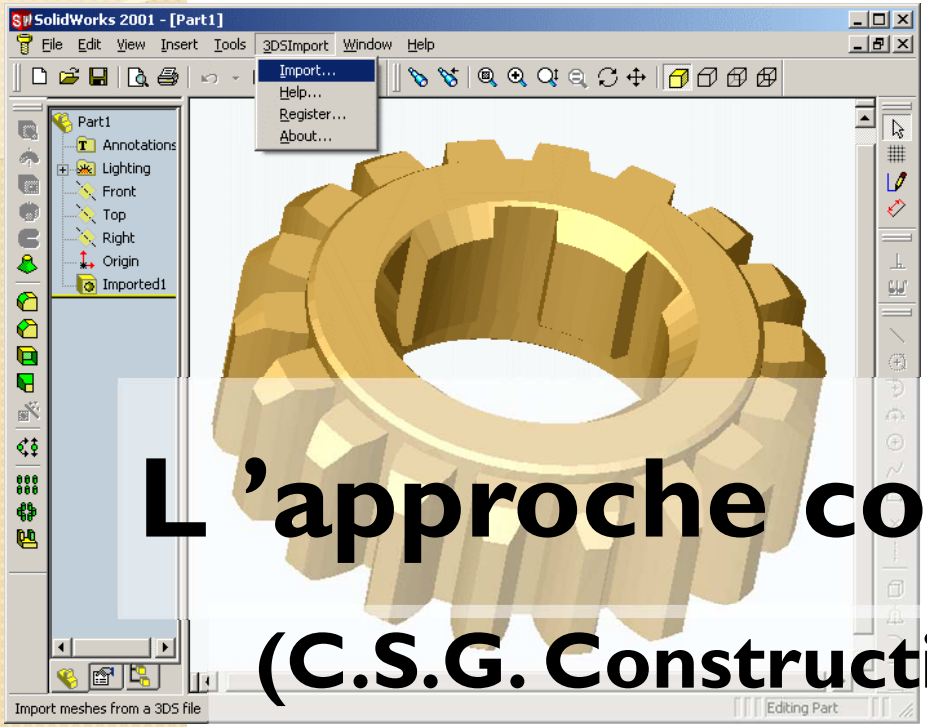
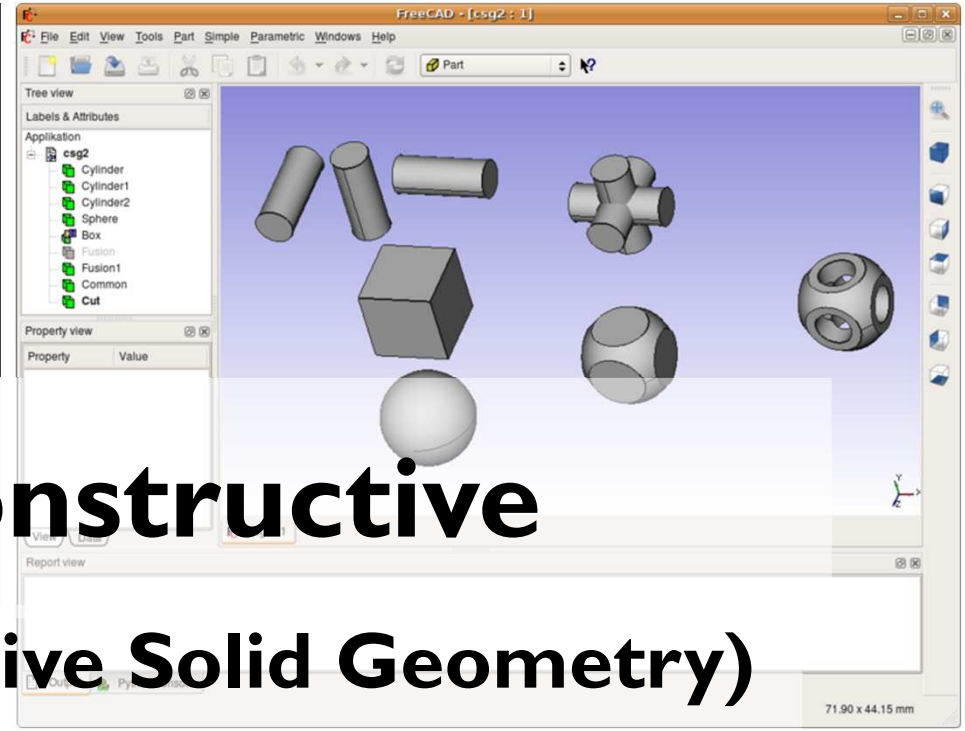
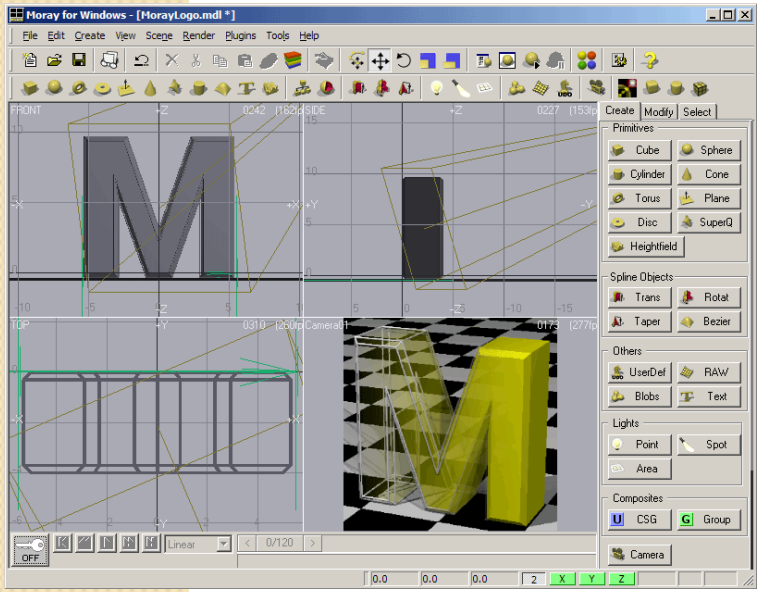
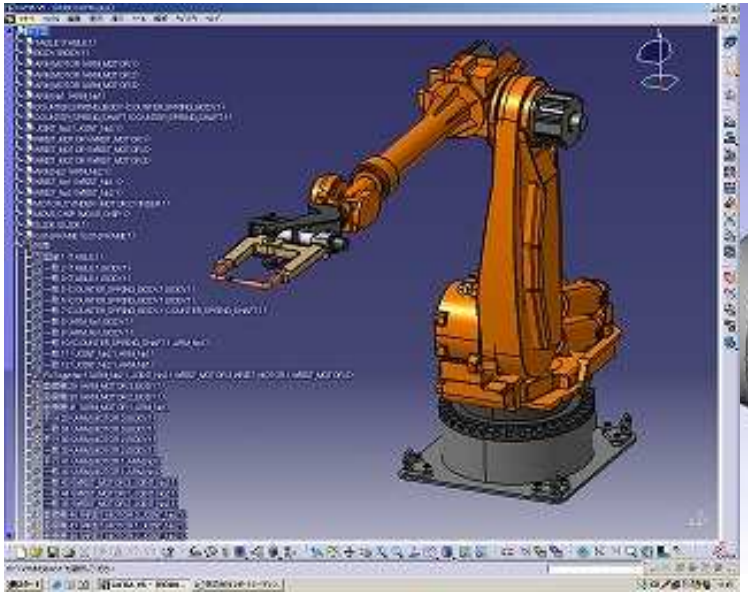


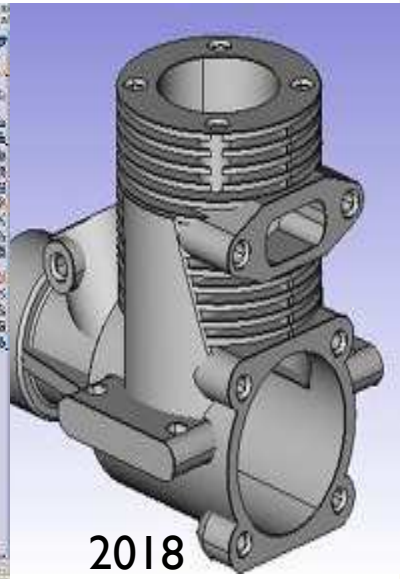


L 'approche constructive

(C.S.G. Constructive Solid Geometry)







Plan du cours

1. Scene Graph

   Les scene graph pour l'animation

2. Arbre CSG & algorithmes et structure

   Principe des algorithmes sur une structure arborescente

3. Primitives & champs de distances

   Fct, Quadriques, volumes élémentaires

4. Opérateurs booléens & variantes

   Union, intersection… Somme…

5. Transformations & CH

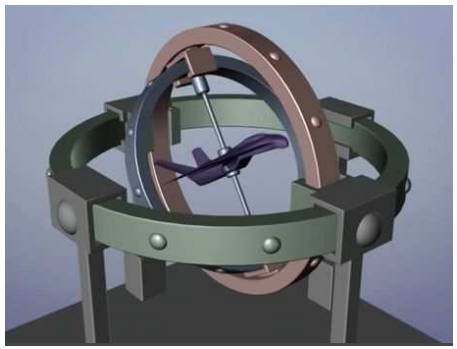
   Rotations et projections en coordonnées homogènes

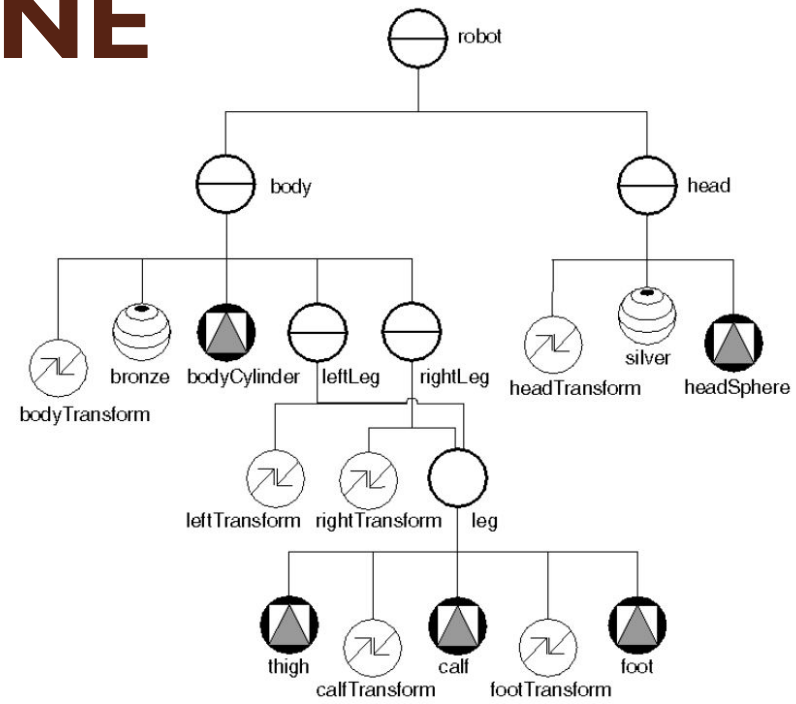
6. Autres approches constructives

   Représentation paramétrique; Courbes polynomiales, et construction arborescentes: fractales

I. LA SCÈNE

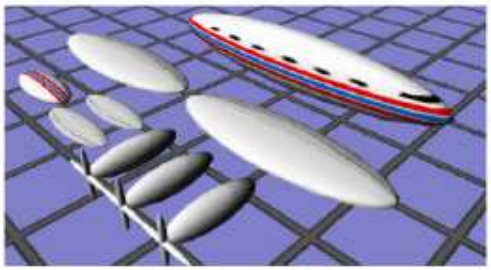
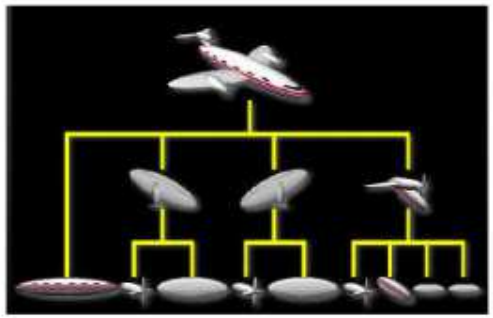
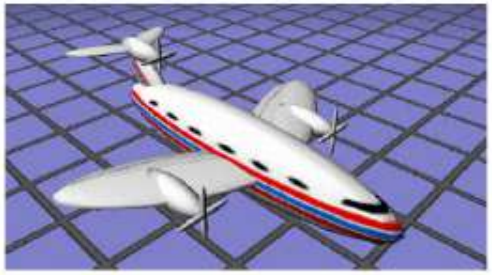
$$R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$





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 : Unity, Ogre3D, Cortona, Unity, CorelDraw…

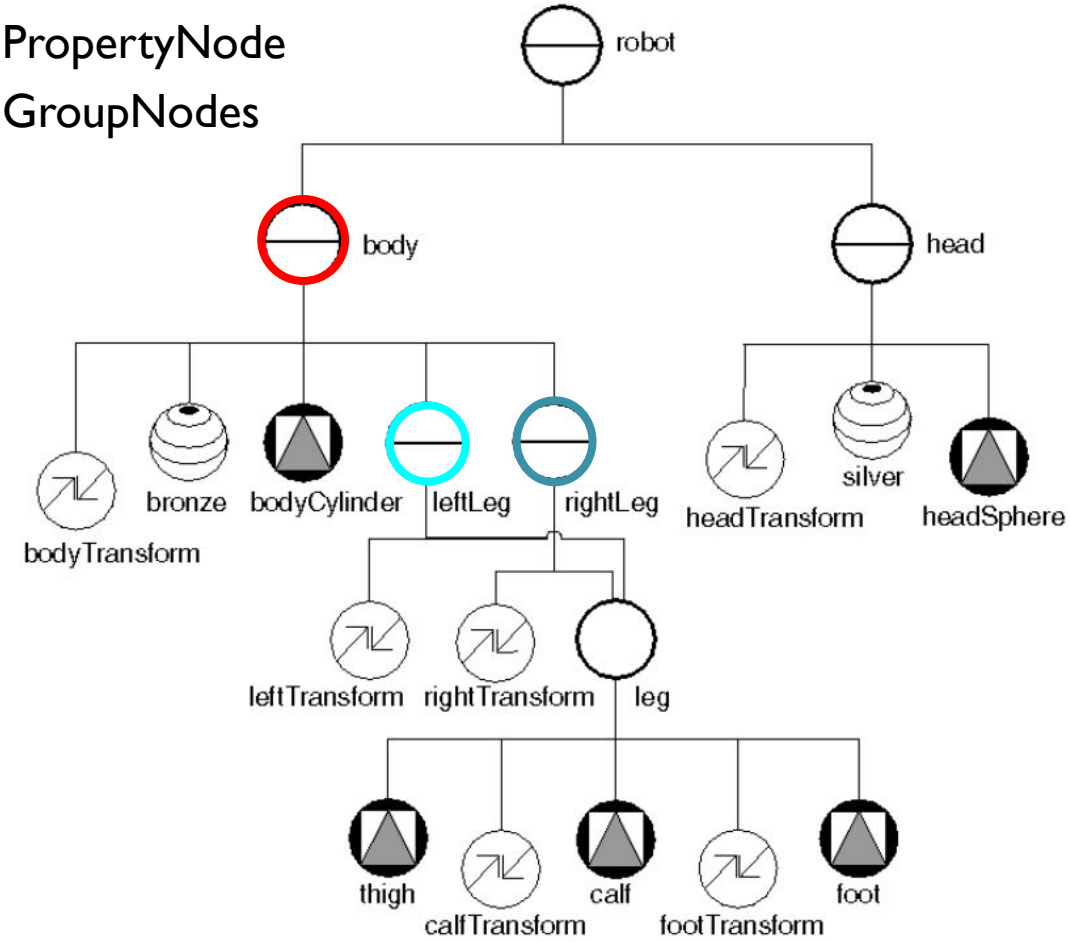




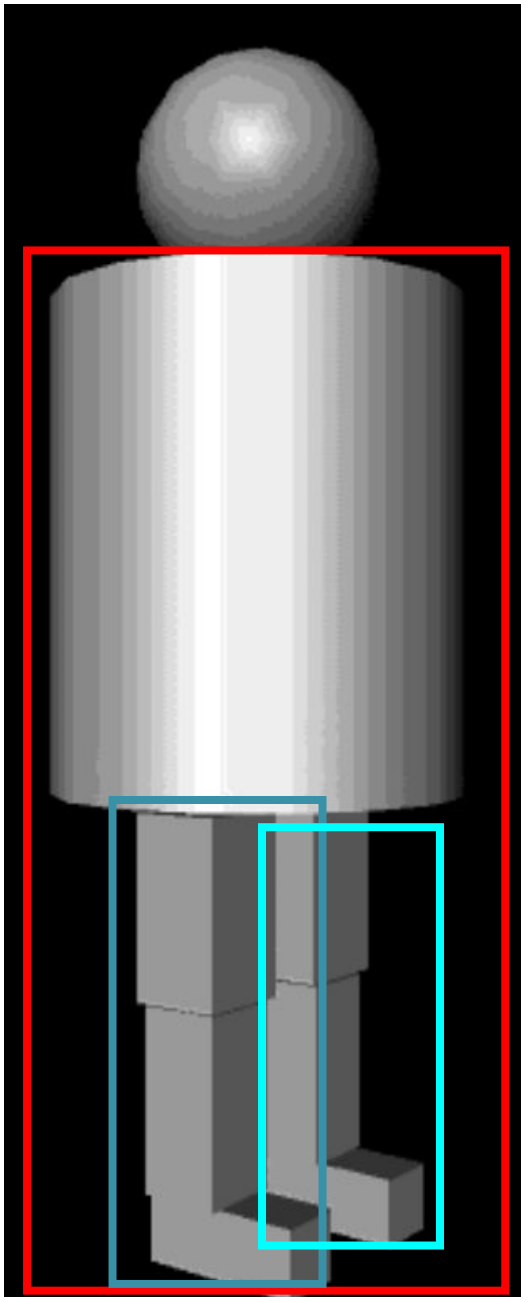
Scene Graph

ShapeNode

PropertyNode

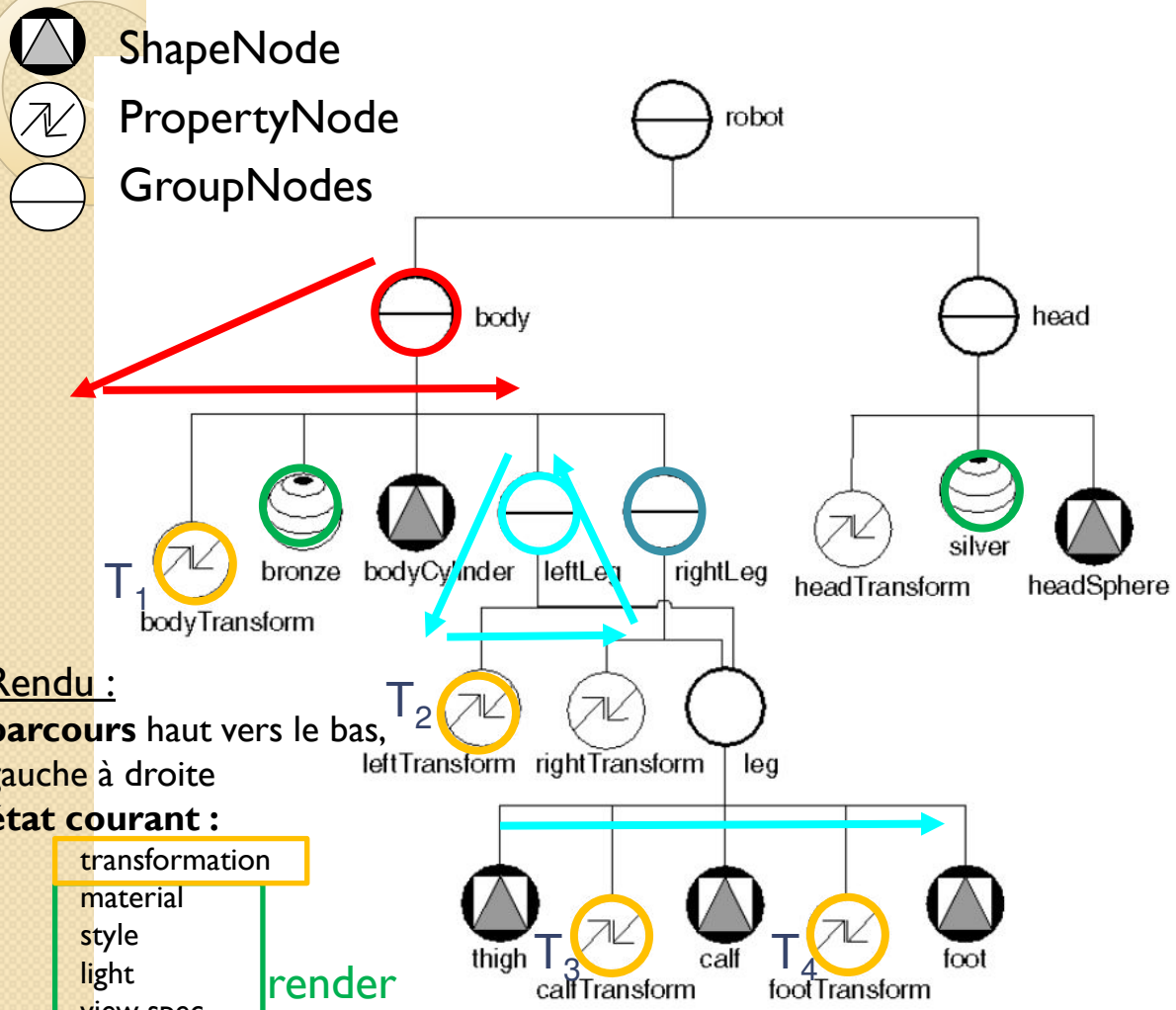
GroupNodes





Exemple Inventor

SG : le parcours de engines



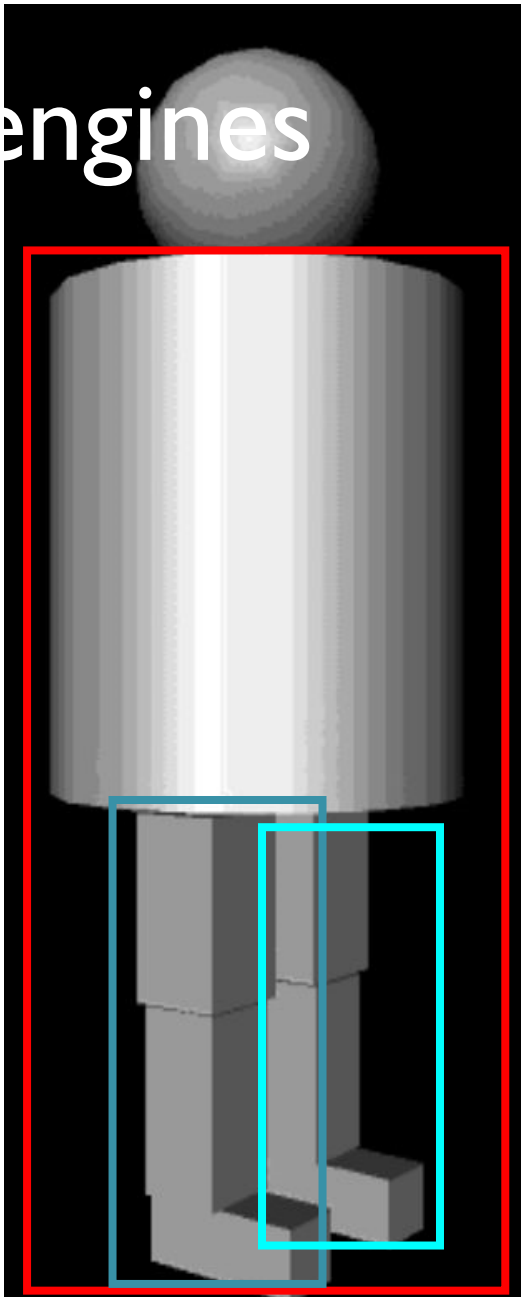
<u>Rendu :</u>

parcours haut vers le bas, gauche à droite

état courant :

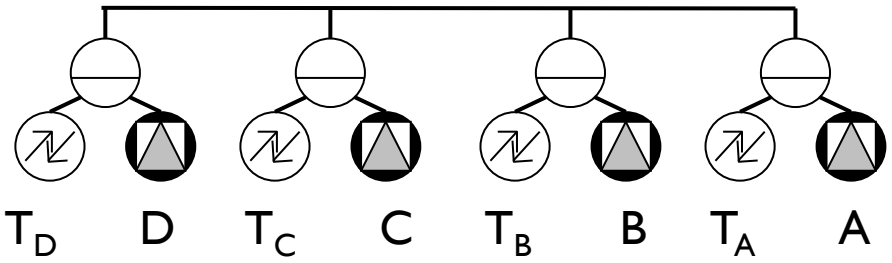
- transformation
- material
- style
- light
- view spec.
- …

