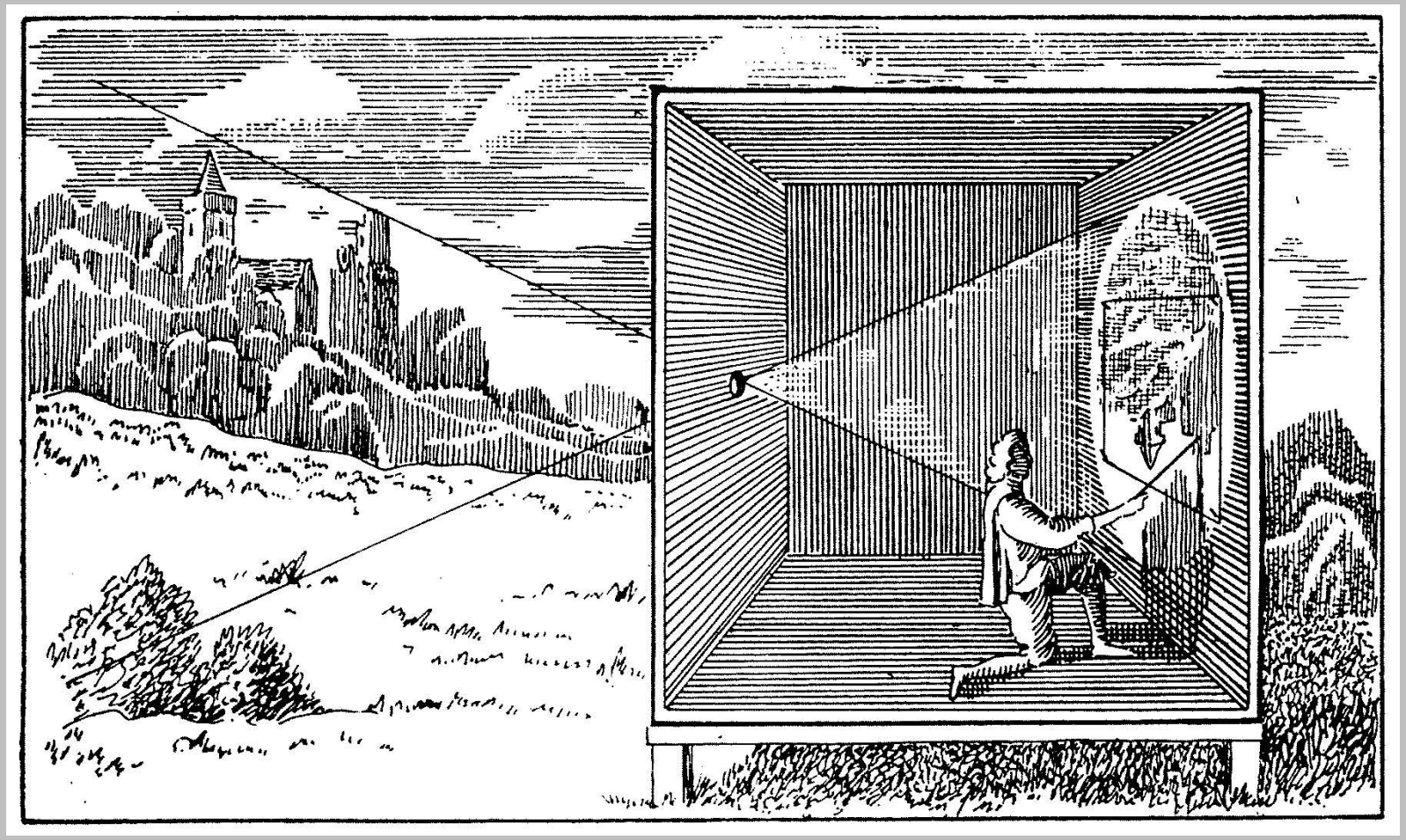
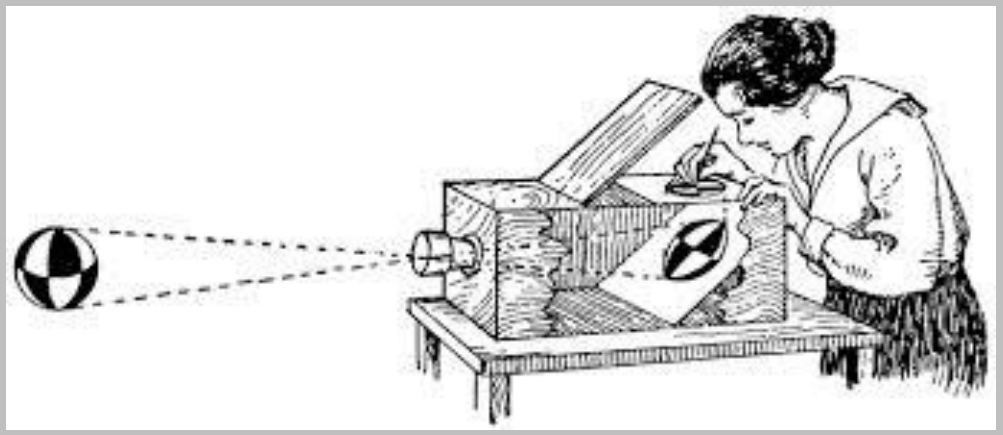
Informatique Graphique 3D

