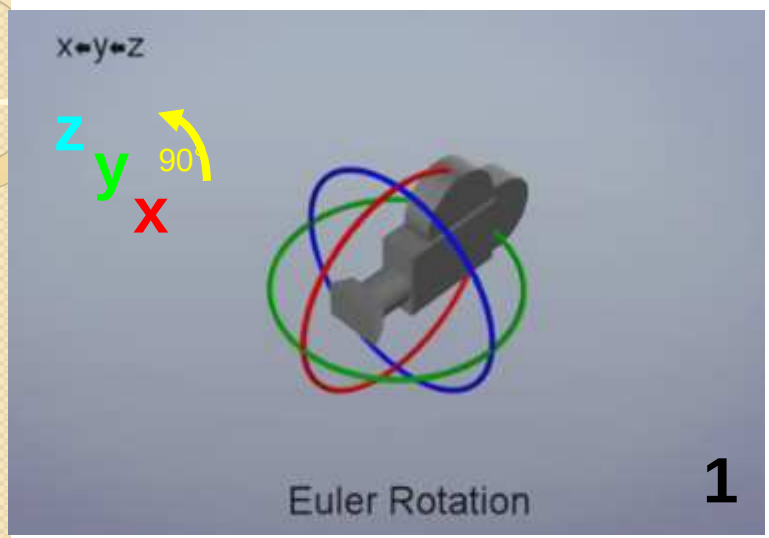
Rotation suivant Z
Rotation suivant Y
Rotation suivant X

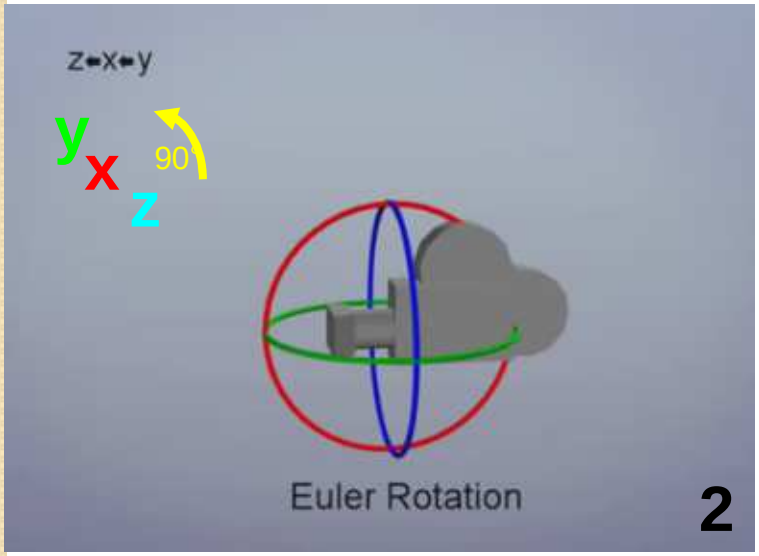
Gimbal lock (blocage de cardan)



EXTRAIT : Guerrilla CG project

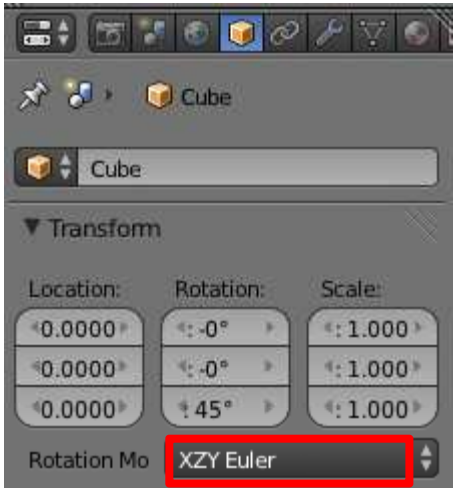
Animation d'une caméra

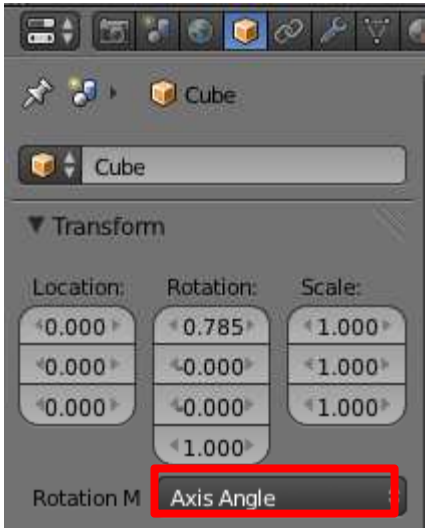


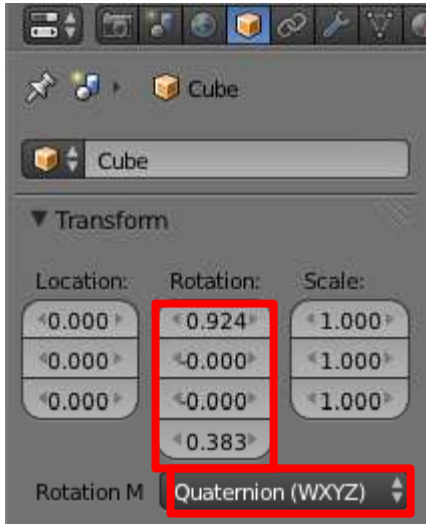


Quelle est la « bonne » caméra :
La caméra 1 ou la 2 ?

Modes de rotation dans Blender

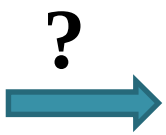


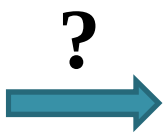




XYZ Euler :
$\alpha_x, \alpha_y, \alpha_z$
en degré (°)

$\alpha_z = 45°$



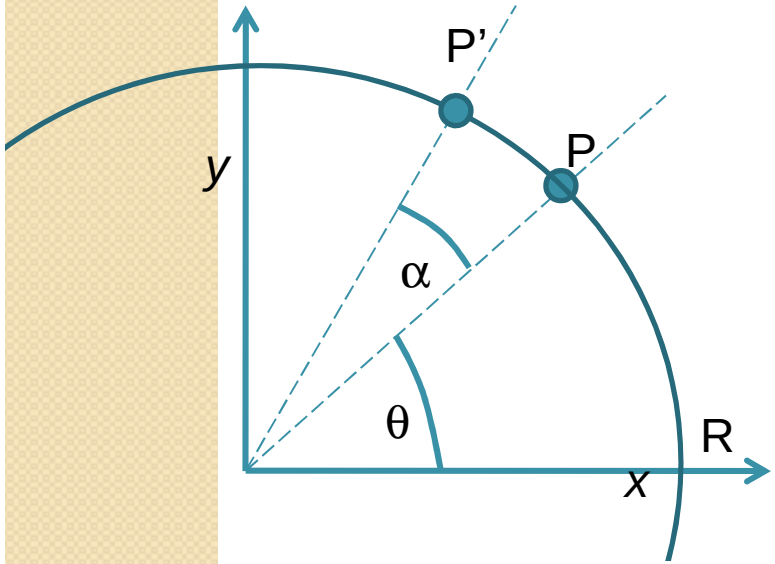


Rotation le cas 2D (rappels)

$$R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

$$P' = R_z(\alpha)(P) = \begin{pmatrix} x' = x\cos(\alpha) - y\sin(\alpha) \\ y' = x\sin(\alpha) + y\cos(\alpha) \end{pmatrix}$$

Composition : $R_{oz}(\alpha) o\ R_{oz}(\beta) = R_{oz}(\alpha+\beta)$



En coordonnées polaires

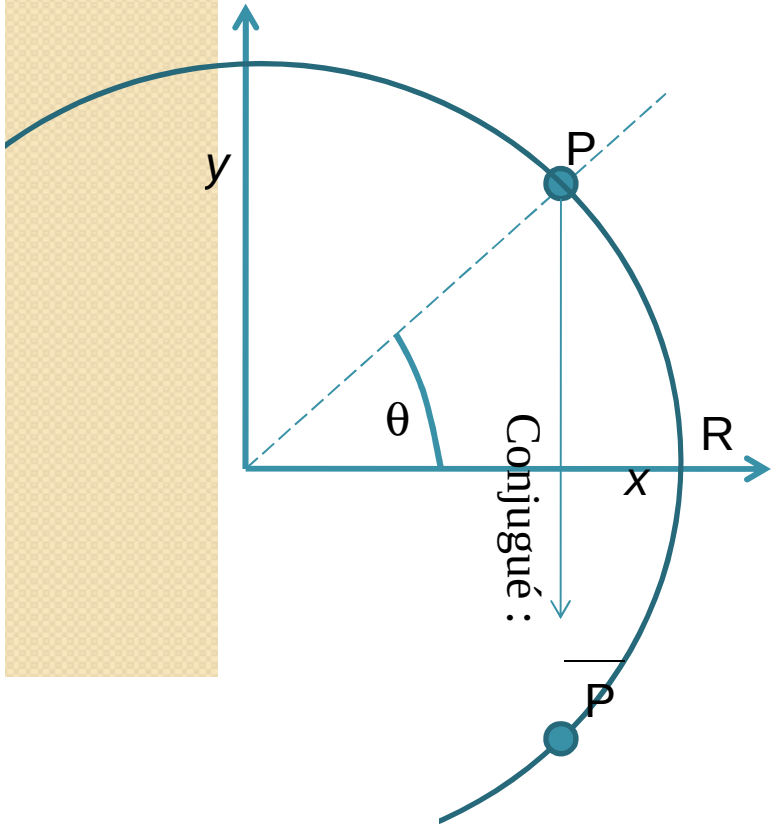
$$P = \begin{pmatrix} x = r\cos(\theta) \\ y = r\sin(\theta) \end{pmatrix}$$

$$P' = \begin{pmatrix} x' = r\cos(\alpha+\theta) \\ y' = r\sin(\alpha+\theta) \end{pmatrix}$$

Coordonnées 2D et complexes

Représentation complexe : $P\ (\ x + i\ y\ )$

$P\ (\ r\ (\ cos(\theta) + i\ sin(\theta))\ )$



Conjugué : $\overline{P}\ (\ x - i\ y\ )$ $\qquad \|P\| = \sqrt{P.\overline{P}}$

Multiplication :

$P_1.P_2 = x_1.x_2 - y_1.y_2 + i\ (x_1.y_2 + y_1.x_2)$

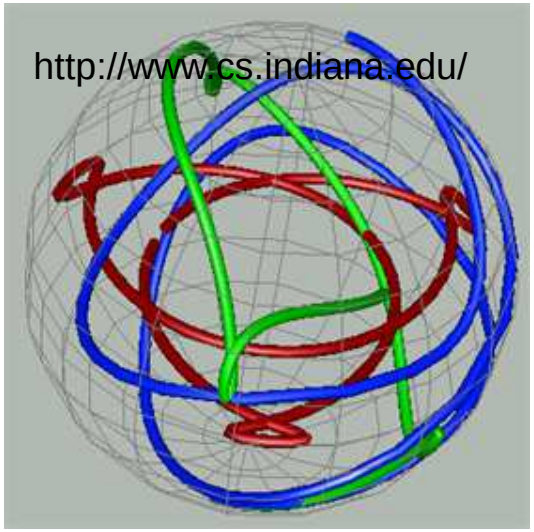
$P_1.P_2 = r_1.r_2\ (\ cos(\alpha_1+\alpha_2) + i\ sin(\alpha_1+\alpha_2)\ )$

rotation d'angle α

$\Leftrightarrow$

multiplication par un complexe d'arg α et de norme 1

Quaternion (hypercomplexes)



1. $Q = a*1 + x\,i + y\,j + z\,k = (a, \vec{V})$
2. $i^2 = j^2 = k^2 = -1$
3. $i*j=k,\ j*k=i,\ k*i=j,\ i*j= -\,j*i$ …
4. $Q_1 ** Q_2 = (a_1a_2 - \vec{V}_1.\vec{V}_2,\ a_1\vec{V}_2 + a_2\vec{V}_1 + \vec{V}_1 \wedge \vec{V}_2)$
5. $\overline{Q} = (a, -\vec{V})$
6. Norme : $\|Q\| = (a^2 + \vec{V}.\vec{V})^{1/2}$
7. Un vecteur : $Q_V = (0, V)$,
8. Une rotation d'angle α et d'axe $\vec{N}$: $Q_R = (\cos(\alpha/2),\ \sin(\alpha/2)\,\vec{N})$
9. Rotation vectorielle : $Q_{V'} = Q_R ** Q_V ** \overline{Q_R}$
10. Composition $Q_{R'} = Q_{R1} ** Q_{R2}$ …

http://la-dimension4.com/quaternions.html

1.3 Coordonnées Homogènes

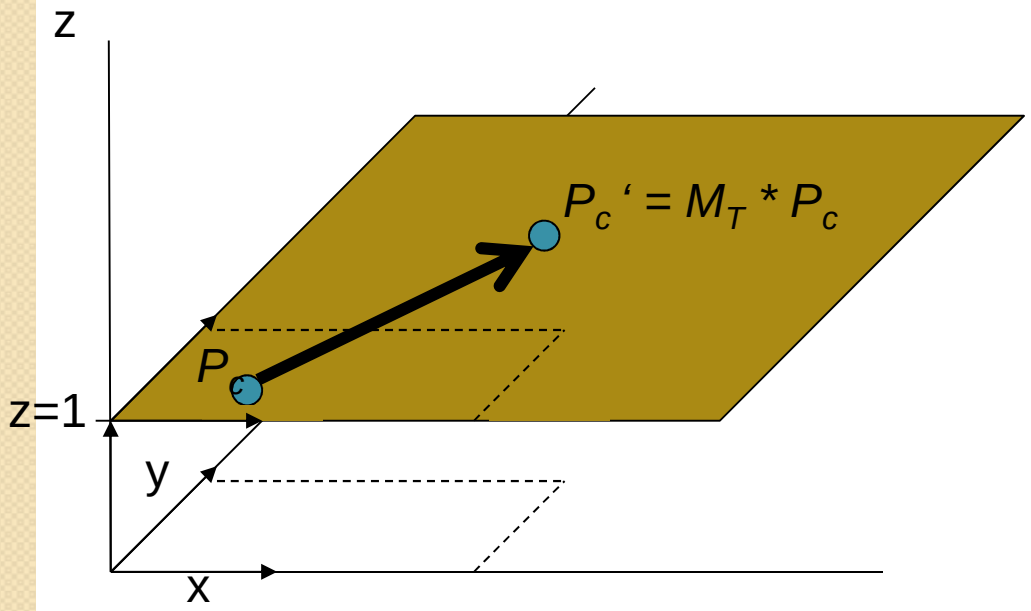
Matrice de transformation dans R^{d+1}

d=2

$$Pc = \begin{pmatrix} x \\ y \end{pmatrix} \longrightarrow Pc = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \longrightarrow Pc = \begin{pmatrix} x/1 \\ y/1 \end{pmatrix}$$

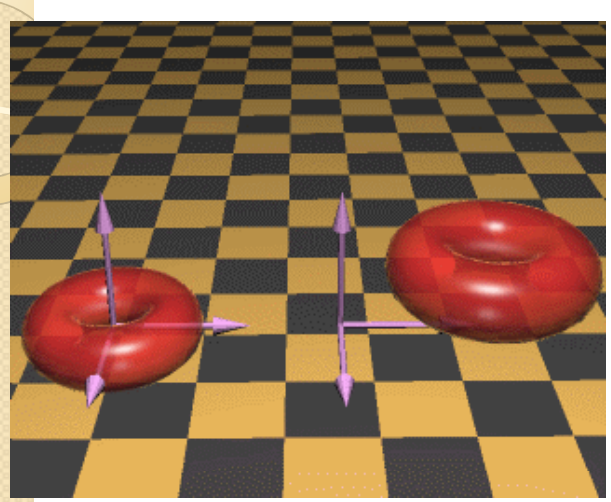
Coordonnées Homogènes
géométrie projective

$R_z(\theta) =$ Matrice de rotation ?



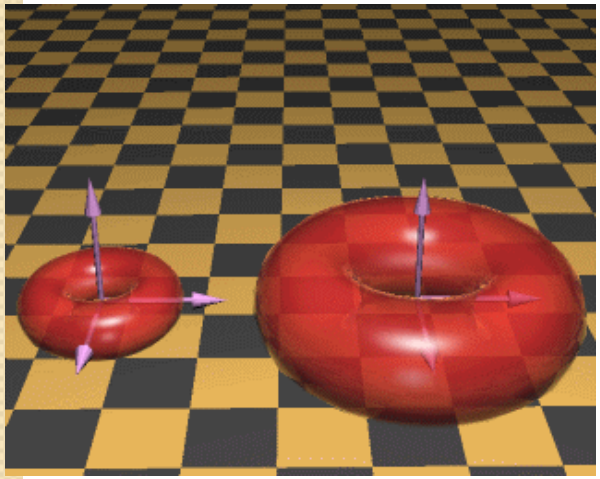
Translation ?

Translation et mise à l'échelle (d=3)



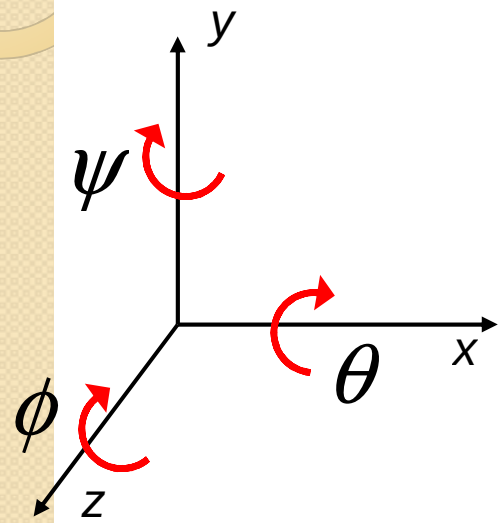
$$M_T = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad P' = M_T P$$

M_T *(x,y,z,1) = (x+t_x,y+t_y,z+t_z,1) ➔ (x+t_x,y+t_y,z+t_z)



$$M_S = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad P' = M_S P$$

Rotations (d=3)



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

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

$$P' = RP = \left(R_z . R_y . R_x \right) P$$

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

Matrice homogène de transformation

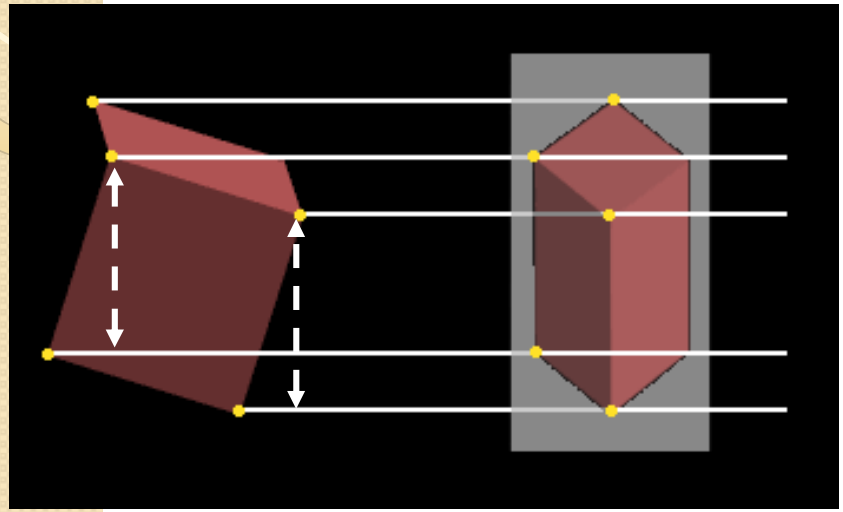
$$M_H = \begin{bmatrix} R_{3,3} & T \\ 0_3^t & 1 \end{bmatrix} \qquad P' = M_H . P$$

M_H : matrice orthonormée

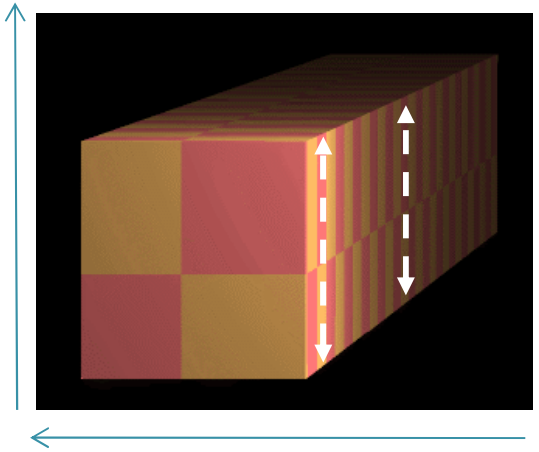
$$M_H^{-1} = \begin{bmatrix} R_{3,3}^T & -(R^T) . T \\ 0_3^t & 1 \end{bmatrix}$$

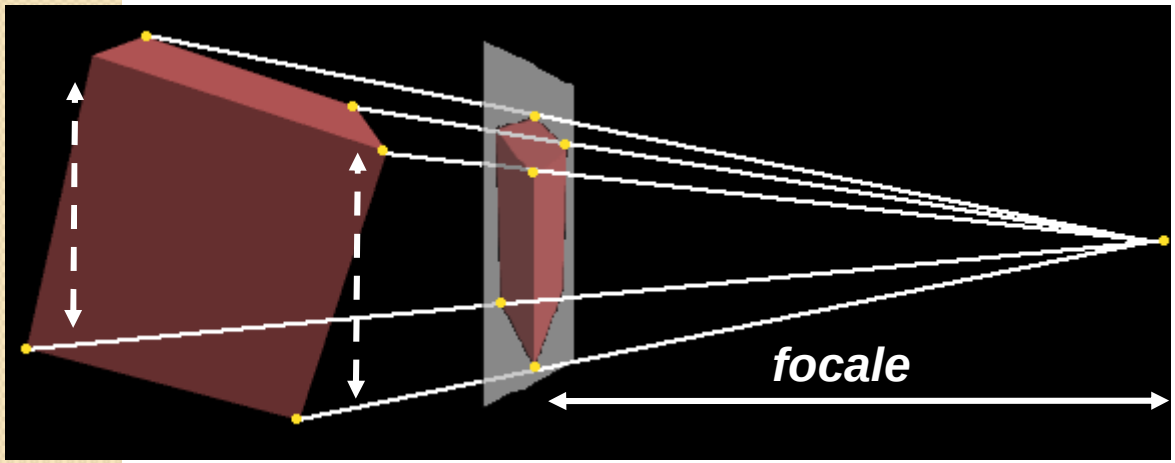
On retrouve facilement les angles de rotation à partir de M_H