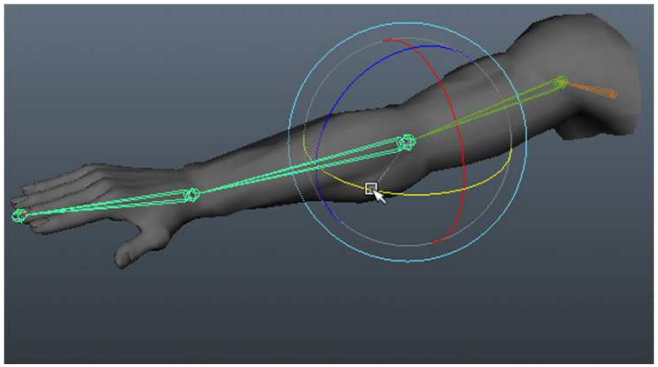
render

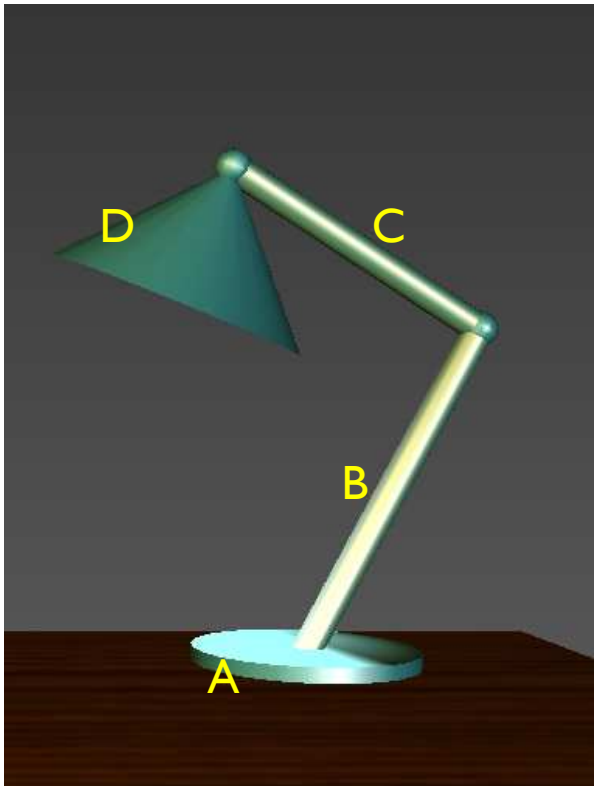


Positionnement et orientation absolus

Corriger l'arbre pour pouvoir animer la lampe

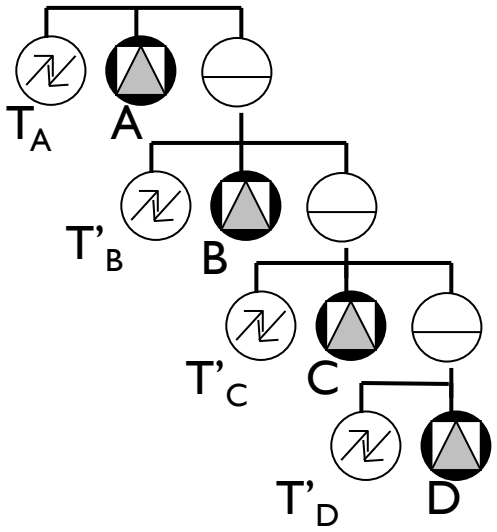






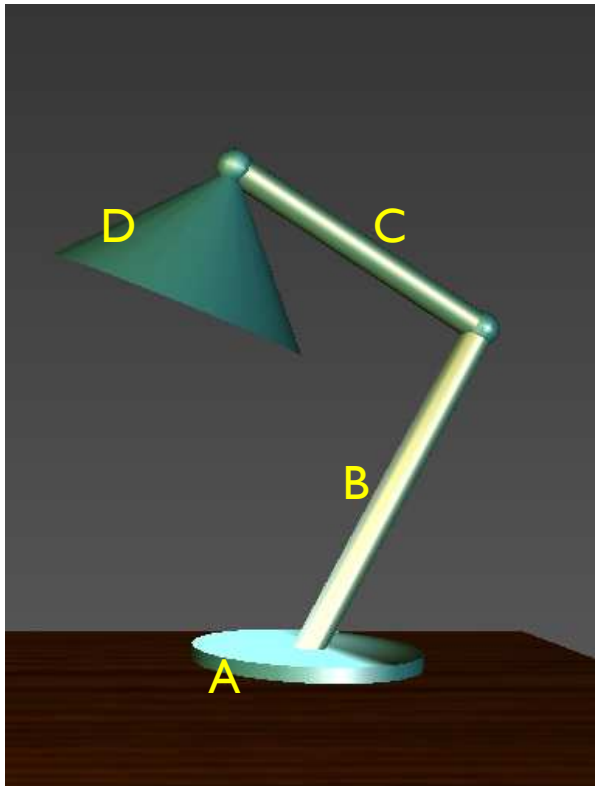
Positionnement et orientation relatifs

Arbre pour pouvoir animer la lampe



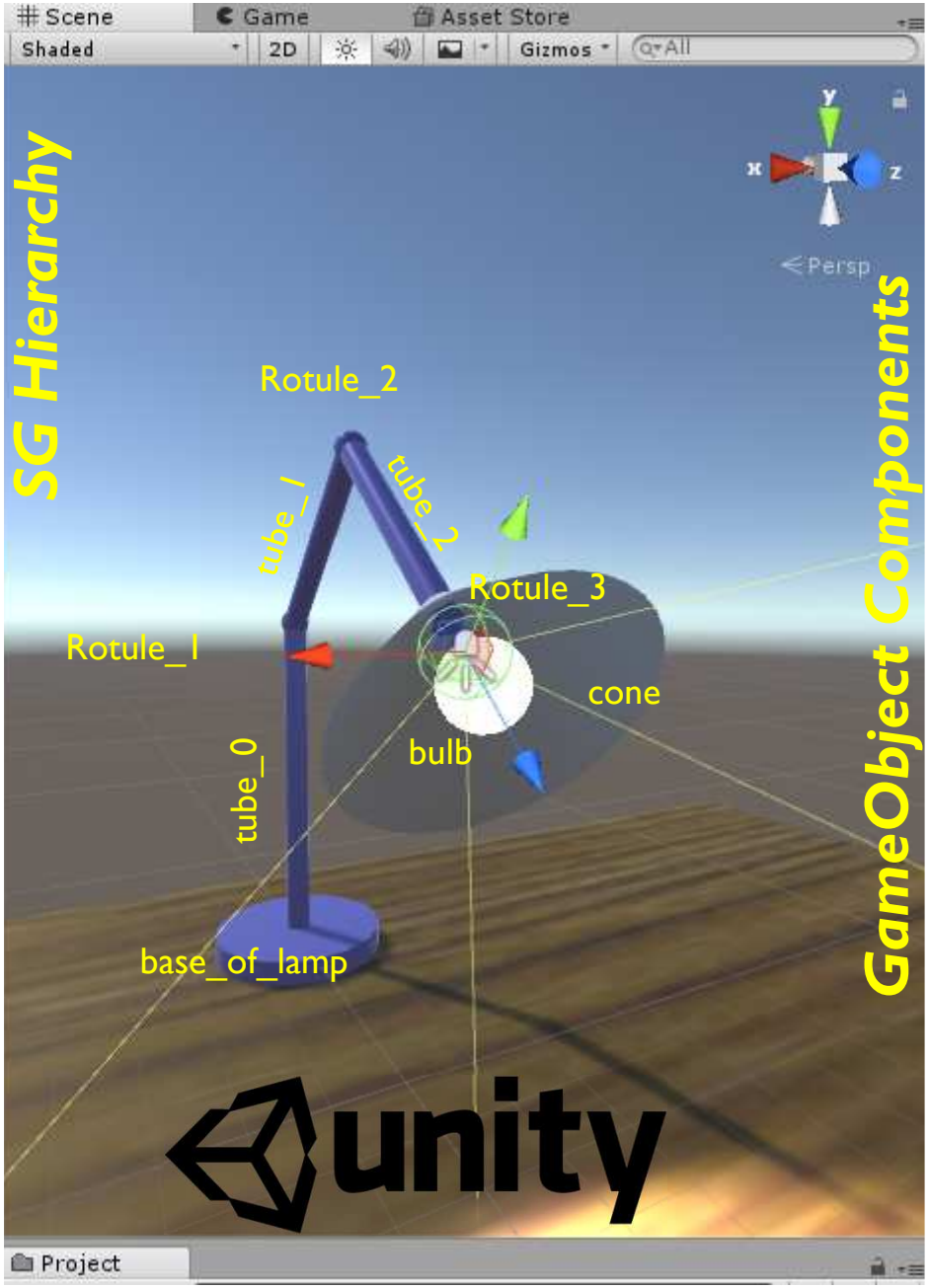
$T_A = T'_A$

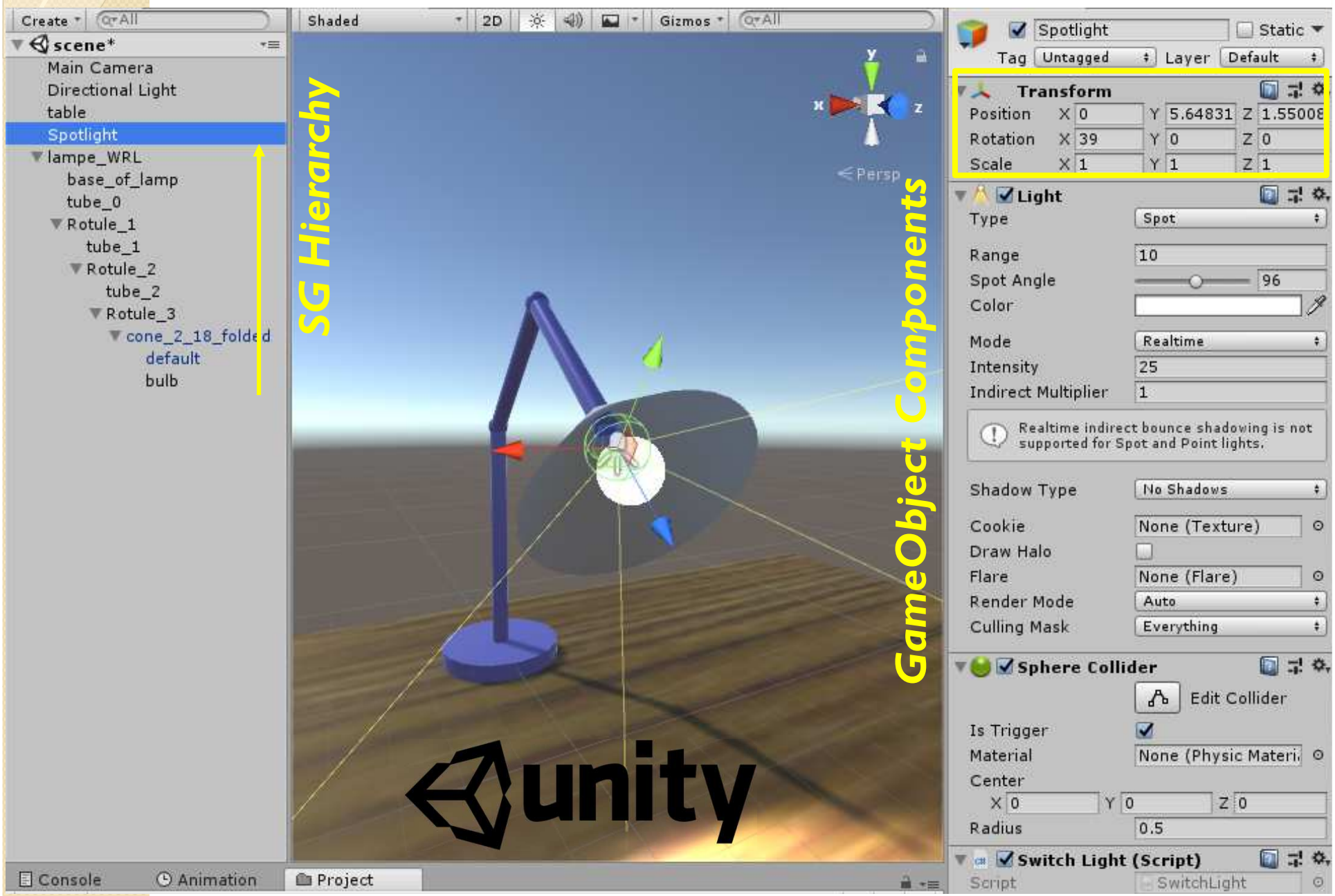
$T_B = T_A T'_B$

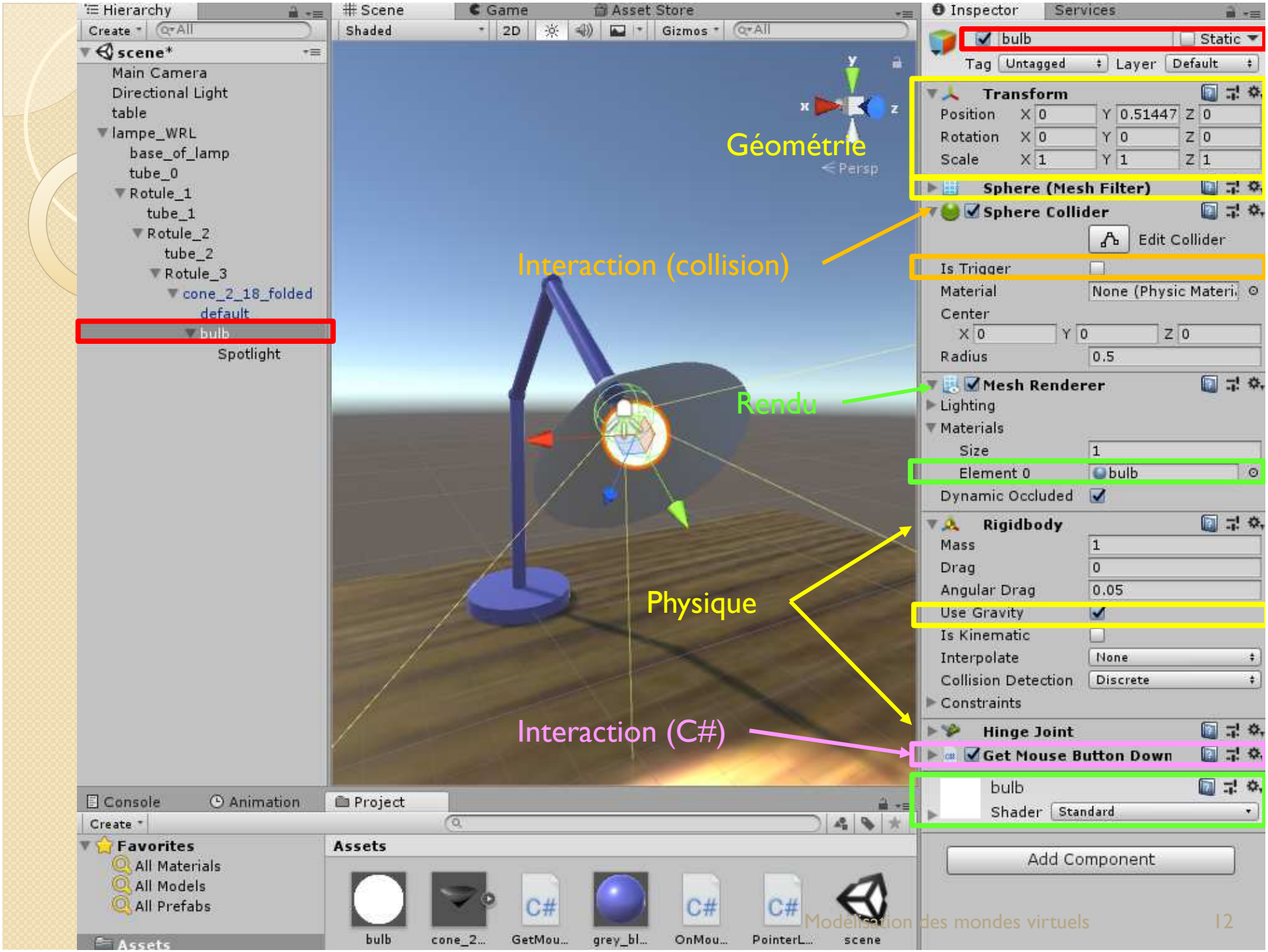


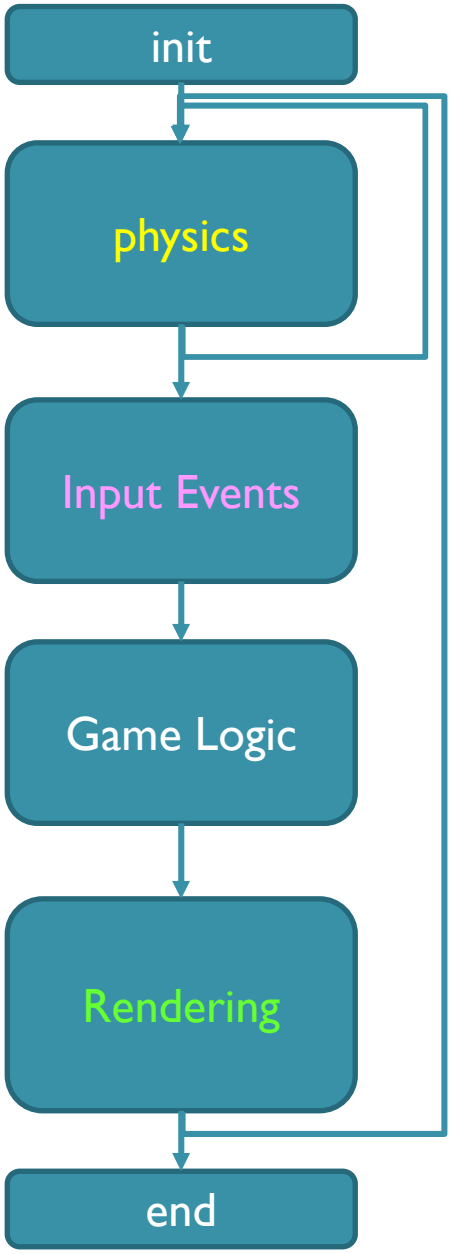
Calculez les transformations T'_Y à partir des T_Y

Montrez que : $T'_Y = (T'_X)^{-1} T_Y$


Scene
Game
Asset Store
Shaded
2D
Gizmos
All
Persp
SG Hierarchy
GameObject Components
Rotule_2
tube_1
tube_2
Rotule_3
Rotule_1
cone
bulb
tube_0
base_of_lamp
unity
Project


SG Hierarchy
GameObject Components
Create
All
scene*
Main Camera
Directional Light
table
Spotlight
lampe_WRL
base_of_lamp
tube_0
Rotule_1
tube_1
Rotule_2
tube_2
Rotule_3
cone_2_18_folded
default
bulb
Shaded
2D
Gizmos
Persp
unity
Console
Animation
Project
Spotlight
Static
Tag
Untagged
Layer
Default
Transform
Position X 0 Y 5.64831 Z 1.55008
Rotation X 39 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


Hierarchy
Create
scene*
Main Camera
Directional Light
table
lampe_WRL
base_of_lamp
tube_0
Rotule_1
tube_1
Rotule_2
tube_2
Rotule_3
cone_2_18_folded
default
bulb
Spotlight
Scene
Game
Asset Store
Shaded
2D
Gizmos
Persp
Géométrie
Interaction (collision)
Rendu
Physique
Interaction (C#)
Inspector
Services
bulb
Static
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
Sphere (Mesh Filter)
Sphere Collider
Edit Collider
Is Trigger
Material
None (Physic Materi
Center
Radius
0.5
Mesh Renderer
Lighting
Materials
Size
1
Element 0
bulb
Dynamic Occluded
Rigidbody
Mass
Drag
Angular Drag
0.05
Use Gravity
Is Kinematic
Interpolate
None
Collision Detection
Discrete
Constraints
Hinge Joint
Get Mouse Button Down
bulb
Shader
Standard
Add Component
Console
Animation
Project
Create
Favorites
All Materials
All Models
All Prefabs
Assets
bulb
cone_2...
GetMou...
grey_bl...
OnMou...
PointerL...
scene



Start()

FixedUpdate()

OnTrigger()
OnCollision()

OnMouse()

UpDate()

LateUpdate()

WaitForEndOfFrame()
OnApplicationPause()

OnApplicationQuit()