Introduction

Histoire de l’imagerie 3D





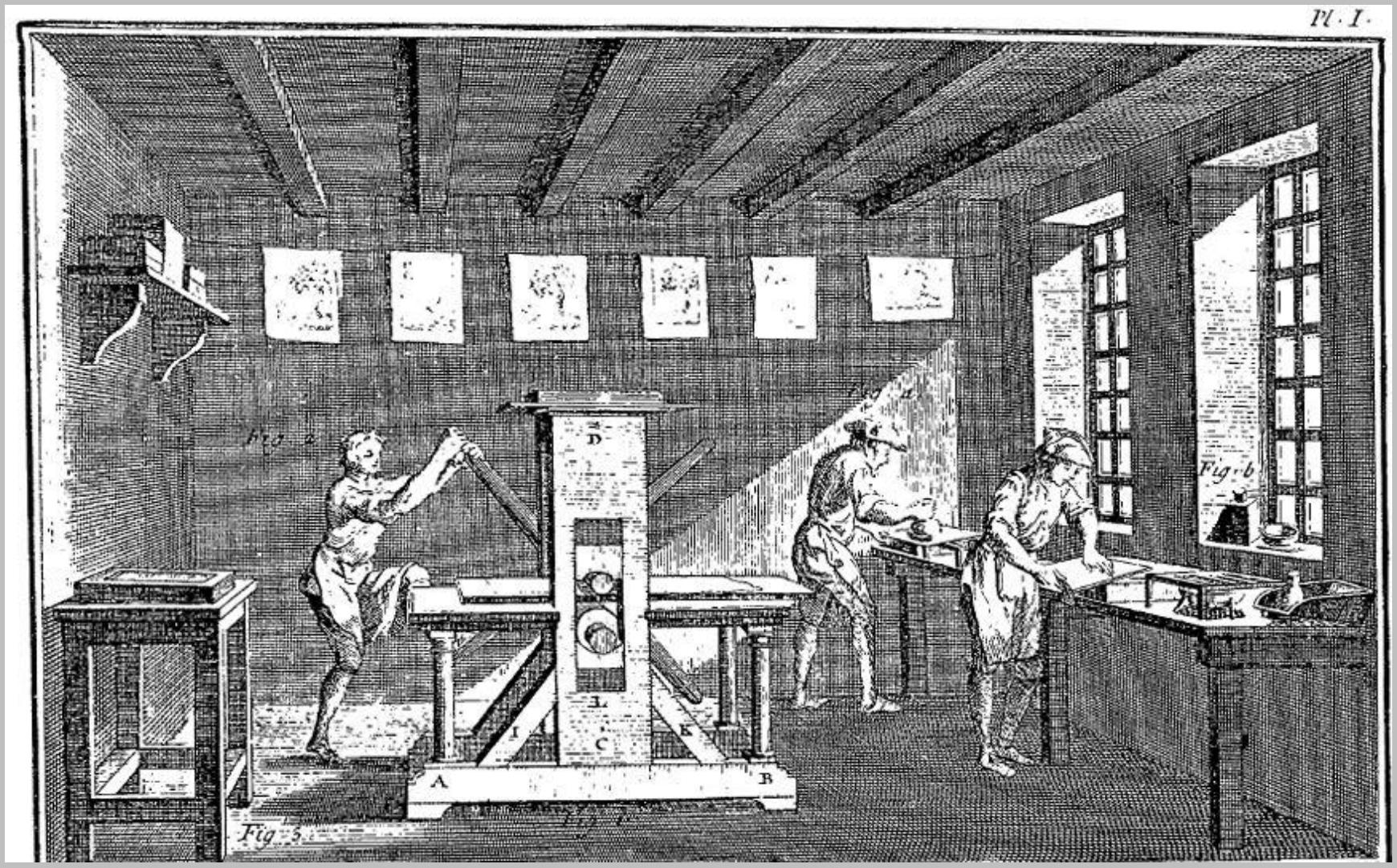
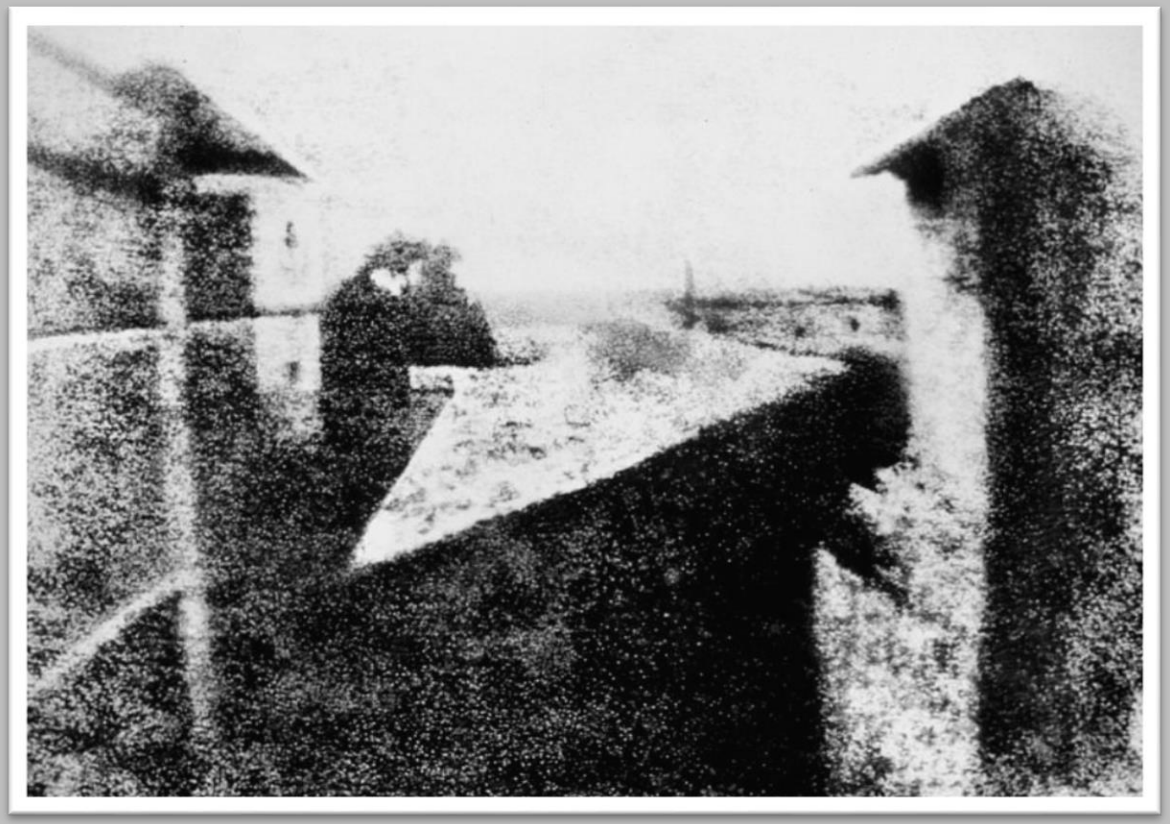