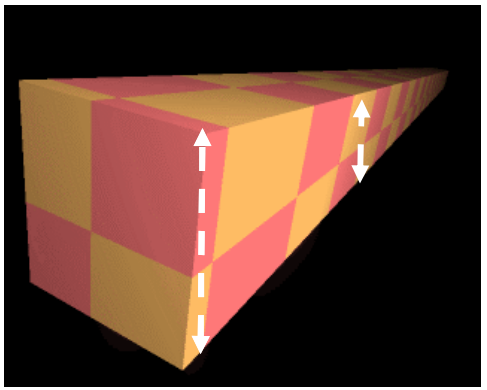
Projections



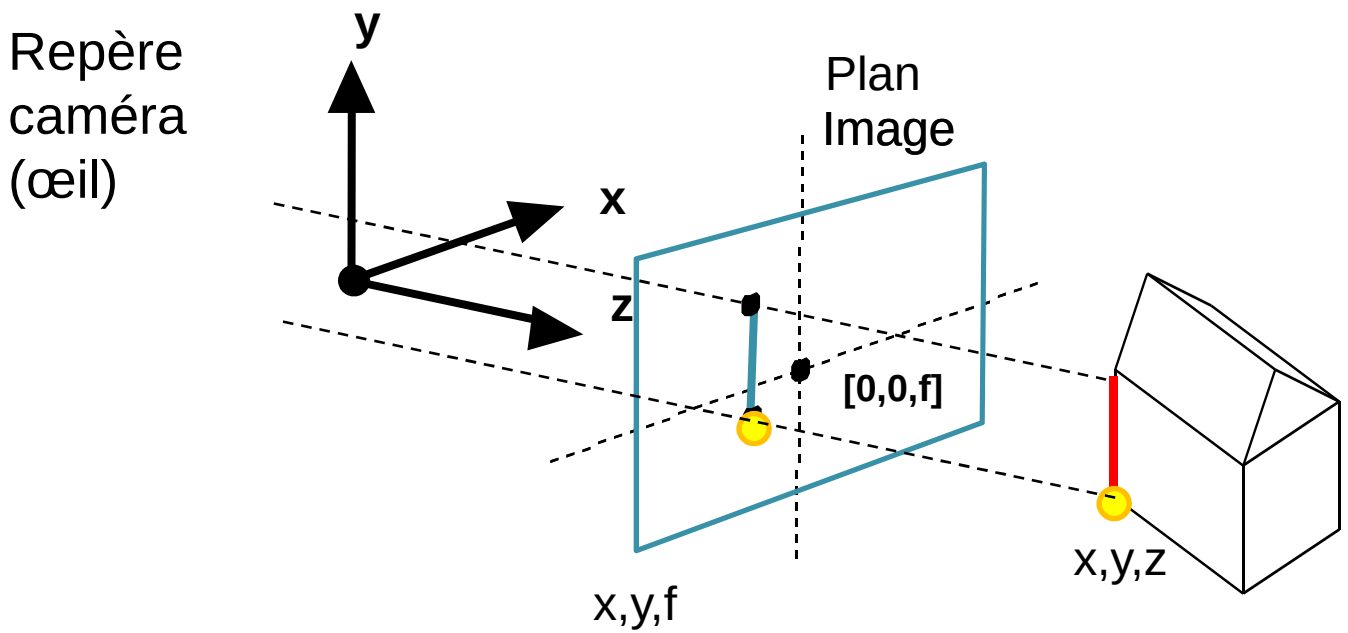
plan image



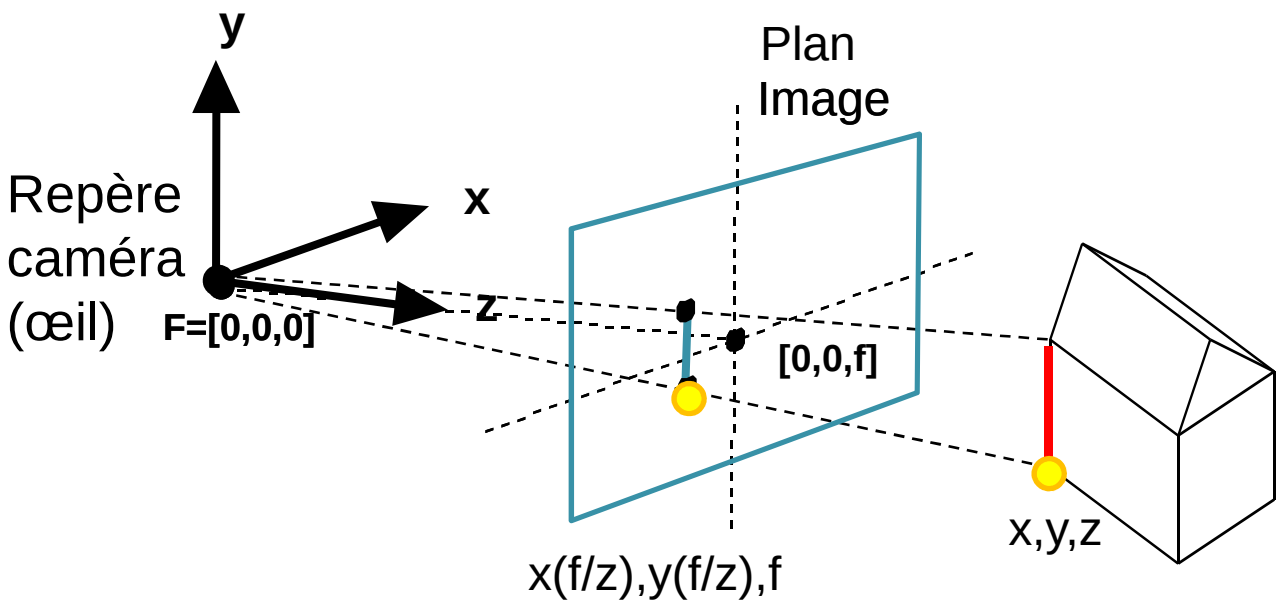


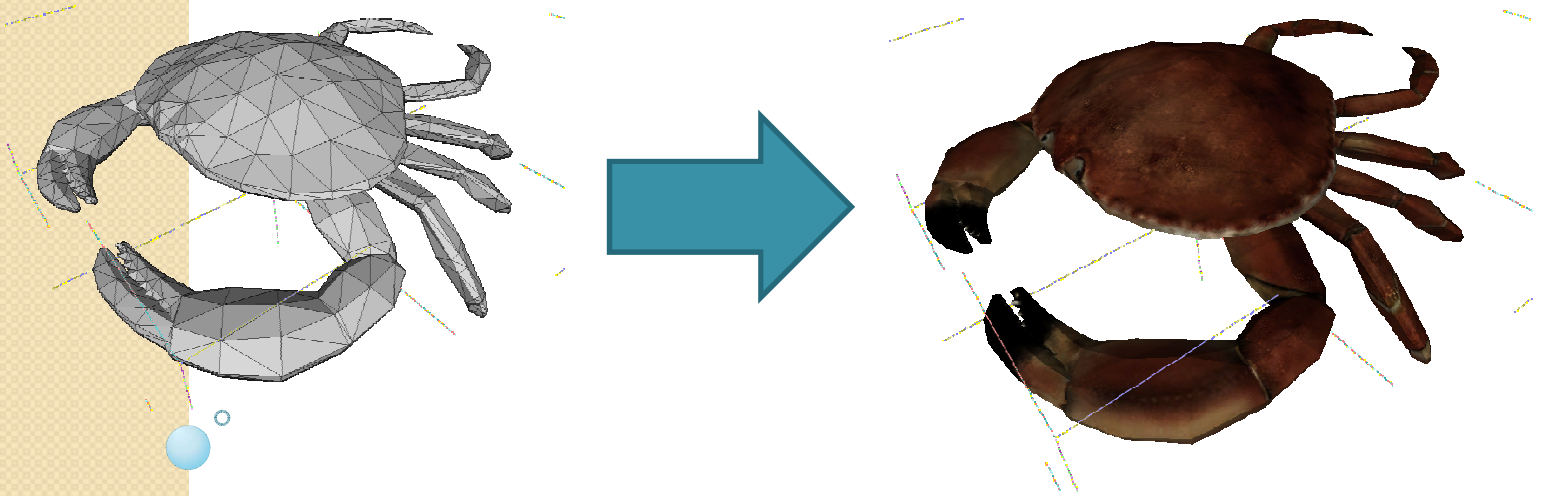


Projection orthographique

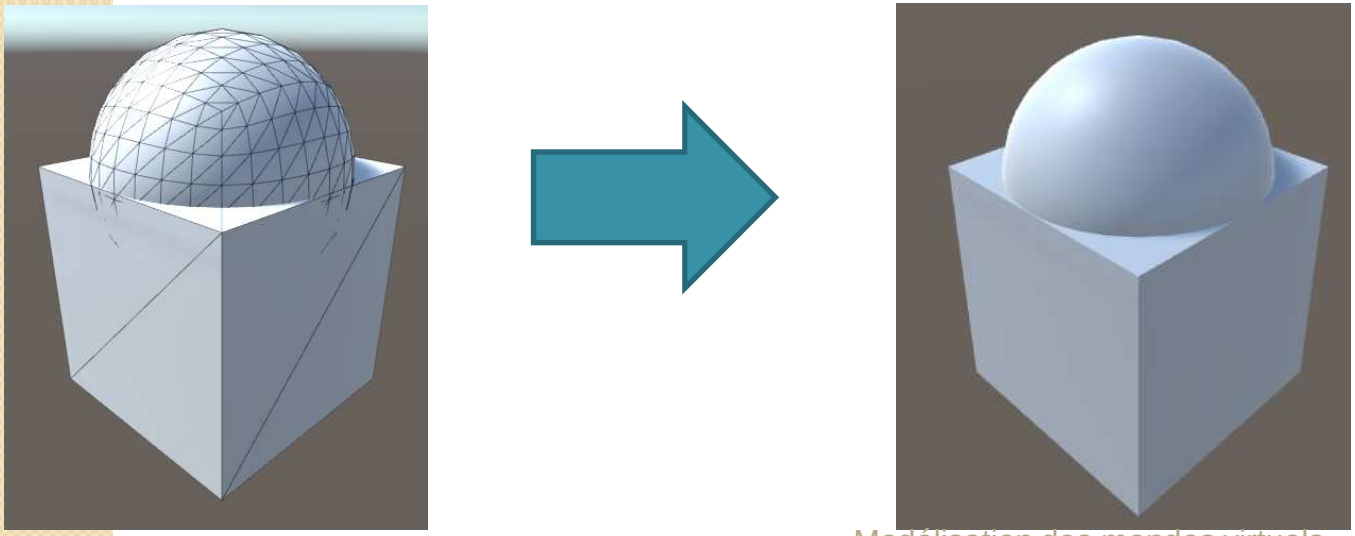


Projection perspective



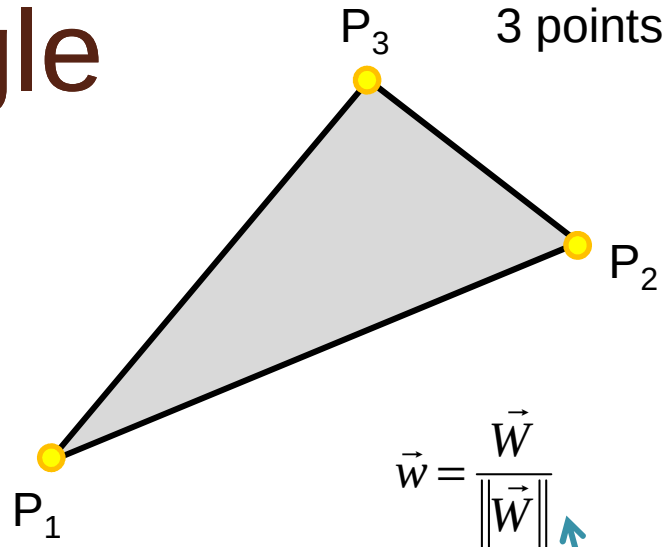


2. DES TRIANGLES À L'IMAGE



2.1 Triangle

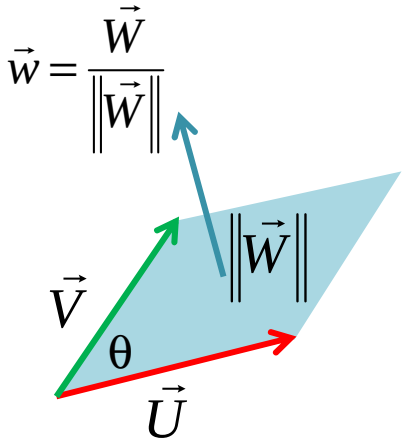
3 points ➔ plan (a priori)



Simplexe,

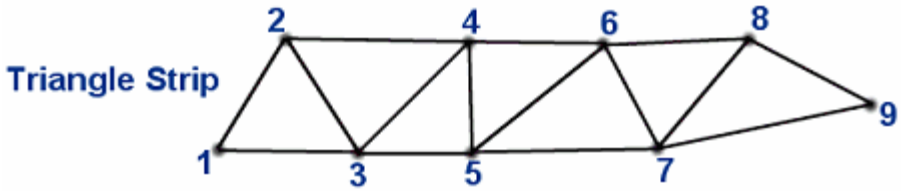
Convexe,

Normale ?



Aire ?

Point dans ?

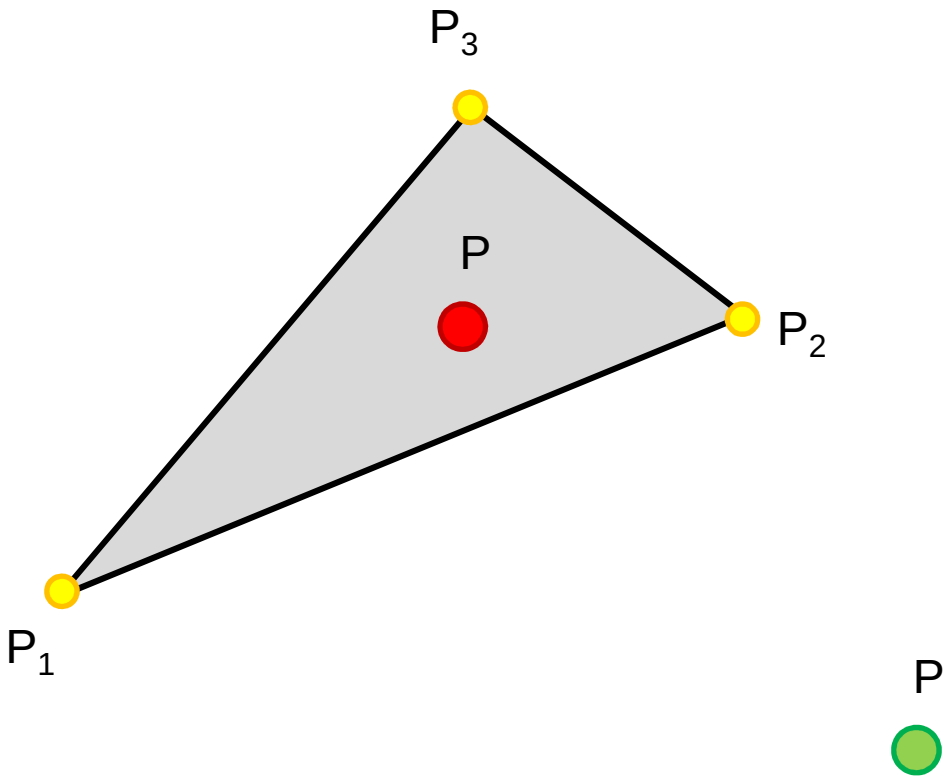


S = { (x_1,y_1,z_1), (x_2,y_2,z_2), (x_3,y_3,z_3), (x_4,y_4,z_4), (x_5,y_5,z_5), (), (), … }

Point dans un triangle ?

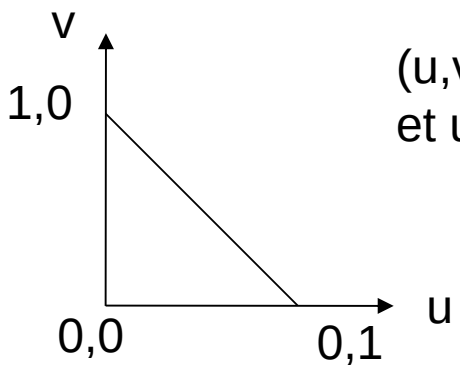
P dans Triangle(P_1,P_2,P_3) ?

En 2D ?
En 3D ?

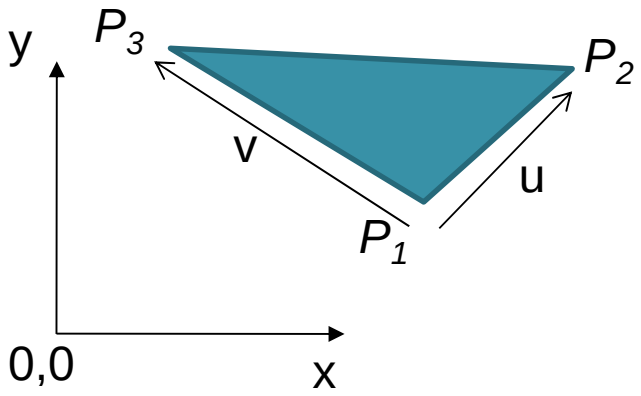


Et si on ne connait pas l'orientation du triangle ?

Représentation paramétrique



(u,v) dans [0:1]x[0:1]
et u+v <= 1

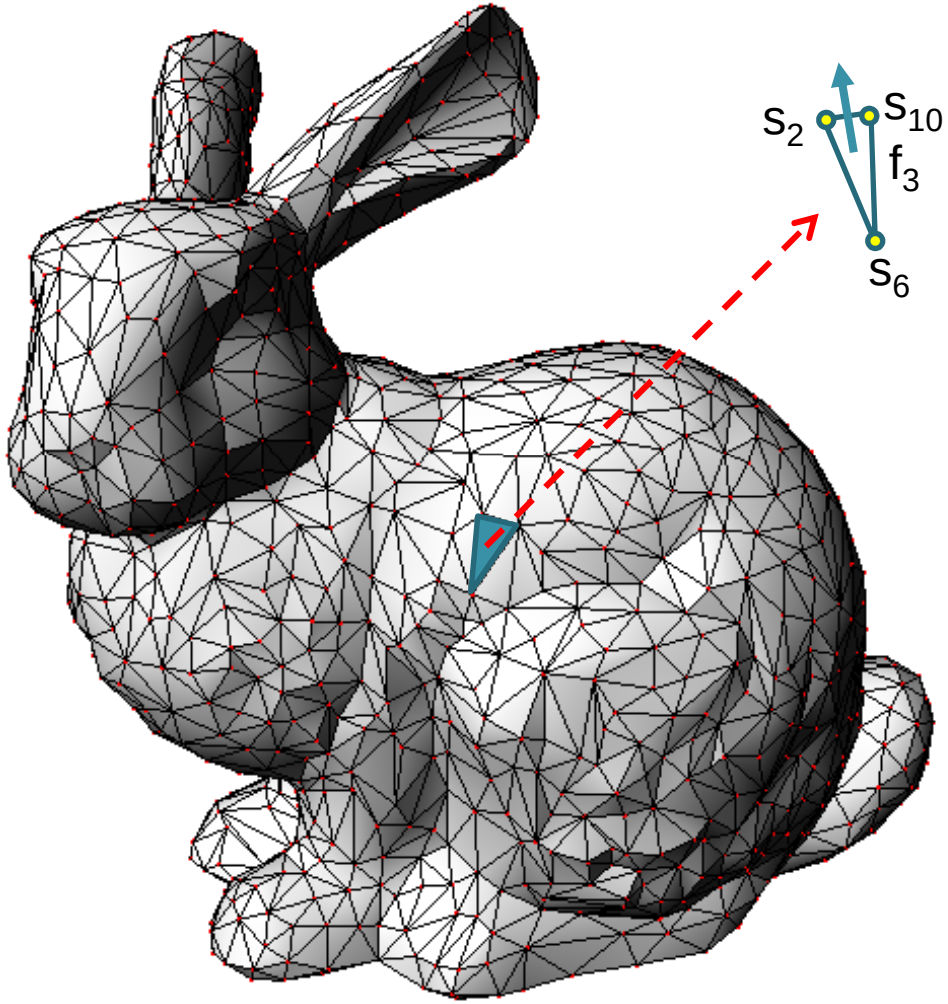


$x(u,v) = (1-u-v)*x_1 + u*x_2 + v*x_3$
$y(u,v) = (1-u-v)*y_1 + u*y_2 + v*y_3$

$$S(u,v) = \sum_{i=0}^{m} P_i \, P_{im}(u,v)$$

$$\begin{pmatrix} x(u,v) \\ y(u,v) \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} \begin{vmatrix} 1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 \\ u \\ v \end{vmatrix}$$

Triangulation frontière



$F = \{ f_1, f_2 \ldots f_6 \}$
$A = \{ a_1, a_2 \ldots a_{12} \}$
$S = \{ s_1, s_2 \ldots s_8 \}$

Frontière:
$fr(v) = \{f_1, f_2 \ldots f_6\}$

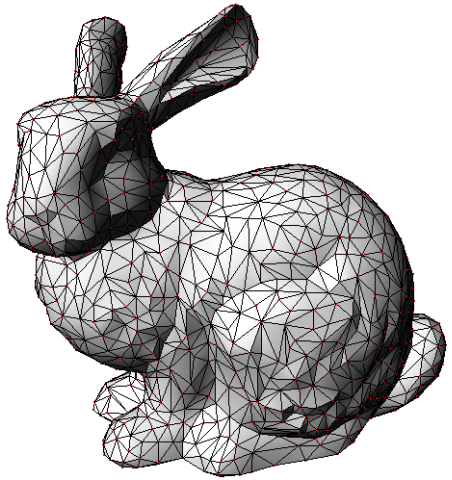
$fr(f_3) = \{a_{14}, a_6, a_{74}\}$
...
$fr(a_{14}) = \{s_2, s_6\}$
$fr(a_6) = \{s_6, s_{10}\}$

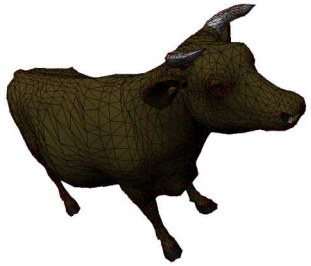
Triangulations :

$b(f_3) = \{s_2, s_6, s_{10}\}$

Card(F)-Card(A)+Card(S) = 2

Formats …





STL

```
solid name
…
facet normal ni nj nk
outer loop
vertex v1x v1y v1z
vertex v2x v2y v2z
vertex v3x v3y v3z
endloop
endfacet
…
endsolid name
```

OFF

```
OFF
# cube.off
# A cube
8 6 12
1.0 0.0 1.0
0.0 1.0 1.0
 -1.0 0.0 1.0
0.0 -1.0 1.0
1.0 0.0 -1.0
0.0 1.0 -1.0
-1.0 0.0 -1.0
0.0 -1.0 -1.0
4 0 1 2 3
4 7 4 0 3
4 4 5 1 0
4 5 6 2 1
4 3 2 6 7
4 6 5 4 7
```

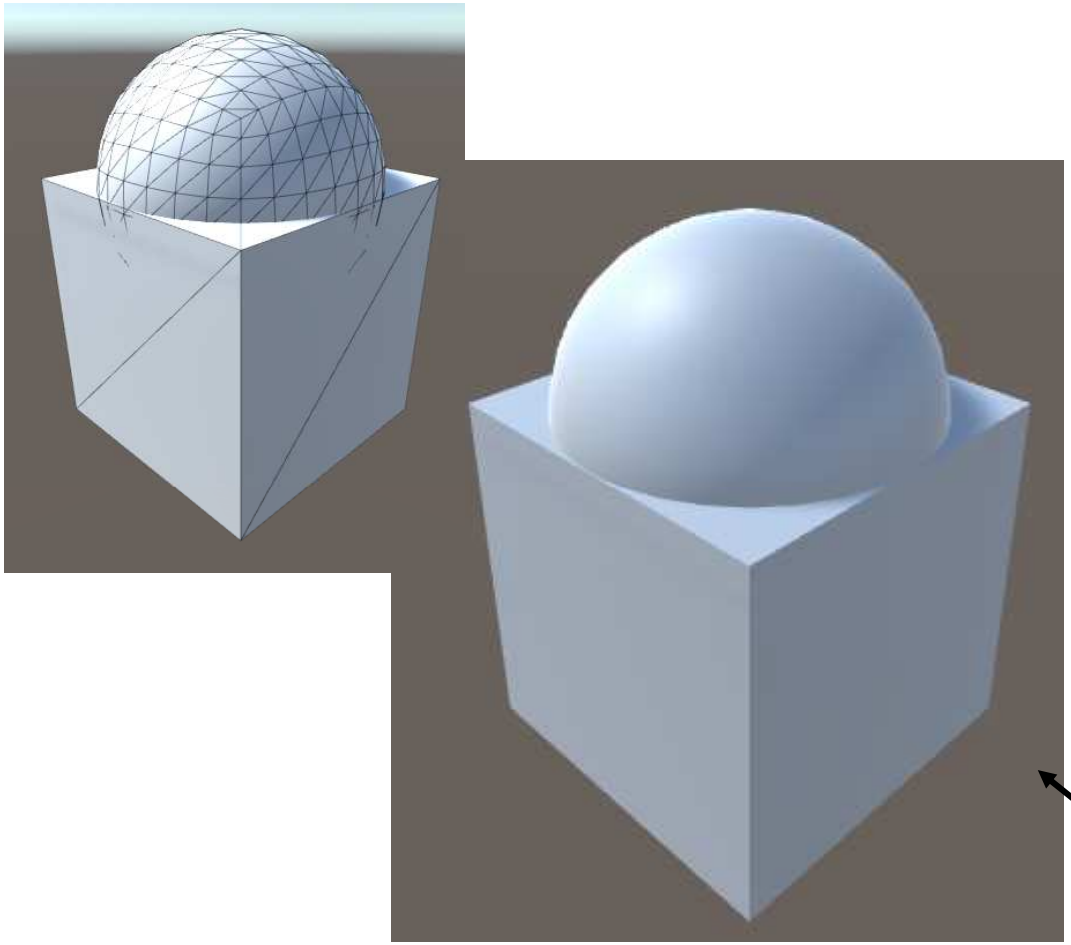
OBJ

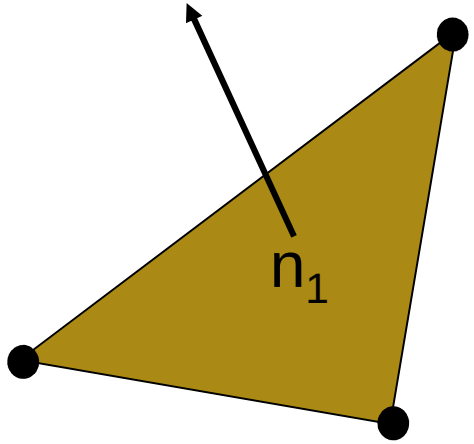
```
# cube.obj
# list of vertices
v   1.0 0.0 1.0
v   0.0 1.0 1.0
v   -1.0 0.0 1.0
v   0.0 -1.0 1.0
v   1.0 0.0 -1.0
v   0.0 1.0 -1.0
v   -1.0 0.0 -1.0
v   0.0 -1.0 -1.0
…
# list of faces
f  1 2 3  4
f  8 5 1  4
…

# mais aussi :
# f  node/norm/texture
f 1/4/1 2/4/3 3/4/8 4/4/5
```