unity

C#

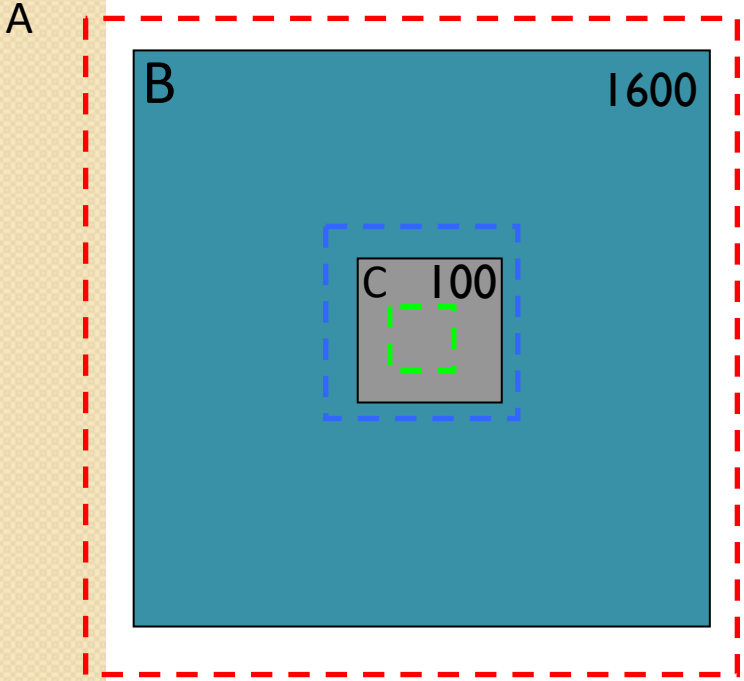
```
…using UnityEngine;

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

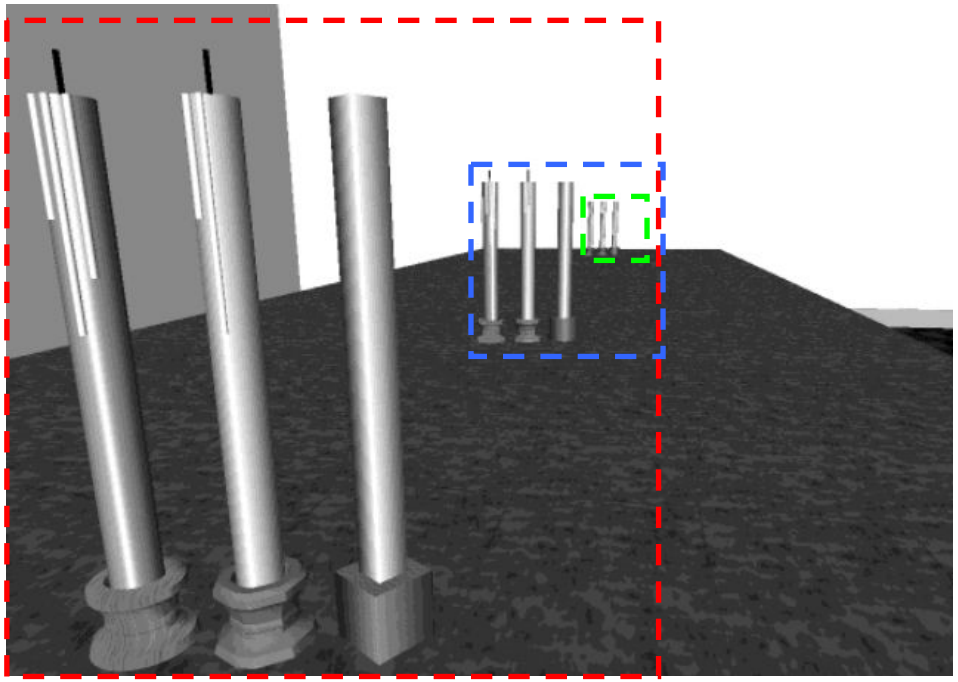
Du GameObject qui contient le script

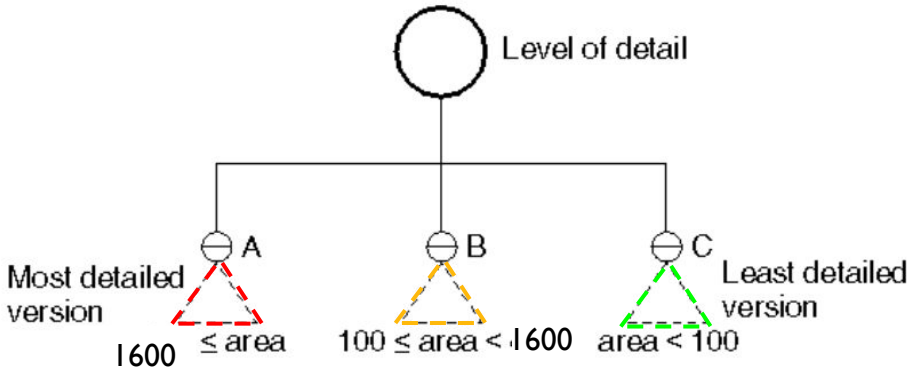
Switch Node : Level Of Detail (LOD)

Mécanisme automatique de rendu multi-résolution

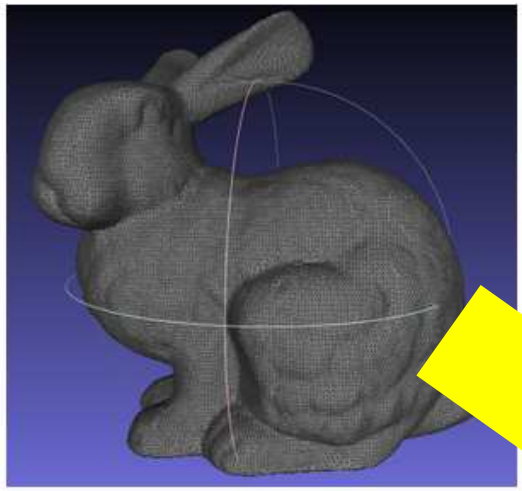


ScreenArea(1600,100)

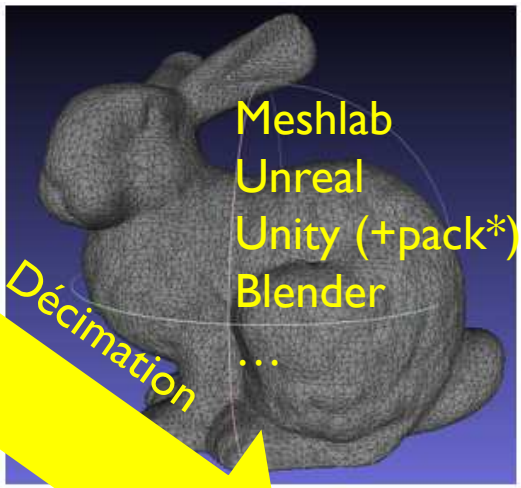




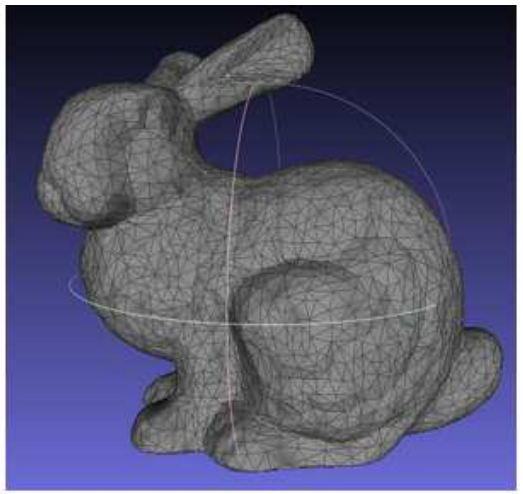
LOD dans la pratique



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



V : 1049… 34, BE : 154, H : 5



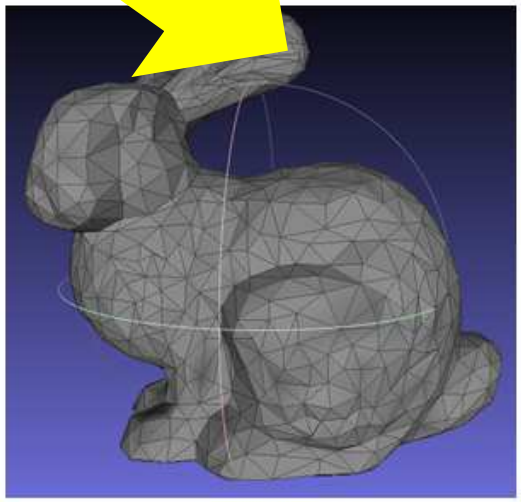
V : 3155, E: 9407, F : 6249, BE : 67, H : 5

MAKE YOUR GAME RUN SMOOTH - Unity LOD Tutorial

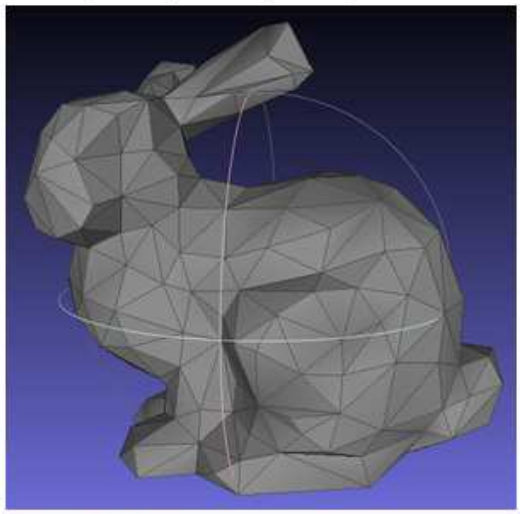
Brackeys

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



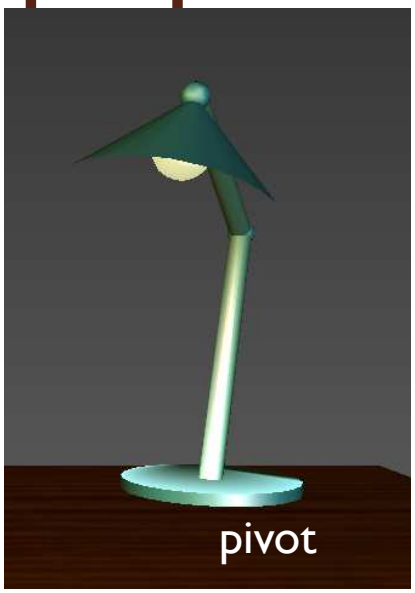
V : 952, E: 2829, F : 1874, BE : 36, H : 5

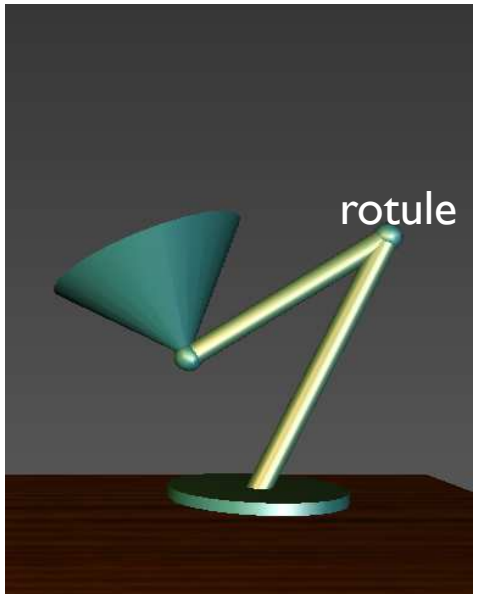


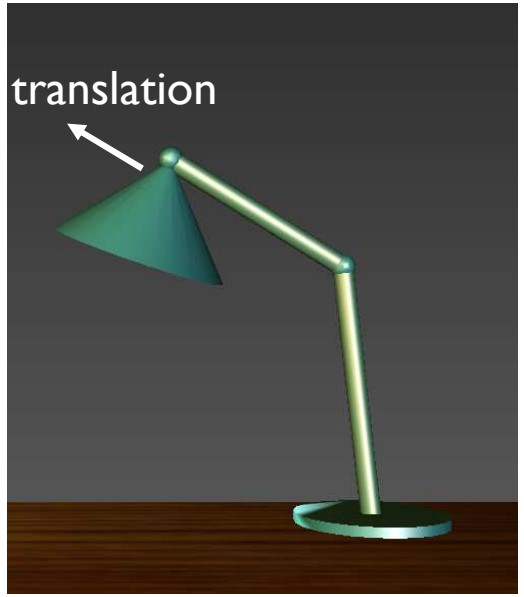
V : 289, E: 854, F : 562, BE : 22, H : 5

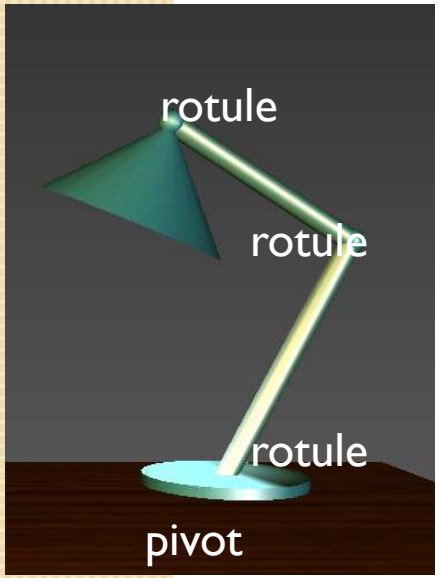
*UltimateGameTools pack

Scène graph pour l'animation



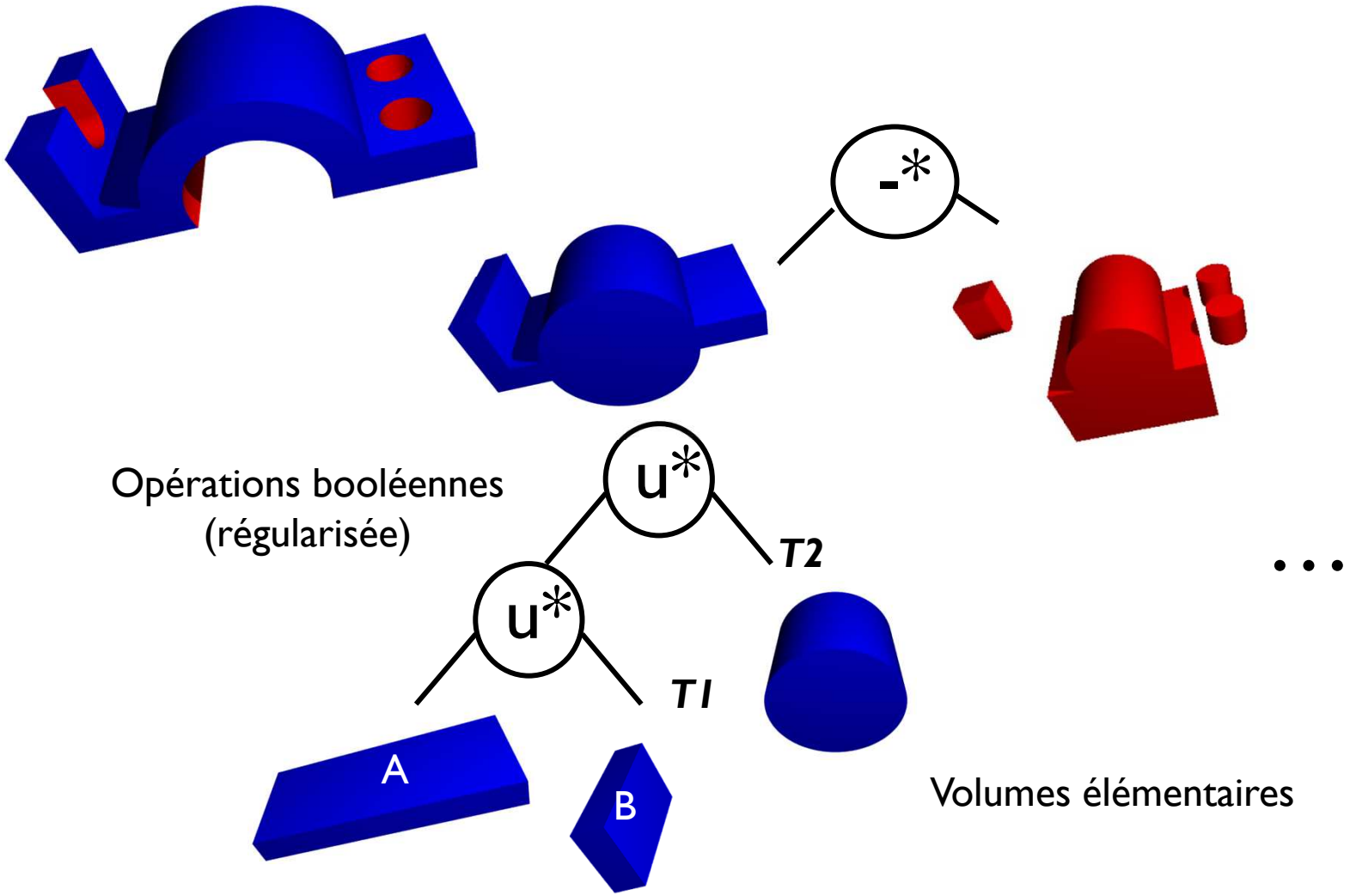




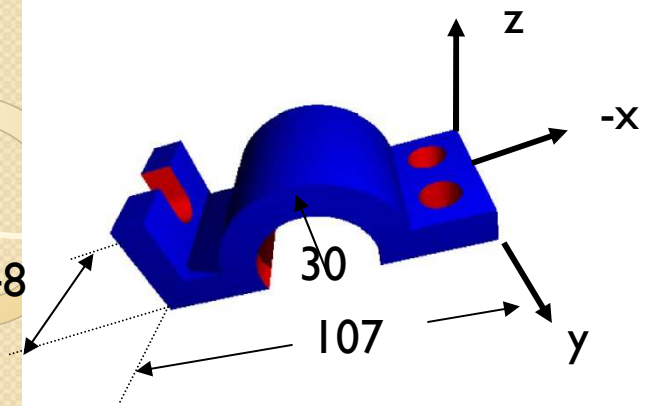


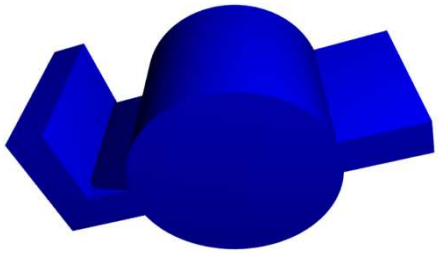
- Une bonne hiérarchie pour l'animation
- Des « Sensor Node » pour l'interaction
- Démo VRML ou Démo UNITY…

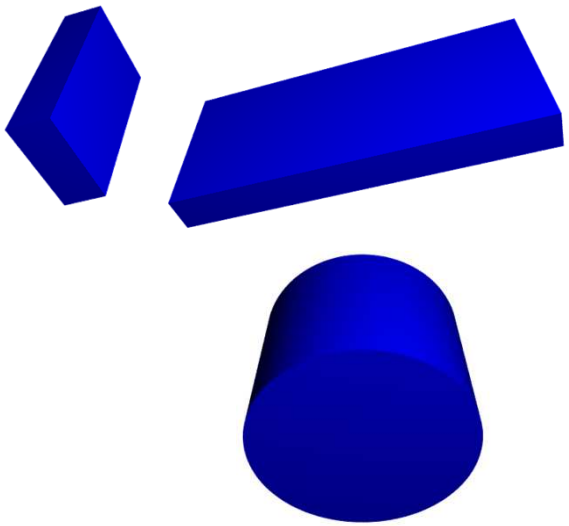
2. Principe des arbres booléens de contruction



Exemple

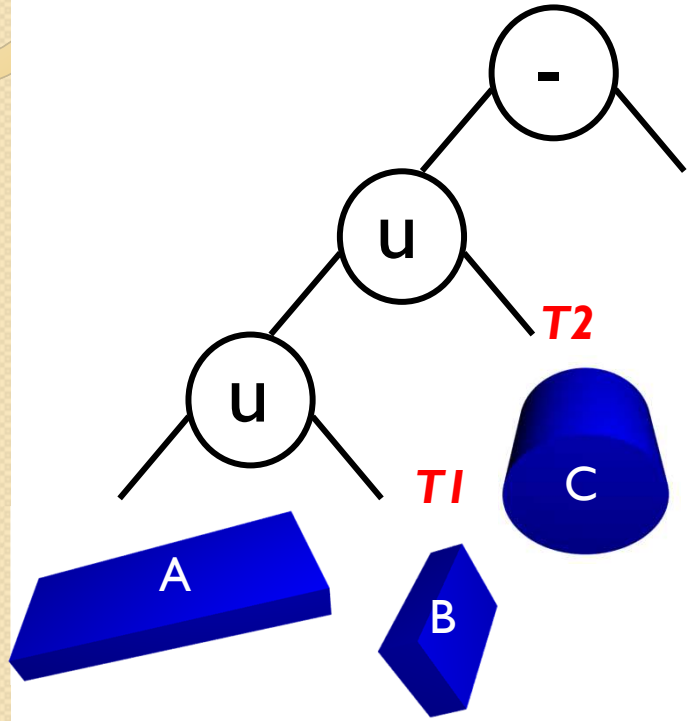






```
Union {
    union {
        box { <0,0,0> <107,48,11> }
        box { <0,0,0> <11,48,25> } translate <96,0,11>
    }
    cylinder { <0,0,0> <0,48,0> 30 translate <53.5,0,5.5> }
}
```

Primitives et transformations



- les primitives
- les transformations
- la structure d'arbre avec opérateurs