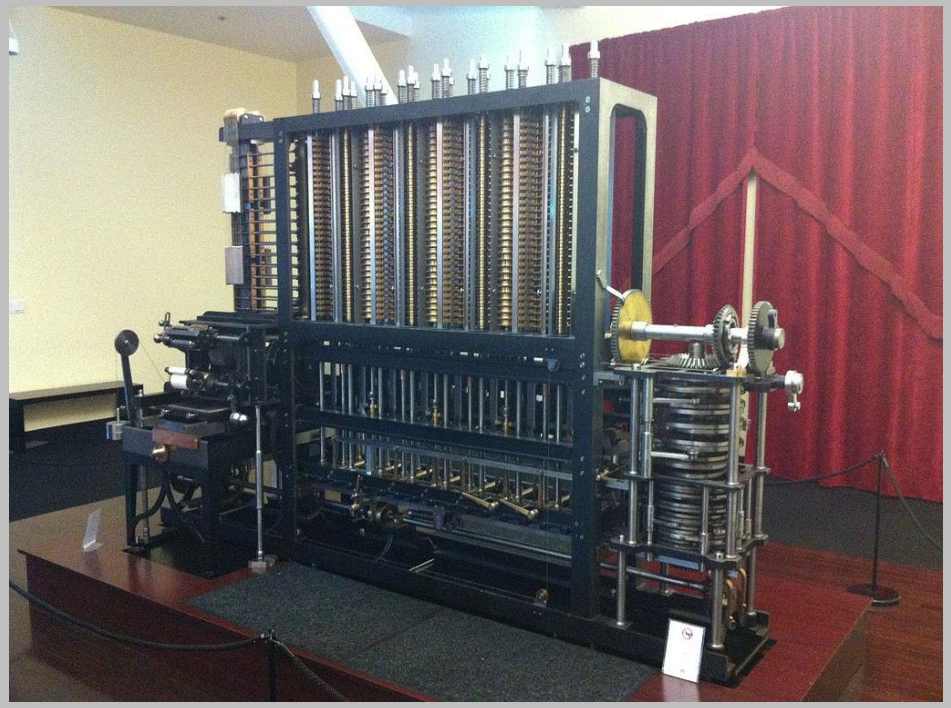
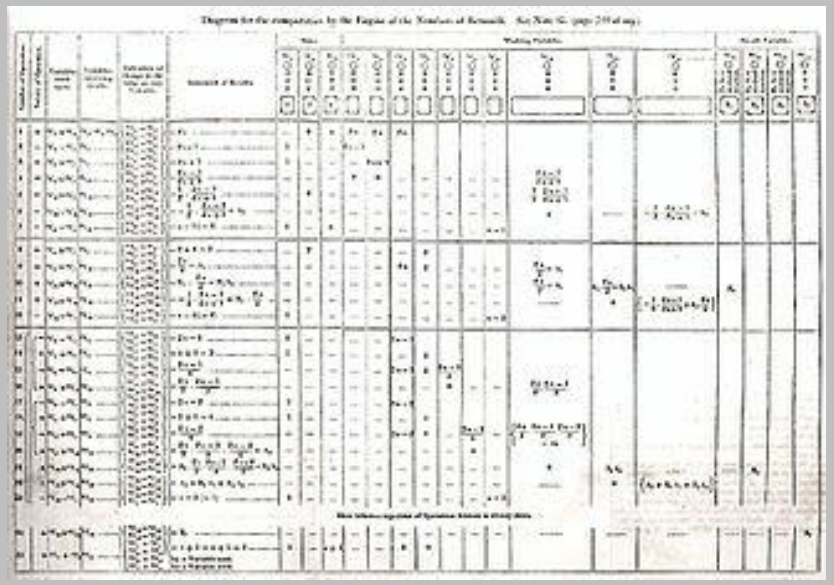
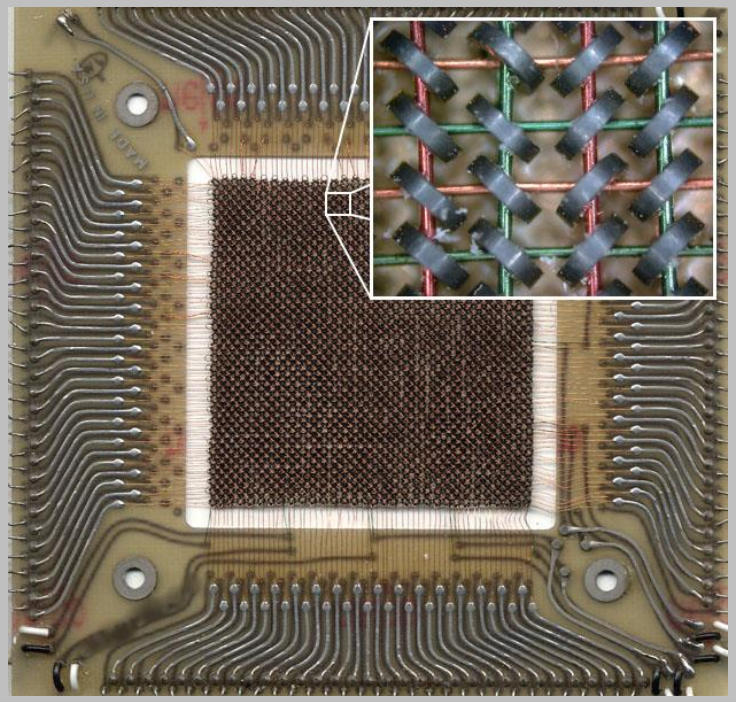