WRL

```
#VRML V2.0 utf8
…
Geometry  IndexedFaceSet {
# optionnel mais utile :
ccw TRUE
solid TRUE
convex TRUE
coord  Coordinate {
point [
# Coordonnees des points
X1 Y1 Z1
X2 Y2 Z2
…
]}
coordIndex [
# Les faces
1 2 3 4 -1
5 6 2 1 -1
…
]
}
```

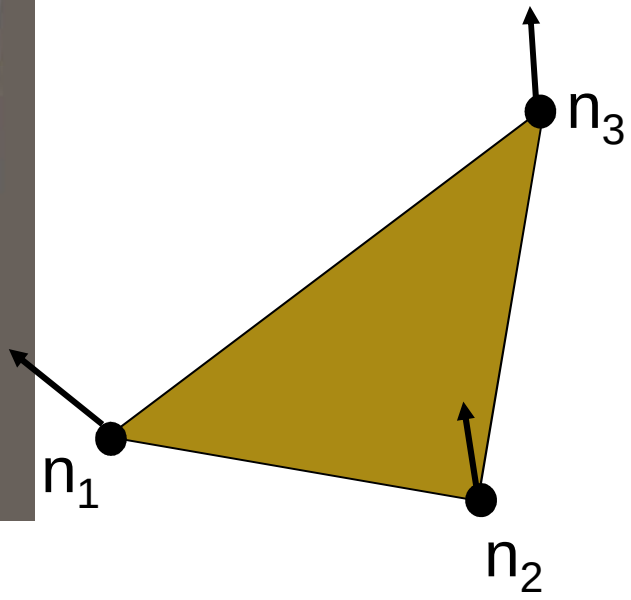
Faces planes et/ou courbes





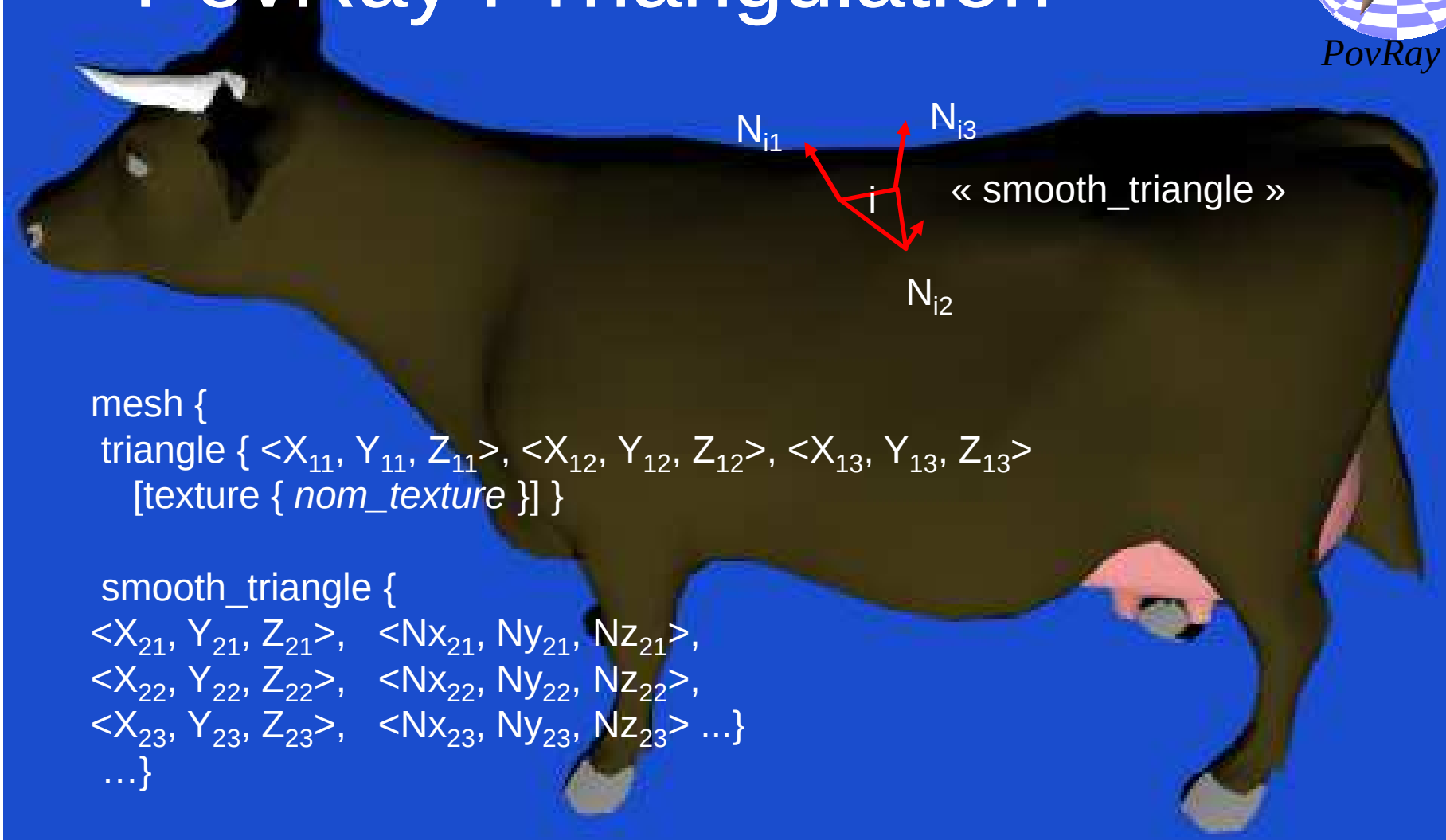
Surface plane

Surface courbe



Combien d'information faut-il stocker dans le 1er modèle ?
Dans le second ? Comment mélanger les 2 approches ?

PovRay : Triangulation



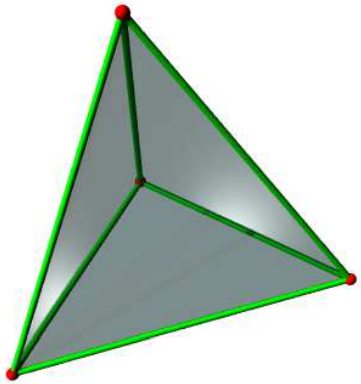
mesh {
 triangle { <X_{11}, Y_{11}, Z_{11}>, <X_{12}, Y_{12}, Z_{12}>, <X_{13}, Y_{13}, Z_{13}>
  [texture { *nom_texture* }] }

 smooth_triangle {
<X_{21}, Y_{21}, Z_{21}>, <Nx_{21}, Ny_{21}, Nz_{21}>,
<X_{22}, Y_{22}, Z_{22}>, <Nx_{22}, Ny_{22}, Nz_{22}>,
<X_{23}, Y_{23}, Z_{23}>, <Nx_{23}, Ny_{23}, Nz_{23}> ...}
 ...}

Fichier : cow2.pov

Triangulation frontière

Relation d'Euler : $t-a+s=2$



4 - 6 + 4 = 2

Triangulation : $2a=3t$

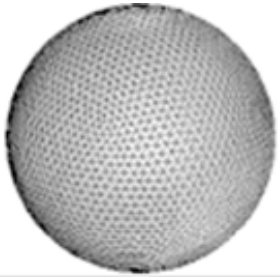
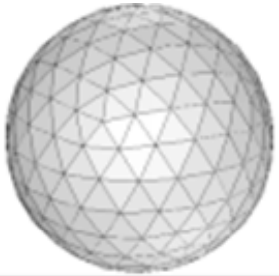
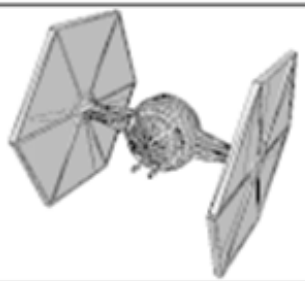
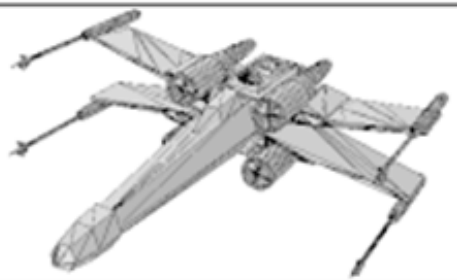
2*6 = 3*4

$t = 2s-4$

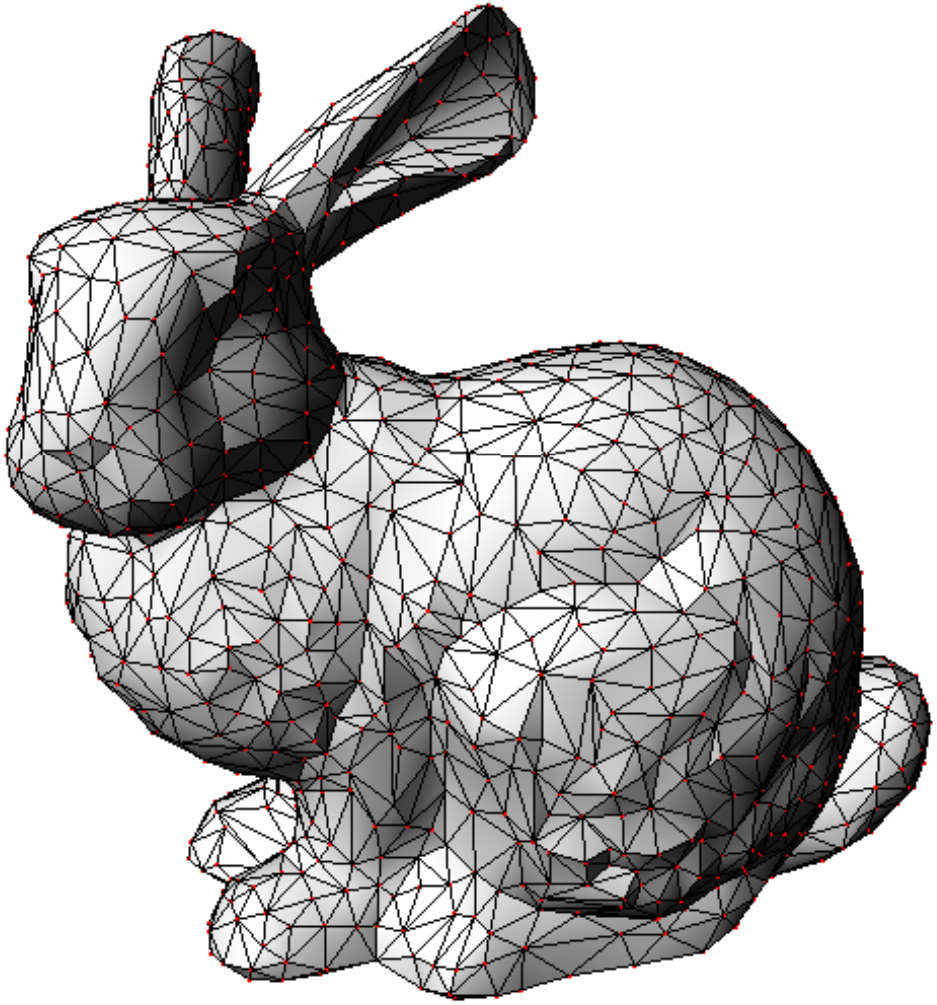
4=2*4-4

EXERCICE : Relation d'Euler

Que pouvez-vous déduire des cardinaux des triangulations ci-dessous ?

	
4002 vertices 12000 edges 8000 triangles 114469 bytes	252 vertices 750 edges 500 triangles 6346 bytes
	
2014 vertices 5944 edges 3827 triangles	3099 vertices 9190 edges 6076 triangles

Volume d’une triangulation ?

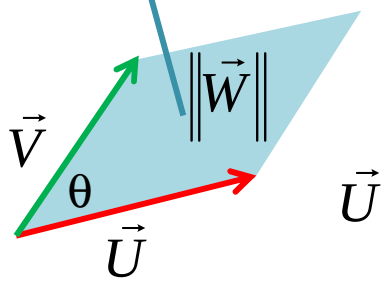


Produit Vectoriel (3D)

Rappels de géométrie (1)

$$\vec{W} = \vec{U} \wedge \vec{V}$$

$$\vec{w} = \frac{\vec{W}}{\|\vec{W}\|}$$



$$\|\vec{W}\| = \|\vec{U}\|\|\vec{V}\|\sin(\theta)$$

$$\vec{U} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}$$

$$W = \begin{pmatrix} w_x = u_y.v_z - u_z.v_y \\ w_y = u_z.v_x - u_x.v_z \\ w_z = u_x.v_y - u_y.v_z \end{pmatrix}$$

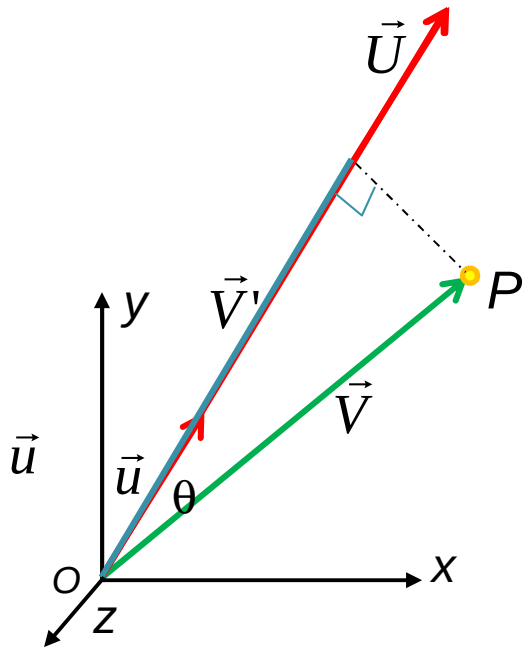
$$M_u V = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} -u_z v_y + u_y v_z \\ u_z v_x - u_x v_z \\ -u_y v_x + u_x v_y \end{pmatrix}$$

Scilab : W=cross (U,V)

Produit scalaire (3D)

Rappels de géométrie (2)

$$w = \vec{U}.\vec{V}$$



*Scilab : w=U'*V*

$$w = \|\vec{U}\|\|\vec{V}\|\cos(\theta)$$

$$\|\vec{V}\|^2 = \vec{V}.\vec{V}$$

$$w = {}^tU * V$$

Projection d'un vecteur *V* sur l'axe $\vec{u}$

$$\vec{V}' = \|\vec{V}\|\cos(\theta)\vec{u}$$

Écriture matricielle de la projection :

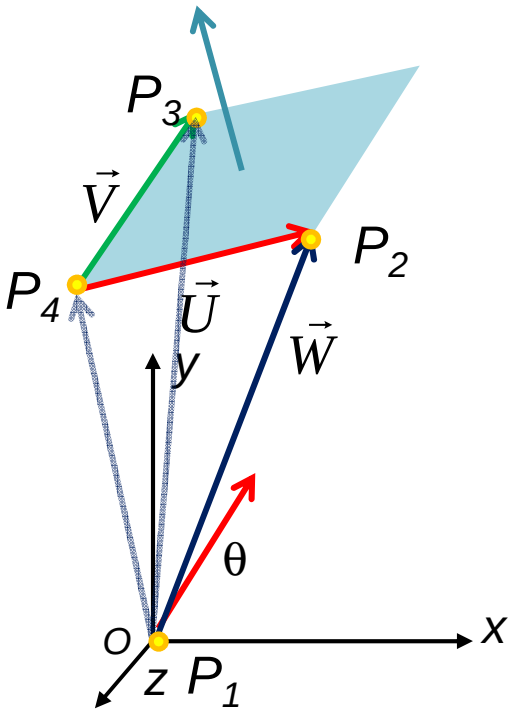
$$w = u_x.v_x + u_y.v_y + u_z.v_z$$

$$P_u * V = \begin{pmatrix} u_x^2 & u_x.u_y & u_x.u_z \\ u_y.u_x & u_y^2 & u_y.u_z \\ u_z.u_x & u_z.u_y & u_z^2 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} w.u_x \\ w.u_y \\ w.u_z \end{pmatrix}$$

P_u est la matrice de projection sur l'axe $\vec{u}$

Produit mixte (3D)

Rappels de géométrie (3)



$$m = \vec{W}.(\vec{U} \wedge \vec{V})$$

Écriture matricielle :

$$\vec{W}.(\vec{U} \wedge \vec{V}) = Det\begin{pmatrix} w_x & u_x & v_x \\ w_y & u_y & v_y \\ w_z & u_z & v_z \end{pmatrix}$$

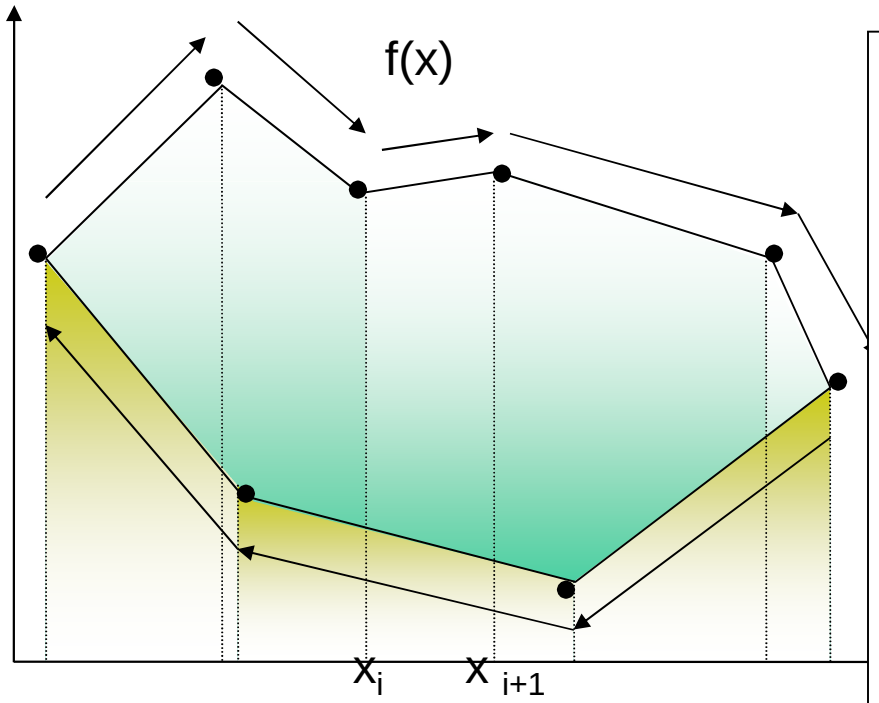
Volume du tétraèdre :

$$vol(P_1, P_2, P_3, P_4) = \frac{1}{6}\left(\vec{W}.(\vec{U} \wedge \vec{V})\right)$$

Calcul de l ’aire

Approche triviale

$$\sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)$$



Exemple cas linéaire :

$f(x)= (x-x_i)(y_{i+1}–y_i)/(x_{i+1}–x_i)+ y_i$

Ou encore

$F(x)=Ax+B$

avec $A=(y_{i+1}–y_i)/(x_{i+1}–x_i)$

et $B=(x_{i+1}\, y_i–x_i y_{i+1})/(x_{i+1}–x_i)$

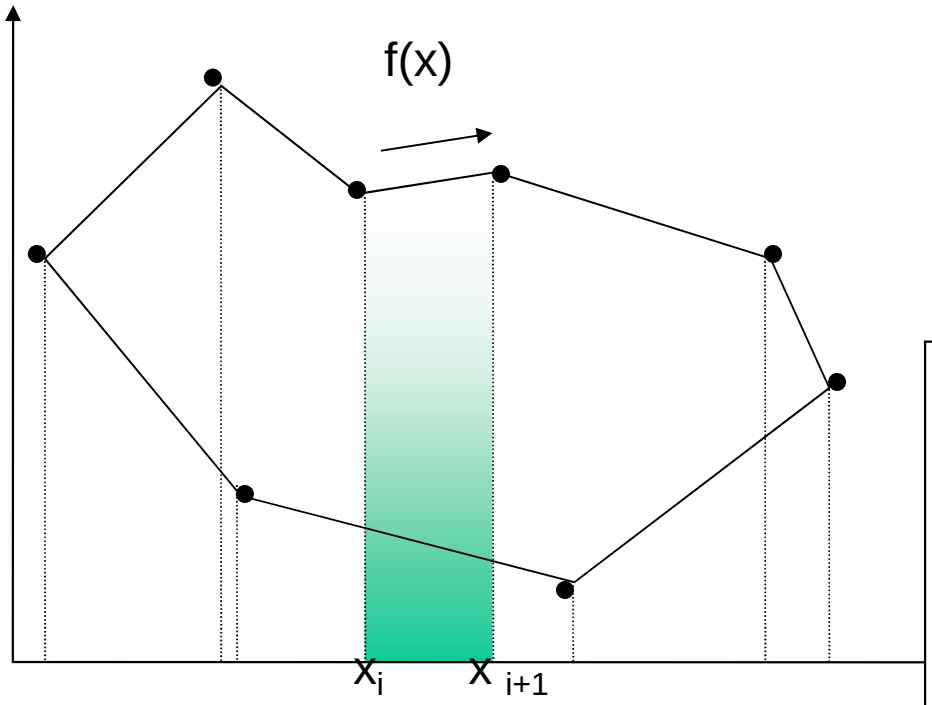
$$\int_{x_i}^{x_{i+1}} (Ax + B)dx =$$

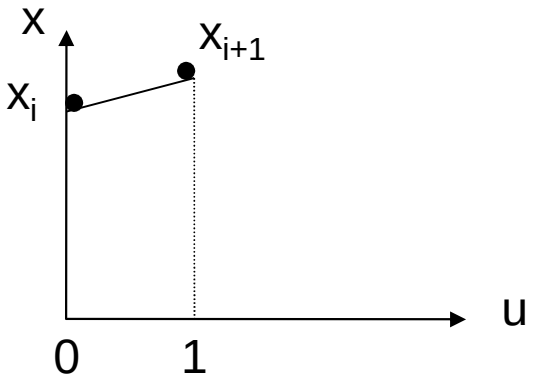
$$\frac{1}{2}(y_{i+1} + y_i)(x_{i+1} - x_i)$$

Calcul de l'Aire

Avec changement de variable

$$\int_{x=x_i}^{x_{i+1}} f(x)dx = \int_{u=0}^{1} f(t(u))t'(u)du$$





x = t(u), u dans [0:1]

<u>Exemple cas linéaire</u> :