```
Union {
    union {
      box { <0,0,0> <107,48,11> }
      box { <0,0,0> <11,48,25> } translate <96,0,11>
      }
     cylinder { <0,0,0> <0,48,0> 30 translate <53.5,0,5.5> }
}
```

Principe des algorithmes de base

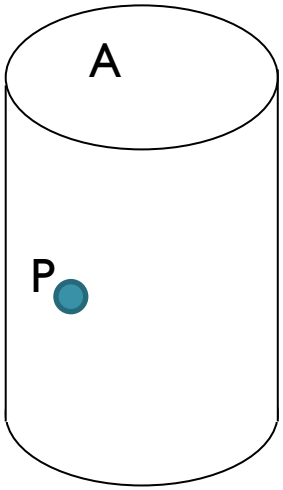
- Questions géométriques élémentaires

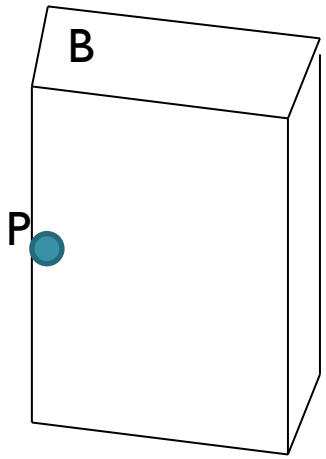
1. Un point (x,y,z) est-il à l 'intérieur ou l 'extérieur d 'un objet ?
2. Position de l 'intersection d 'une droite avec le modèle ?
3. Aire ou volume d'un objet ?

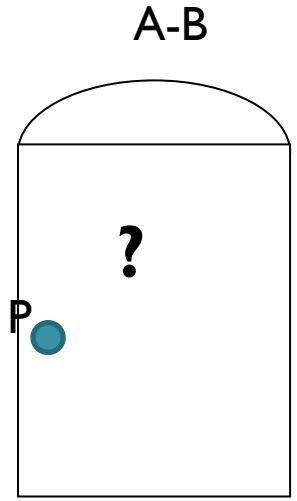
...

Localisation d'un point

Principe : réflexion sur un exemple





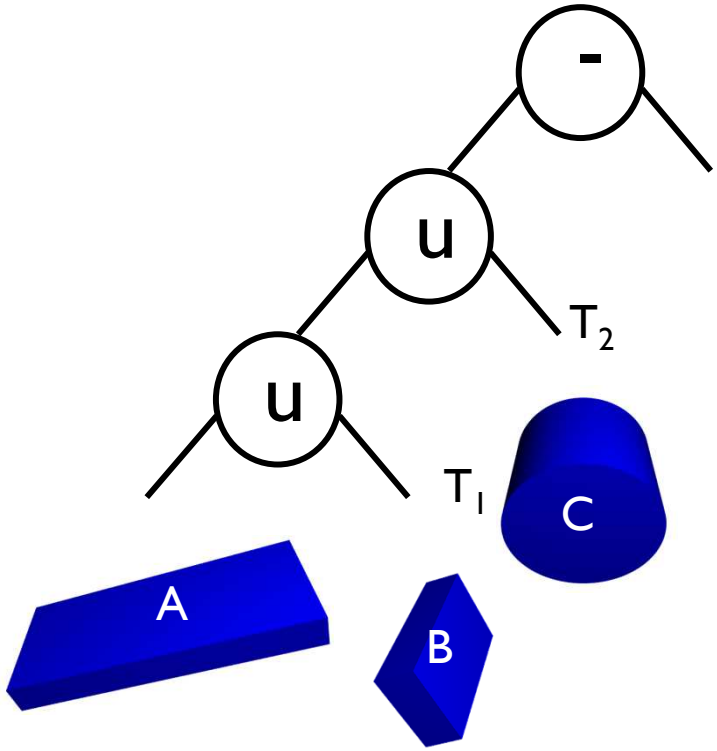


$$P \in A \qquad P \in B \qquad P \in (A - B)\,?$$

$$P \in (A - B) \Leftrightarrow (P \in A) et (P \notin B)$$

Localisation d 'un point

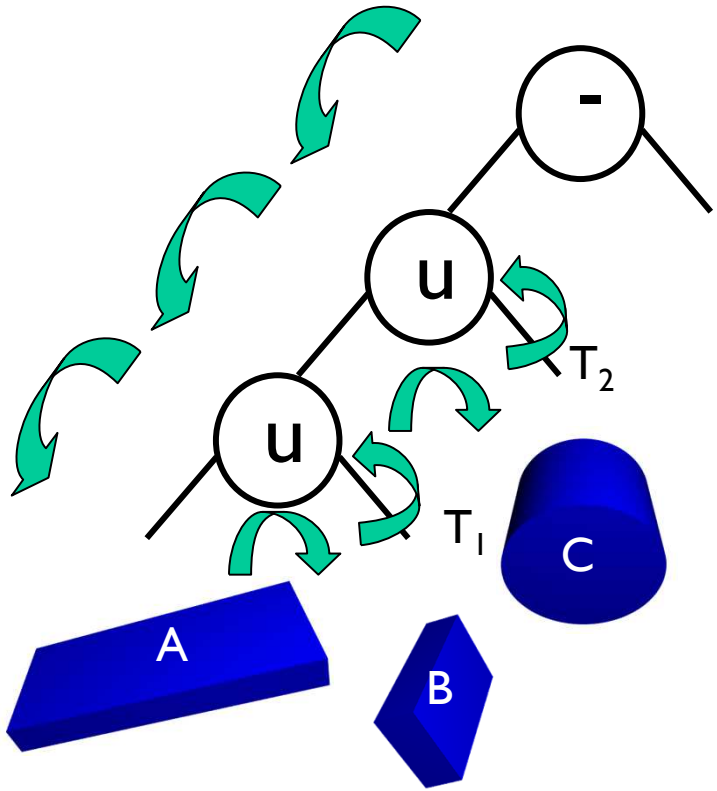
$$O = (((\ldots(A \cup (T_1(B)) \cup T_2(C)) - (\ldots)$$



$$P \in O\,?$$

Localisation d 'un point

$$O = (((\ldots(A \cup (T_1(B)) \cup T_2(C)) - (\ldots)$$



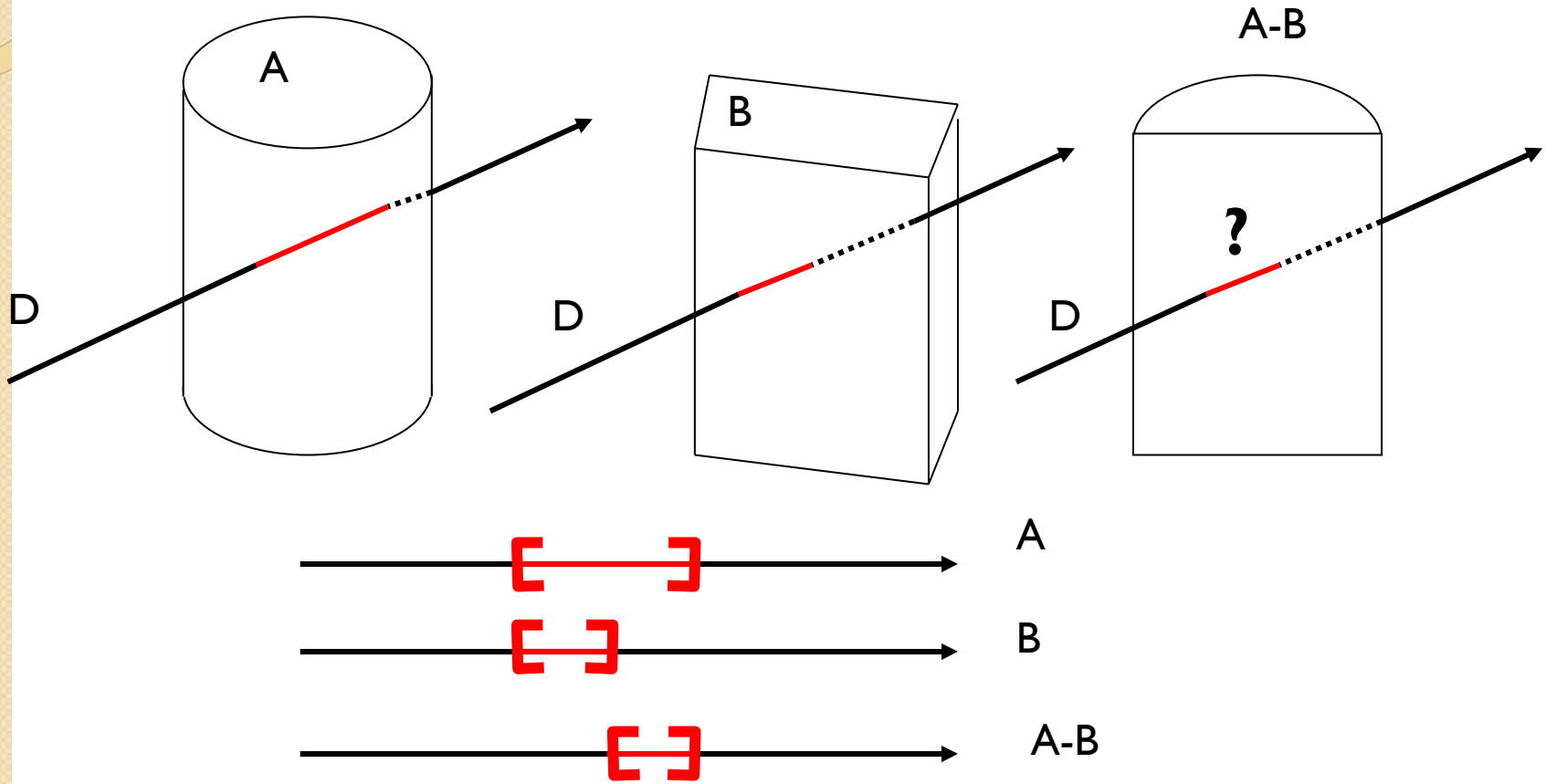
$$P \in O\,?$$

P ∈A ou $(T_1^{-1}(P) \in B)$ ou $(T_2^{-1}(P)$ ∈ C)

$$P \in O \Leftrightarrow (((\ldots(P \in A) ou (T_1^{-1}(P) \in B)) ou (T_2^{-1}(P) \in C)) - (\ldots)$$

Intersection avec une droite

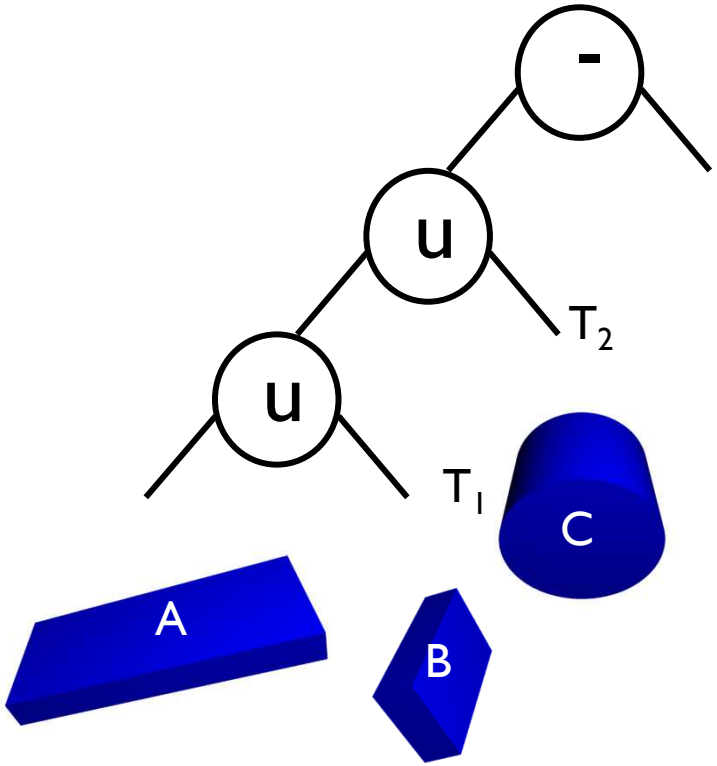
Principe : réflexion sur un exemple



$$D \; n \; (A\text{-}B) = (D \; n \; A)\text{-}(D \; n \; B)$$

Intersection avec une droite

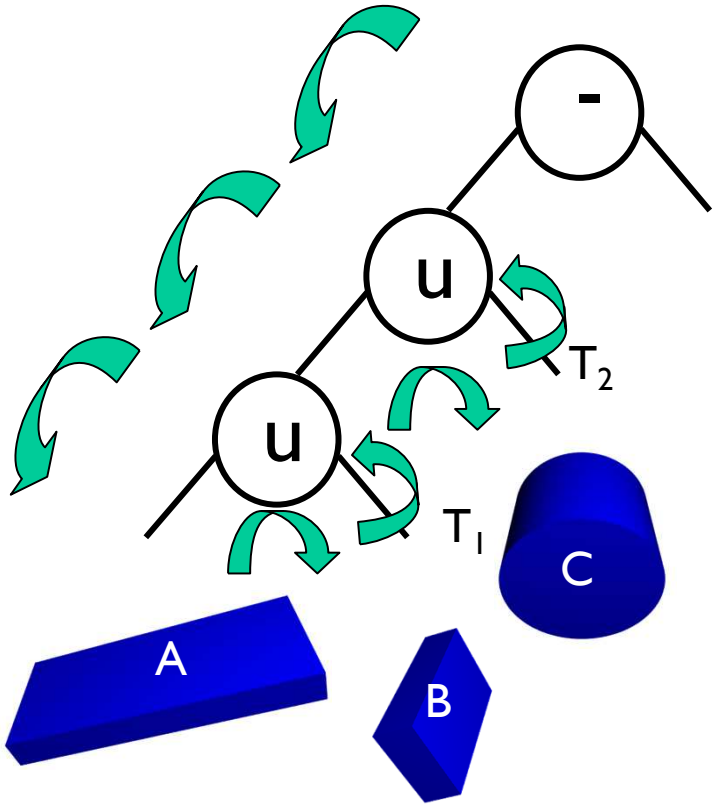
$$O = (((\ldots(A \cup (T_1(B)) \cup T_2(C)) - (\ldots)$$



$$DnO\,?$$

Intersection avec une droite

$$O = (((\ldots(A \cup (T_1(B)) \cup T_2(C)) - (\ldots)$$



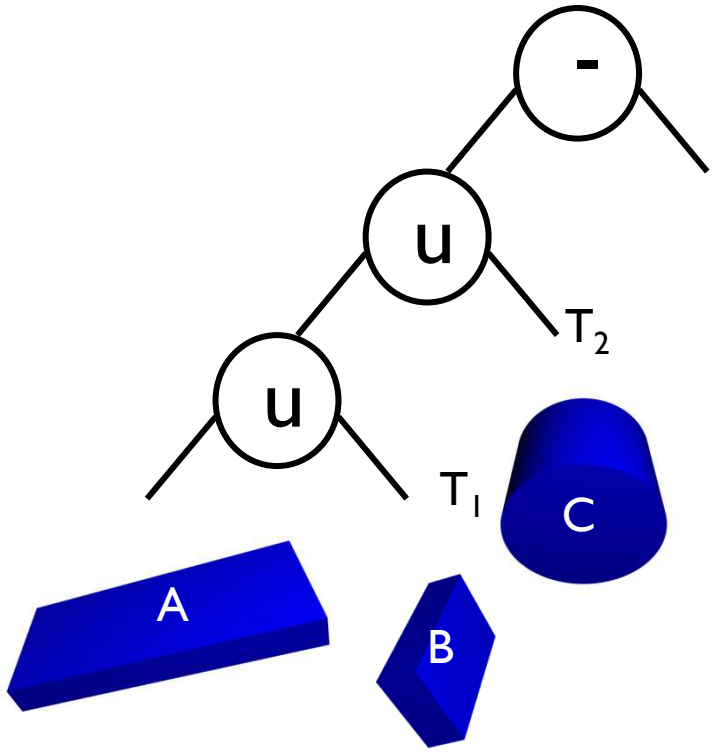
$$DnO\,?$$

D n A u $T_1(T_1^{-1}(D)$ n B) u $T_2(T_2^{-1}(D)$ n C)

$$Dr \cap O = ((\ldots(Dr \cap A) \cup T_1\ (T_1^{-1}(Dr) \cap B)) \cup T_2\ (T_2^{-1}(Dr) \cap C)) - (\ldots)$$

Calcul de la frontière d 'un CSG

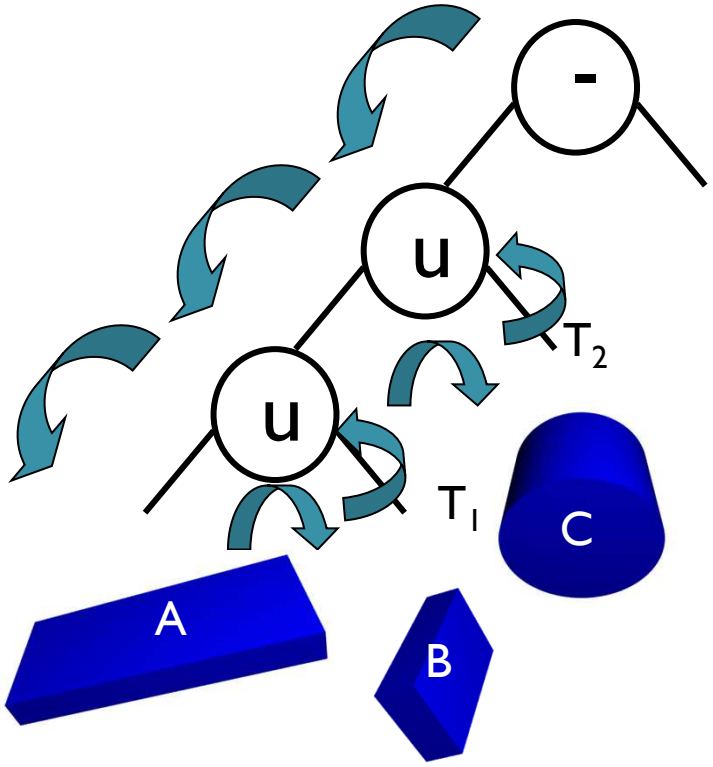
$$O = ((...(A \cup (T_1(B)) \cup T_2(C)) - (...)$$



$$Fr(O)\,?$$

Calcul de la frontière d 'un CSG

$$O = ((...(A \cup (T_1(B)) \cup T_2(C)) - (...)$$



$$Fr(O)\,?$$

Fr(A) u_{Fr} T_1(Fr(B)) u_{Fr} T_2(Fr(C))

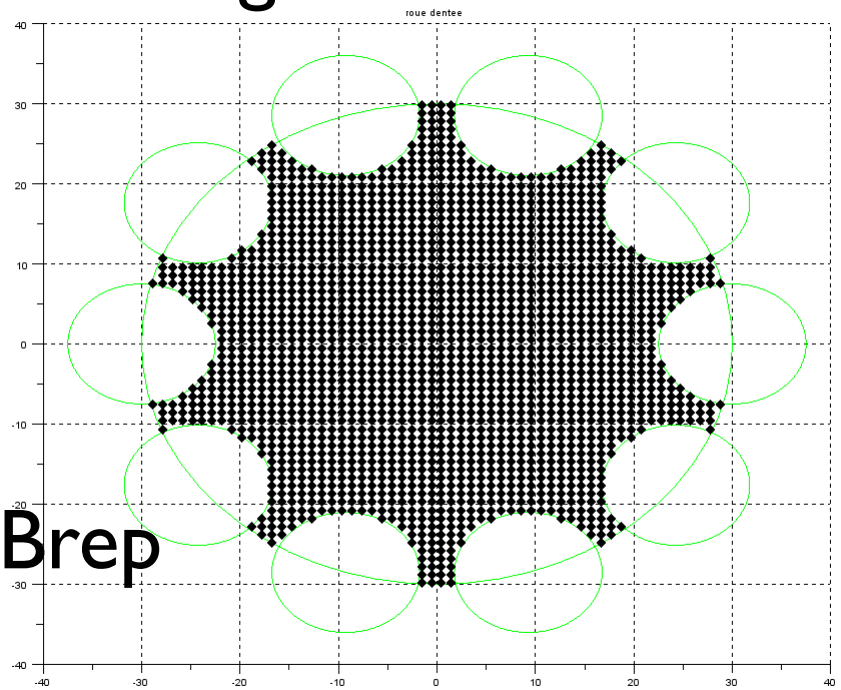
$$Fr(O) = ((...Fr(A) \cup^*_{Fr} (T_1\ (Fr(B)))) \cup^*_{Fr} (T_2\ (Fr(C))) -^*_{Fr} (...)$$

Calcul du volume d'un CSG

Calcul du volume d'un CSG

- Pas de solution simple, générale et précise
- Estimation à partir d 'un tirage aléatoire ou régulier de points

Tirage est régulier :
CSG →énumération spatiale.



- Conversion : CSG →Brep

Représentation informatique

- Pas le détail de l'implémentation
- OOP (C++) : les primitives dans une classe virtuelle avec des méthodes : point_dans(), volume(), intersection_droites()…
- Structure de l'arbre booléen et parcours récursif

Structures & méthodes récursives

- Définition récursive (grammaire)

Arbre	=	*(op, Arbre, Arbre)*
	ou	*(primitive)*
	ou	*(transf, Arbre)*

- Le parcours suit la définition, exemple :

$$Arbre = (\cup, (\cup, A, (T_1(B))), (T_2(C)))$$

op=Arbre(1) *Arbre(2)* *Arbre(3)*

$$Arbre = (\cup, A, (T_1(B)))$$

op=Arbre(1) *Arbre(2)* *Arbre(3)*

- Traitements : primitives, res1 op res2, transf

Algorithme de parcours

DEFINITION DE LA STRUCTURE : à l'aide de « list »

```
function [resF]=ParcoursCSG(Arbre, Fct, argF)
//=============================================
```

$$Arbre = \begin{cases} (op, Arbre, Arbre) \\ (primitive) \\ (transf, Arbre) \end{cases}$$

```
endfunction
```

Algorithme de parcours

DEFINITION DE LA STRUCTURE : à l'aide de « list »