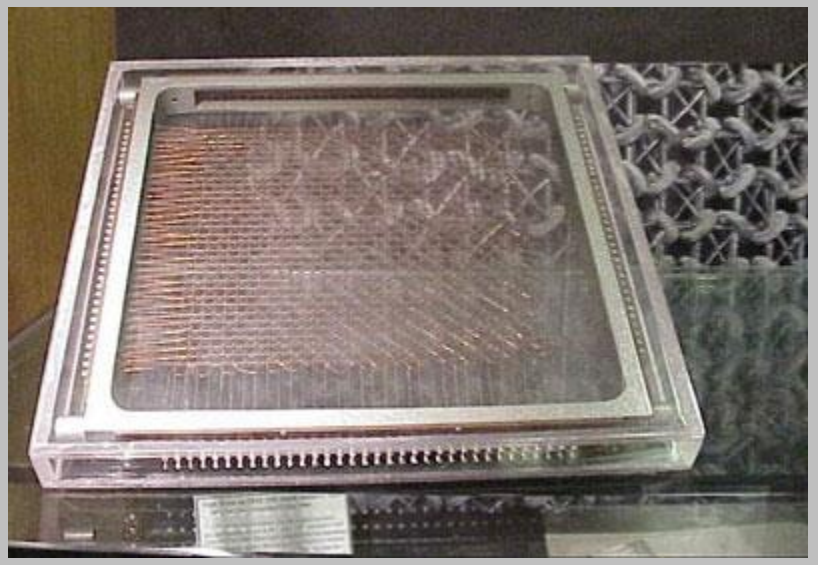
Fig. 2
D
E
C
I
K
A
B
Fig. 5
Fig. 6

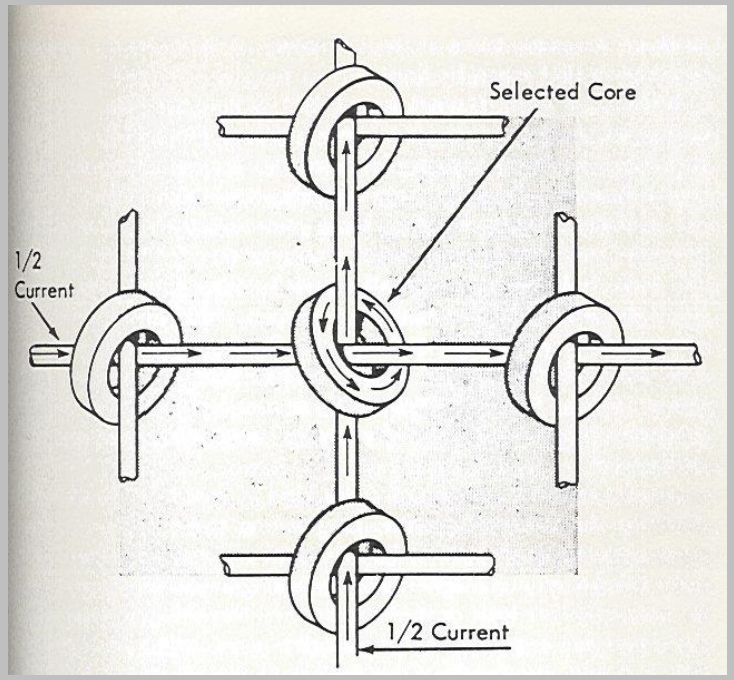








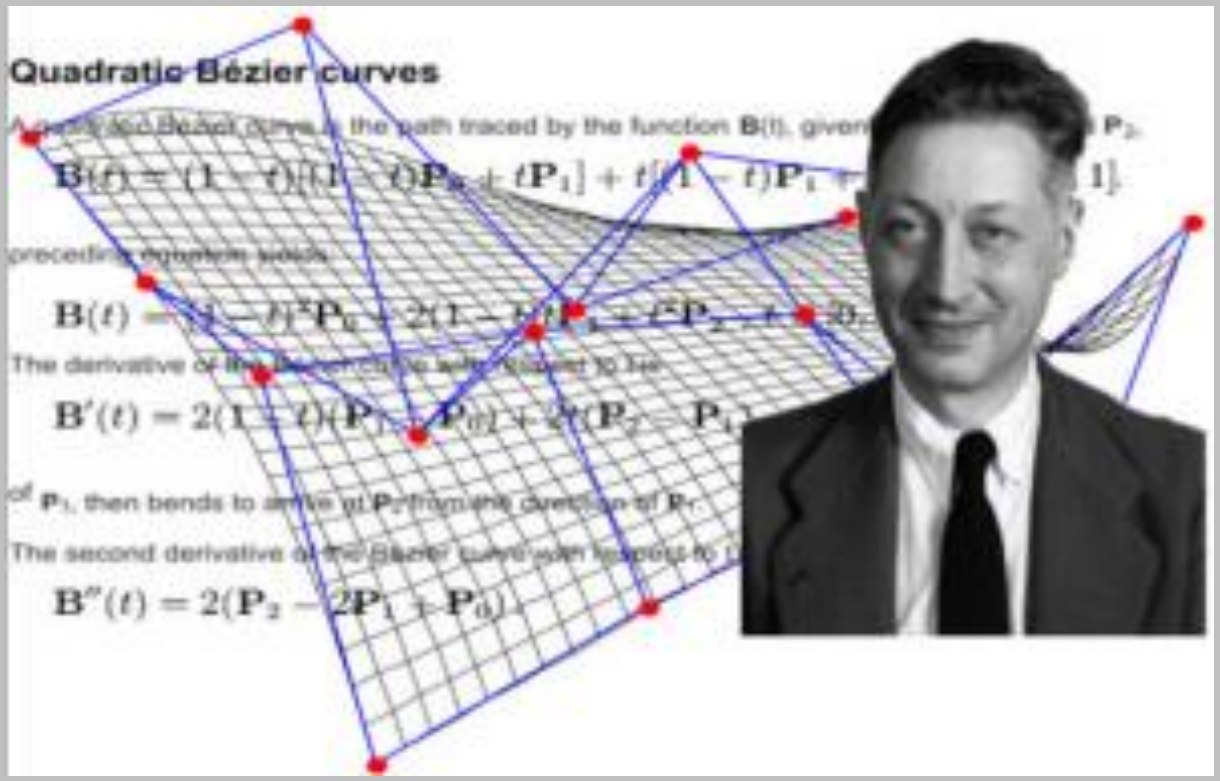