$t(u) = ux_{i+1} + (1-u)x_i$

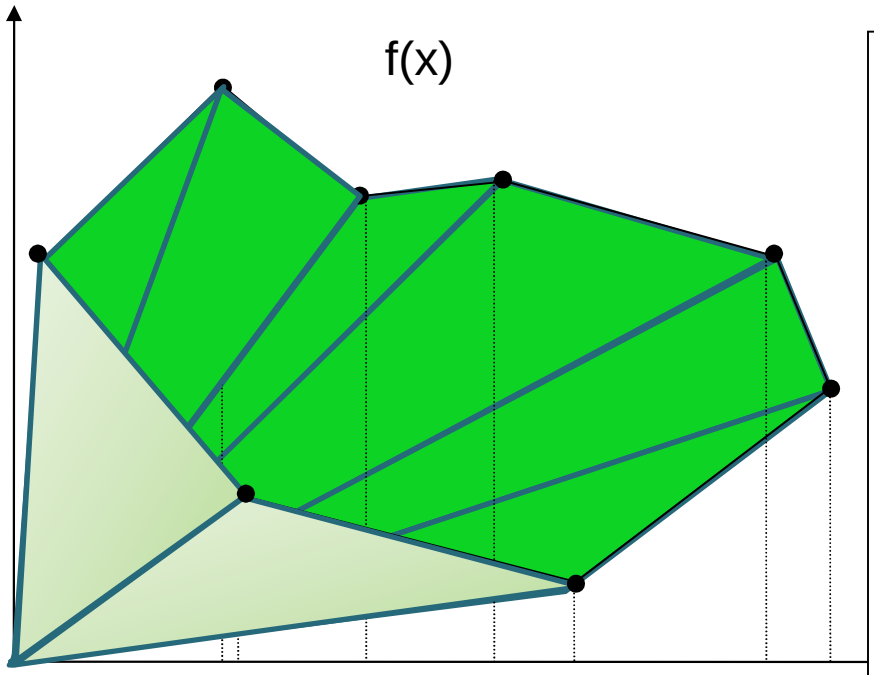
$f(t(u)) = uy_{i+1} + (1-u)y_i$

$$\int (uy_{i+1} + (1-u)y_i)(x_{i+1} - x_i)du =$$

$$\frac{1}{2}(y_{i+1} + y_i)(x_{i+1} - x_i)$$

Calcul de l ’aire

Approche pragmatique



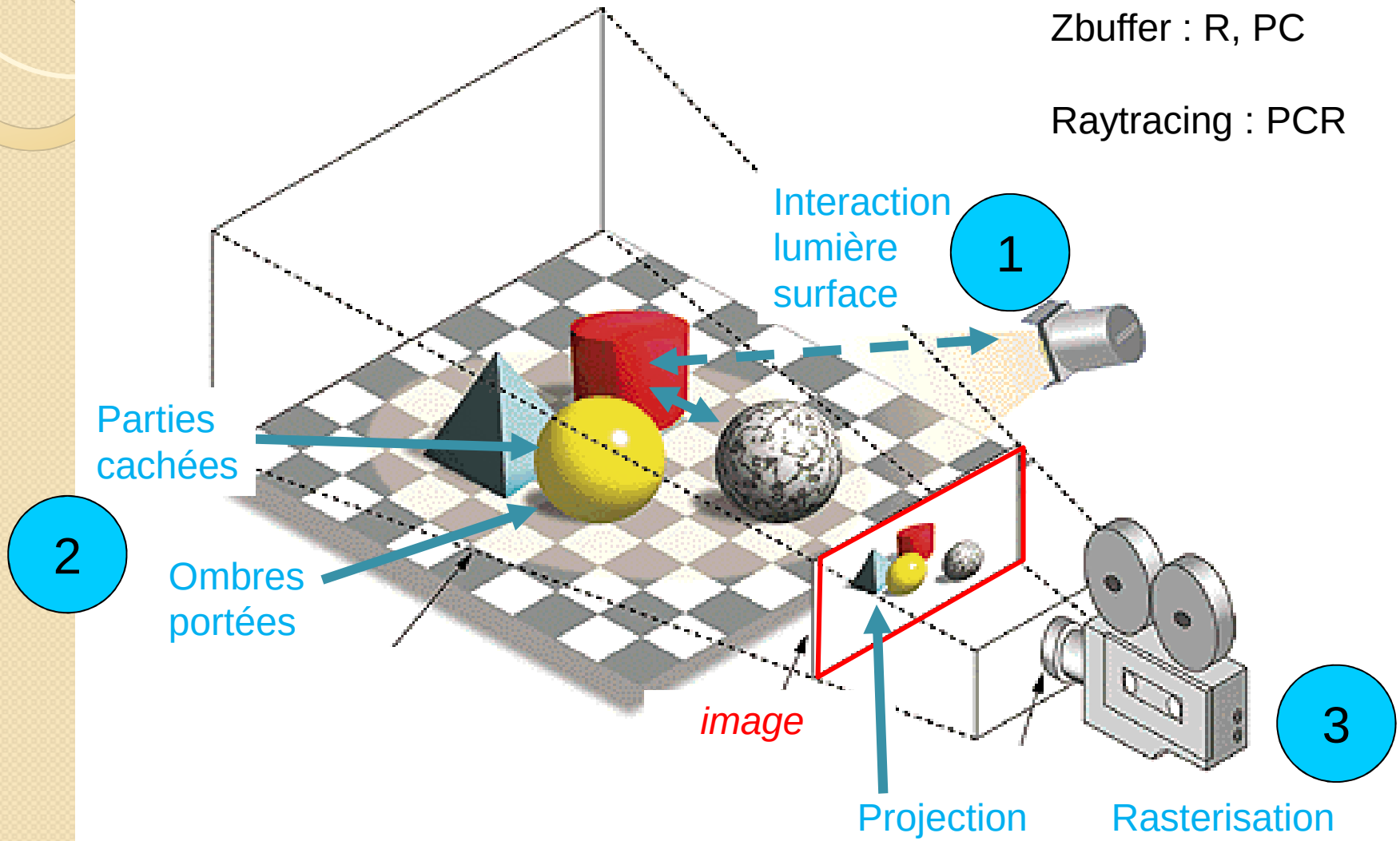
<u>Algorithme</u> :
Calcul du point O
A=0
Pour toutes les sommets du poly
A=A+ ½ PV(OS_i,S_iS_{i+1})

L’image

A.Peintre : PC, R

Zbuffer : R, PC

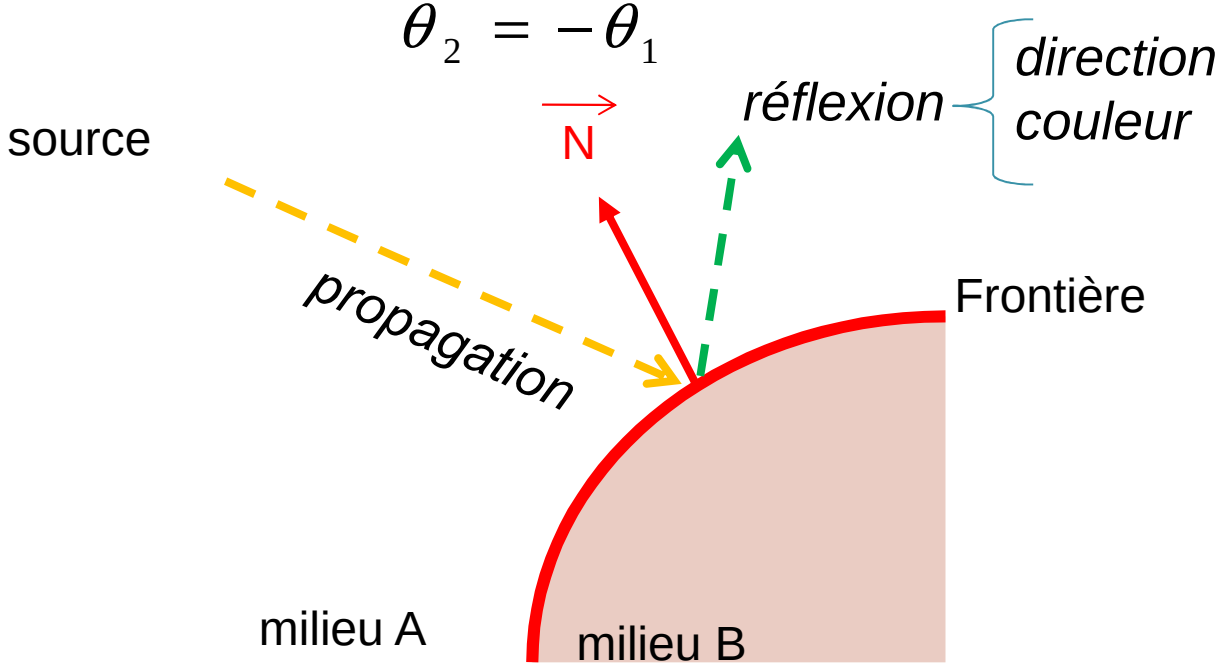
Raytracing : PCR



2.2 Propagation de la lumière

Les lois de Snell-Descarte

$$\theta_2 = -\theta_1$$



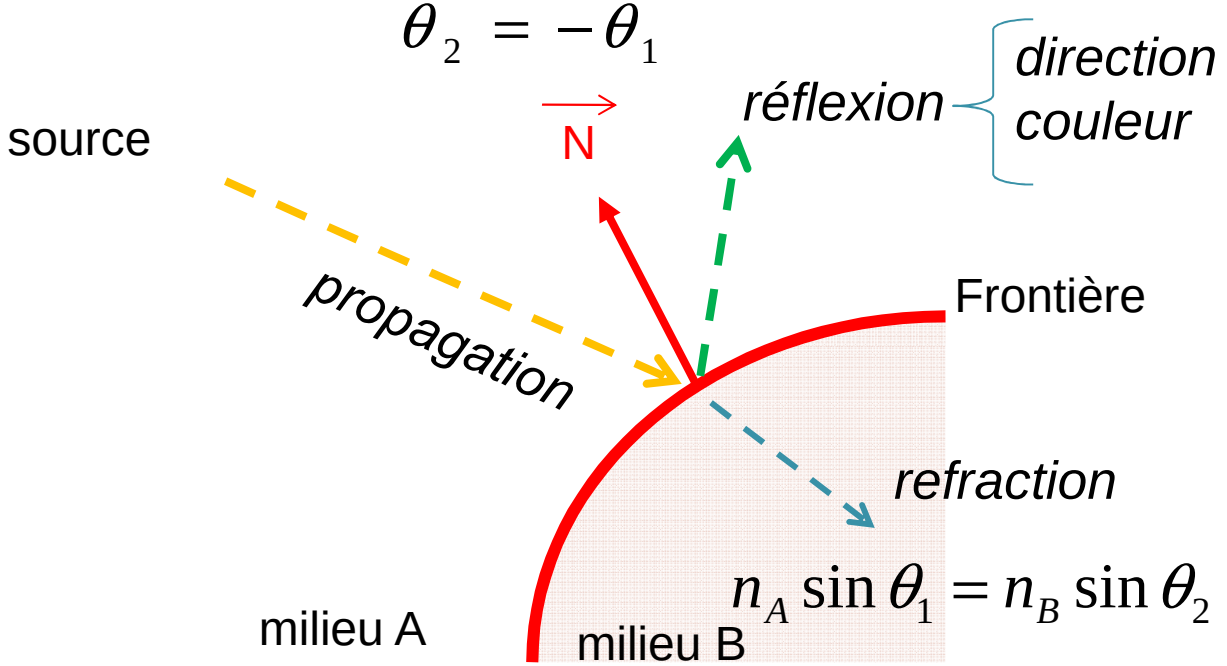
La lumière est l'ensemble des ondes électromagnétiques visibles par l'œil humain

λ = 800 nm  λ = 380 nm

2.2 Propagation de la lumière

Les lois de Snell-Descarte

$$\theta_2 = -\theta_1$$



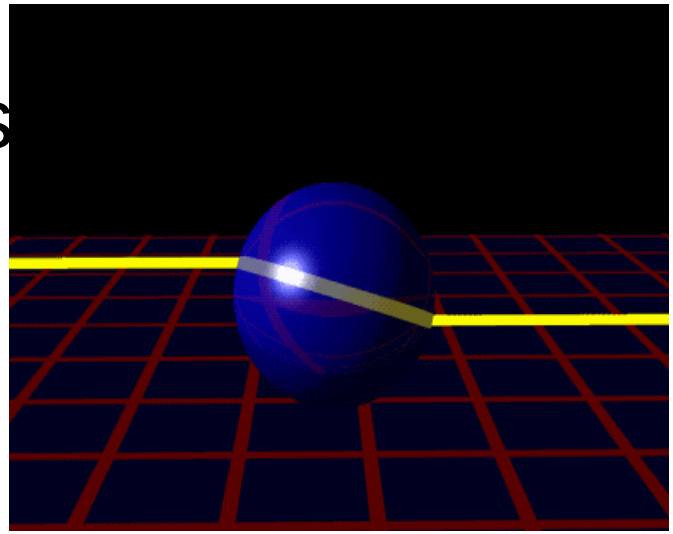
La lumière est l'ensemble des ondes électromagnétiques visibles par l'œil humain

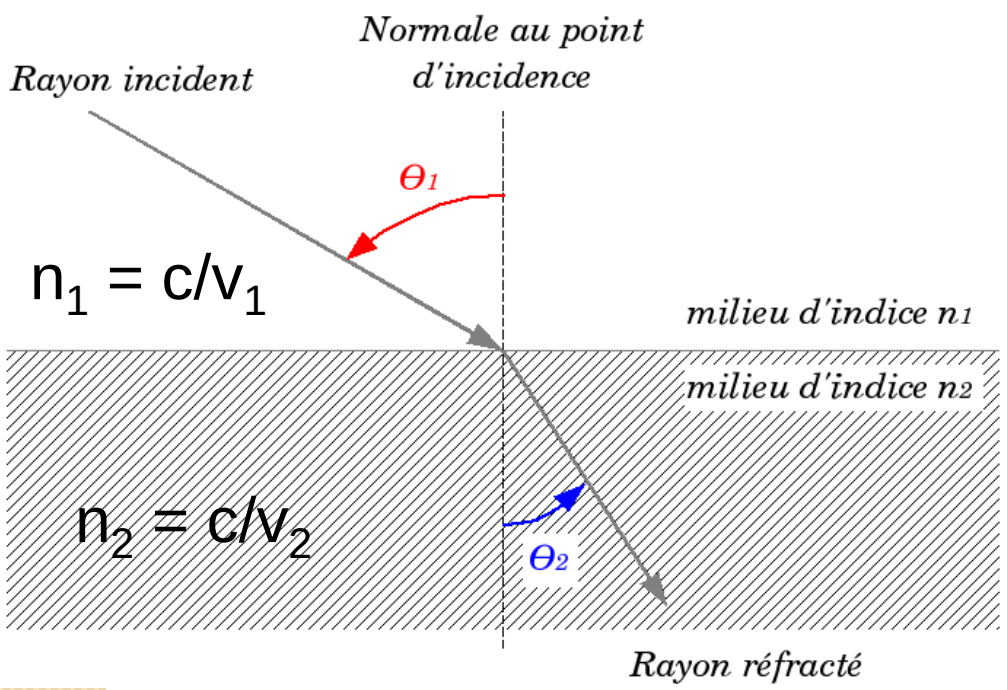
λ = 800 nm  λ = 380 nm

Transparence, réfraction

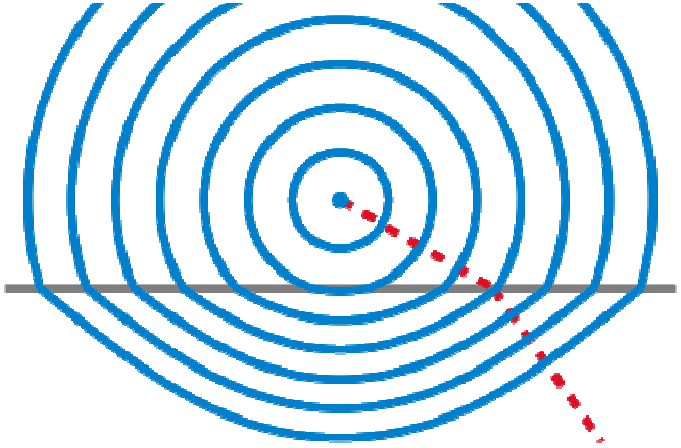
Loi de Snell-Descartes

$$n_1 \sin\theta_1 = n_2 \sin\theta_2$$



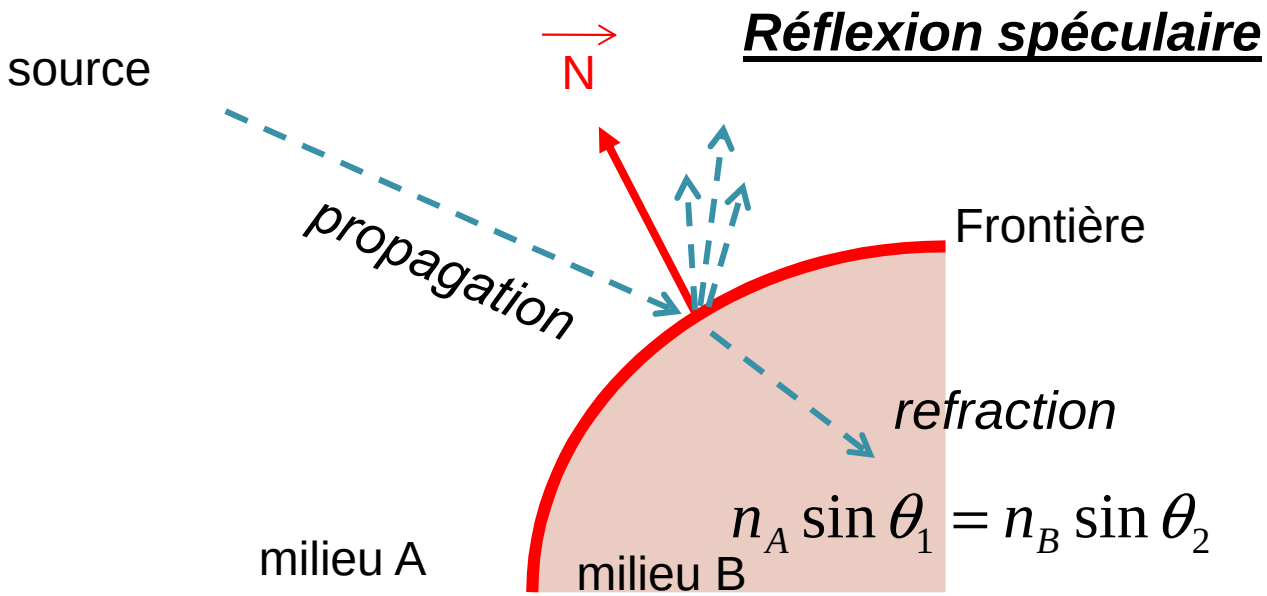


n dépend de λ, T, structure : matériaux biréfringents



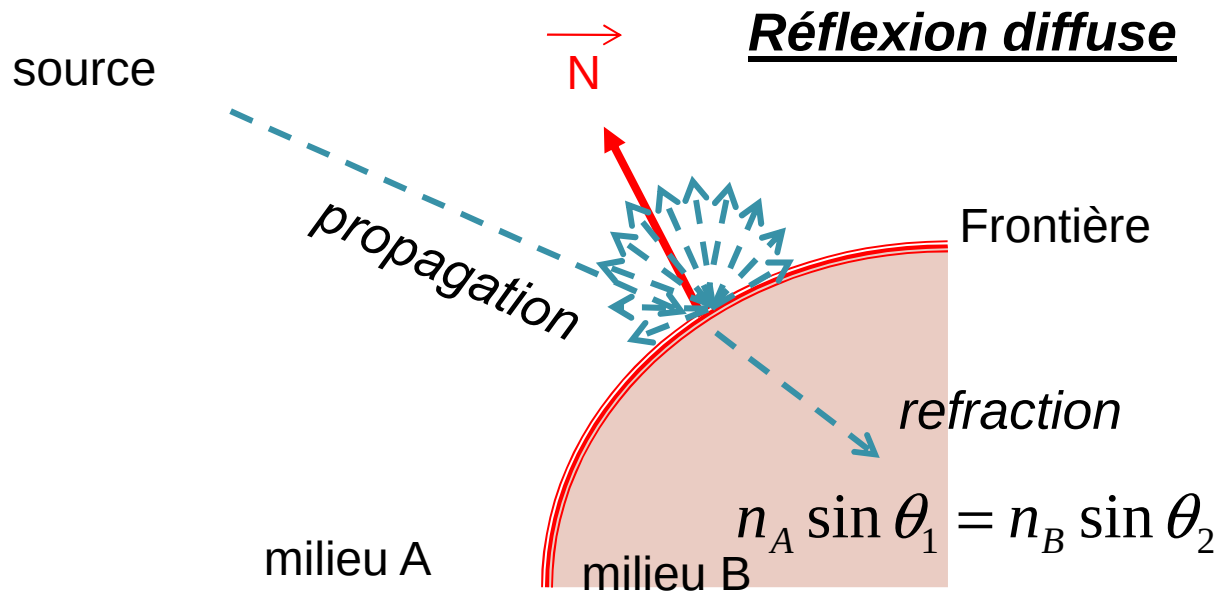
Réflexion spéculaire

$$\theta_2 = -\theta_1$$



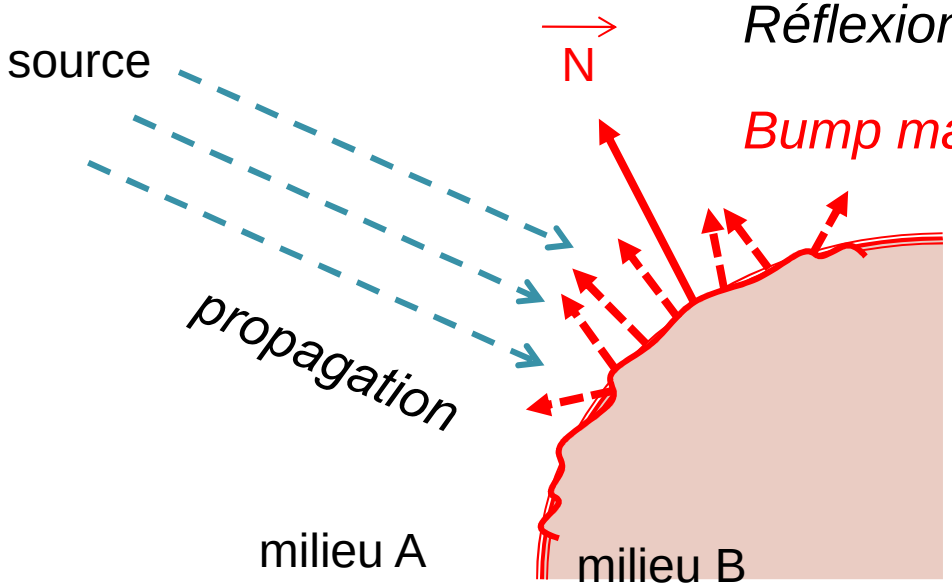
Réflexion diffuse

$$\theta_2 = -\theta_1$$



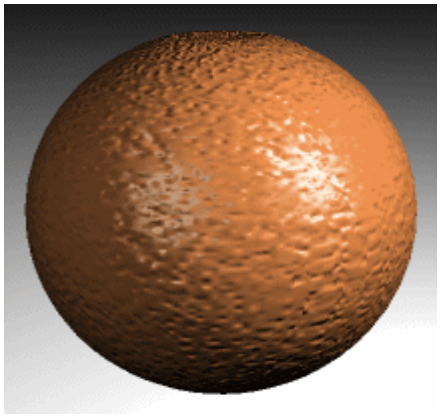
Bump mapping

$$\theta_2 = -\theta_1$$



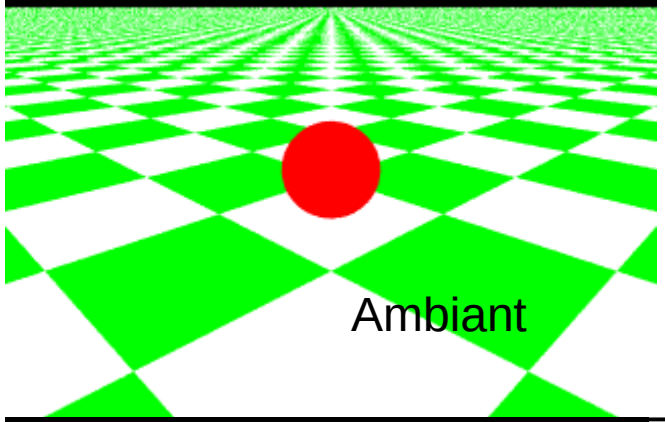
Réflexion spéculaire ou diffuse

Bump mapping



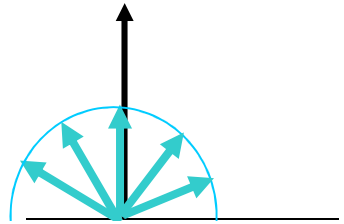
Réflexions

Diffuse la lumière ambiante



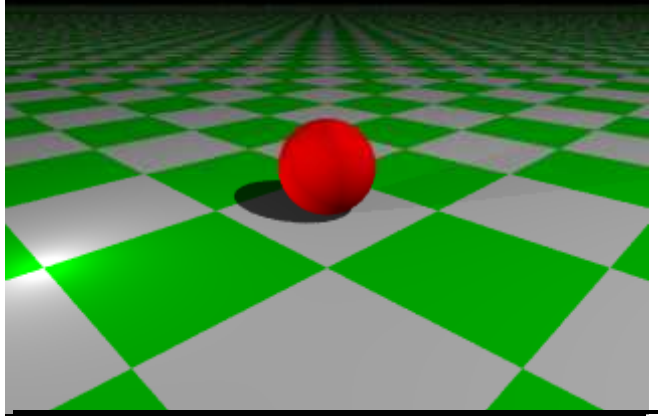
« Radiosité »

$$I_i = K_{ai} I_a$$



Pour K_{ai}=1

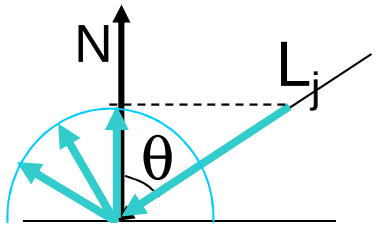
Réflexion diffuse



Matériau Mat

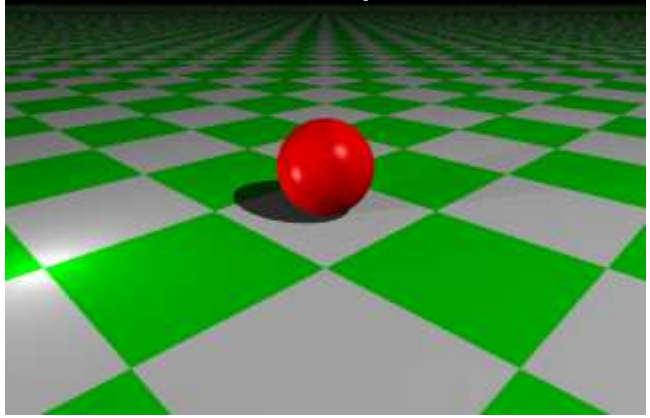
$$I_d = K_d \sum_{sources} \langle N, L_j \rangle I_j$$