```
function [resF]=ParcoursCSG(Arbre, Fct, argF)
//=============================================
if ( feuilleCSG(Arbre) ) then      ...    return; end;
if (OperateurCSG(Arbre)) then
        resFG=ParcoursCSG(Arbre(2),Fct,argF);
        resFD=ParcoursCSG(Arbre(3),Fct,argF);
        ...
        return; end
if (TransformationCSG(Arbre)) then
        ...
        resF=ParcoursCSG(Arbre(2),Fct,argF);
        return; end
endfunction
```

$$Arbre = \begin{cases} (op, Arbre, Arbre) \\ (primitive) \\ (transf, Arbre) \end{cases}$$

Algorithme de parcours

fichier : CSG.sci

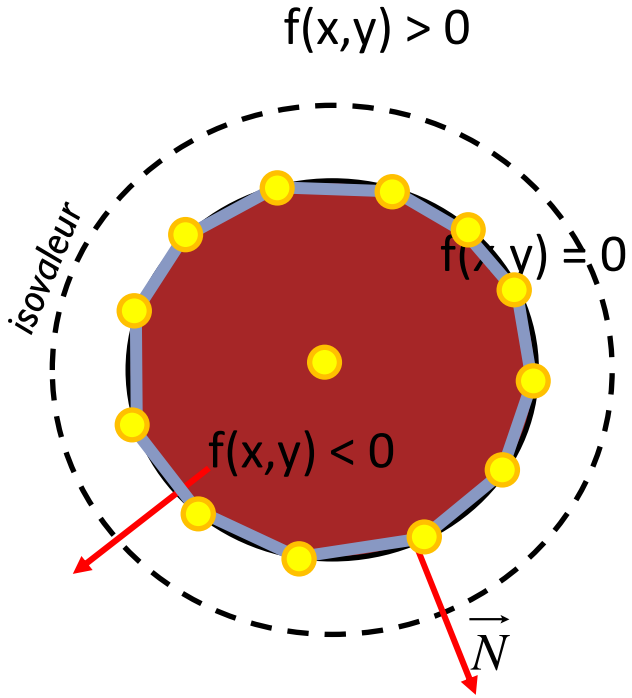
```
function [resF]=ParcoursCSG(Arbre, Fct, argF)
//=====================================================
// Fct = traitement sur l'arbre, exemples Plot, PTDans, Print
// La matrice passe dans l'argument argF
// avec d'autres arguments pour le traitement (attributs graphiques, Points,...)
if ( feuilleCSG(Arbre) ) then resF=FctVE(Arbre,argF); return; end;
if (OperateurCSG(Arbre)) then
        resFG=ParcoursCSG(Arbre(2),Fct,argF);
        resFD=ParcoursCSG(Arbre(3),Fct,argF);
        resF=FctOpCSG(Arbre(1),resFG,resFD,argF);
        return; end
if (TransformationCSG(Arbre)) then
        argF=FctTransf(Arbre(1),argF); // La matrice est accumulee dans argF
        resF=ParcoursCSG(Arbre(2),Fct,argF);
        return; end
endfunction
```

$$Arbre = (-,(\cup,(\cup,A,(T_1(B))),(T_2(C)),...)$$

3. LES « PRIMITIVES » ET LES SURFACES IMPLICITES OU ISO-POTENTIELLES

Exemple 2D : interrogation du modèle

$f(x,y) = (x^2 + y^2)^{1/2} - r$



P(x,y) dans ?

Intersection *Dr* ?

$x(u) = a_0\, u + b_0$
$y(u) = a_1\, u + b_1$

Normale sur la frontière ?

$$\vec{N} = grad(f)$$

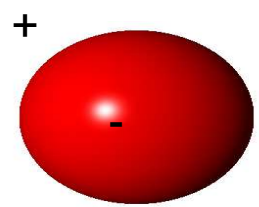
Représentation frontière ?

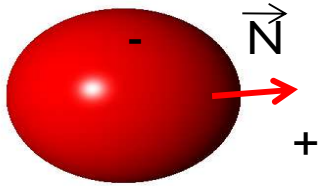
$x(\theta) = r \cos(\theta)$
$y(\theta) = r \sin(\theta)$
Avec θ dans $\{\theta_i\}$

La fonction donne la « distance signée» au cercle

Surfaces algébriques: DS(x,y,z)=0

Positif à l'extérieur





DS(x,y,z) donne la distance à la surface S
Surface frontière : DS(x,y,z) = 0,
intérieur : DS(x,y,z) < 0,
extérieur : DS(x,y,z) > 0

Exemples :

$DS(x,y,z) = (x^2 + y^2 + z^2)^{1/2} - r$

$DS(x,y,z) = z - h$

$DS(x,y,z) = Ax + By + Cz + D$

Surface quadrique : $Ax^2 + By^2 + Cz^2 + 2D\,xy + 2E\,yz + 2F\,xz + 2G\,x + 2H\,y + 2I\,z + J = 0$

Surface quartic :

$(x^2 + y^2 + z^2 + R^2 - r^2)^2 = 4R^2 (x^2 + y^2)$

$$\vec{N} = grad(DS) = \begin{pmatrix} \partial DS/\partial x \\ \partial DS/\partial y \\ \partial DS/\partial z \end{pmatrix}$$

Point dans ? Intersection ? …

Exercice : Normale en un point de la surface d'une sphère

- Identifier les coefficients A, B , C … pour le cas d'une sphère

$$Quadrique :$$

$$Ax^2 + By^2 + Cz^2 + 2D\,xy + 2E\,yz + 2F\,xz + 2G\,x + 2H\,y + 2I\,z + J = 0$$

- Calculer le gradient de la sphère à partir de son équation algébrique implicite

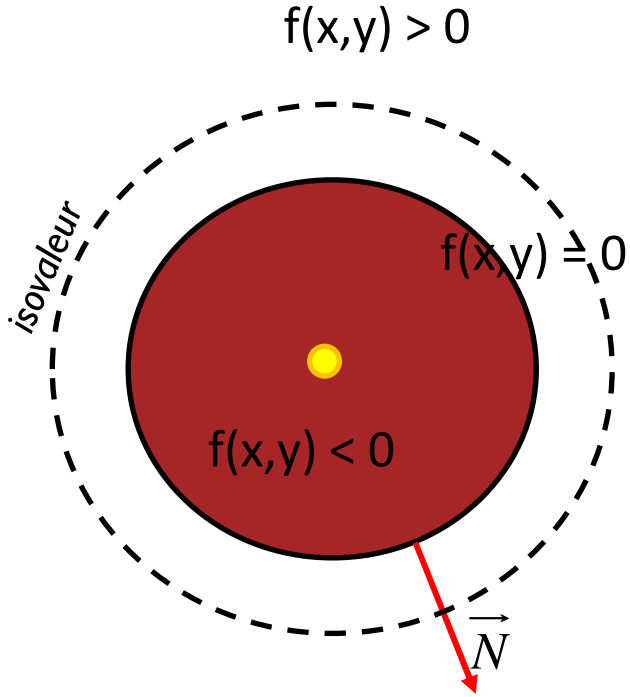
- Vérifier le résultat en utilisant la formule ci-dessous

$$\vec{N} = 2 * \begin{pmatrix} Ax_i + Dy_i + Fz_i + G \\ By_i + Dx_i + Ez_i + H \\ Cz_i + Ey_i + Fx_i + I \end{pmatrix}$$

Autre formalisme : distance et densité

$f(x,y) = (x^2 + y^2)^{1/2} - r$

La fonction donne la « distance signée» au cercle



distance(x,y) = $(x^2 + y^2)^{1/2}$

distance(x,y) <= r (seuil)

densité(x,y) = - distance(x,y)

densité(x,y) >= -r (seuil)

densité(x,y) = 1 - distance(x,y)/r

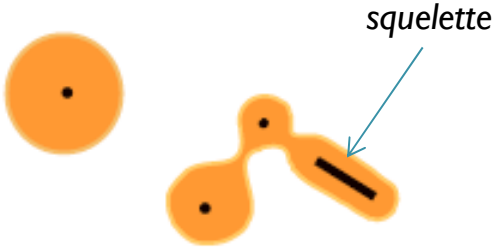
densité(x,y) >= 0 (seuil)

…

Distance Field
« Blob » ou « Metaball »

$$density = strength \cdot \left(1 - \left(\frac{distance}{radius}\right)^2\right)^2 \qquad if\ (density > threshold)\ then\ INSIDE$$

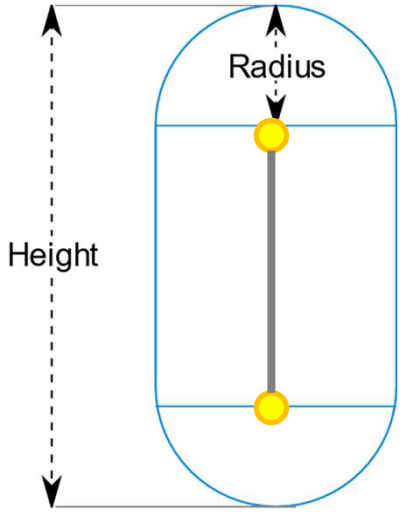
- En 3D
  - Par rapport à un point :
  - Par rapport à un axe :
  - Par rapport à un segment :



$$DS(x,y,z) = S*(1-(d_E(x,y,z)/R)^2)^2 - T$$

*Fonction de densité (ou de potentiel) de Murakami

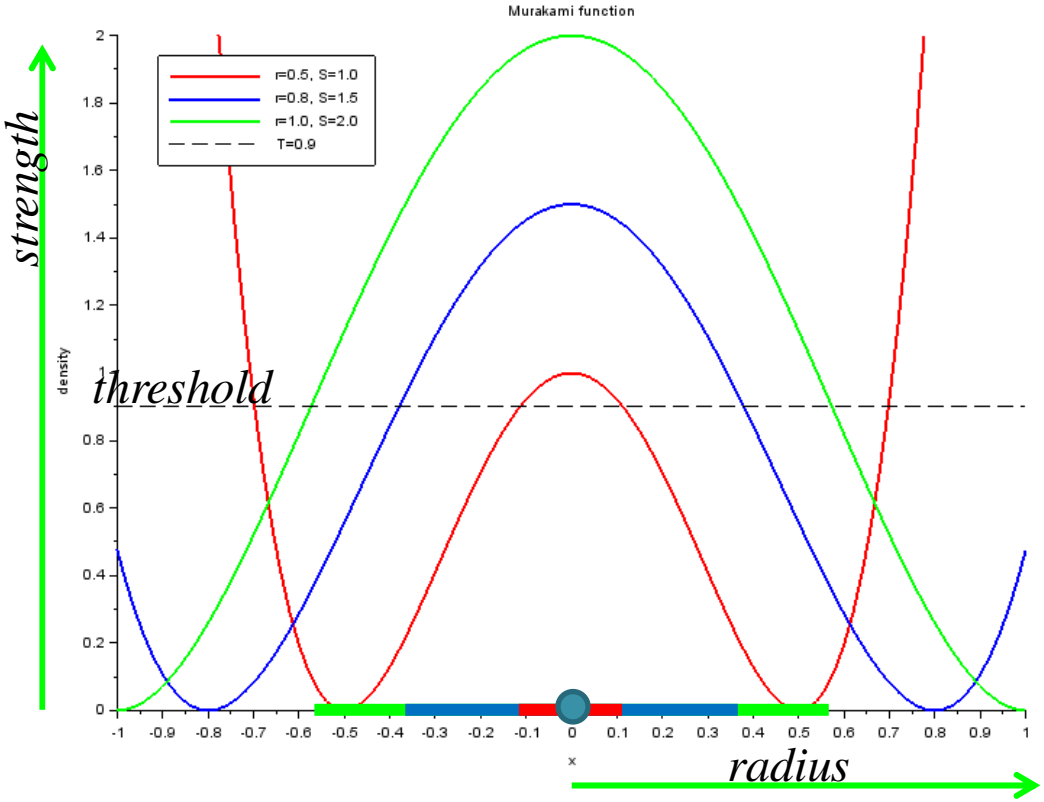
Courbe équidistance à un segment ?



Distance Field
« Blob » ou « Metaball »

$$density \overset{*}{=} strength \cdot \left(1 - \left(\frac{distance}{radius}\right)^2\right)^2$$

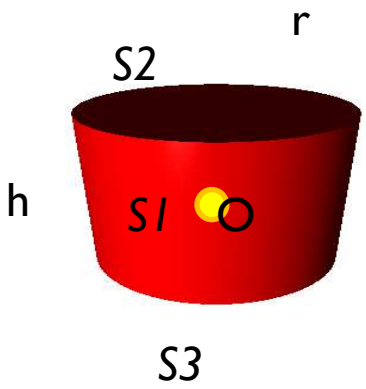
if (density > threshold) then INSIDE



*Fonction de densité (ou de potentiel) de Murakami

Comment décrire un cylindre ?

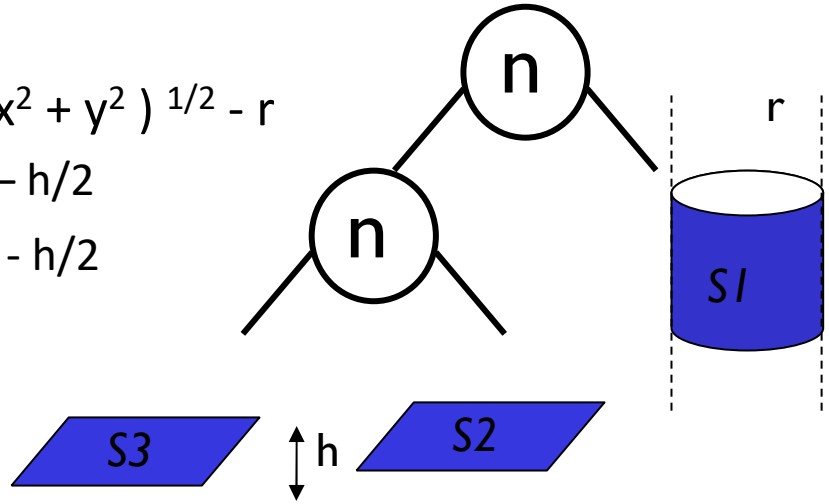
surfaces algébriques ➔ ½ espaces



$DS1(x,y,z) = (x^2 + y^2)^{1/2} - r$

$DS2(x,y,z) = z - h/2$

$DS3(x,y,z) = -z - h/2$

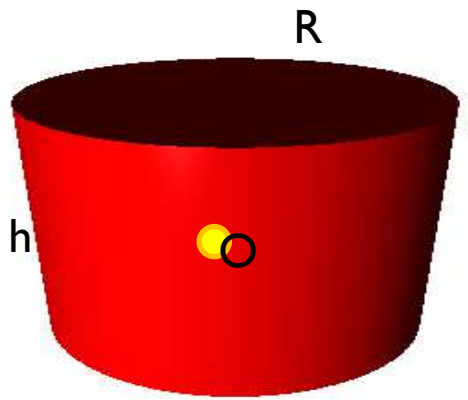


Position du point x,y,z ?

function [**resF**]=PTdansCylindre(**Cylindre**, **PT**)

Dans ?
Hors ?
Sur ?

PT=(x,y,z)

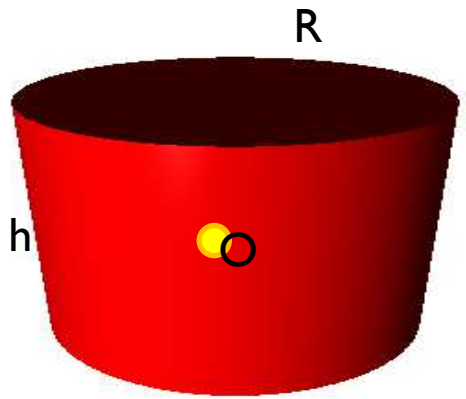


Position du point x,y,z ?

function [**resF**]=PTdansCylindre(**Cylindre**, **PT**)

Dans ?
Hors ?
Sur ?

PT=(x,y,z)

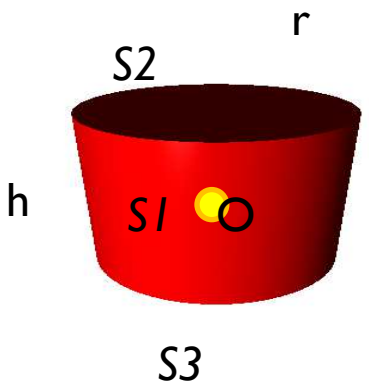


$x^2+y^2 <= R^2$
et
$-h/2 <= z <= h/2$

Représentation
algébrique implicite

Position du point x,y,z ?

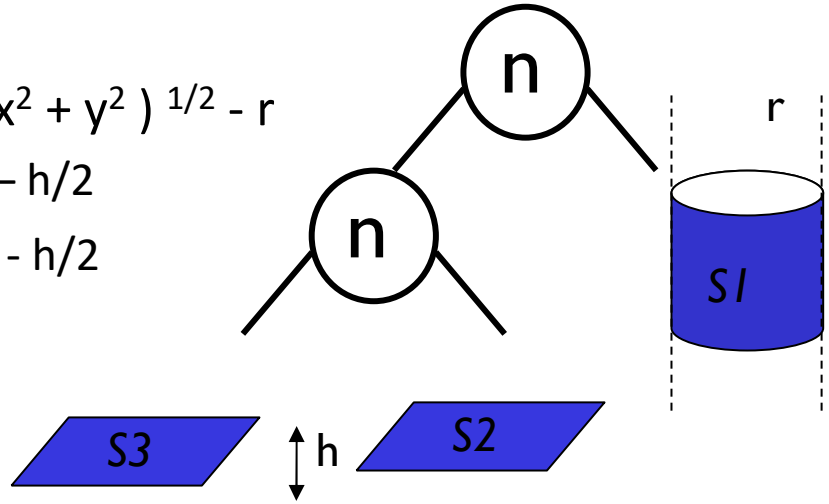
surfaces algébriques ➔ ½ espaces



$DS1(x,y,z) = (x^2 + y^2)^{1/2} - r$

$DS2(x,y,z) = z - h/2$

$DS3(x,y,z) = -z - h/2$

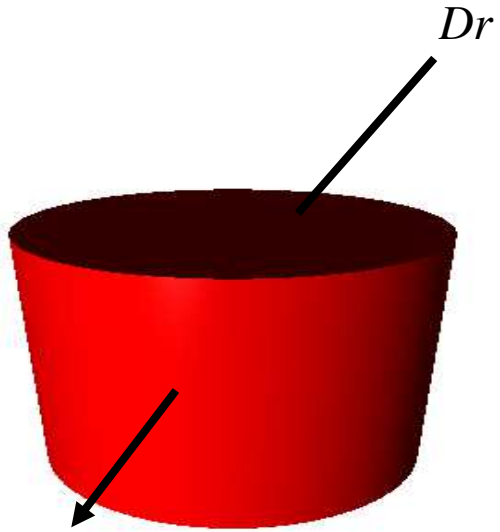


Donner la proposition qui exprime l'appartenance d'un point au cylindre ?

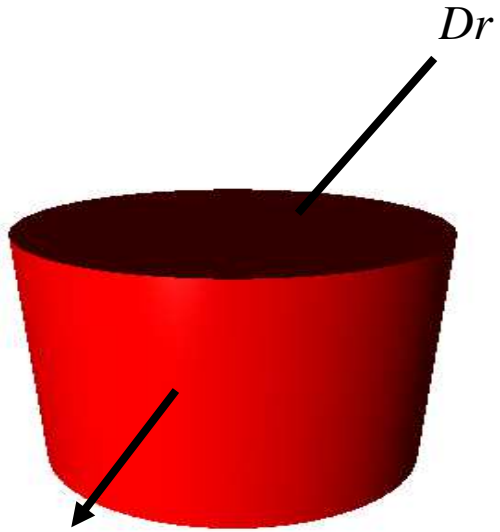
P(x,y,z) € Cyl? ⇔(DS1(x,y,z) <= 0) **ET** (DS2(x,y,z)<=0) **ET** (DS3(x,y,z)<=0)

⇔(DS_{cyl}(x,y,z) <= 0) avec DS_{cyl} (x,y,z)= **max**(DS1(x,y,z),DS2(x,y,z),DS3(x,y,z))

Intersection avec une droite



Intersection avec une droite



Plan : $Ax+By+Cz+D=0$

Droite : $x(t)=a_1t+a_2$
$y(t)=b_1t+b_2$
$z(t)=c_1t+c_2$

=> équation linéaire :

$$t = \frac{Aa_1+Bb_1+Cc_1}{Aa_2+Bb_2+Cc_2+D}$$

Cylindre : $x^2+y^2=R^2$

Droite : $x(t)=a_1t+a_2$
$y(t)=b_1t+b_2$
$z(t)=c_1t+c_2$

=> équation second degré

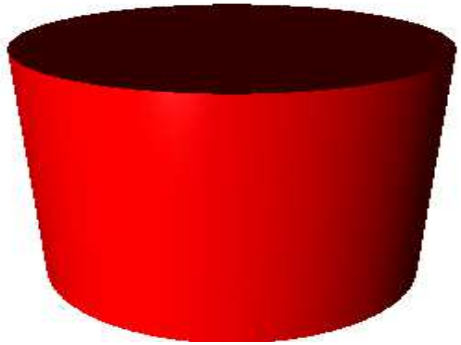
Vérifier que les solutions $\{t_1,t_2,t_3,t_4\}$ appartiennent au volume

$$Dr \cap Cylindre = (Dr \cap DS1) \cap (Dr \cap DS2) \cap (Dr \cap DS3)$$

Implicite ➔ Paramétrique

cas simple : paramétrage naturel, courbure constante…

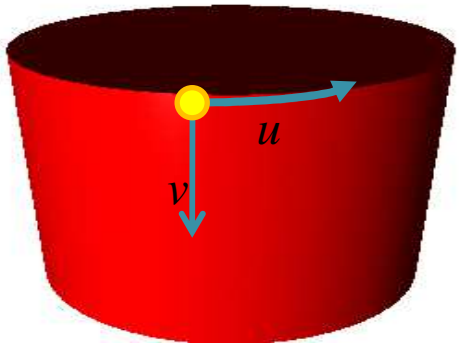
$F(x,y,z) = 0 \rightarrow \{ x(u,v), y(u,v), z(u,v) \}$



$F(x,y,z) = 0$

cylindre { h, r}

$F=\{ x(u,v), y(u,v), z(u,v) \}$



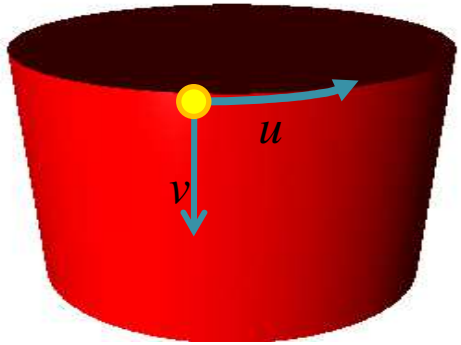
Représentation paramétrique

$x(u,v) = r \cos(u*\pi)$
$y(u,v) = r \sin(u*\pi)$ pour $-1 < u <= 1$
$z(u,v) = v*h/2$ pour $-1 <= v <= 1$

Calcul de la frontière

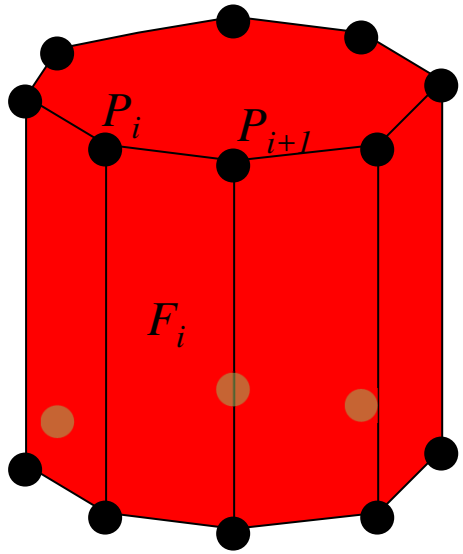
Conversion ➔ Brep/Triangulation

$F(x,y,z) = 0$ ➔ $\{ x(u,v), y(u,v), z(u,v) \}$



cylindre { h, r}

Polyèdre à face planes



Représentation paramétrique

$x(u,v) = r \cos(u*\pi)$
$y(u,v) = r \sin(u*\pi)$ pour $-1 <= u <= 1$
$z(u,v) = v*h/2$ pour $-1 <= v <= 1$

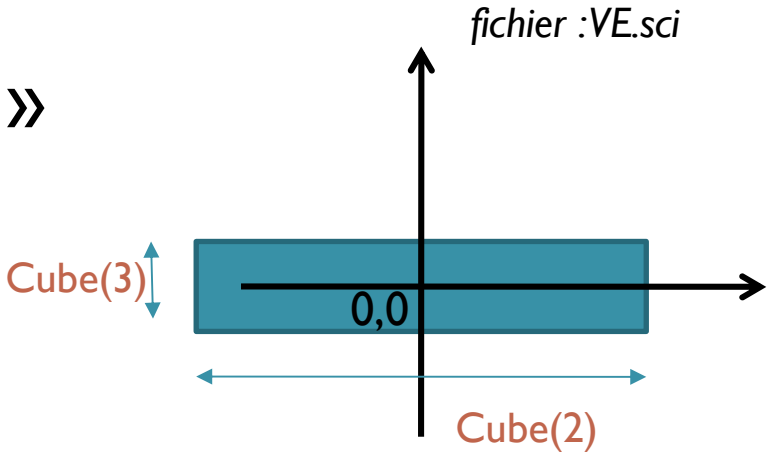
$P_i = (x_i, y_i, z_i) = (r \cos(u_i*\pi), r \sin(u_i*\pi), v_i*h/2)$

$F_i = \{P_i, P_j, P_{j+1}, P_{i+1}\}$

Localisationd'un point

- Exemple du « Cube »

```
Cube=list("Cube",20,2); // une primitive geometrique
```

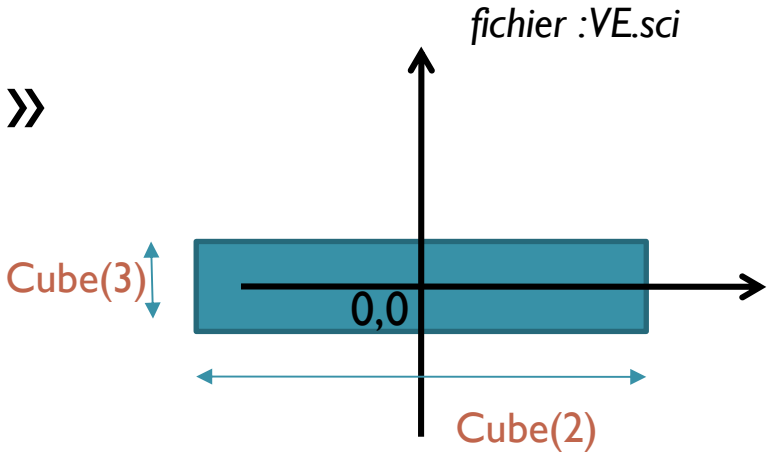



```
function [VF]=PTdansCube(Cube, PT)
 //======================
 // Cube(1)="Cube", Cube(2)=largeur, Cube(3)=hauteur,
 // PT = [x,y] du point a tester (deja dans le repere local de la primitive)
dim=length(PT); VF=%T;
if ( Cube(1) ~= "Cube" ) then abort; end
for i=1:1:dim
   if( (PT(i) > Cube(i+1)/2) ) then VF=%F; return; end;
   if( (PT(i) < -Cube(i+1)/2) ) then VF=%F; return; end;
end
endfunction
```

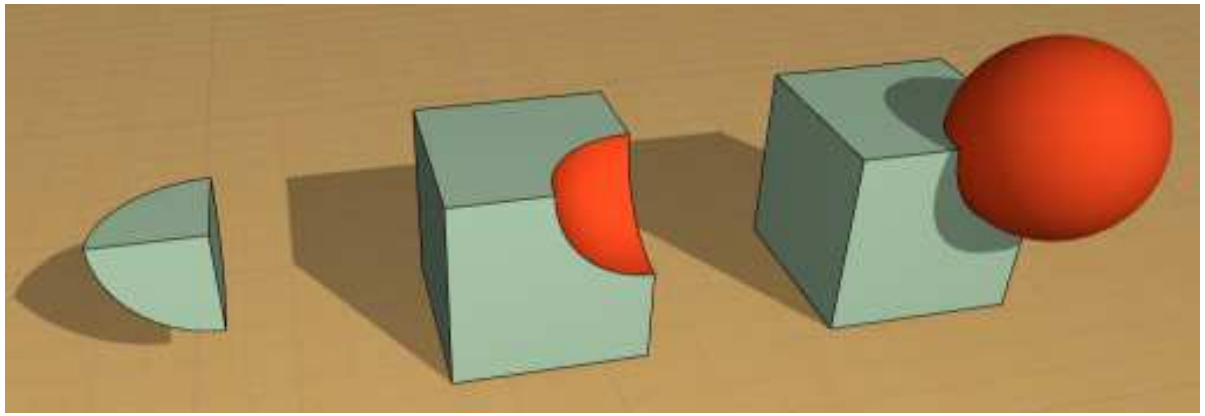
Calcul de la frontière

- Exemple du « Cube »

```
Cube=list("Cube",20,2); // une primitive geometrique
```

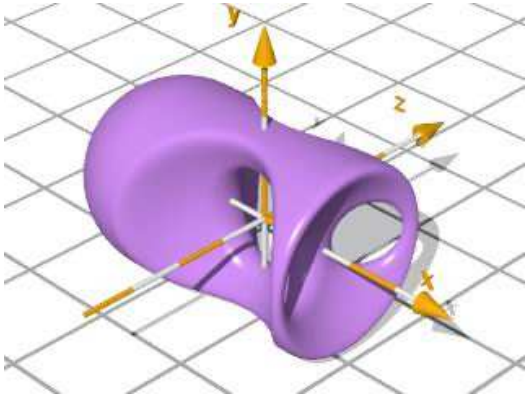



```
function [err]=PlotCube(Cube, matM, AtGr)
 //===================================
// Cube(1)="Cube", Cube(2)=largeur, Cube(3)=hauteur,
// matM : matrice Homogene a appliquer au Cube
// AtGr = couleur du trace
 err=%T
 if ( Cube(1) ~= "Cube" ) then abort; end
// Le Cube est transforme en polygone
X=[-Cube(2)/2;Cube(2)/2;Cube(2)/2;-Cube(2)/2;-Cube(2)/2;]';
Y=[-Cube(3)/2;-Cube(3)/2;Cube(3)/2;Cube(3)/2;-Cube(3)/2]';
[XY]=trPoints([X;Y],matM); // Applique la matrice des transformations aux points
plot2d(XY(1,:),XY(2,:),style=AtGr)
endfunction
```



4. OPÉRATIONS BOOLÉENNES & VARIANTES

- Opérations booléennes régularisées
- Fonctions de « mélange » : somme… blob et metaball



Algorithme de parcours

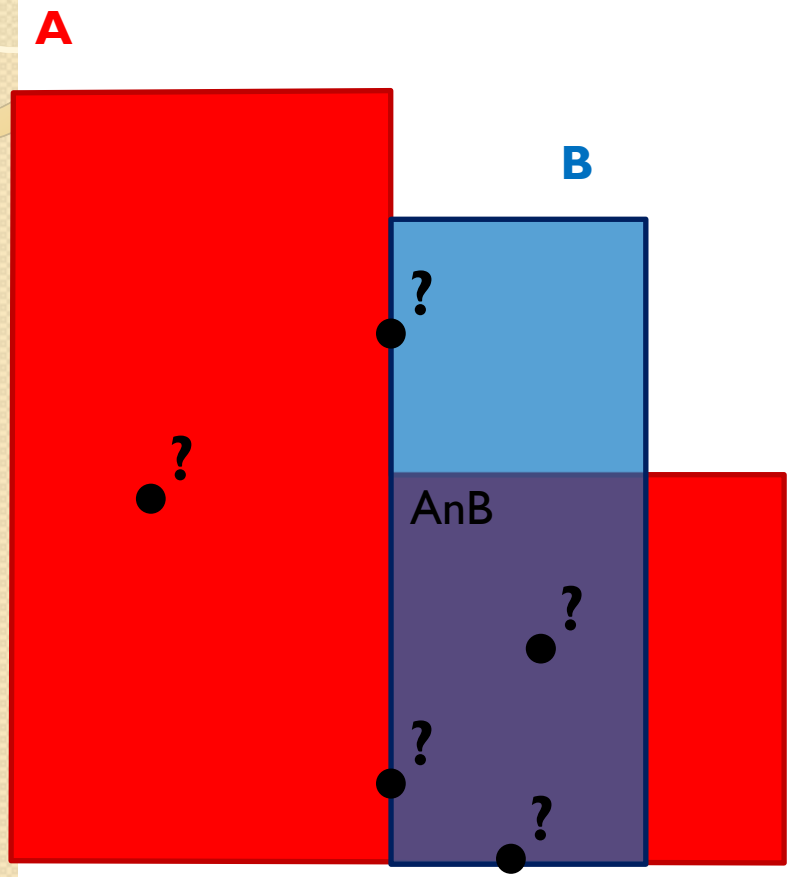
Combinaison des résultats de 2 sous arbres

fichier : CSG.sci