Selected Core
1/2
Current
1/2 Current

- 1948 - ENIAC

- 1961 – Spacewar! Premier jeu vidéo

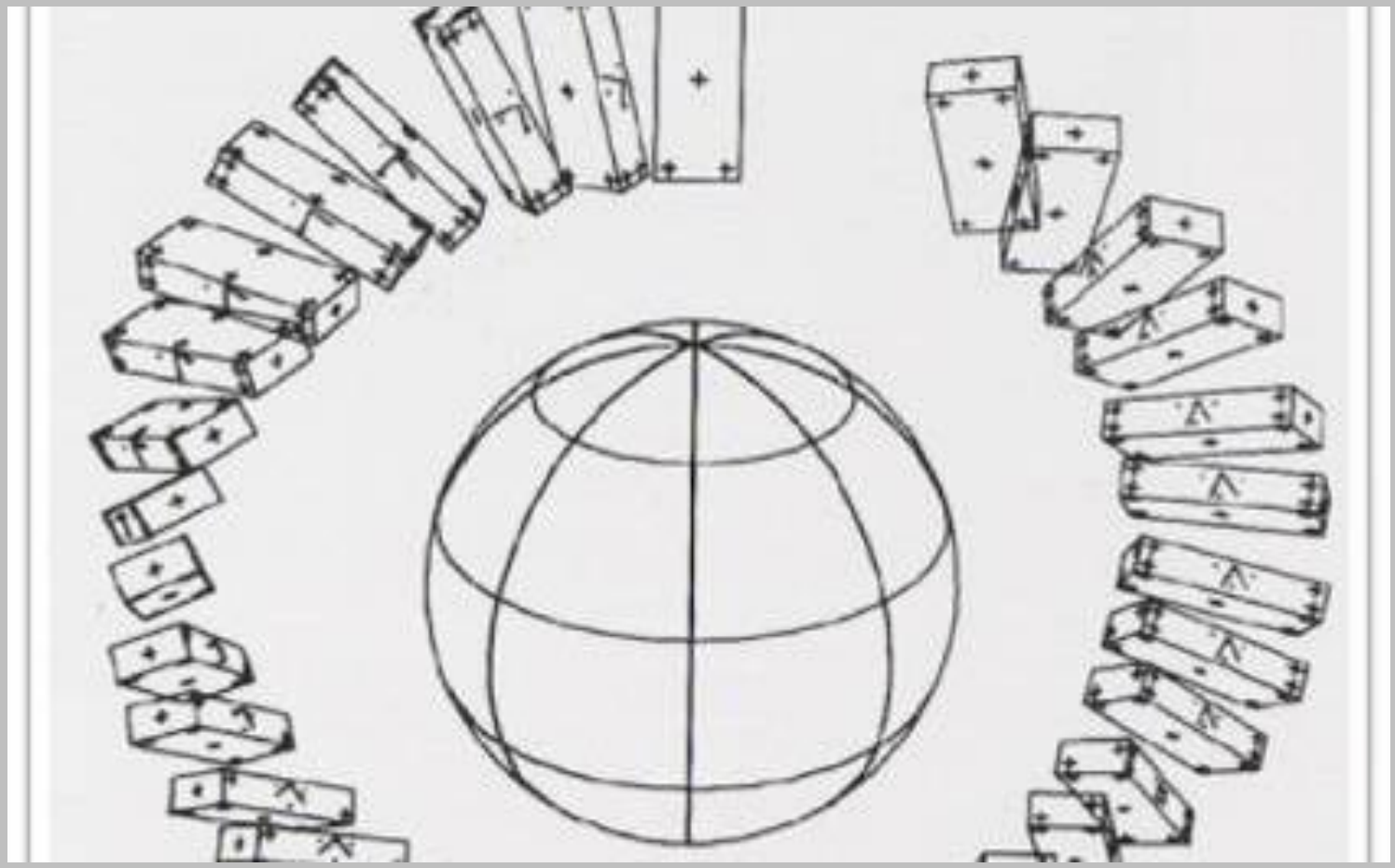
- 1962 – Courbe de Bézier