N normale à la surface
Lj rayon de la lumière j



Pour K_d=1

Réflexion spéculaire

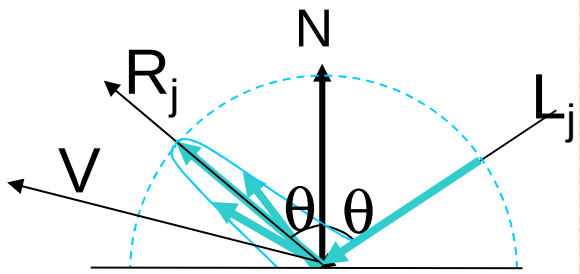


Brillant/Lisse

$$I_s = \sum_{sources} K_s \langle R_j, V \rangle^n I_j$$

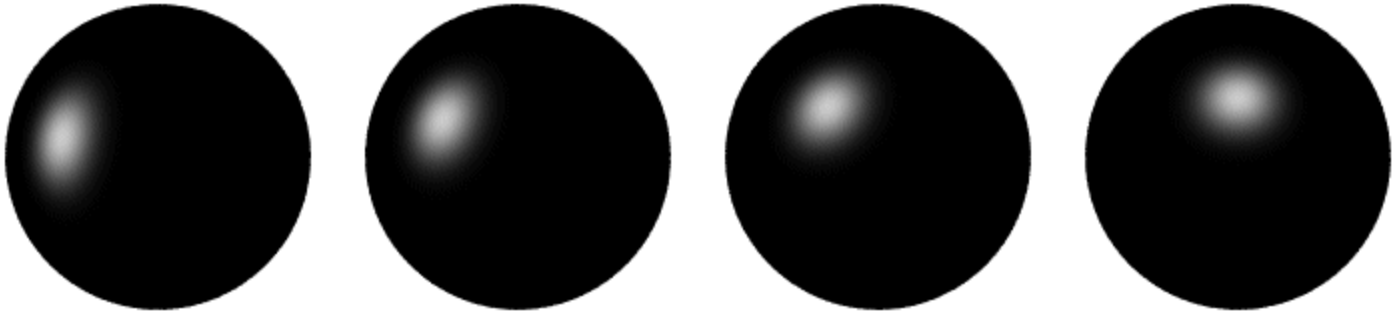
$$R_j = L_j - 2\langle N, L_j \rangle N$$

V direction de l'observateur

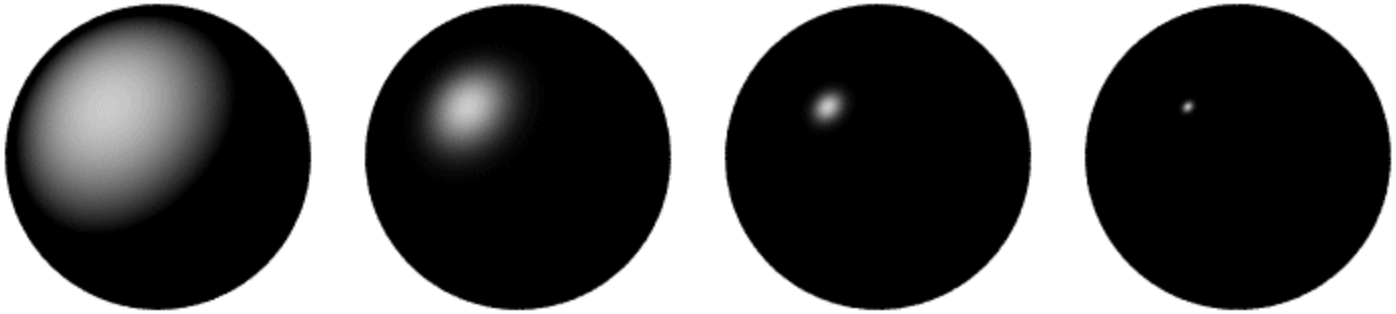


Pour K_s=1

Modèle de Phong



Si on déplace une source lumineuse



Si on change la brillance (n)

Matériaux

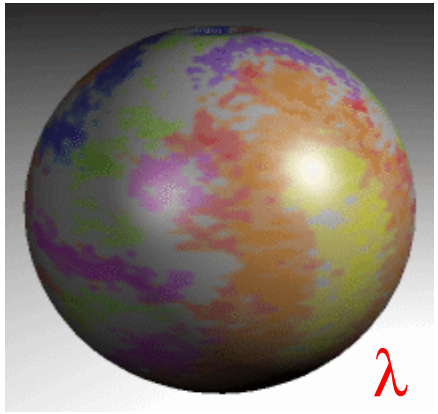
Métal : K_a = , K_d =
K_s = n= …

Verre : K_a = , K_d =
K_s = n= …

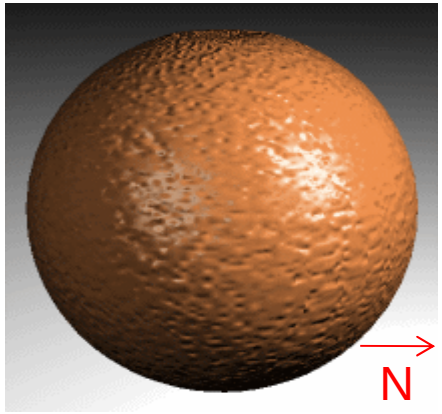
Nuage : K_a = , K_d =
K_s = …

Bois : K_a = , K_d =
K_s = n= …

2.3 Répartition : les textures



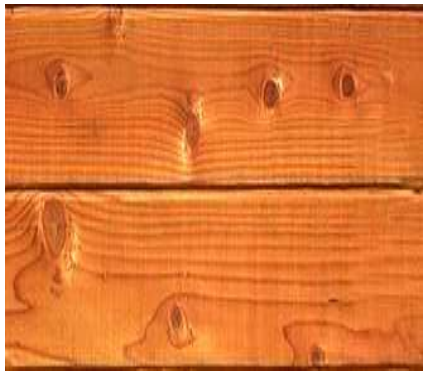
Color mapping



Bump mapping

Reflexion map ?
Why not ?

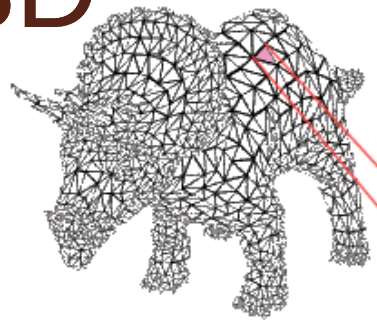
Motif unique



Motif répété

Perturbateurs :
aléas…

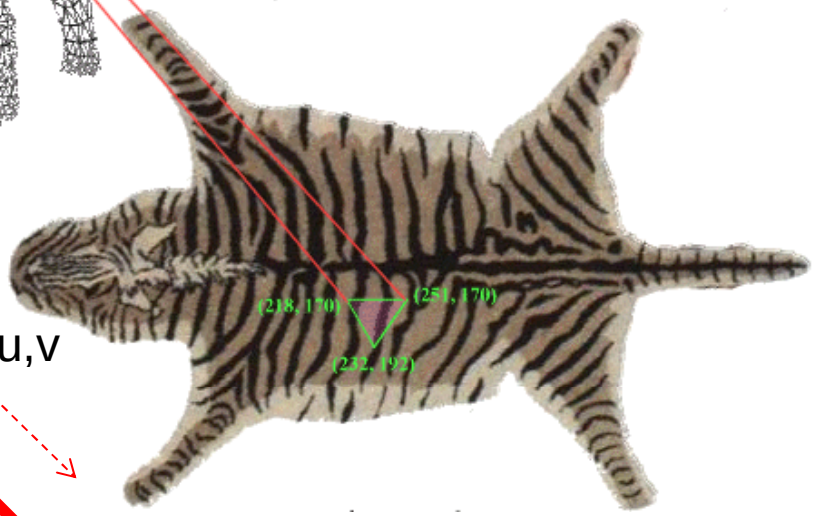
Mapping en 3D



Pour chaque triangle du modèle 3D prendre la texture sur la « photo »

3D

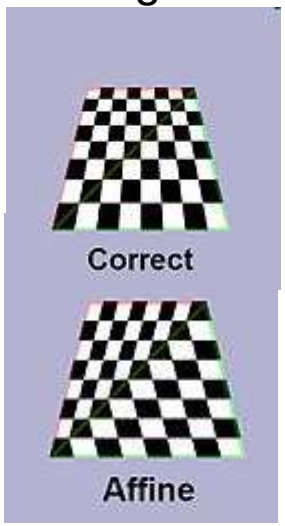
x,y,z



u,v

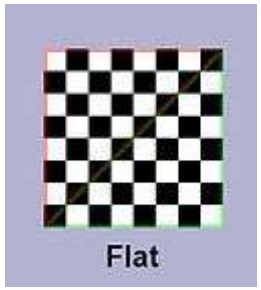
i,j

Image



Projection sur le plan de l'image

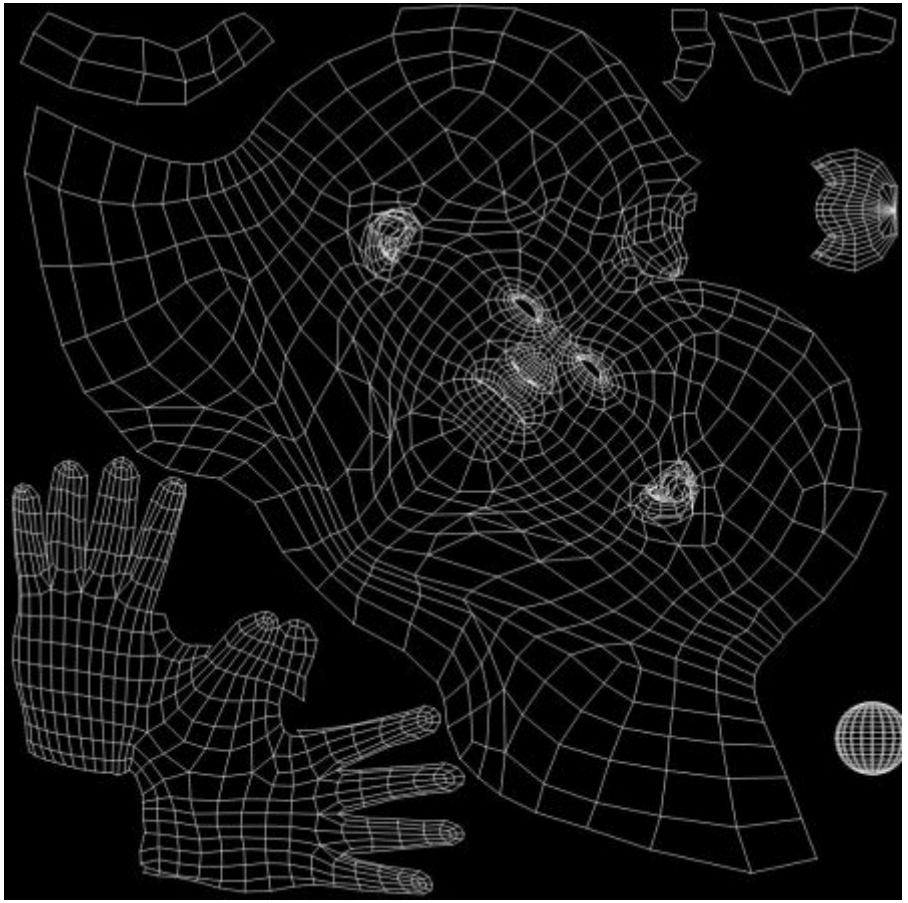
erreur



Texture

Pendant la rasterisation, on interpole les coordonnées sur la carte de texture

Texture : pigment mapping



Texture, dans la pratique…

[Blender 2.7 Tutorial #13 : UV Mapping (Unwrapping for Image Textures)]