```
function [resF]=ParcoursCSG(Arbre, Fct, argF)
//=====================================================
// Fct = traitement sur l'arbre, exemples Plot, PTDans, Print
// La matrice passe dans l'argument argF
// avec d'autres arguments pour le traitement (attributs graphiques, Points,...)
if ( feuilleCSG(Arbre) ) then resF=FctVE(Arbre,argF); return; end;
if (OperateurCSG(Arbre)) then
        resFG=ParcoursCSG(Arbre(2),Fct,argF);
        resFD=ParcoursCSG(Arbre(3),Fct,argF);
        resF=FctOpCSG(Arbre(1),resFG,resFD,argF);
        return; end
if (TransformationCSG(Arbre)) then
        argF=FctTransf(Arbre(1),argF); // La matrice est accumulee dans argF
        resF=ParcoursCSG(Arbre(2),Fct,argF);
        return; end
endfunction
```

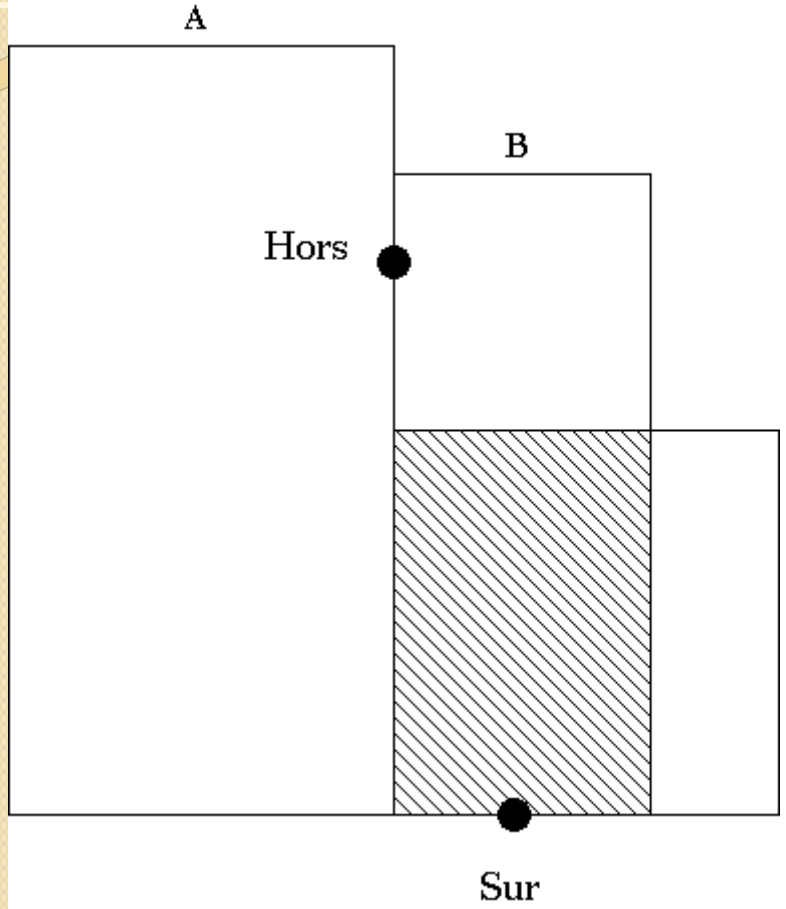
G \ D	Dans	Hors
Dans	?	?
Hors	?	?

Localisation d ’un point sur AnB



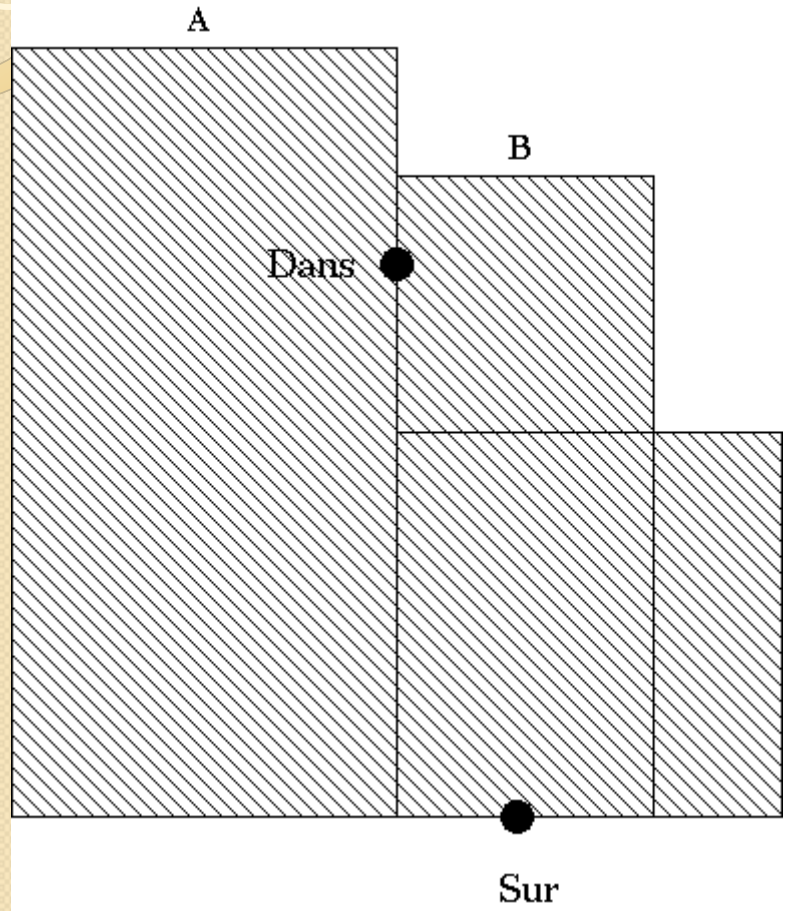
A \ B	Dans	Sur	Hors
Dans			
Sur			
Hors			

Localisation d ’un point



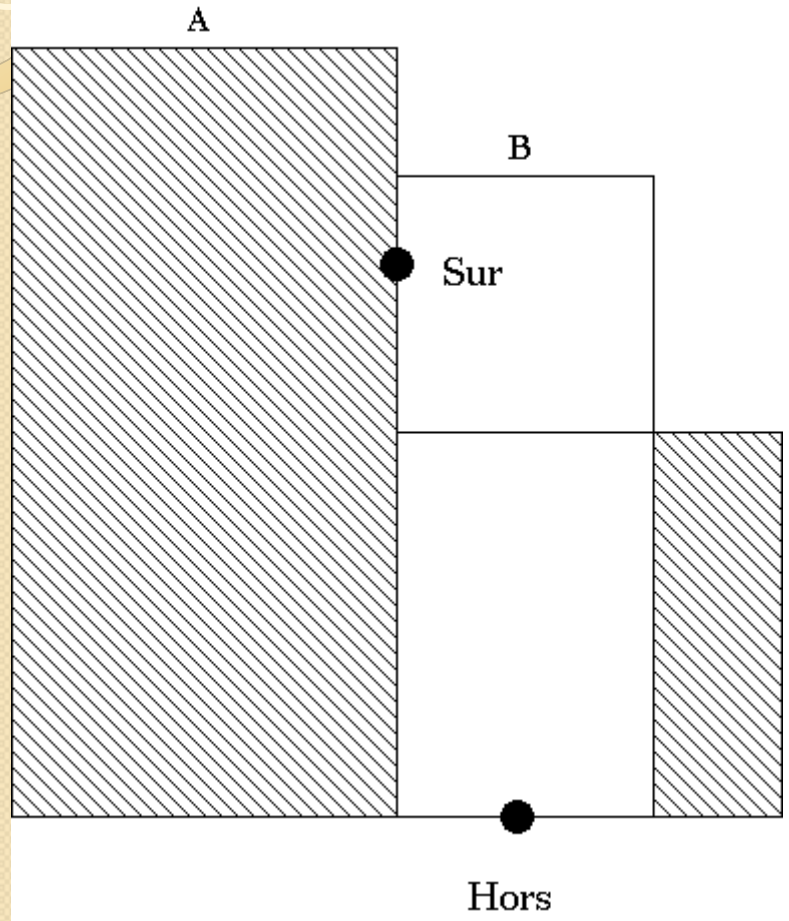
AnB	Dans	Sur	Hors
Dans	Dans	Sur	Hors
Sur	Sur	Sur/Hors	Hors
Hors	Hors	Hors	Hors

Localisation d ’un point



AuB	Dans	Sur	Hors
Dans	Dans	Dans	Dans
Sur	Dans	Sur/Dans	Sur
Hors	Dans	Sur	Hors

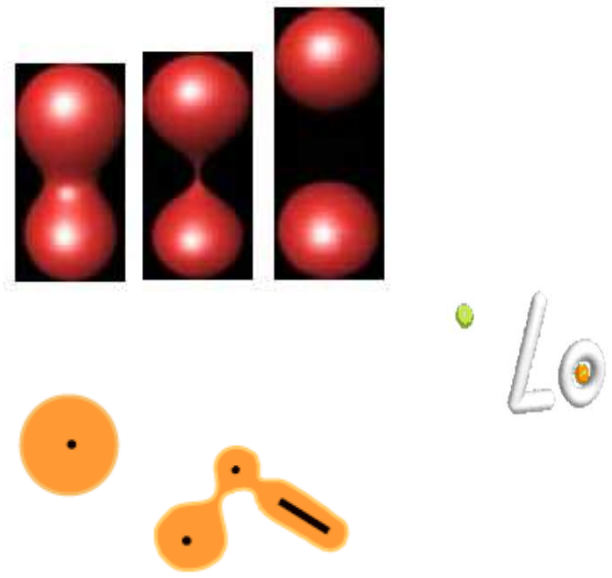
Localisation d ’un point

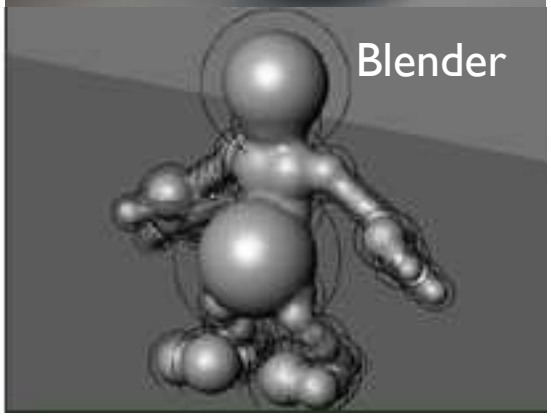


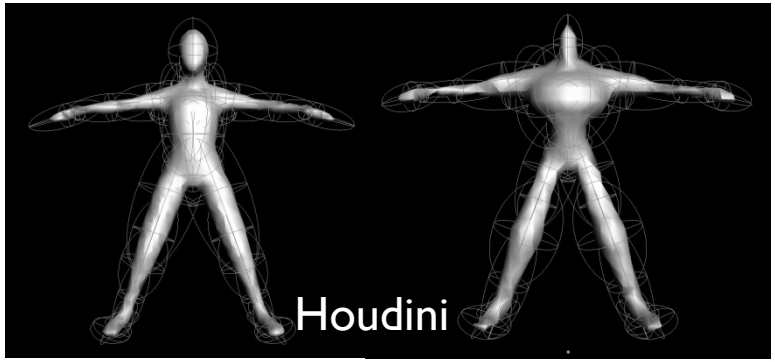
A-B	Dans	Sur	Hors
Dans	Hors	Sur	Dans
Sur	Hors	Sur/Hors	Sur
Hors	Hors	Hors	Hors

VARIANTES
« Blob » ou Metaball

Surfaces de potentiel d'une somme
de champs scalaire



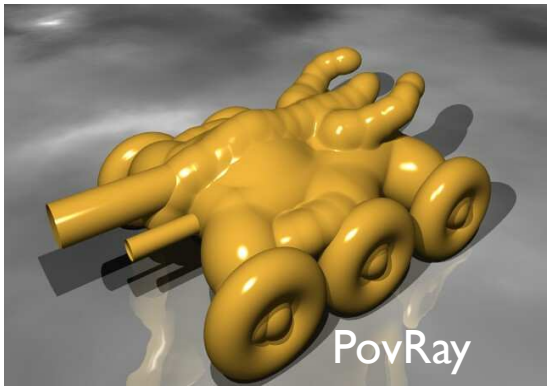




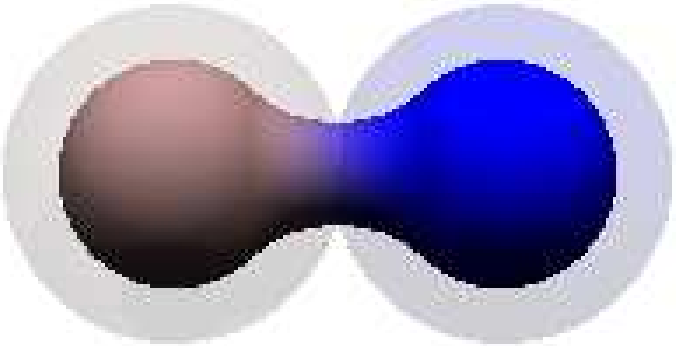
Houdini, Side Effect Software Inc.

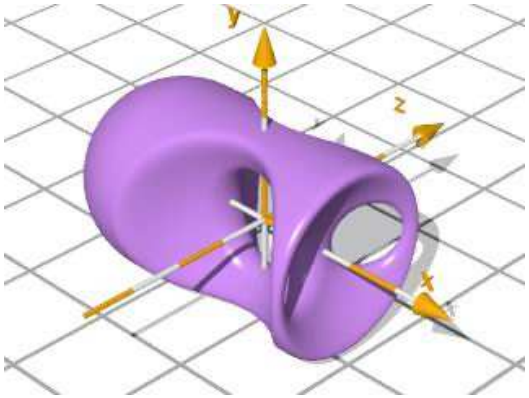
Assemblage

Exemples PovRay



blob {
threshold *seuil_force*
sphere { < *P1_centre*>, *rayon_sphere1* , strength *force1_centre*}
sphere { < *P2_centre*>, *rayon_sphere2* , strength *force2_centre*}
cylinder {…}
…
}



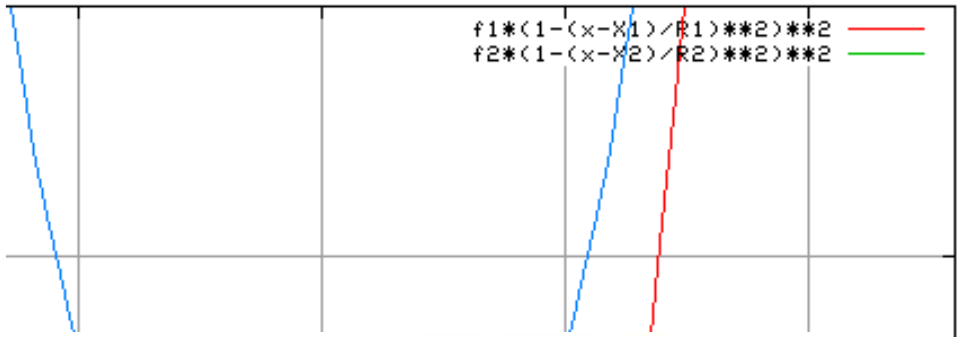


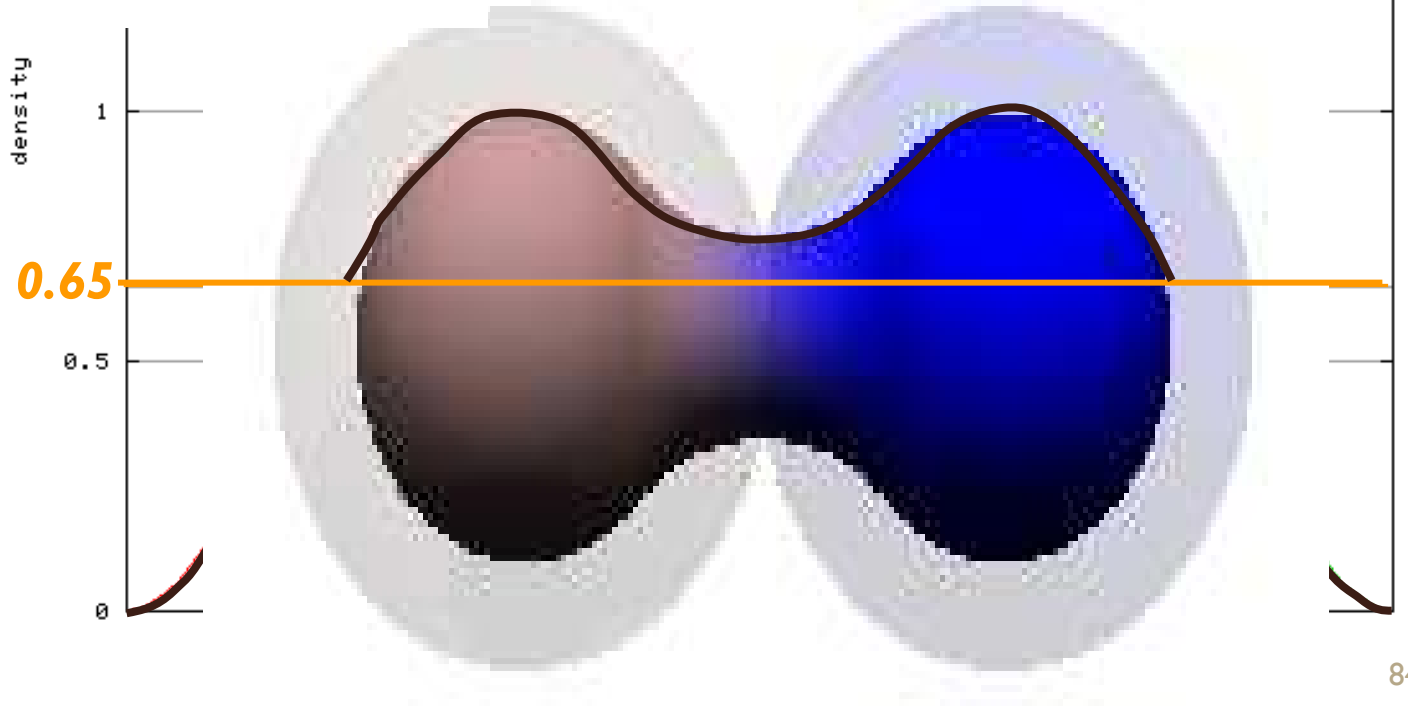
force_centre < 0

Fonction « BLOB »

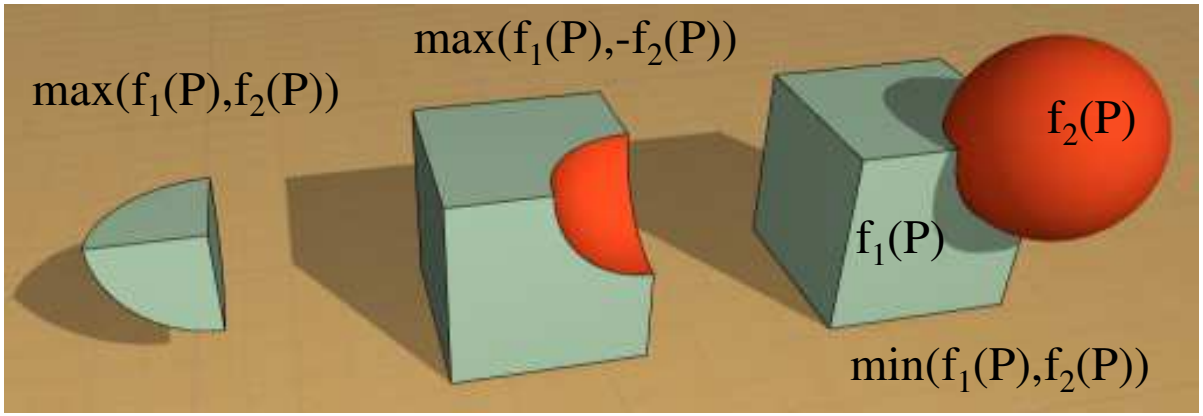