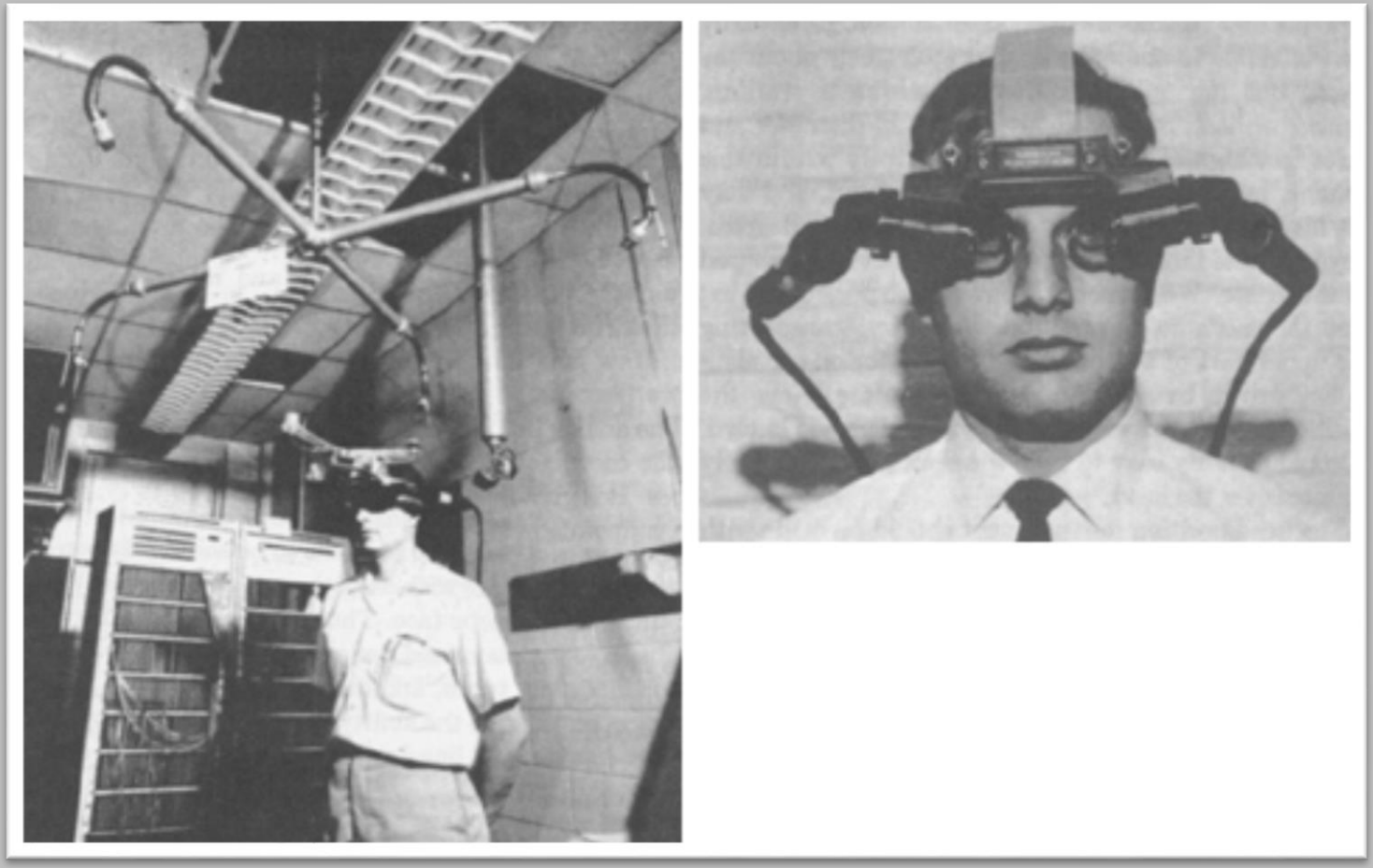
- 1963 – La souris

- 1963 - Sketchpad: premier logiciel de CAO

- 1963 – premier film généré par ordinateur



- 1968 – HMD par Sutherland