BornCG 2014

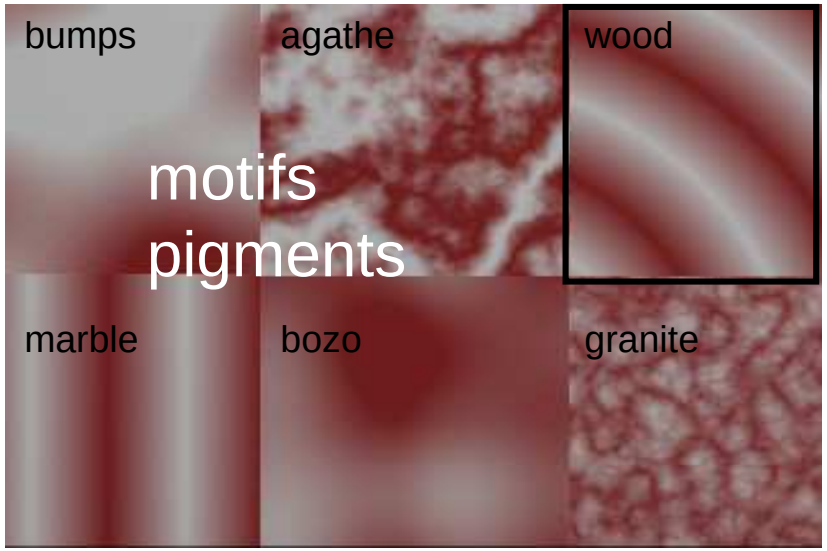
Exemple

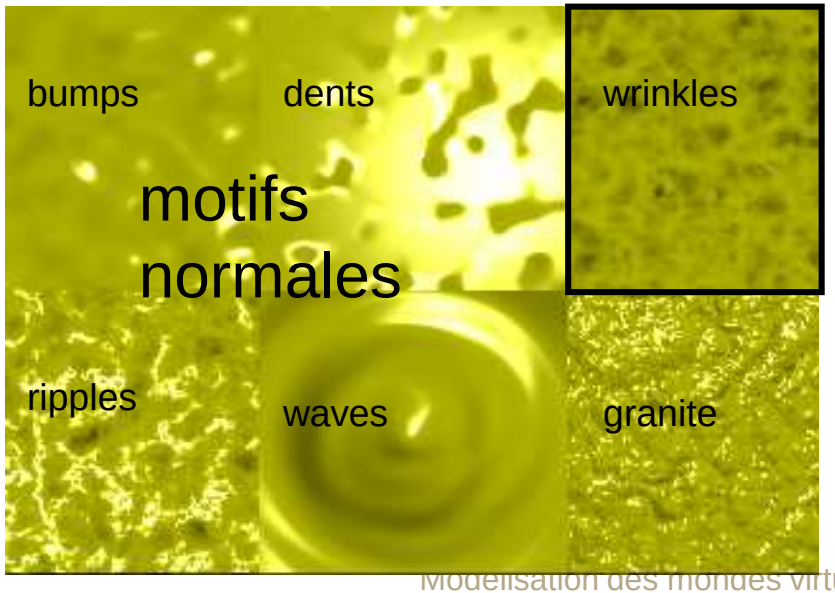
Répétition, taille, aléas…

pigments

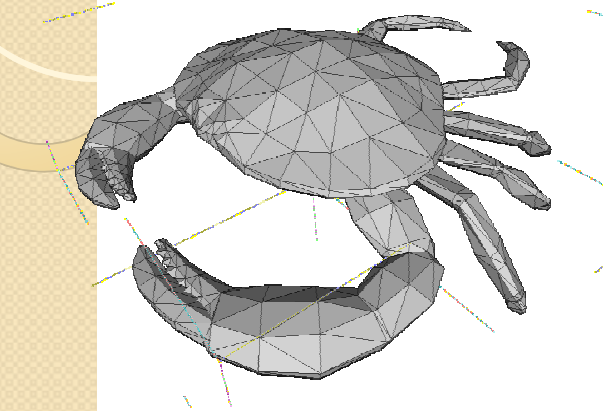
normales

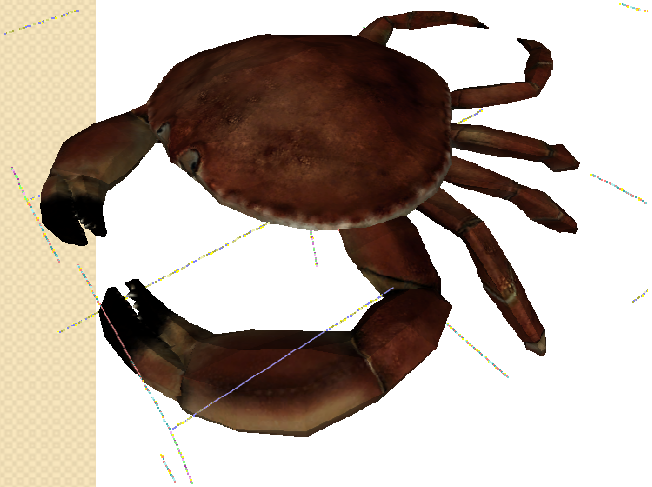
Surface échelle méso

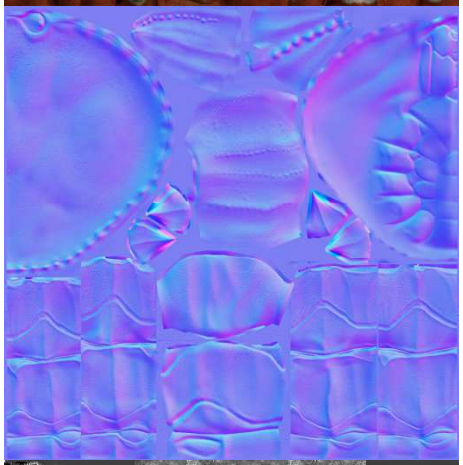


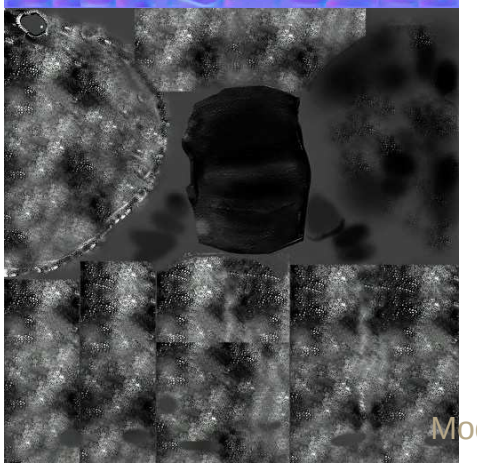


Exemple

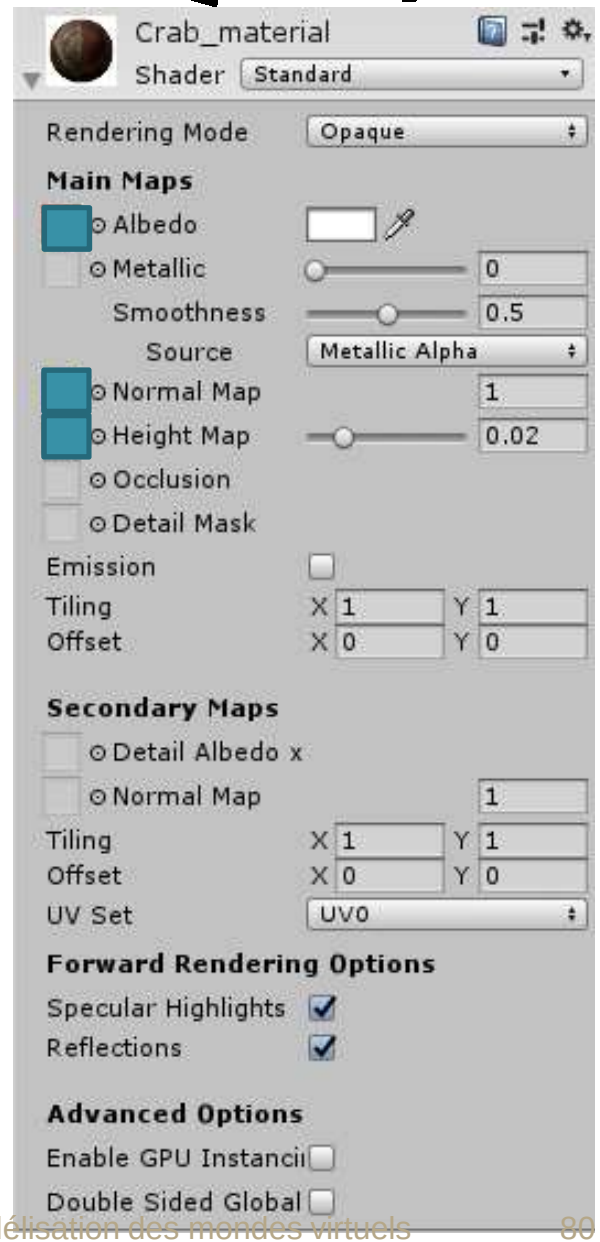




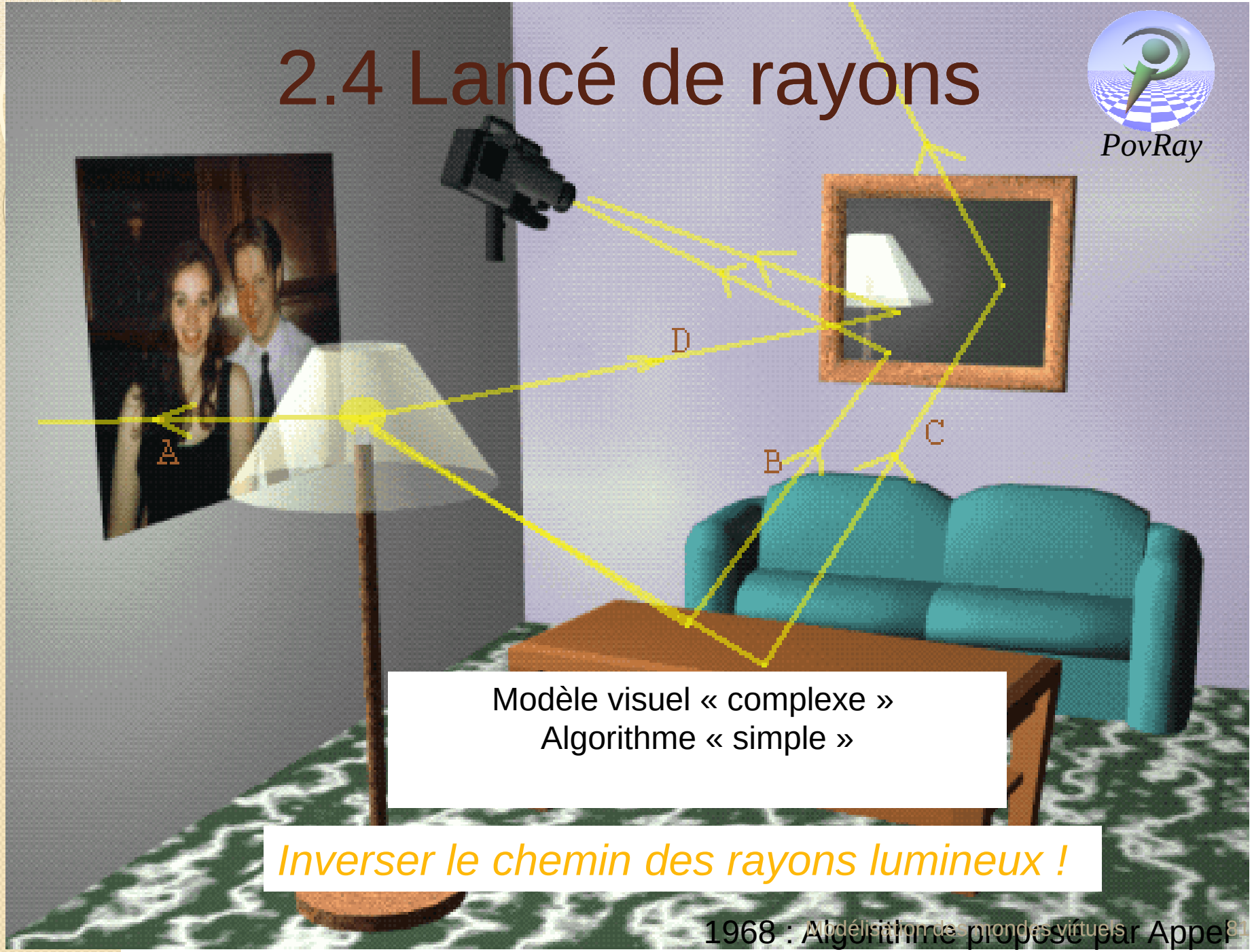





unity


Crab_material
Shader
Standard
Rendering Mode
Opaque
Main Maps
Albedo
Metallic
0
Smoothness
0.5
Source
Metallic Alpha
Normal Map
1
Height Map
0.02
Occlusion
Detail Mask
Emission
Tiling
X 1
Y 1
Offset
X 0
Y 0
Secondary Maps
Detail Albedo x
Normal Map
1
Tiling
X 1
Y 1
Offset
X 0
Y 0
UV Set
UV0
Forward Rendering Options
Specular Highlights
Reflections
Advanced Options
Enable GPU Instancii
Double Sided Global

2.4 Lancé de rayons



Modèle visuel « complexe »
Algorithme « simple »

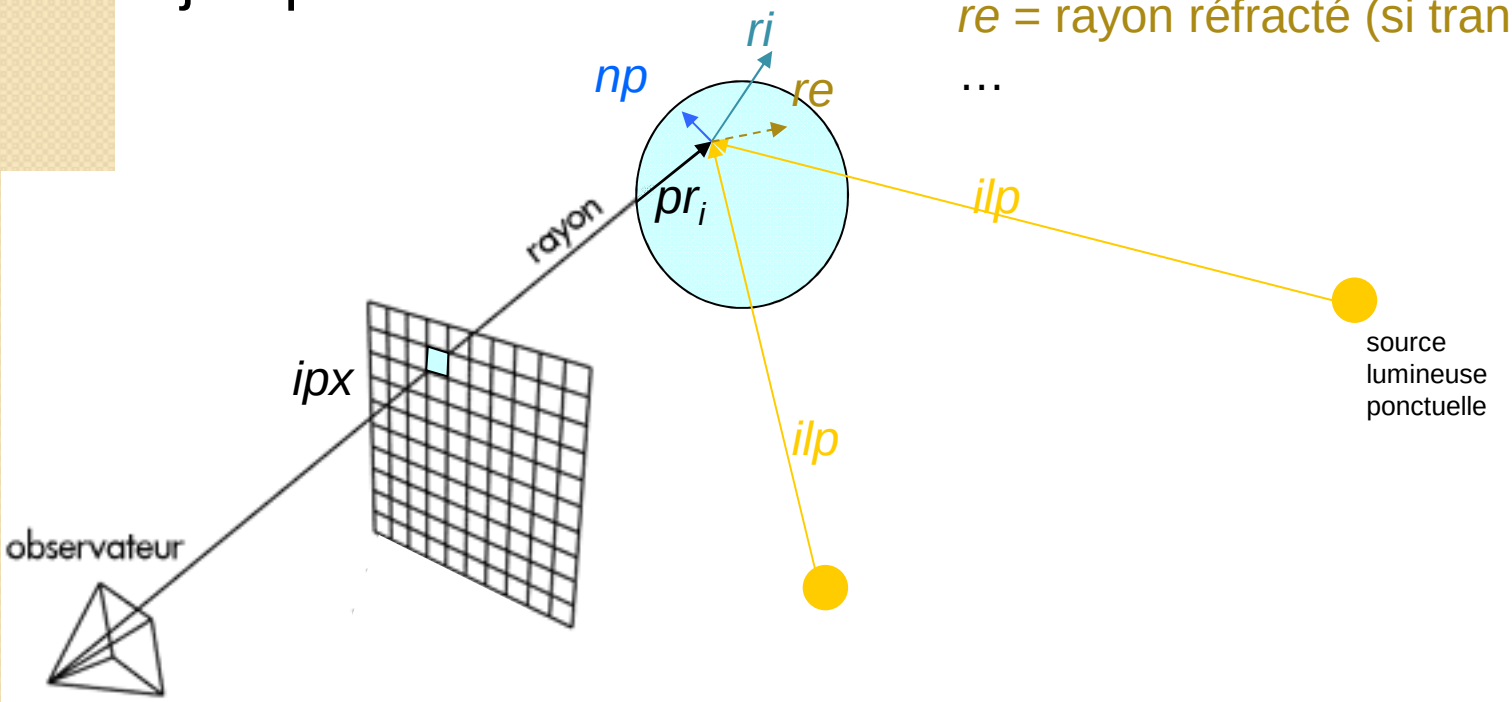
Inverser le chemin des rayons lumineux !

1968 : Algorithme proposé par Appel

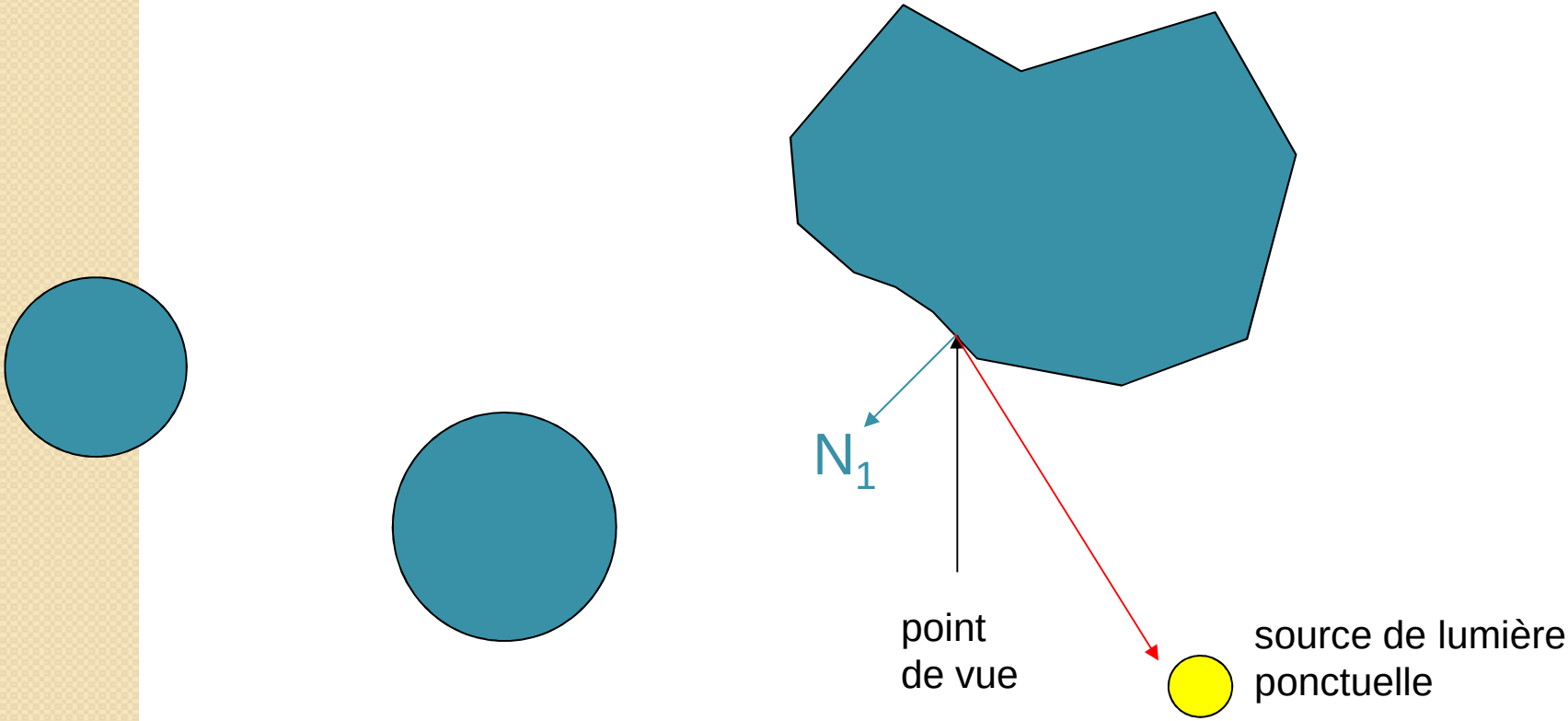
Lancé de rayon (Ray tracing)

Principe : modéliser le trajet des rayons de la lumière jusqu'à l'oeil

ipx = intensité lumineuse du pixel
r = lancé d'un rayon partant de *ipx*
pr_i = intersection de r avec la géométrie
np = normale en pr_i
ilp=éclairage de la surface en pr_i
ri = rayon réfléchi en pr_i (si miroir)
re = rayon réfracté (si transparence)
…

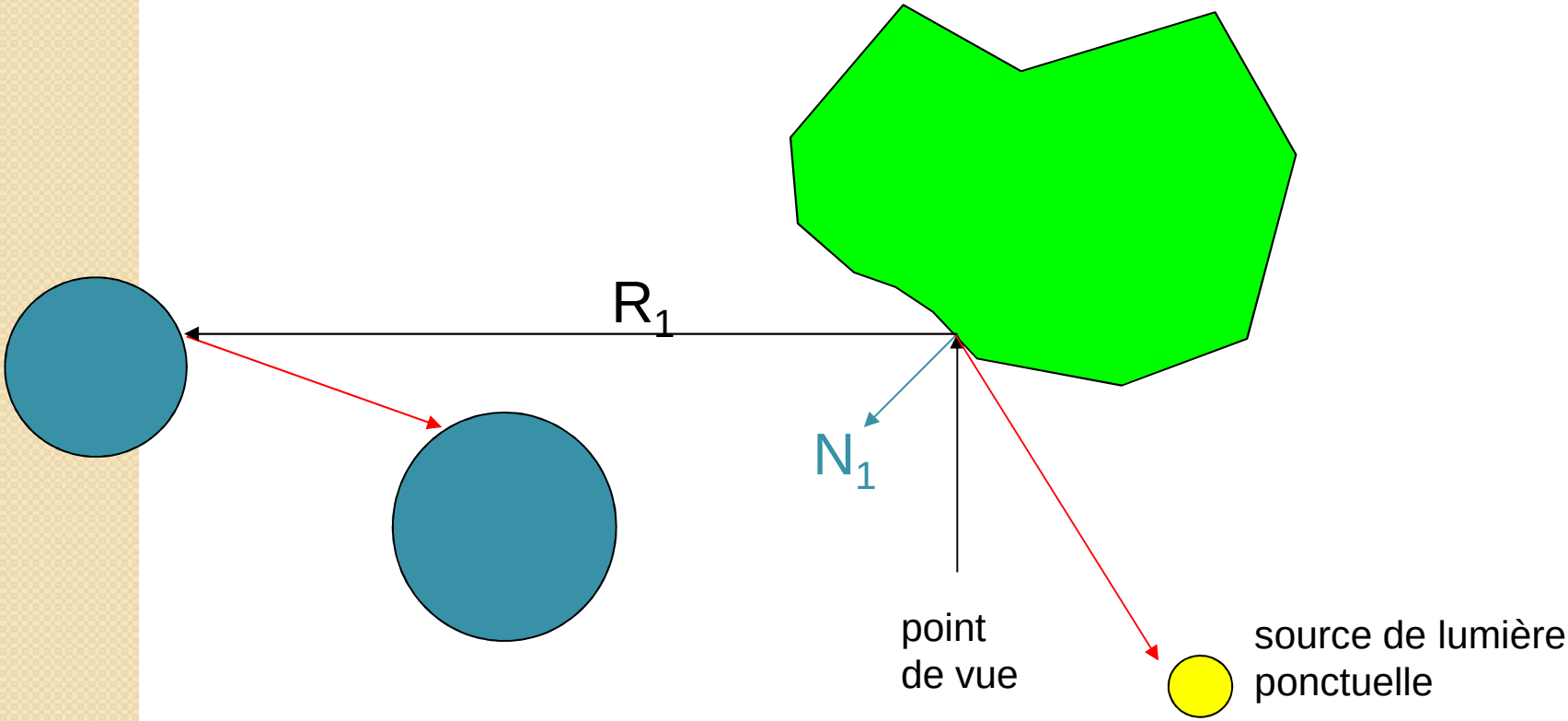


Principe : lancé de rayon



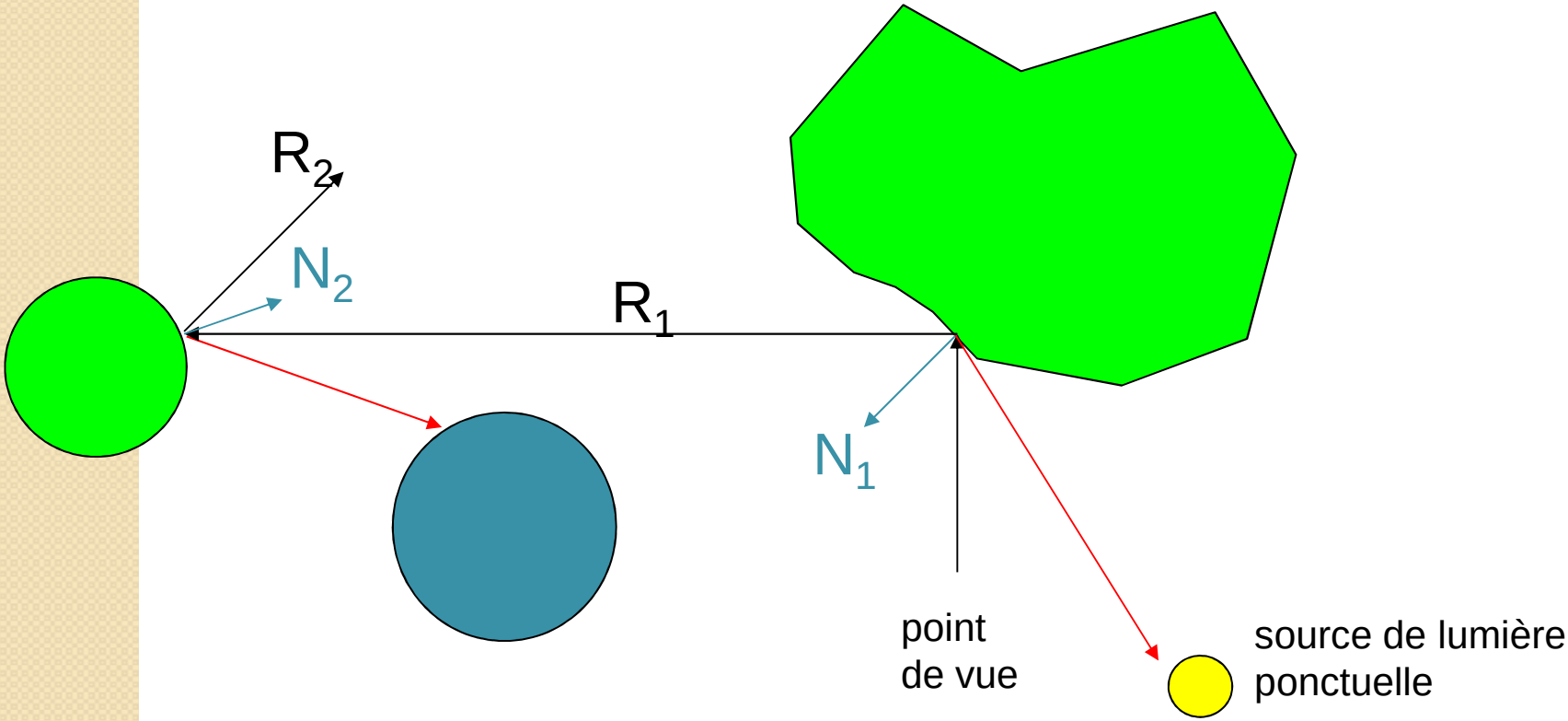
from CG principles & practice Foley, Van Dam …

Principe : lancé de rayon



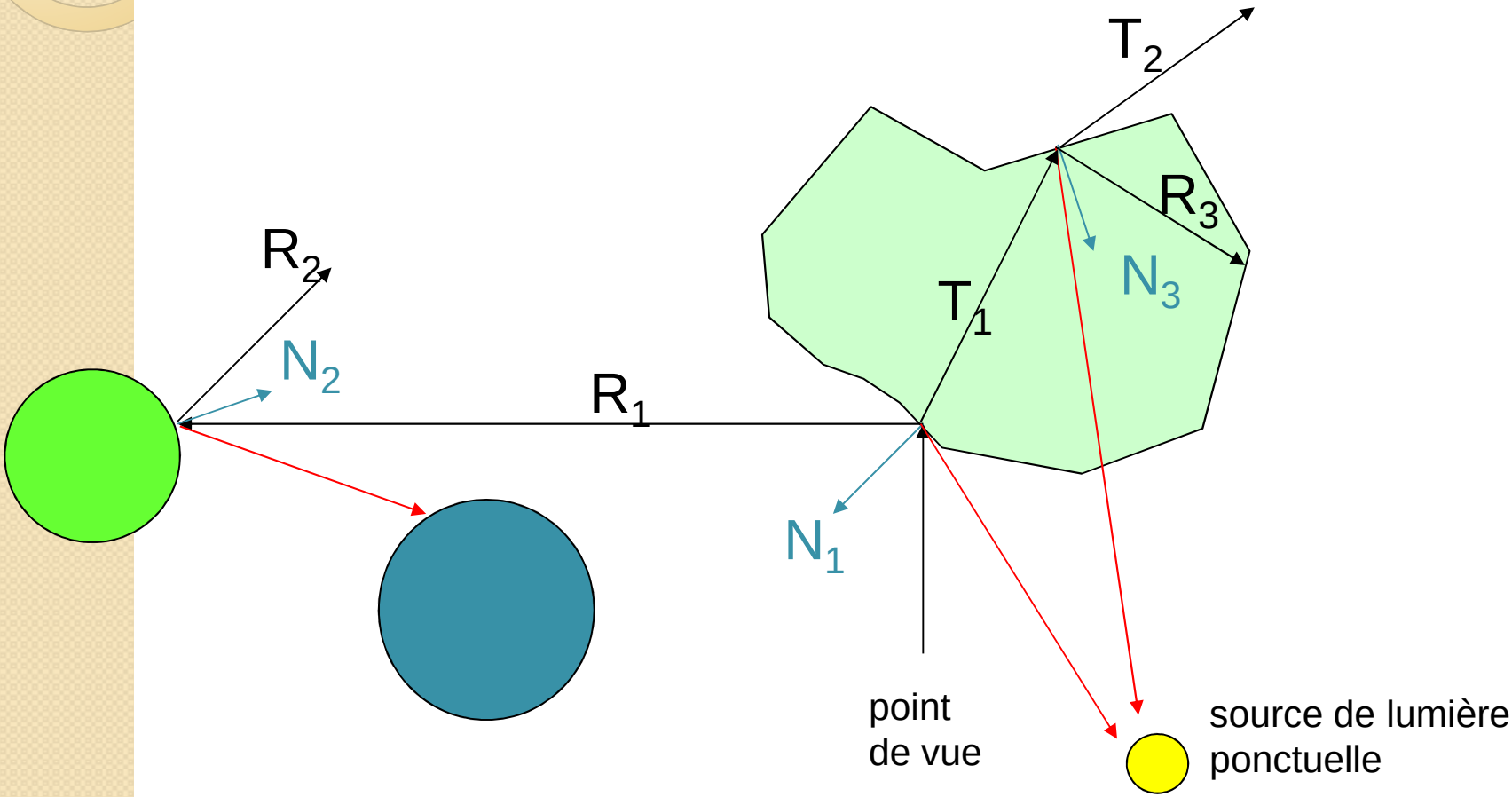
from CG principles & practice Foley, Van Dam …