```
blob {
   threshold 0.65
   sphere { < 0,0,0>, 0.8 , strength 1.0}
   sphere { < 1,0,0>, 0.8 , strength 1.0}
}
```

D= strength(1-(distance/radius)^2)^2





Différents types d'opérations



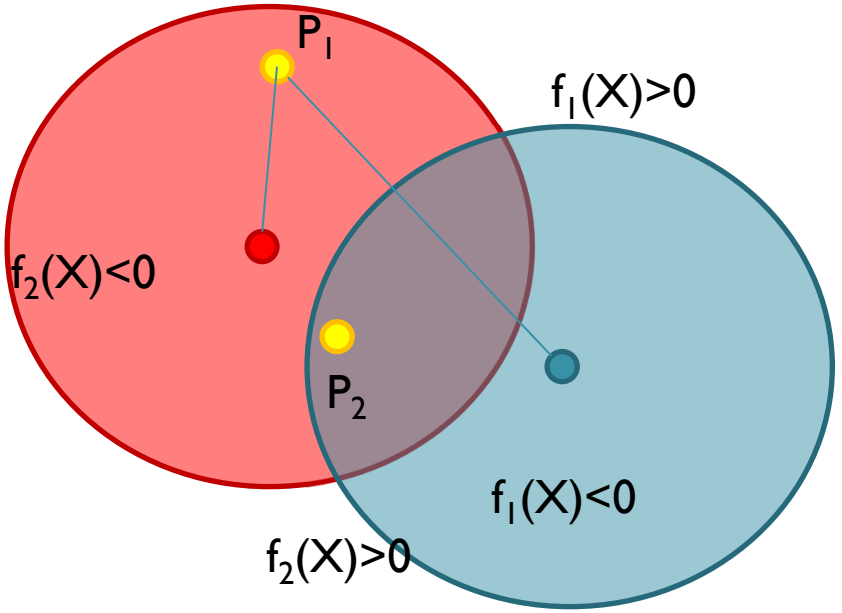
- Fonctions de mélange : $f(P)=\Sigma f_i(P)$ …

- R-fonctions [Pasko] :

$$\bigcup_{\alpha}(F_1,F_2) = \frac{1}{1+\alpha}(F_1+F_2+\sqrt{F_1^2+F_2^2-2\alpha F_1F_2})$$

$$\bigcap_{\alpha}(F_1,F_2) = \frac{1}{1+\alpha}(F_1+F_2-\sqrt{F_1^2+F_2^2-2\alpha F_1F_2})$$

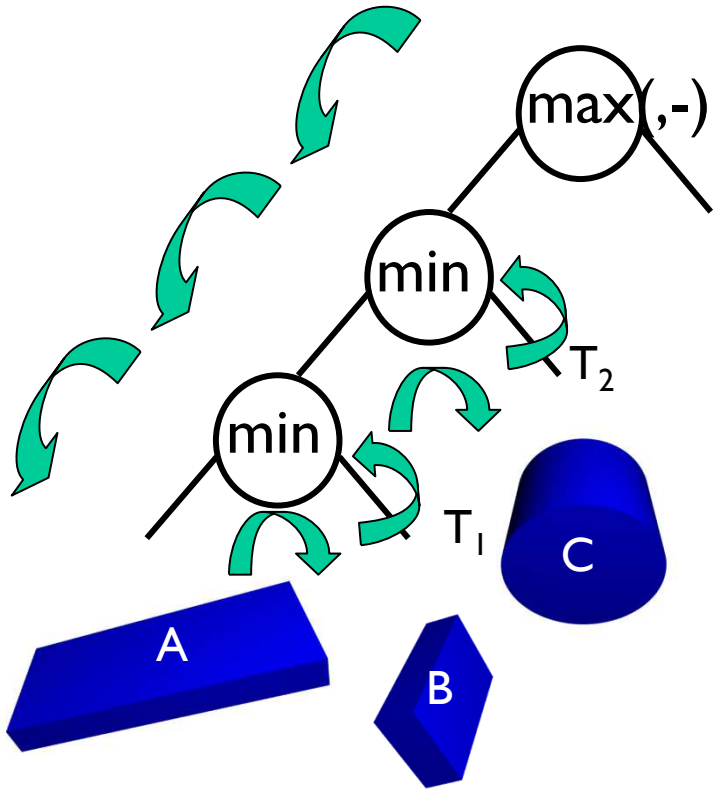
Avec paramètre α [0,1]



* *voir J.L.Mari 2006*

Localisation d 'un point

$$O = (((\ldots(A \cup (T_1(B)) \cup T_2(C)) - (\ldots)$$

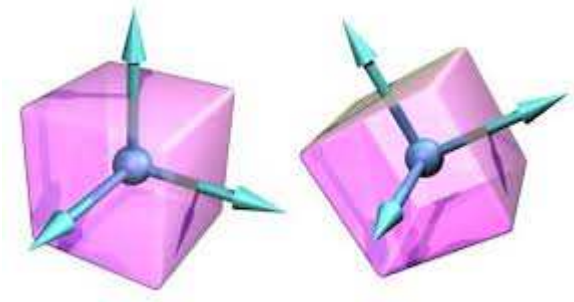


$$P \in O\,?$$

min(min($DS_A(P)$, $DS_B(T_1^{-1}(P))$) , $DS_C(T_2^{-1}(P)$?)) ..... > 0

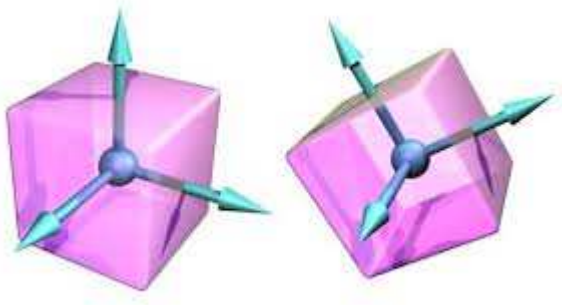
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

5. TRANSFORMATIONS ET COORDONNÉES HOMOGÈNES



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

SKIP

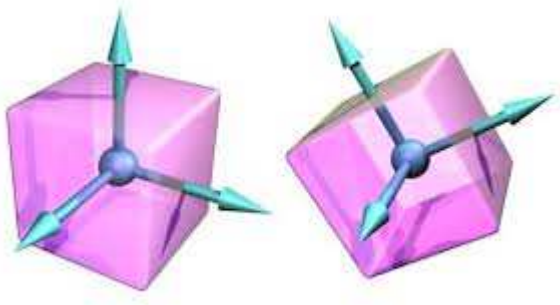


Rotations d'axes x, y, z

$$R_x(\theta) = \quad ?$$

$$R_z(\alpha) = \quad ?$$

$$R_y(\psi) = \quad ?$$

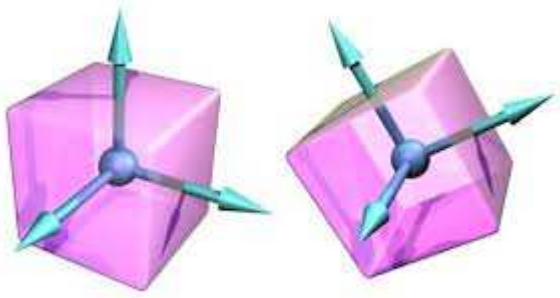


Rotations d'axes x, y, z

$$R_x(\theta) = \quad ?$$

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

$$R_y(\psi) = \quad ?$$

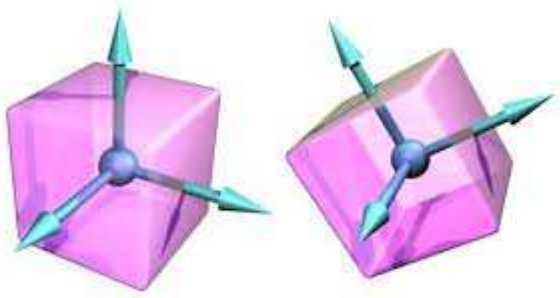


Rotations d'axes x, y, z

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

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

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



Rotations d'axes x, y, z

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

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

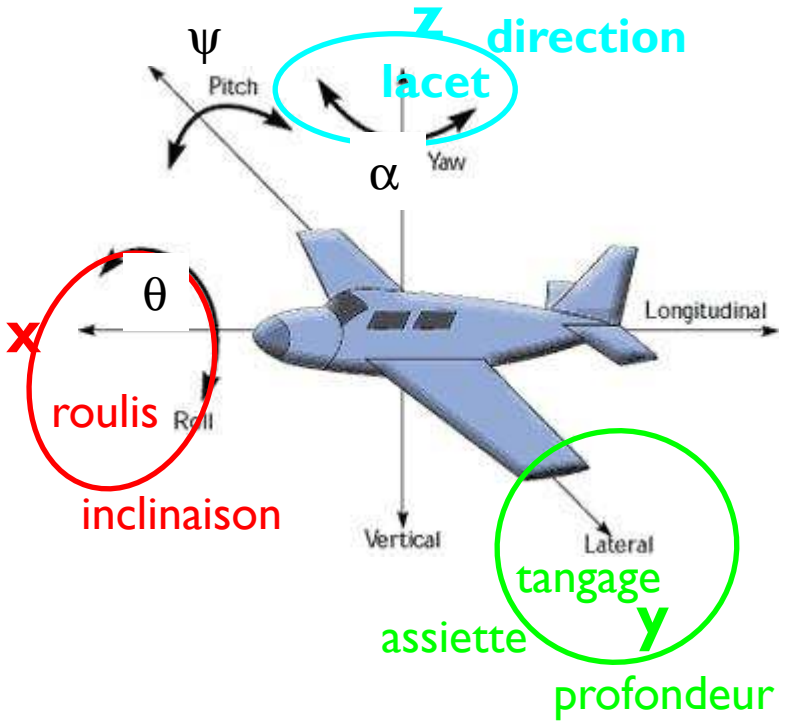
$$R_y(\psi)=\begin{pmatrix}\cos(\psi) & 0 & \sin(\psi)\\ 0 & 1 & 0\\ -\sin(\psi) & 0 & \cos(\psi)\end{pmatrix}$$

Composition des rotations = Multiplication des matrices
Elle n'est pas commutative ➔ choisir un ORDRE

La rotation suivant *x* puis y puis *z* dans **le repère global** (l'ordre est important) correspond à la composition :

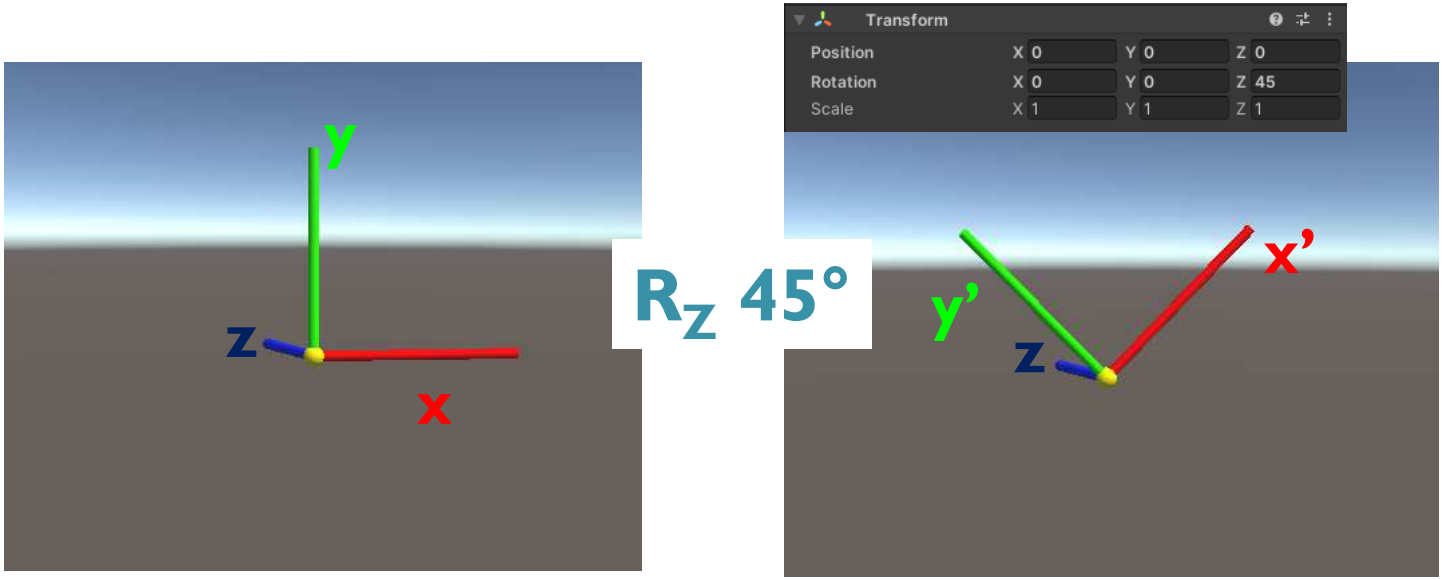
$$R_z(\alpha)R_y(\psi)R_x(\theta)$$

Les angles d'Euler (le repère local)



Les angles d'Euler (le repère local)

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



La rotation suivant *x* puis y' puis *z*" dans **le repère local** correspond à la composition :

$$R_{z''}(\alpha)R_{y'}(\psi)R_x(\theta) \quad = \; R_x(\theta)R_y(\psi)R_z(\alpha)$$

C'est-à-dire la rotation suivant *z* puis y puis *x* dans **le repère global : l'ordre inverse**

« Démonstration »

Exemple :

- Dans le repère local : $R_{y'}(\psi) \circ R_x(\theta)\ P(x,y,z)$
- $R_{y'}(\psi) = ?$

$$R_x(\theta) \circ R_y(\psi) \circ R_x(-\theta)\ P'(x,y,z)$$

On ramène le résultat dans le repère global

On applique la transformation

On ramène le point dans le repère local de l'objet

D'où : $R_{y'}(\psi) \circ R_x(\theta)\ P(x,y,z) = R_x(\theta) \circ R_y(\psi) \circ \cancel{R_x(-\theta) \circ R_x(\theta)}\ P(x,y,z)$

Les rotations dans le système d'Euler (local) sont équivalente aux rotations par rapport à des axes fixes mais dans l'ordre inverse :

$$R_{z'}(\alpha) \circ R_{y'}(\psi) \circ R_{x'}(\theta) = R_x(\theta) \circ R_y(\psi) \circ R_z(\alpha)$$

Matrices de transformations

- rotations 2D, 3D : $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

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

- translations $T = \begin{pmatrix} tx \\ ty \\ tz \end{pmatrix}$

- changement d'échelle $S = \begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & Sz \end{pmatrix}$

SKIP

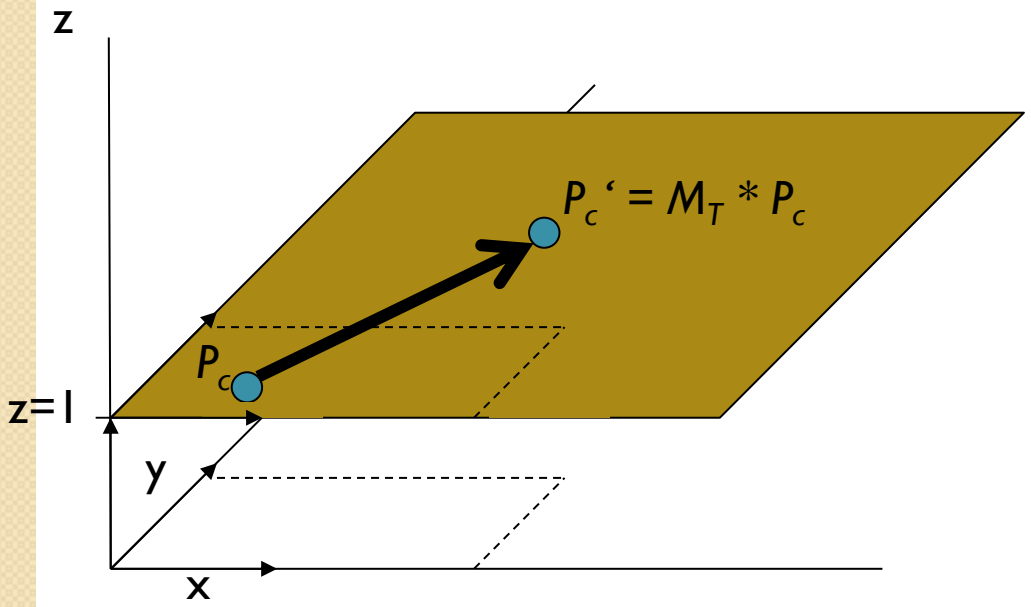
Coordonnées Homogènes

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



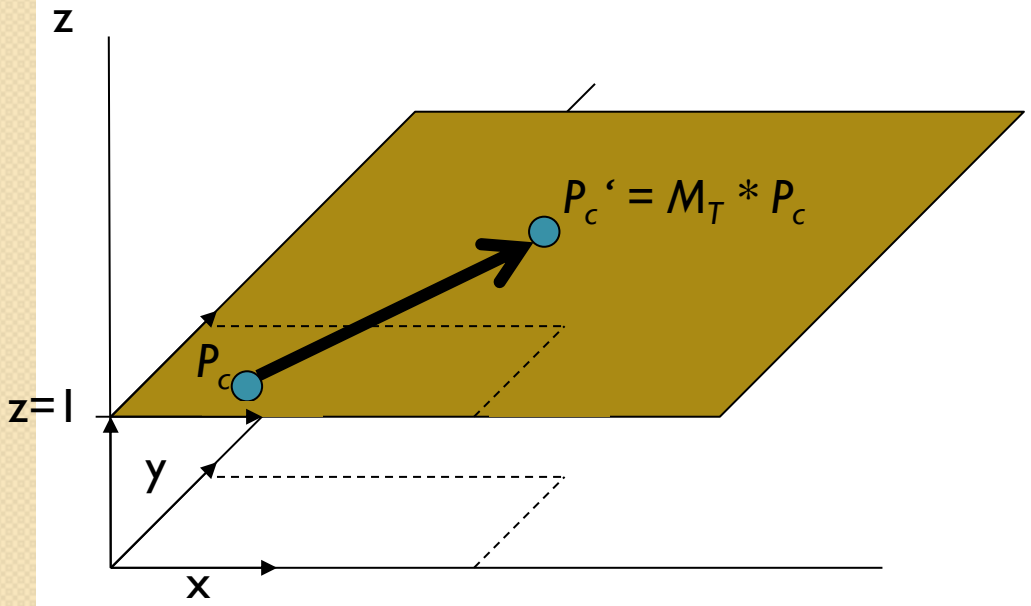
Coordonnées Homogènes

Matrice de transformation dans R^{d+1}

d=2

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

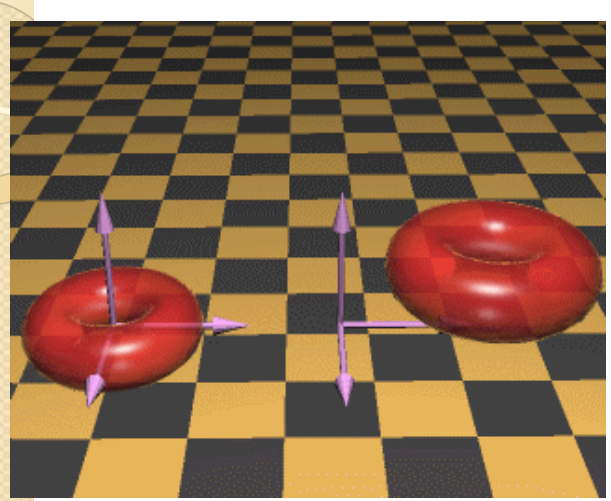
Coordonnées Homogènes
géométrie projective



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

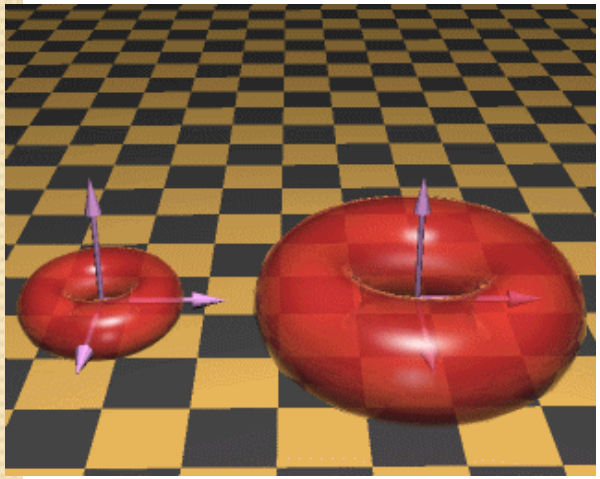
$$M_T = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Translation et mise à l'échelle (d=3)



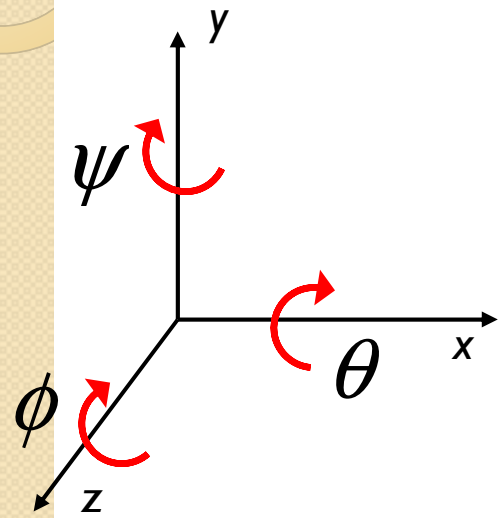
$$M_T = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad P' = M_T P$$

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

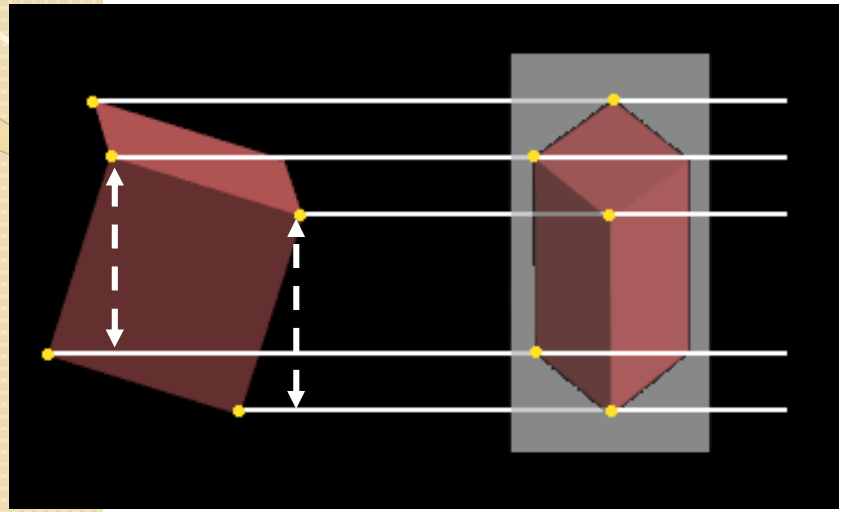
$$M_H = \begin{bmatrix} R_{3,3} & T \\ 0_3^t & 1 \end{bmatrix} \qquad P' = M_H . P$$

M_H : matrice orthonormée

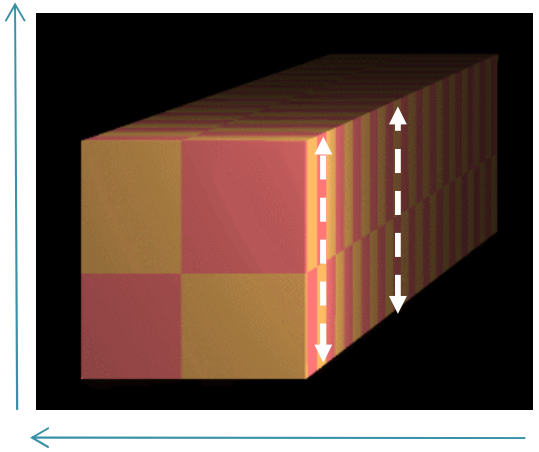
$$M_H^{-1} = \begin{bmatrix} R_{3,3}^T & -R.\ T \\ 0_3^t & 1 \end{bmatrix}$$

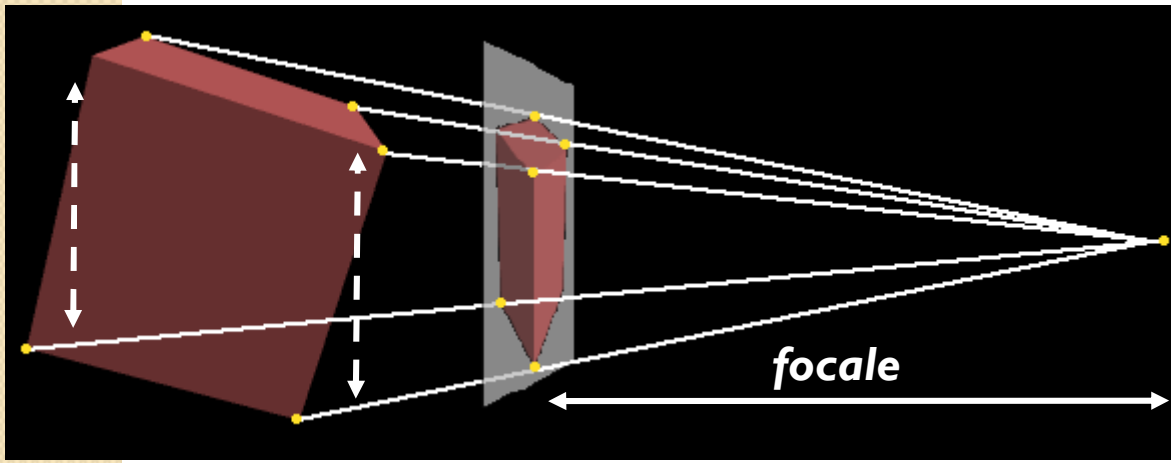
On retrouve facilement les angles de rotation à partir de M_H

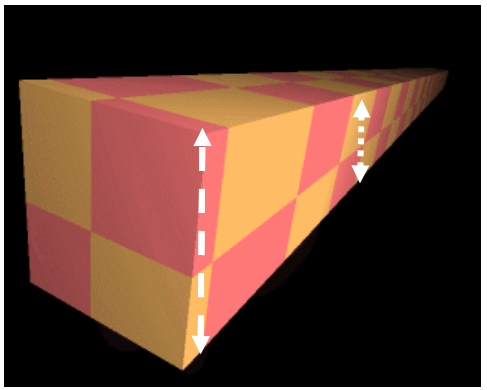
Projections



plan image

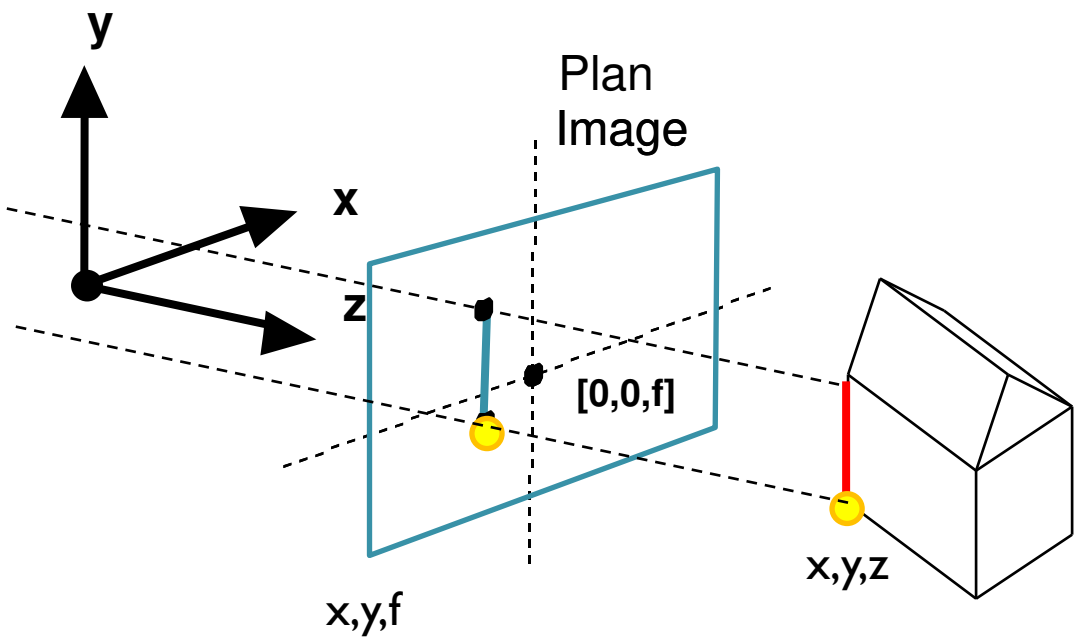






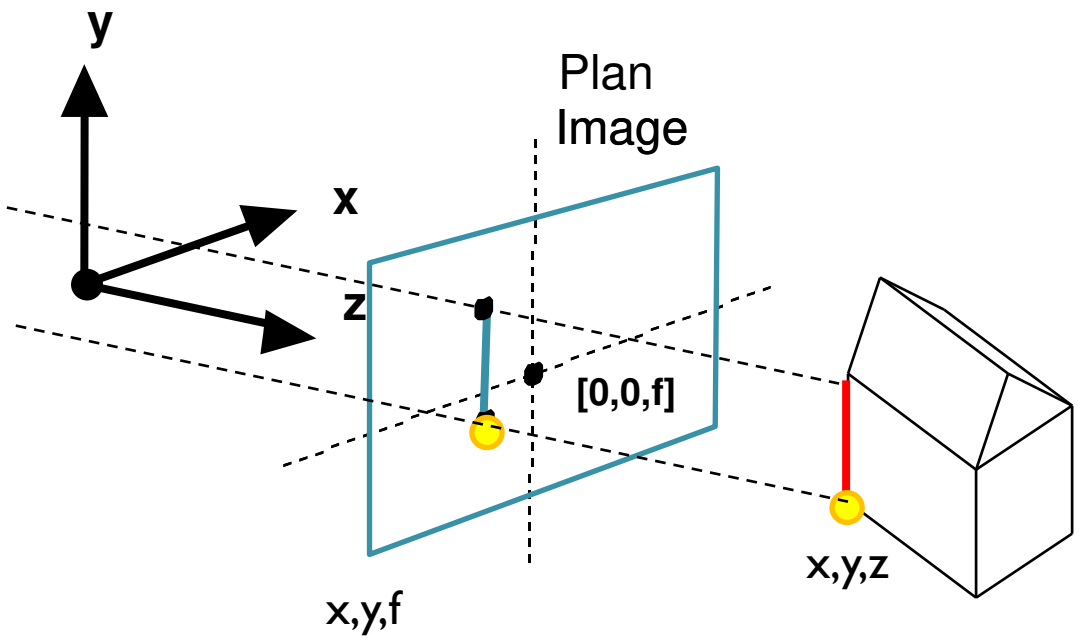
Projection orthographique

Repère caméra (œil)



Projection orthographique

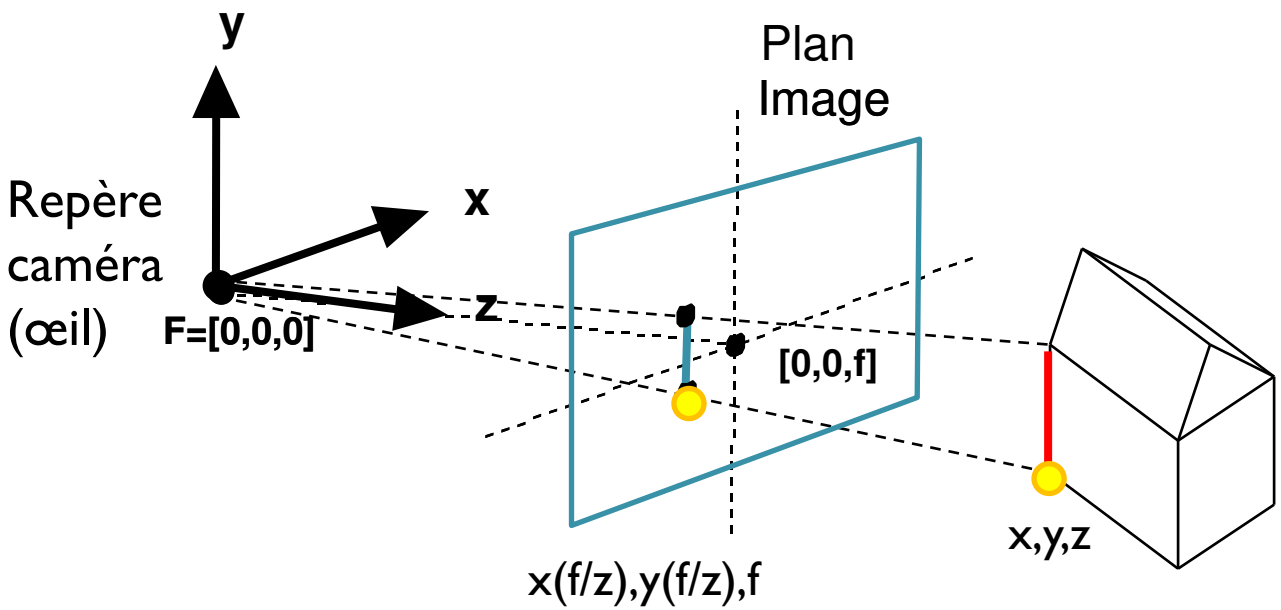
Repère caméra (œil)



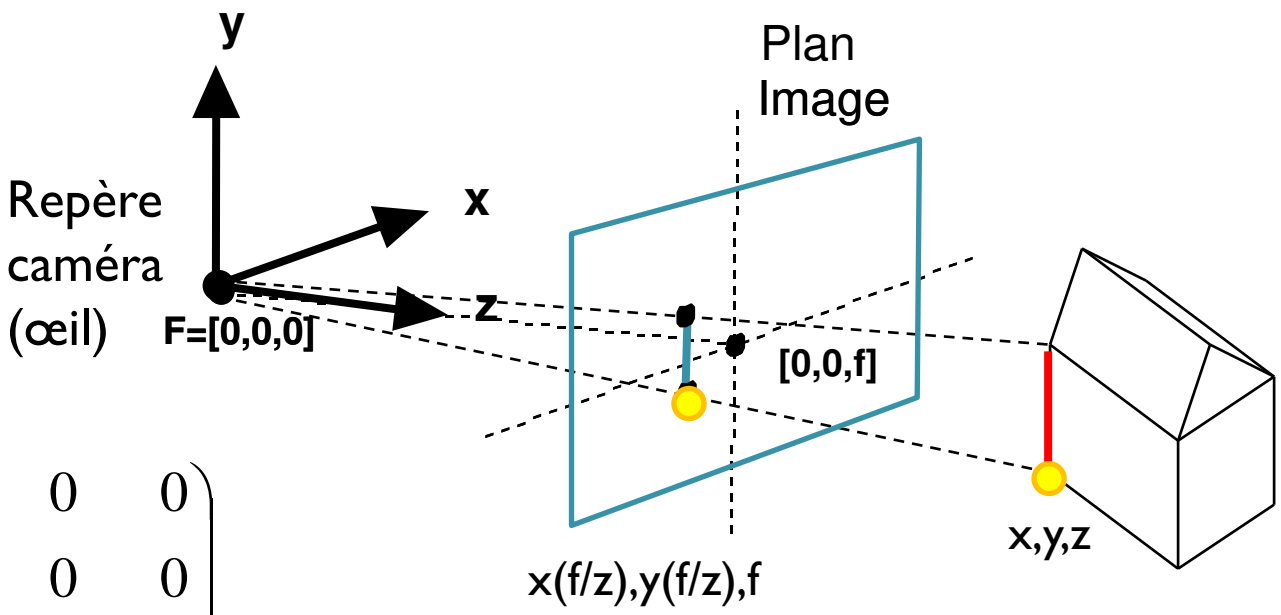
$$M_{per} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & f \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

M_{per} *(x,y,z,1) = (x,y,f,1) qui donne en 3D : (x, y, f)

Projection perspective

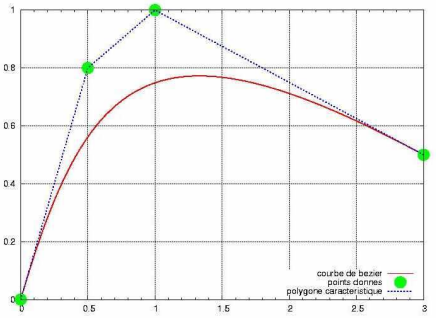


Projection perspective



$$M_{per} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix}$$

M_{per} *(x,y,z,1) = (x,y,z,z/f) qui donne en 3D : (x*(f/z), y*(f/z), f)



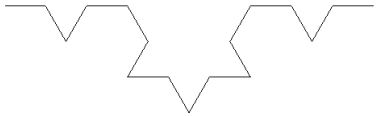
points➔ courbes ➔ surfaces

6. AUTRES APPROCHES CONSTRUCTIVES

L*-system

0→F

F → F+F-F-F+F

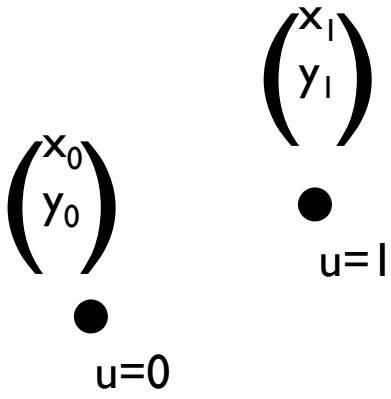


*Lindenmayer

Courbes polynomiales : principe

Une courbe polynomiale paramétrique contrôlée par les points

$$\begin{cases} x(u) = a_0 * u + b_0 \\ y(u) = a_1 * u + b_1 \end{cases}$$



Résolution système linéaire

$$\begin{pmatrix} x(u) \\ y(u) \end{pmatrix} = \begin{pmatrix} x_1 - x_0 & x_0 \\ y_1 - y_0 & y_0 \end{pmatrix} * \begin{pmatrix} u \\ 1 \end{pmatrix}$$

$$X(u) = \sum_{i=0}^{m} X_i \, P_{im}(u)$$

P_{01} P_{11}

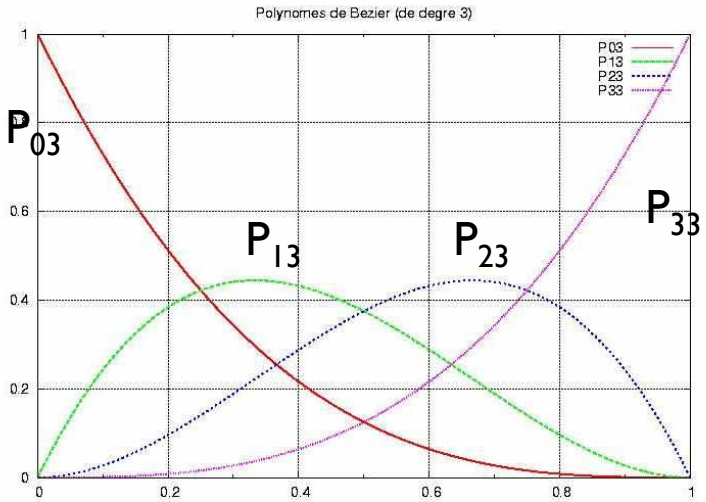
$$\begin{pmatrix} x(u) \\ y(u) \end{pmatrix} = \begin{pmatrix} x_0 & x_1 \\ y_0 & y_1 \end{pmatrix} * \begin{pmatrix} 1-u \\ u \end{pmatrix}$$

$$\begin{pmatrix} x(u) \\ y(u) \end{pmatrix} = \begin{pmatrix} x_0 & x_1 \\ y_0 & y_1 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ u \end{pmatrix}$$

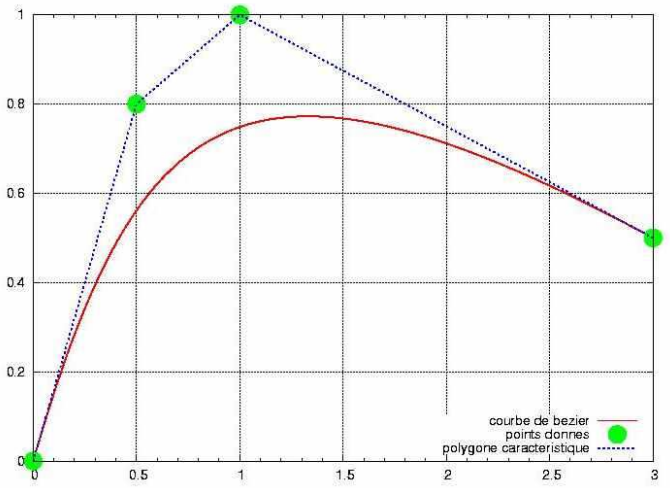
Exemple : la courbe de Bézier

$$X(u) = \sum_{i=0}^{m} X_i P_{im}(u)$$

$$P_{im}(u) = (m!/((m-i)!\, i!))\, u^i (1-u)^{m-i}$$



$$\begin{pmatrix} x(u) \\ y(u) \end{pmatrix} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \end{pmatrix} \begin{vmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{Vmatrix} 1 \\ u \\ u^2 \\ u^3 \end{Vmatrix}$$



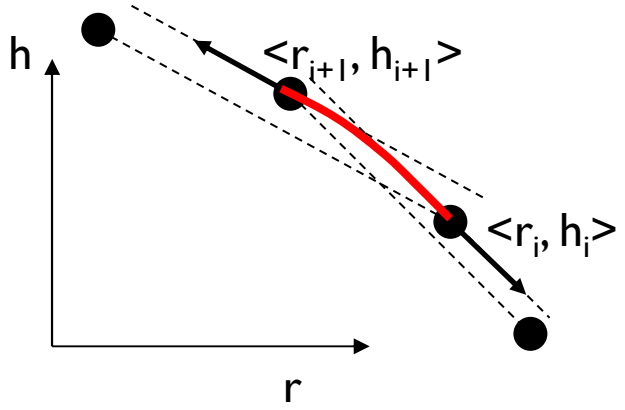
Constructeurs : S.O.R

S.O.R : Surface de révolution autour d 'un axe

PovRay