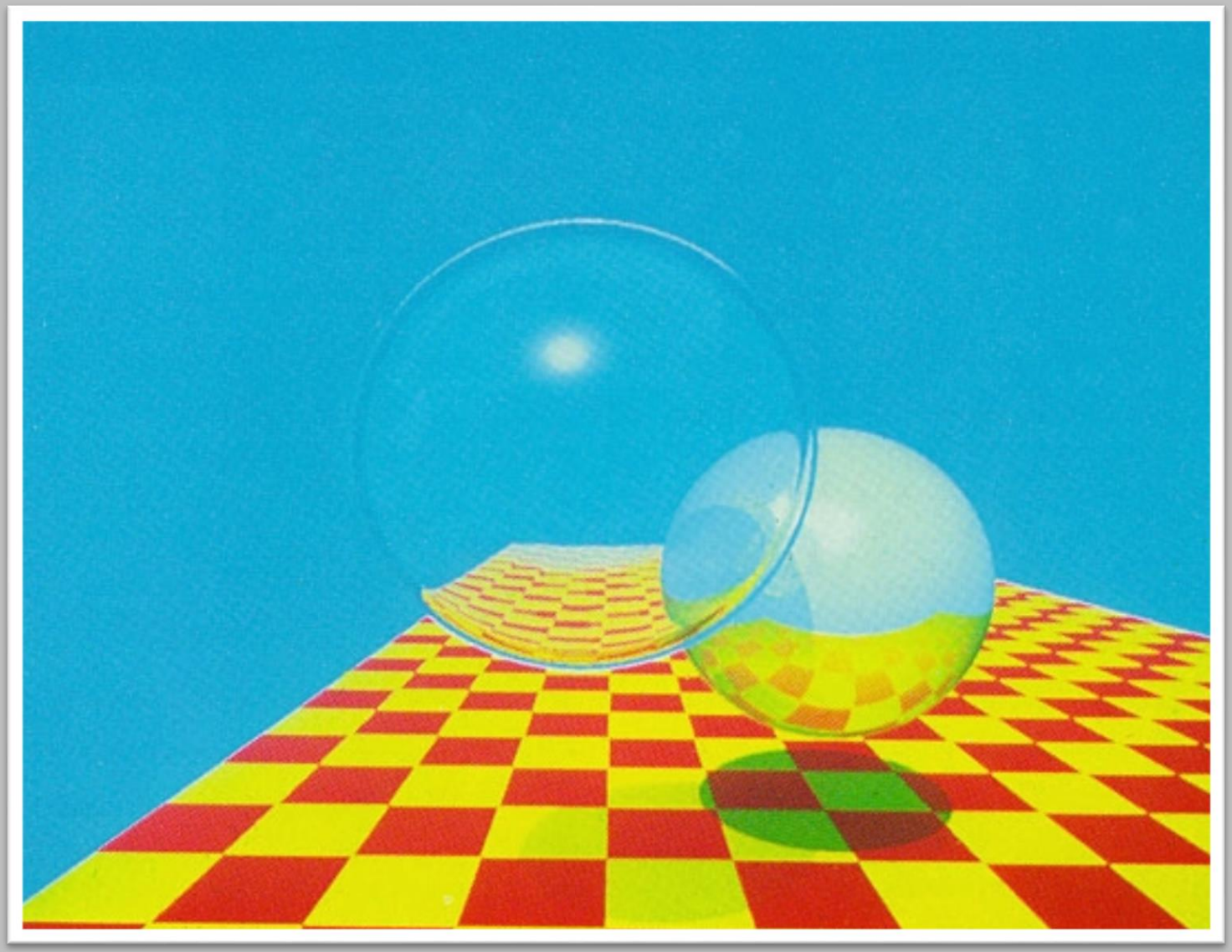


- 1971 – Ombrage de Gouraud

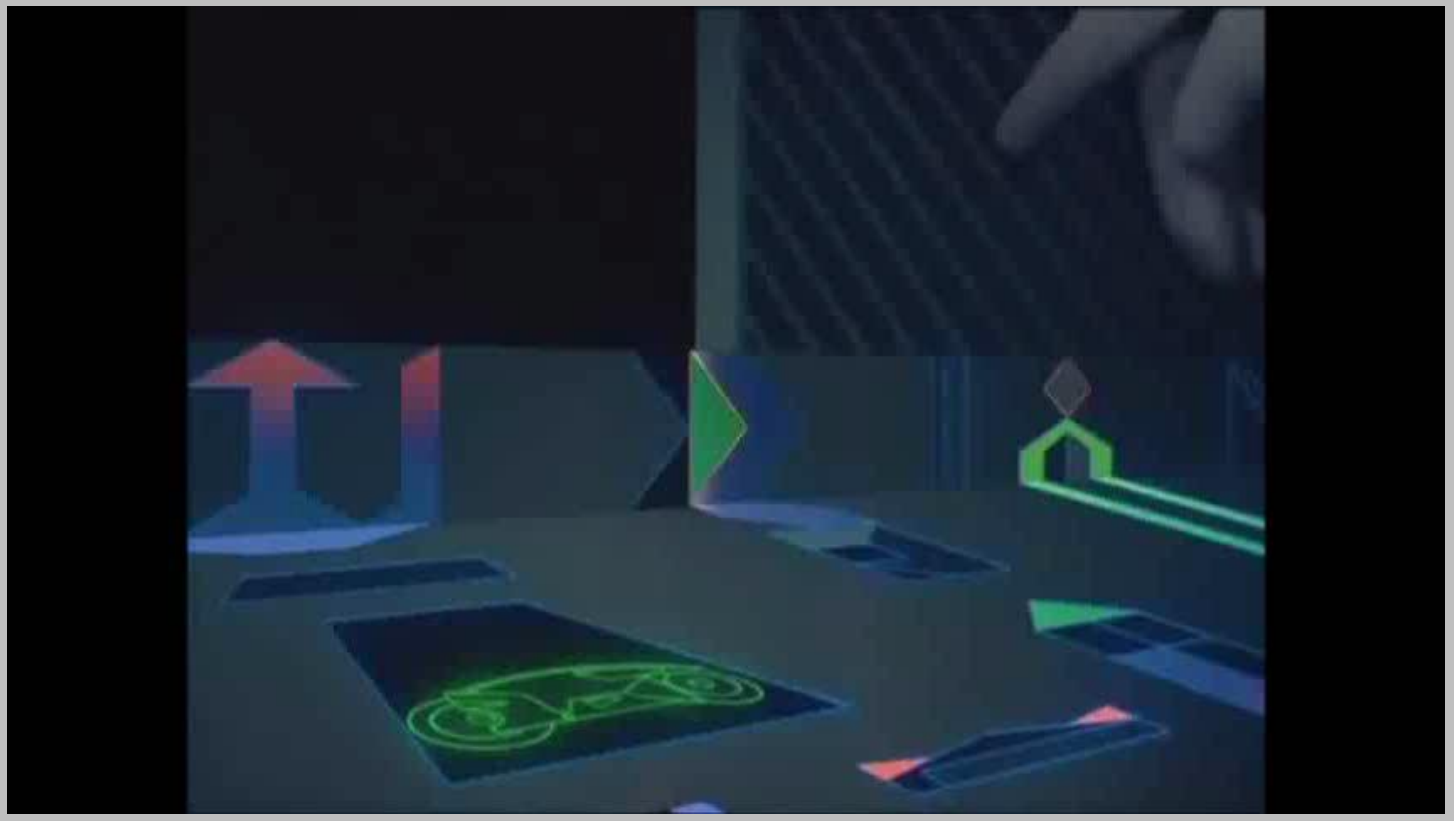
- 1972 – premier film avec rendu ombré par Ed Catmull