Principe : lancé de rayon



from CG principles & practice Foley, Van Dam …

Principe : lancé de rayon

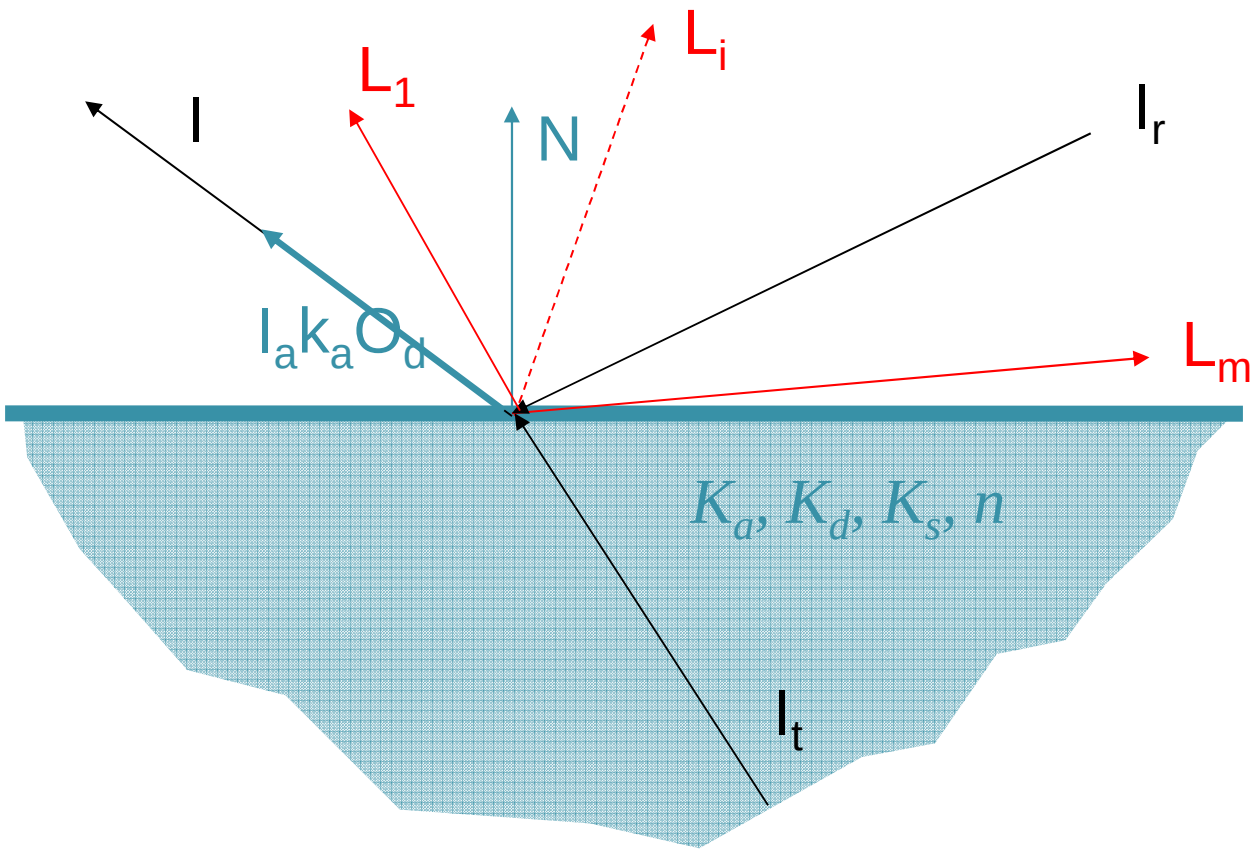


from CG principles & practice Foley, Van Dam …

Définition récursive

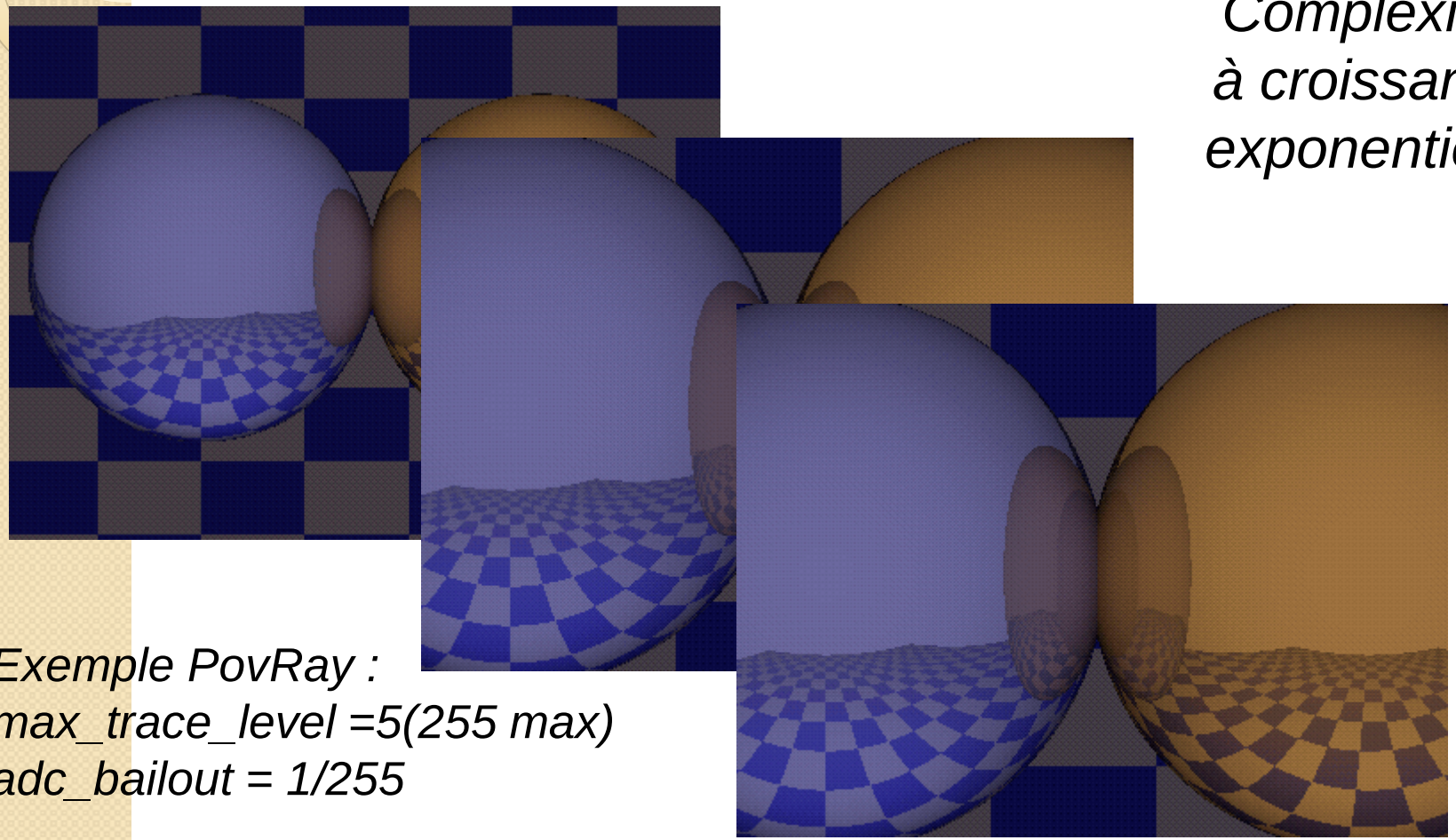
$$I_\lambda = I_{a\lambda} K_a O_{d\lambda} + \sum_{i=1}^{m} S_i \, f_{att\_i} \, I_{p\lambda i} \left[K_{di} O_{d\lambda} \left(\vec{N}.\vec{L}_i \right) + K_s \left(\vec{R}_i.\vec{V} \right)^n \right] + K_r I_{r\lambda} + K_t I_{t\lambda}$$

ambiante — sources — diffuse — spéculaire — refléchie — réfractée



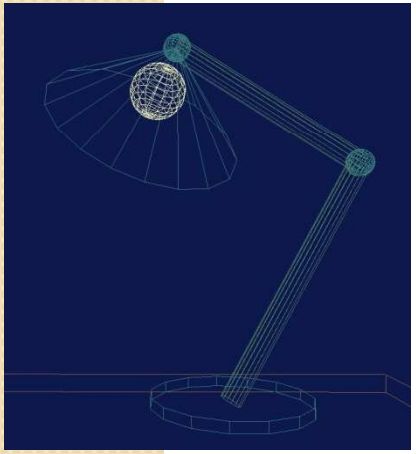
Jusqu'où ?

Complexité à croissance exponentielle

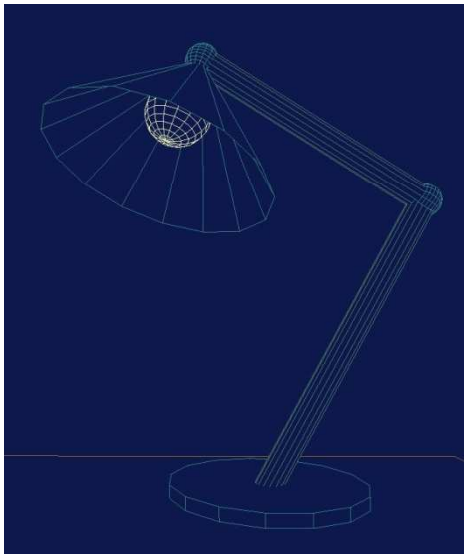


Exemple PovRay :
max_trace_level =5(255 max)
adc_bailout = 1/255

2.5 Parties cachées & rendu



Wireframe



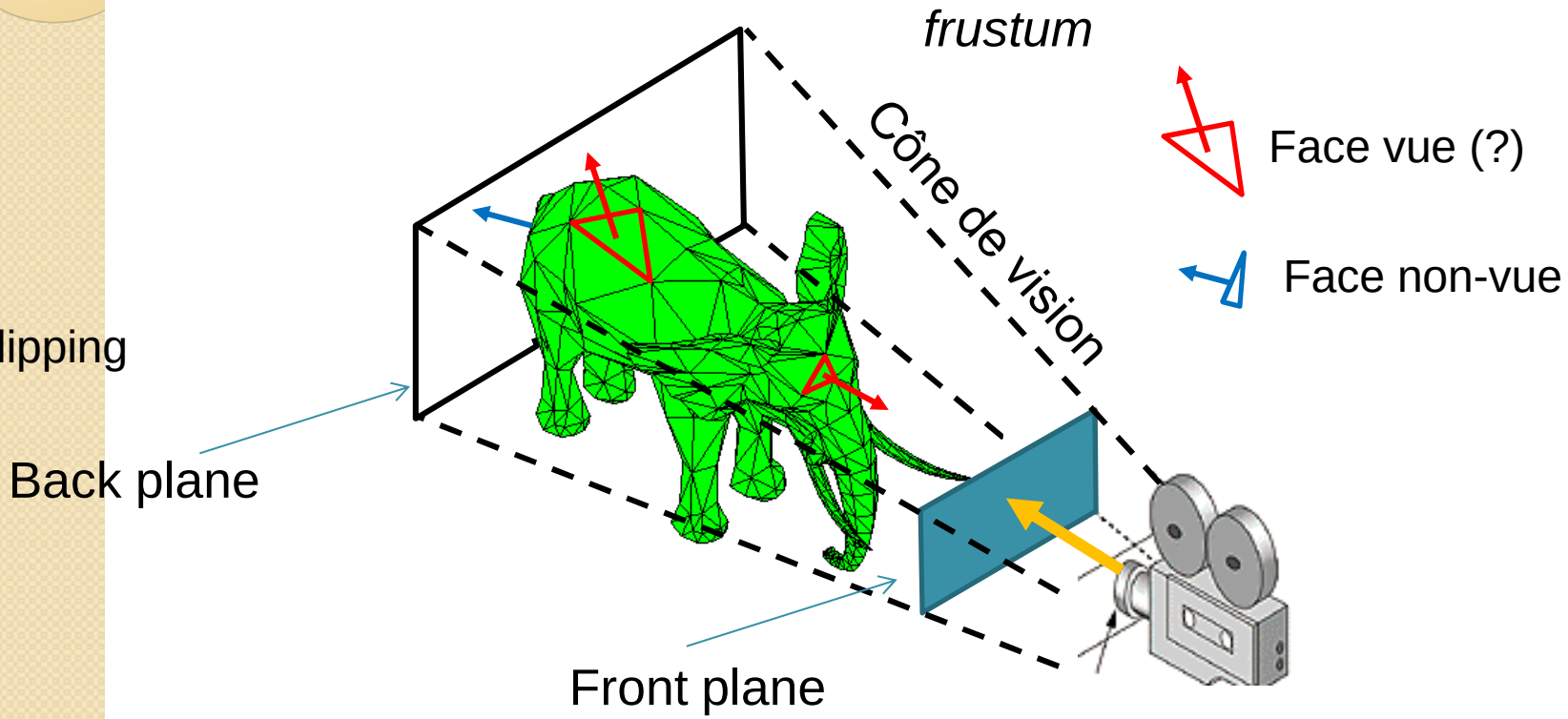
Hidden Lines

Flat Shading

Antialiasing

Gouraud Shading

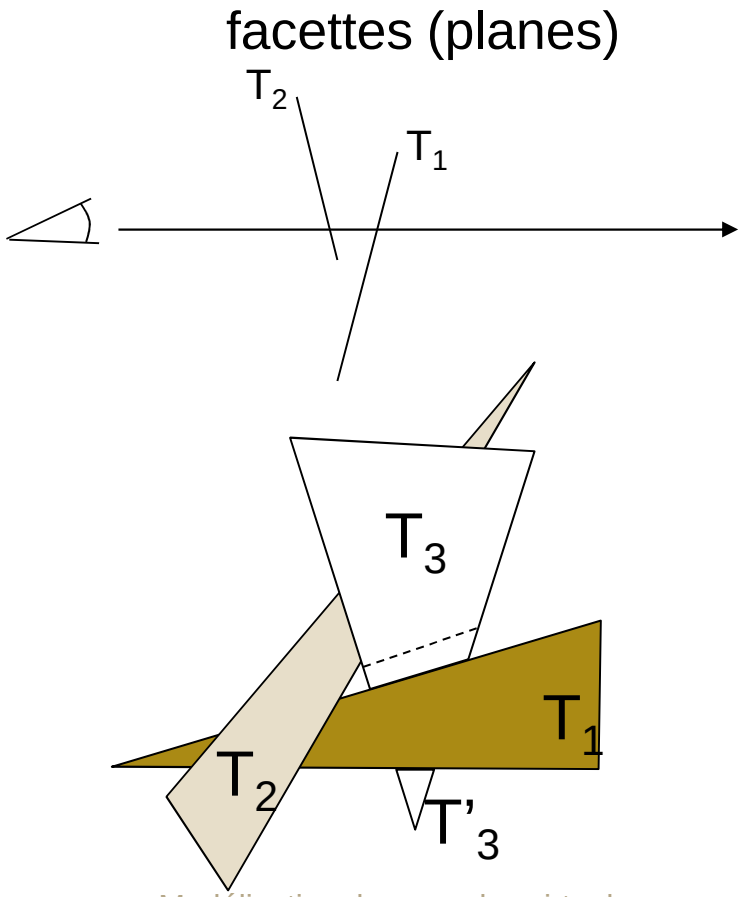
Le Clipping
et le Back face culling



Dans quel cas est-ce suffisant ?

Algorithme du peintre: ordre de profondeur

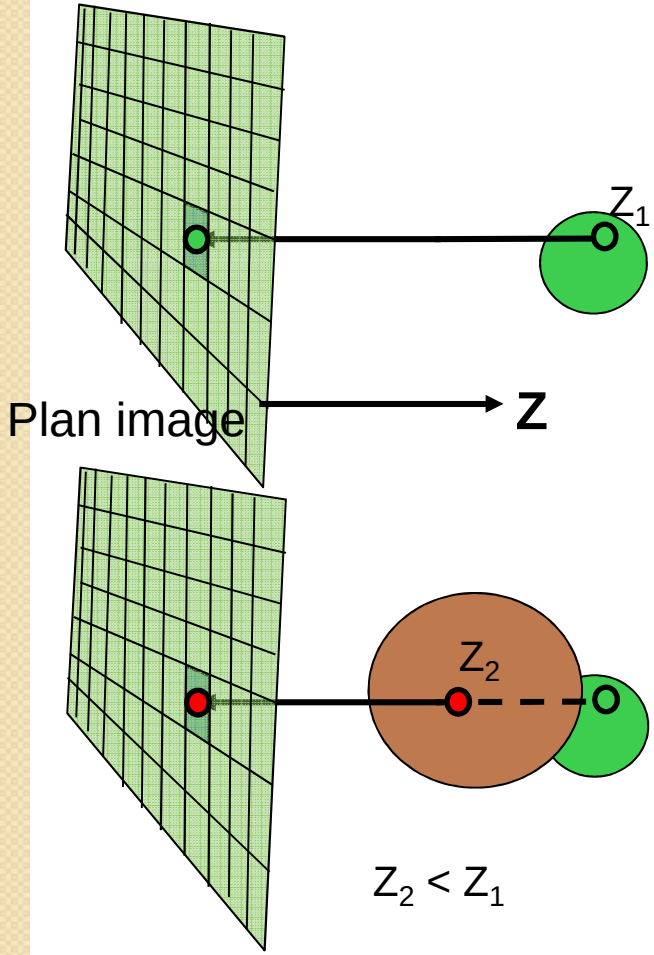
- Liste ordonnée de facettes
  - trier les facettes suivant leur profondeur : Z-max
  - afficher en Z-max décroissant
  - ambiguïtés : problème du recouvrement cyclique…



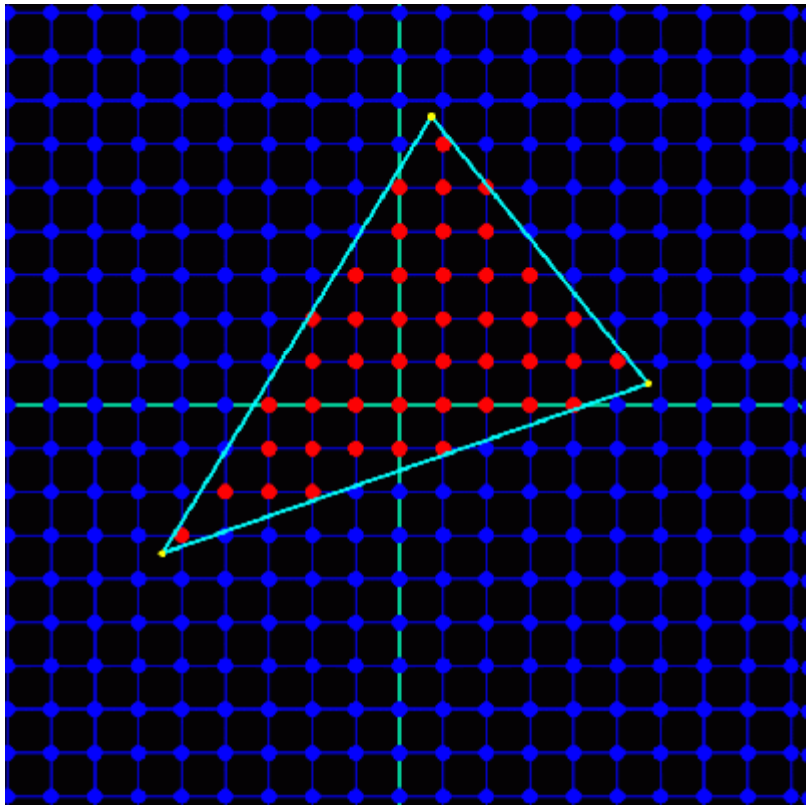
$T_2 < T_1$

Z-buffer* : ordre de profondeur

Distance min(Z_i) par pixel?



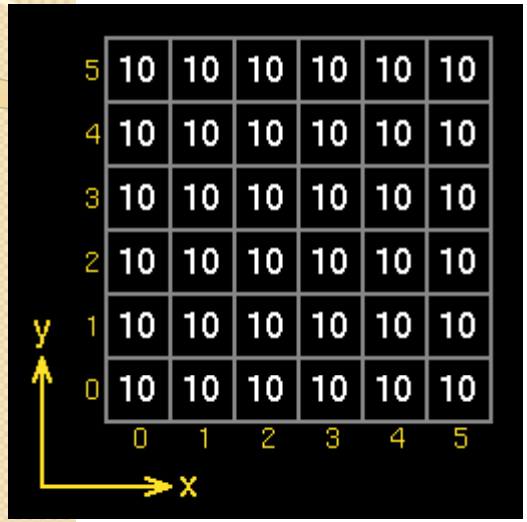
Pendant la « pixelisation »

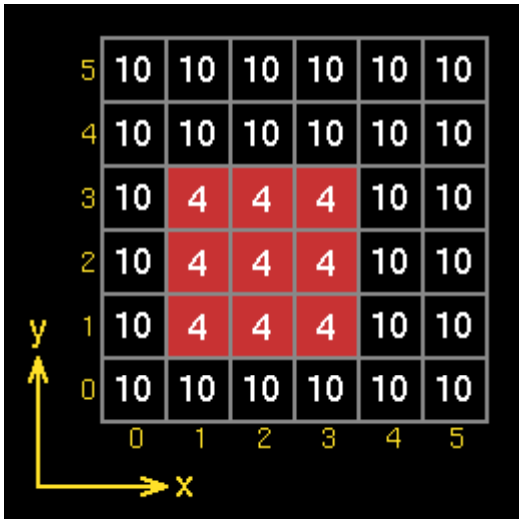


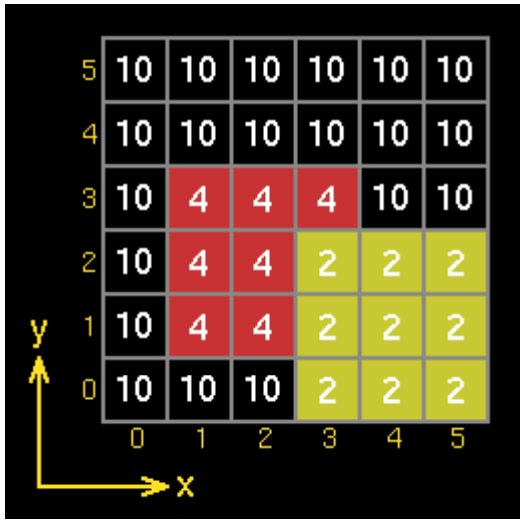
A la résolution écran

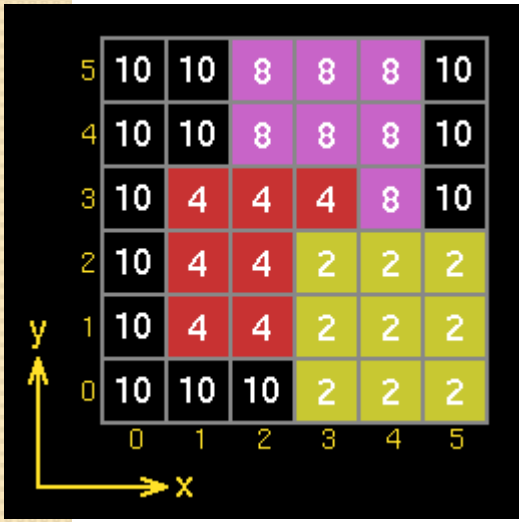
**Edwin Catmull 1974*

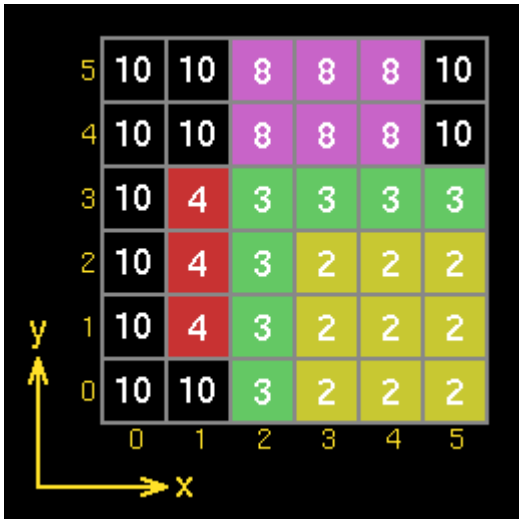
Z-buffer (exemple)











Z-buffer (algorithme*)

Stocker l'intensité des pixels mais aussi leur profondeur

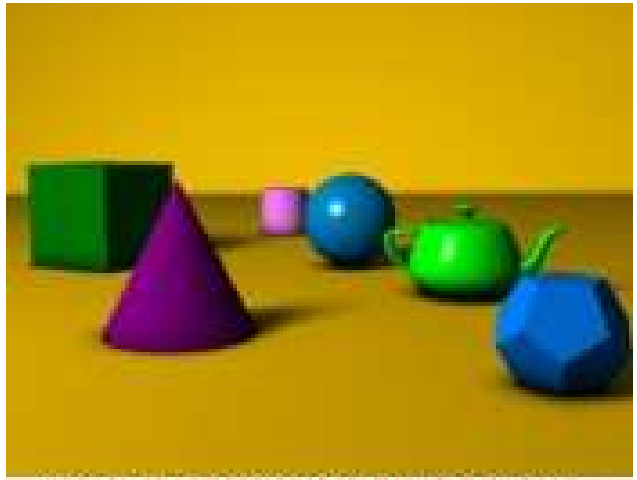
Idée

- Si Z du pixel < Z mémorisée on mémorise

Avantage : algorithme câblé (16 à 32 bits)