```
sor {
  nb_pt,
  <r_1, h_1>
...
  <r_i, h_i>
  open
}
```

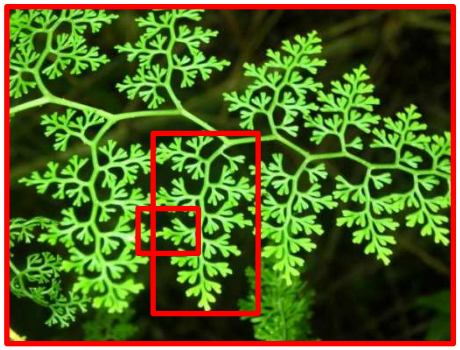
$r(h)^2 = a\,h^3 + b\,h^2 + c\,h + d$

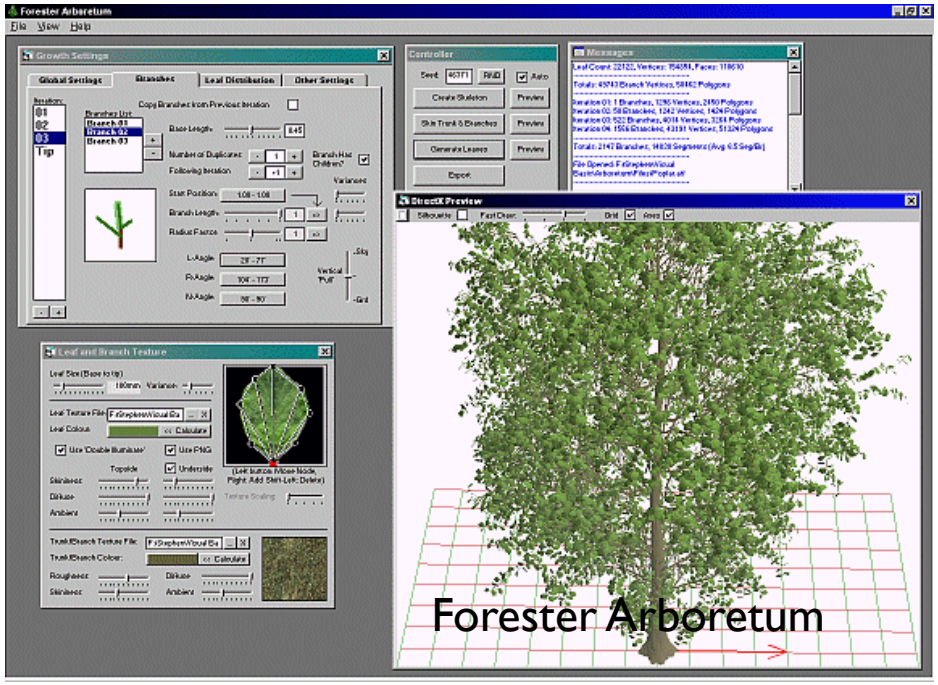


Fractales, L-system et approches procédurales

- Les fractales dans la nature
- Auto-réplication parfaite ou avec aléas…
- L-system : DOLsystem / SOL

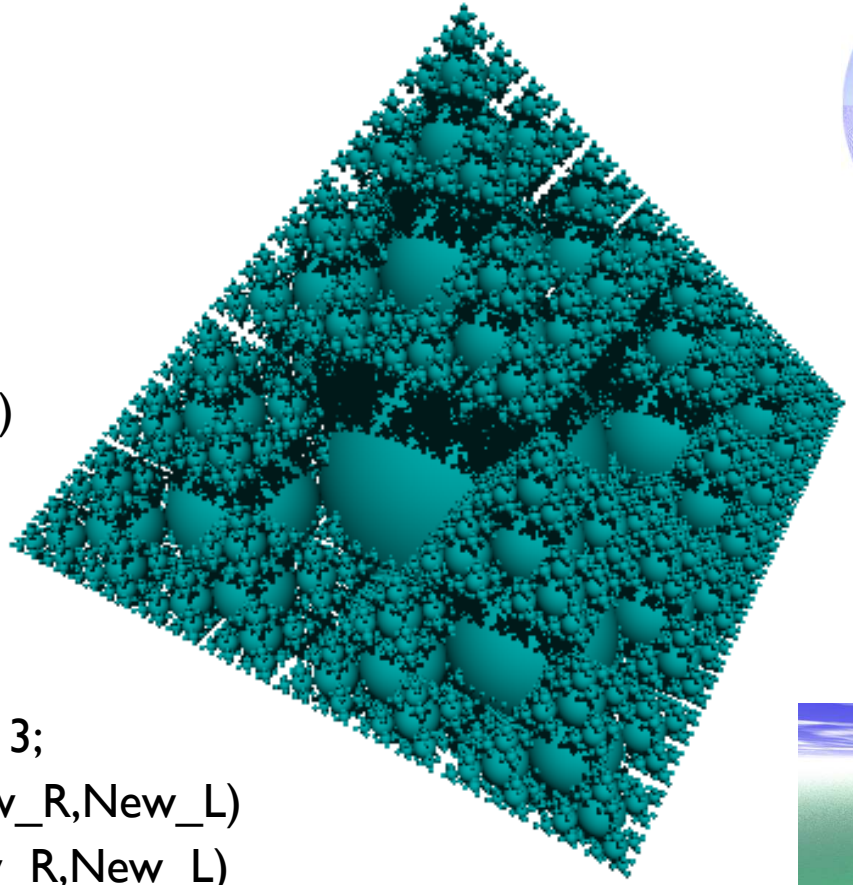
Deterministic-O-context / Stochastic-O-context



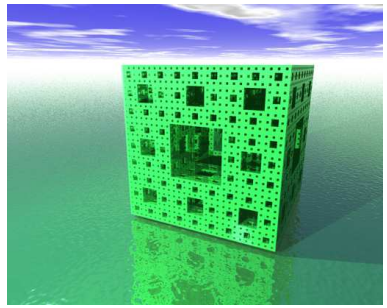


Logiciels : Forest Arboretum, Virtual Terrain Project, plug-in Maya, Lightwave etc…

Exemple




```
#macro Pyramid(X,Y,Z,R,L)
 sphere { <X,Y,Z>,R}
  #if (L > 0)
    #local New_L = L - 1;
    #local New_R = R / 2;
    #local Pos   = New_R * 3;
    Pyramid(X+Pos,Y,Z,New_R,New_L)
    Pyramid(X-Pos,Y,Z,New_R,New_L)
    Pyramid(X,Y+Pos,Z,New_R,New_L)
    Pyramid(X,Y-Pos,Z,New_R,New_L)
    Pyramid(X,Y,Z+Pos,New_R,New_L)
  #end
#end
```



Courbe et flocon de Von Koch

Courbe de Von Koch

+ tourner à gauche de 90°

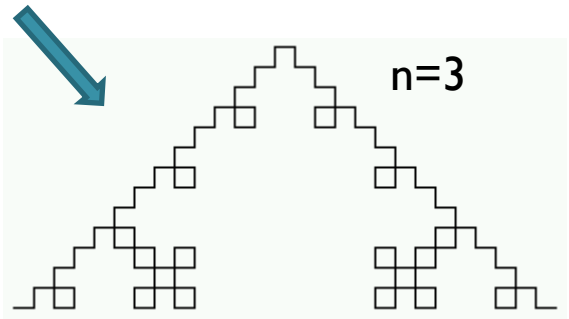
- tourner à droite de 90°

F = segment

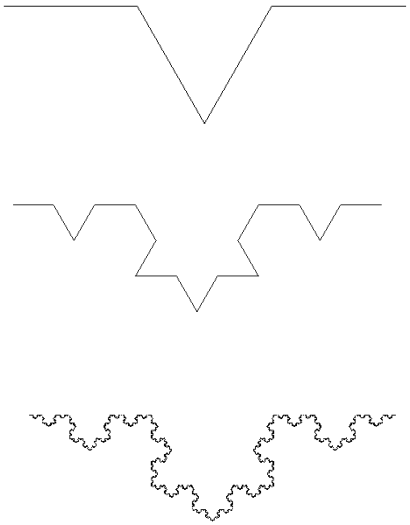
0→F

F → F+F-F-F+F

n=1 



Flocon de Von Koch



Modélisation des arbres

STAGE 2017: Raphaël BELUS chez E-on Software

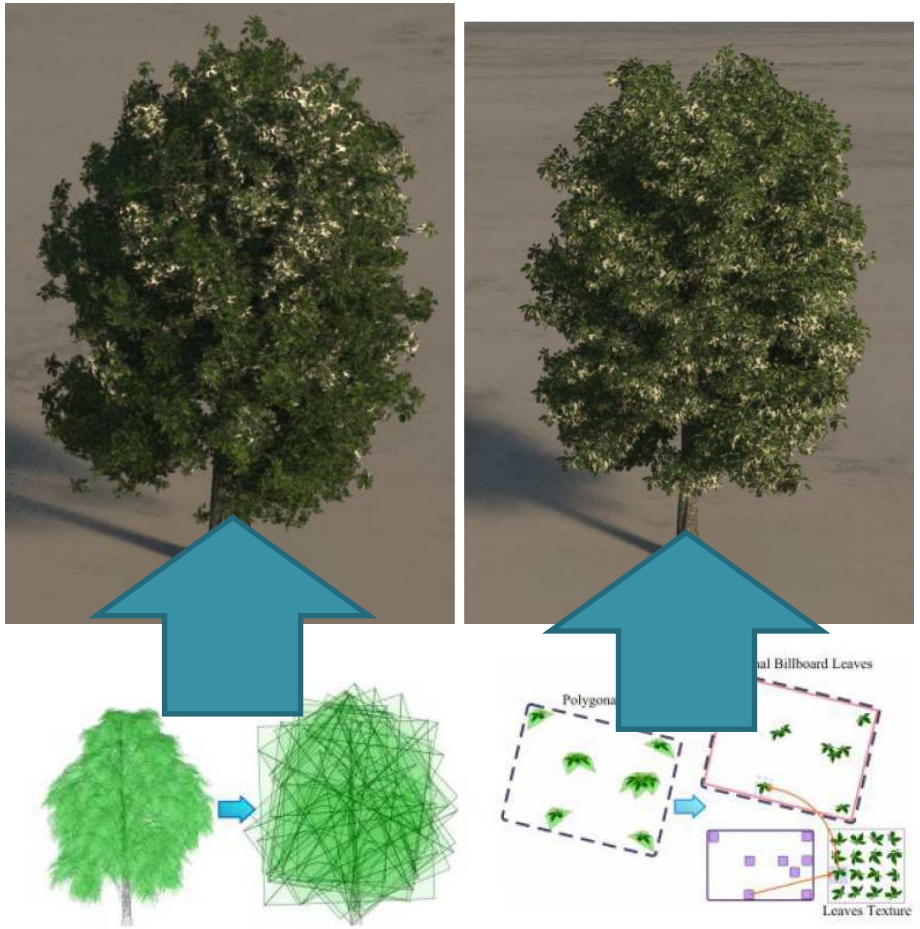


FIGURE 3.9: Description du modèle de Garcia, à gauche un schéma montre la conversion d'un arbre géométrique basique vers un arbre avec un nuage d'imposteurs, à droite un schéma montre la conversion des feuilles géométriques à de simples textures

Historique :

Ulam 1962, Honda 1971,
Lindenmayer 1974,
…
Garcia 2007, Coté 2009…

Logiciels :

SpeedTree,
The Grove (Plug-in)
ngPlant, Xfrog(PI)

PlantFactory (e-on)

Exemples :VRML (X3D), Inventor, Java3D,…

www.web3d.org

vrmlview (VRML 1.0)
www.sim.no

cortona (VRML 2.0)
www.cortona3d.com

coin3D (VRML97+Inventor)
www.coin3d.org

Java3D java3d.j3d.org
www.xj3d.org

SGI : inventor V2.1 VRML 1.0
VRML 2.0 (VRML 97 version iso)
X3D

Exemples OpenSource

Systèmes interactifs :

VRML, Inventor, java3D…

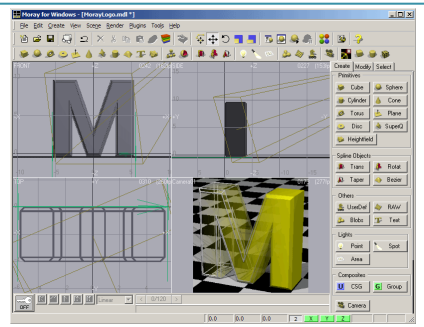
Coin3D -> VRMLview

OpenInventor (VGS)…

Browser HTML5 – X3D

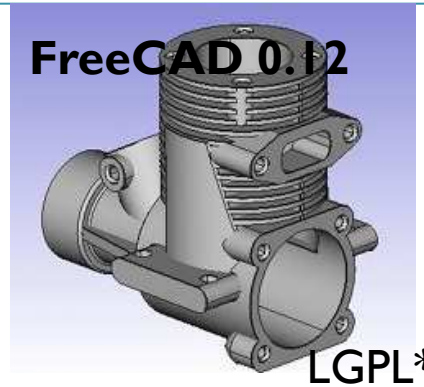
Synthèse d'images, Anim.:

Pov-Ray, Moray, Blender…



CAO :

FreeCAD, BrlCAD…



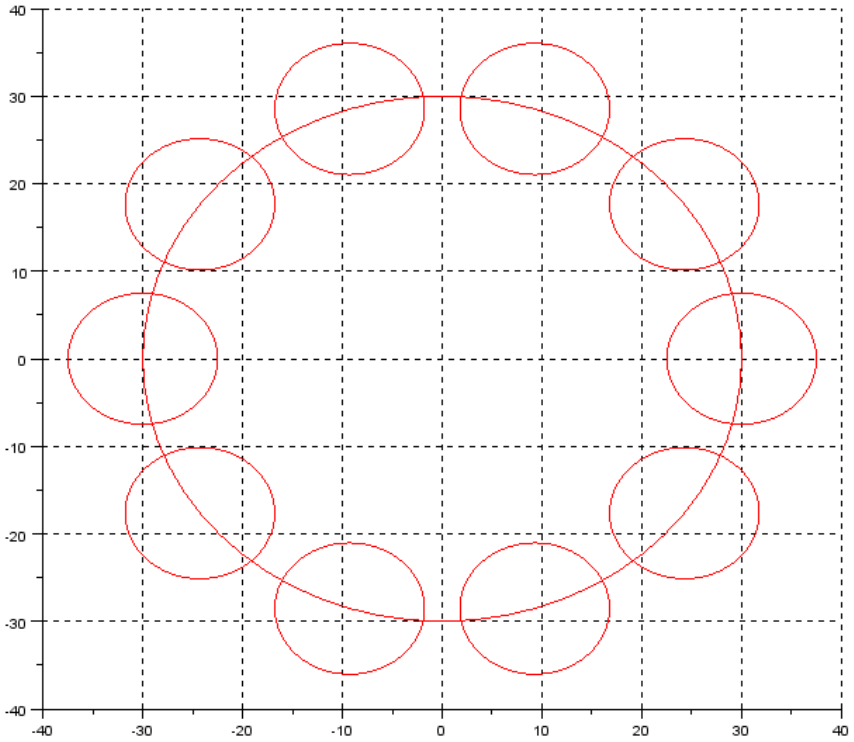
TP n°4 : C.S.G

- Parcours récursif des arbres
- PTdansCSG :

  Estimation de l'aire
- Ajout d'une primitive :

  Triangle
- Ajout d'une méthode :

  Boite d'encombrement



TP n°4 : C.S.G & list

```
// TYPE sous SCILAB
A=(10+(5+27)/32)
              11
// une expression est evalue
As=string(A)
              11
typeof(A)
              constant
A+10
              21
typeof(As)
              string
As+10
              Error 144
As+ "+10"
              11+10
```

```
// Evaluation d'une chaine
S="(10+(5+27)/32)"
typeof(S)
              string
B=evstr(S)
              11
```

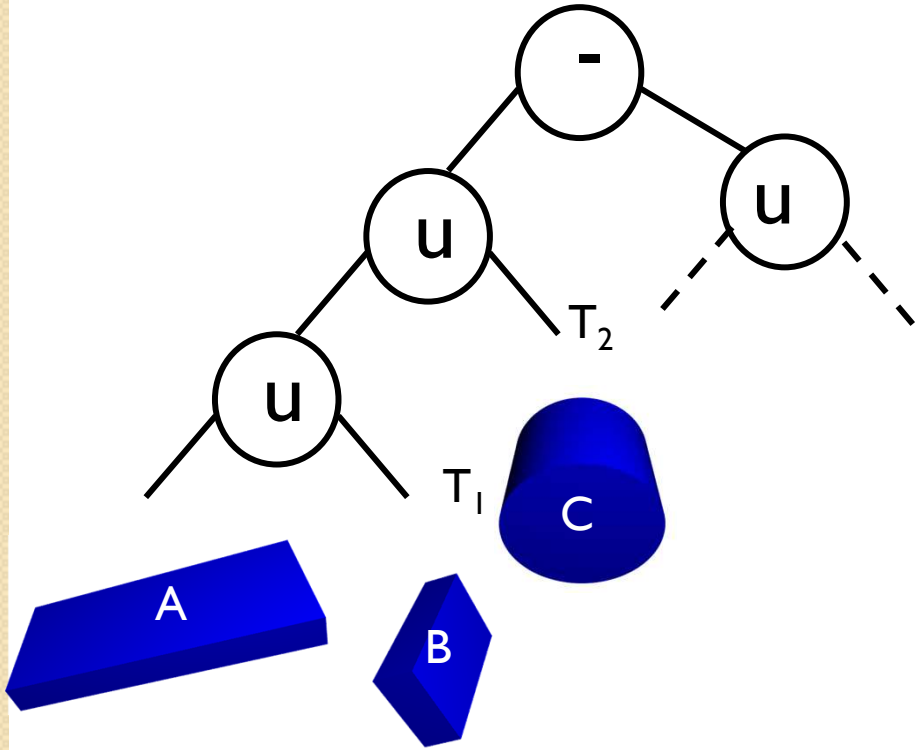
$$Arbre = (\cup, A, (T_1(B)))$$

```
// List = plusieurs types d'objets
L=list("10+",list("5+27"),"/",32) ;
L(1)
               10+
typeof(L)
              list
typeof(L(1))
              string
typeof(L(4))
              constant
```

```
// Un arbre CSG
A=list("Sphere",30);
B=list("Sphere",10);
T1=list("Translation", [0,10]);
Arbre=list("+",A,list(T1,B));
```

Exercice : structure d'arbre

- Représenter un arbre avec une liste



Exercice : parcours d'arbre

- Parcours à l'aide de la récursion