Inconvénient

- taille mémoire (codage)
- problème d'aliasing

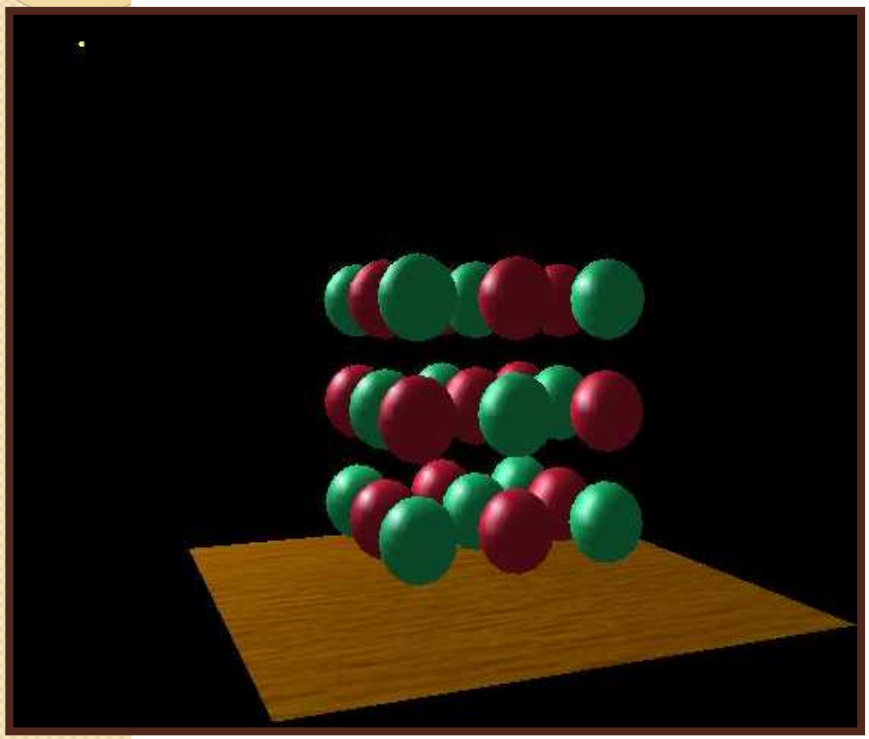


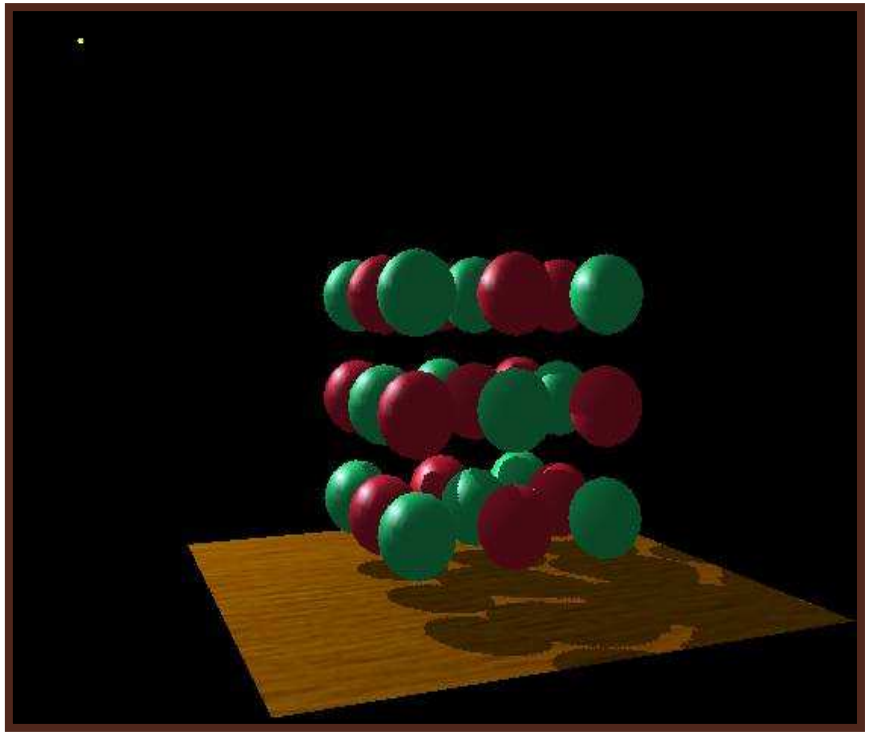
A simple three dimensional scene

Z-buffer representation

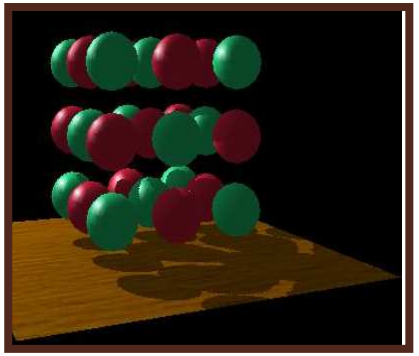
Edwin Catmull 1974

Ombres portées





Ombres portées

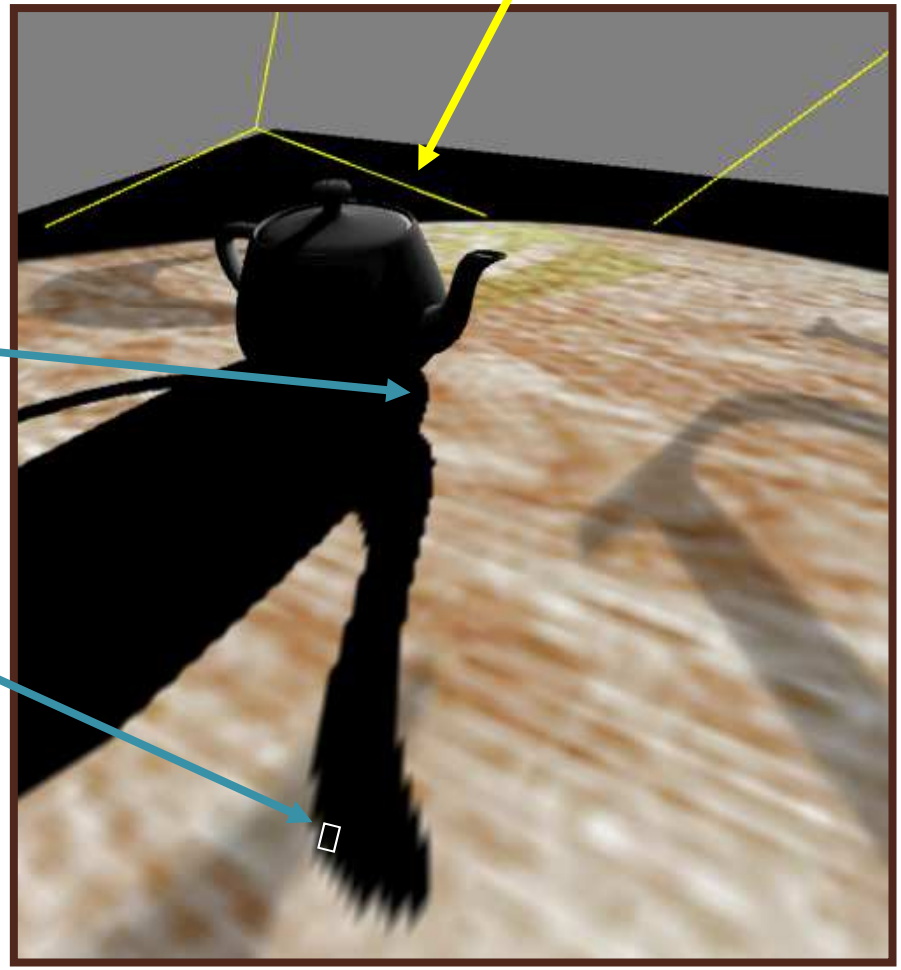


- Principe
  - réutiliser l'algorithme d'élimination des faces cachées depuis l'origine de la source lumineuse
  - créer des polygones d'ombre qui seront traités comme les autres facettes lors du second passage
- Coût
  - rendu en deux passes au lieu d'une

Texture d'ombre

problème d’ « aliasing » avec le Z-buffer

source de lumière



Contours très précis prés de la source

Contour très grossier loin de la source

Un seul pixel dans le Zbuffer de la source

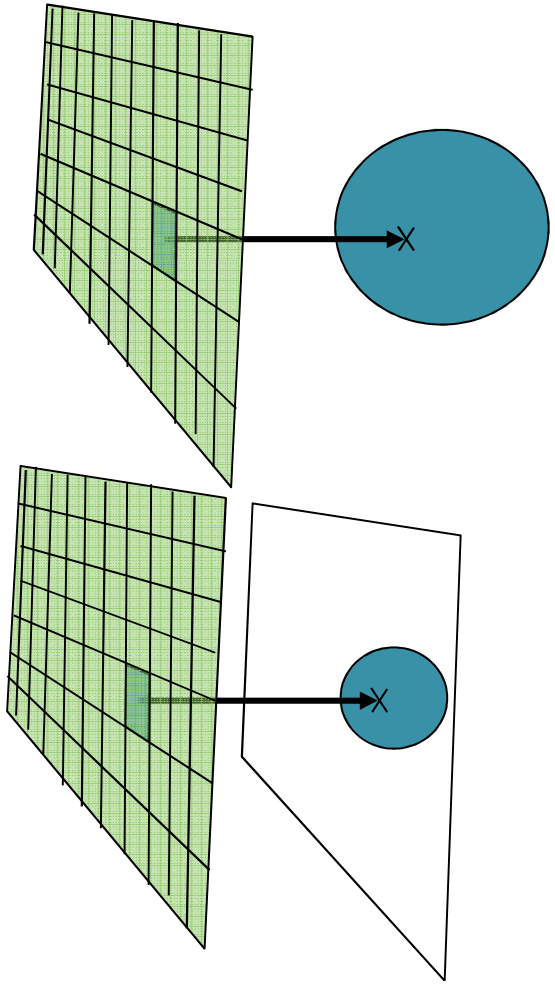
Solution : Irregular Z-buffer

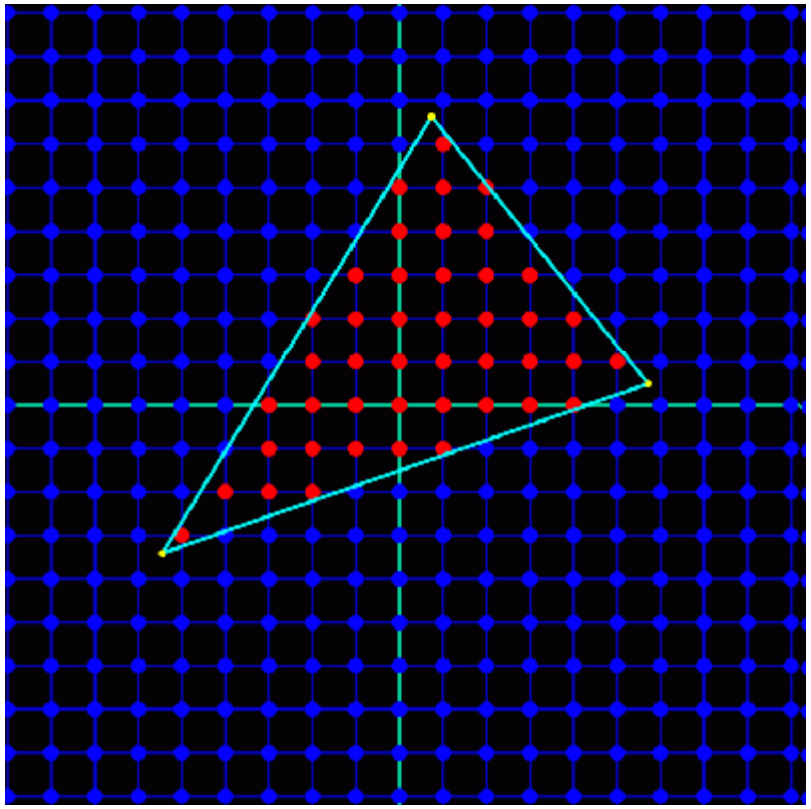
2.6. Rasterization

- Couleur du pixel ?

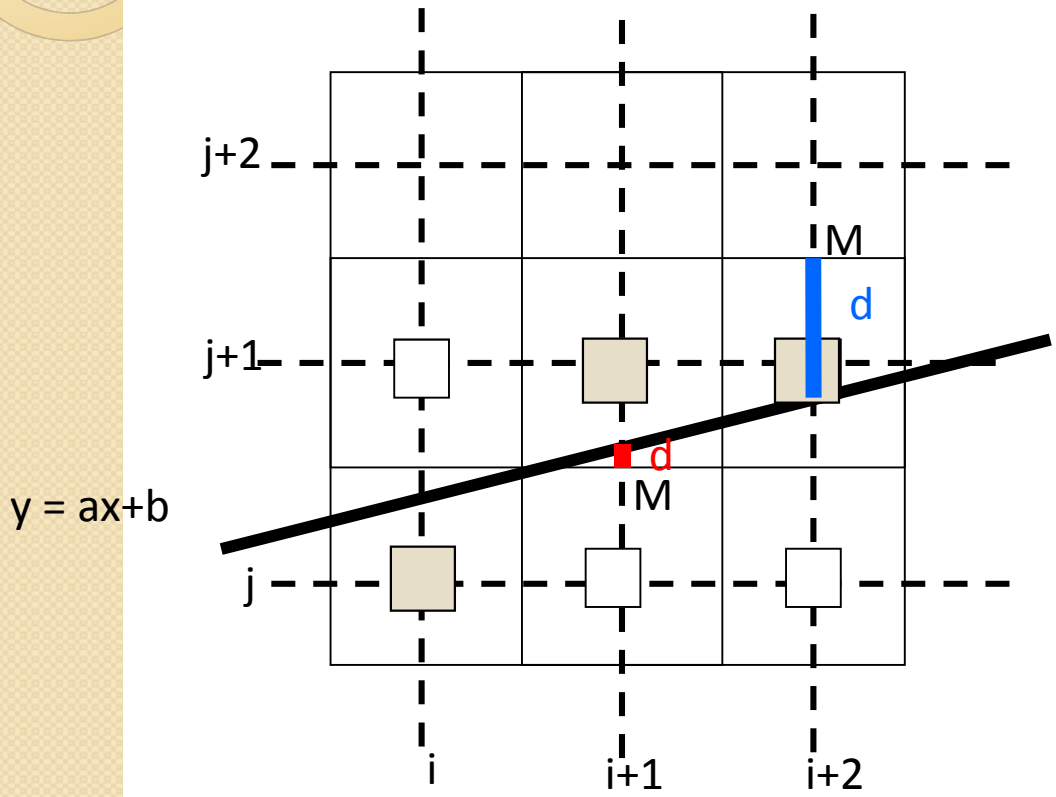
3D

2D





Tracé d'un segment
(Mid-point Algorithm)

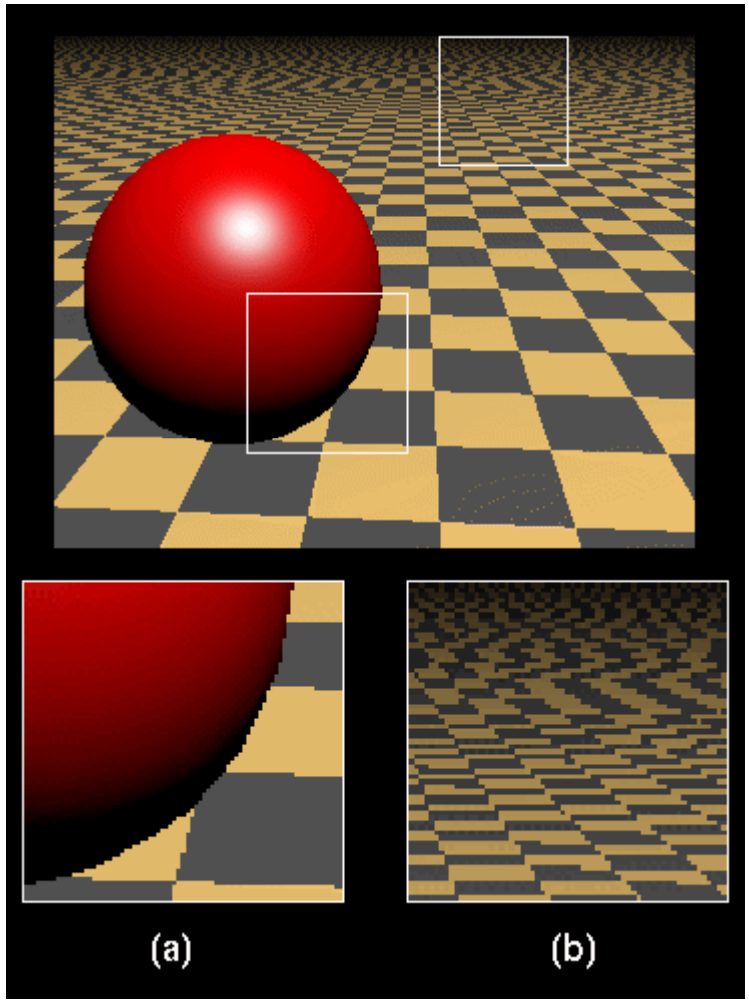


Cas : 0<a<1

```
WritePixel(i0,j0)
d=(a*(i0+1)+b)-(j0+1/2)
For(i=i0+1;i<=i1;i++){
    Si (d<0) d=d+a
    Sinon    j=j+1
             d=d+a-1
    WritePixel(i,j)
}
```

Effets d' « aliasing »

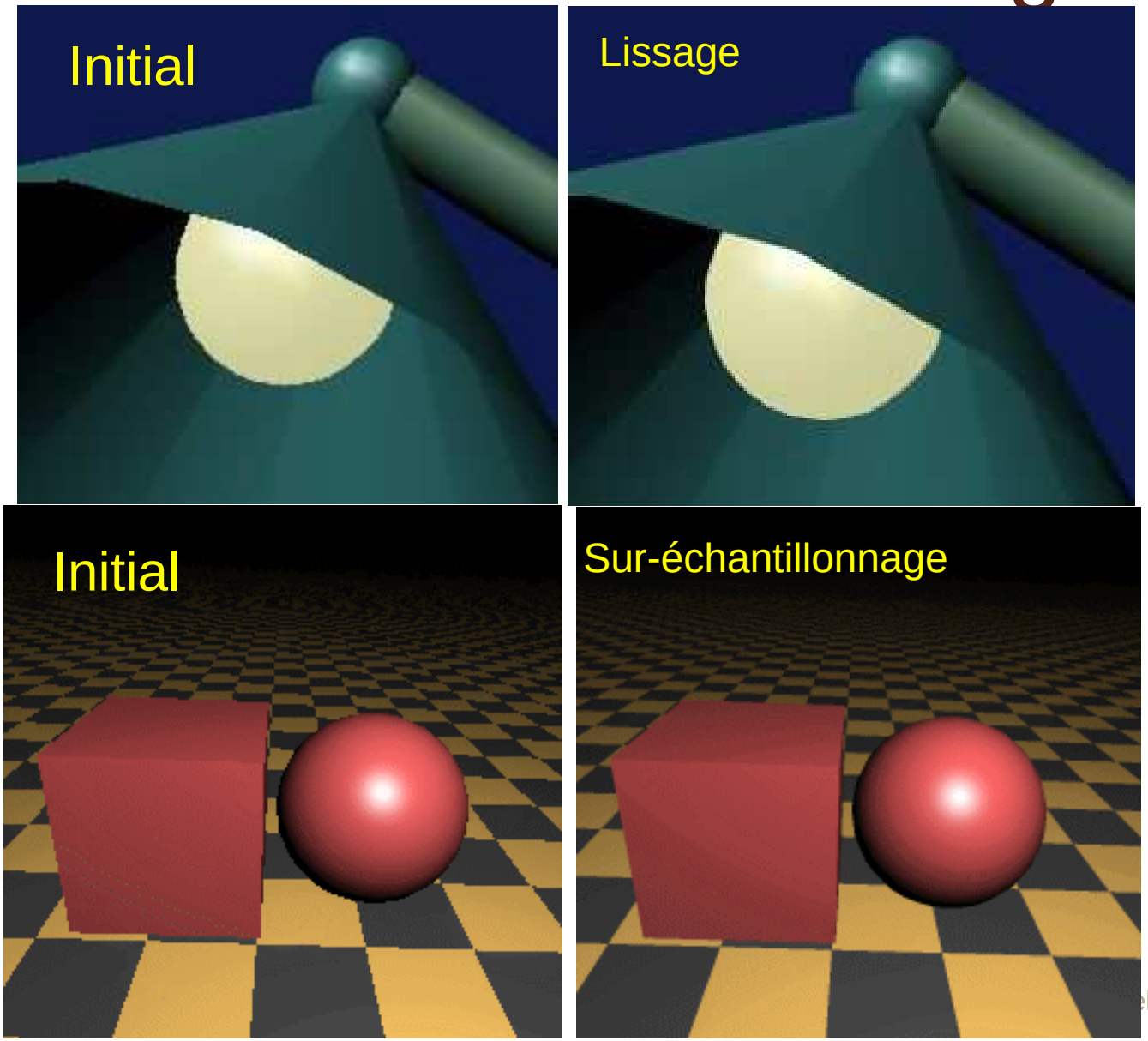
(a) frontières d'objet
(b) textures



Effets:

- *effets d'escalier sur les images*
- *petits objets entièrement ou partiellement masqués*
- *moirés sur les textures*

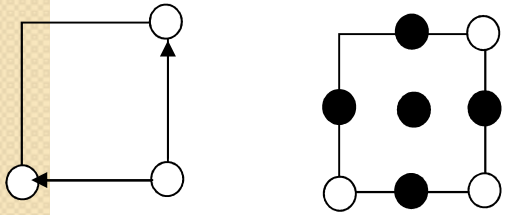
Méthodes « Anti-aliasing »



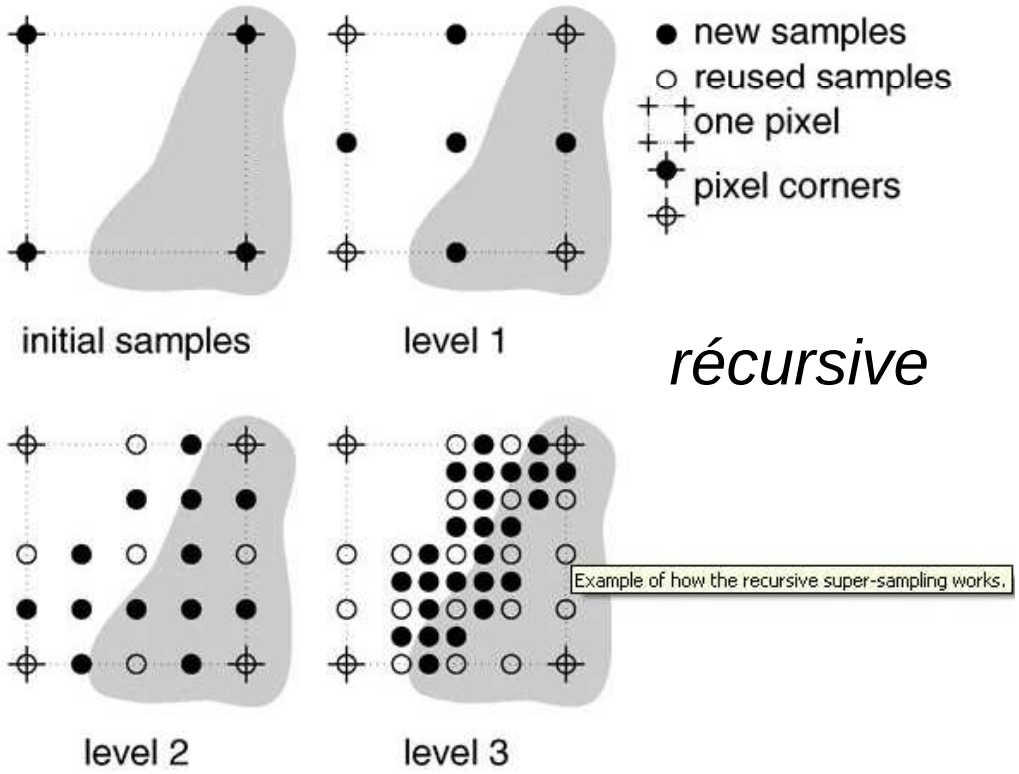
Antialiasing Adaptatif
(sous PovRay)

2 méthodes adaptatives

- diff=abs(R_1-R_2)+abs(G_1-G_2)+abs(B_1-B_2)
- seuil : [0:3], 0.3
- profondeur : 3



non-récursive



récursive