- 1976 – Environment mapping par J. Blinn

- 1980 – Lancer de rayon - Turner Whitted

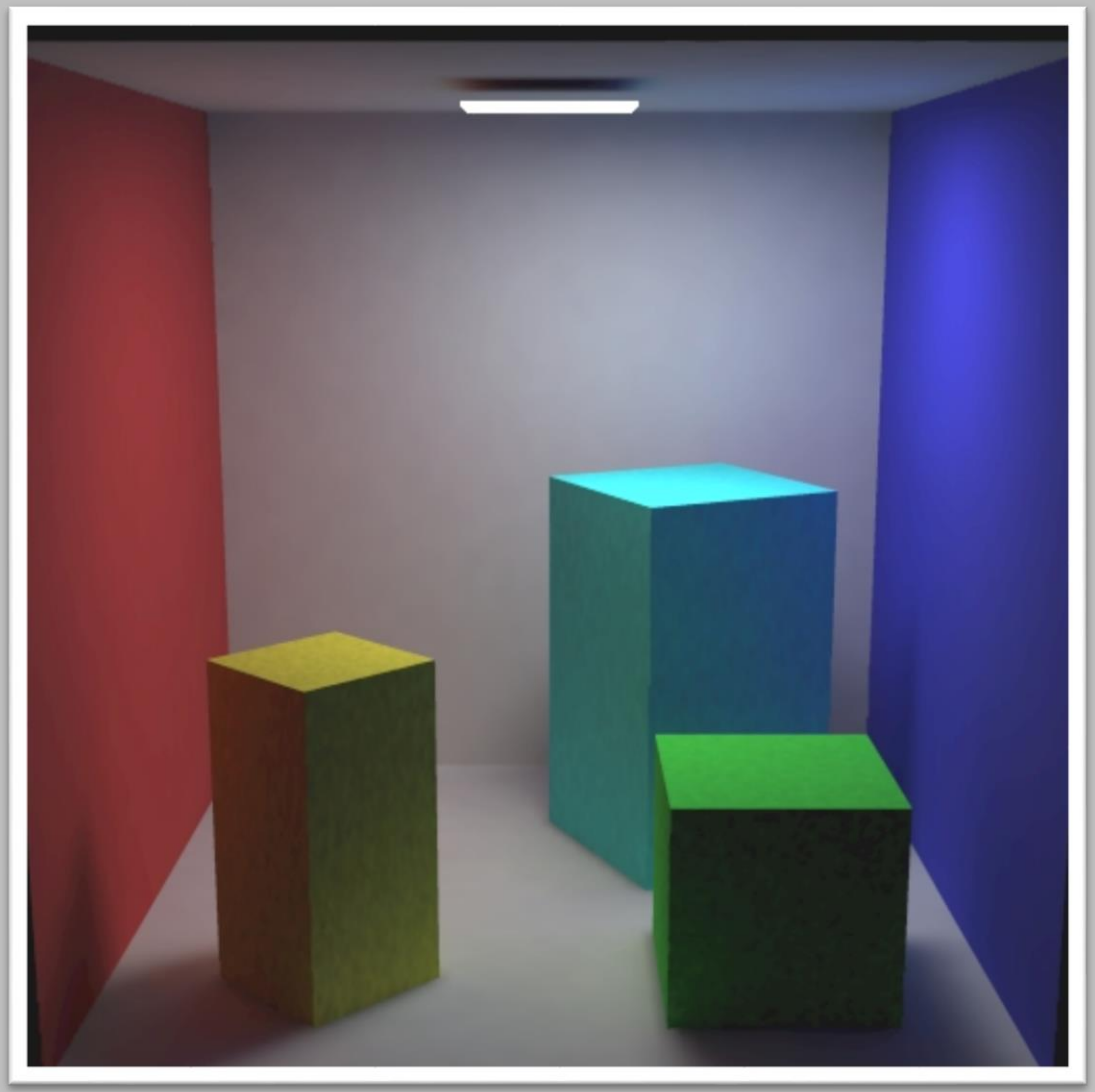


- 1980 – Film Tron



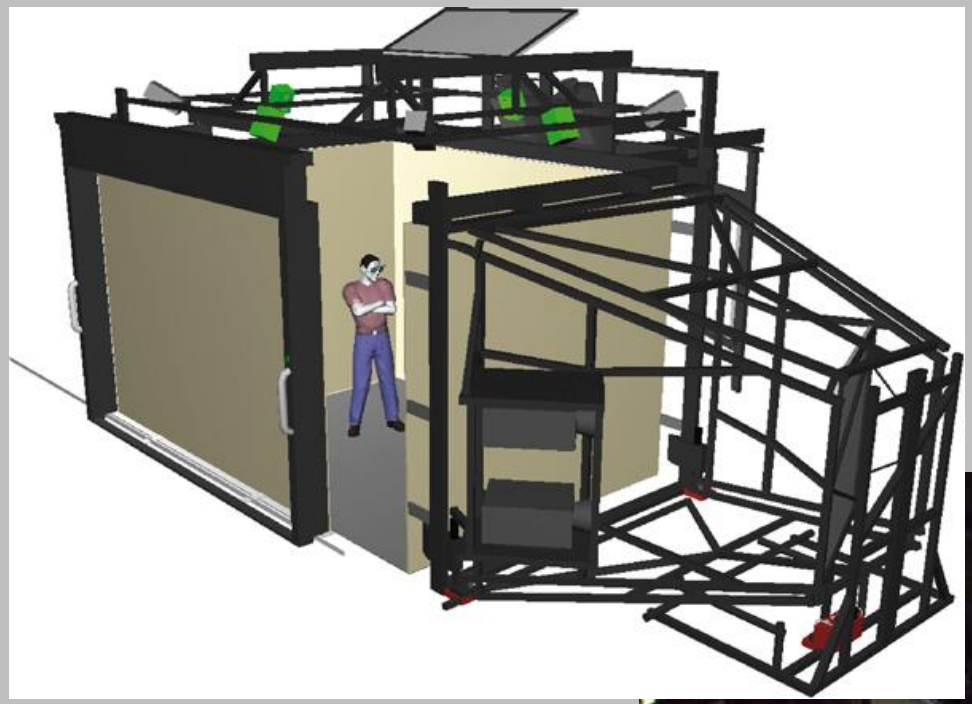
- 1982 – Première intégration réel/virtuel utilisant l'environment mapping - Gene Miller

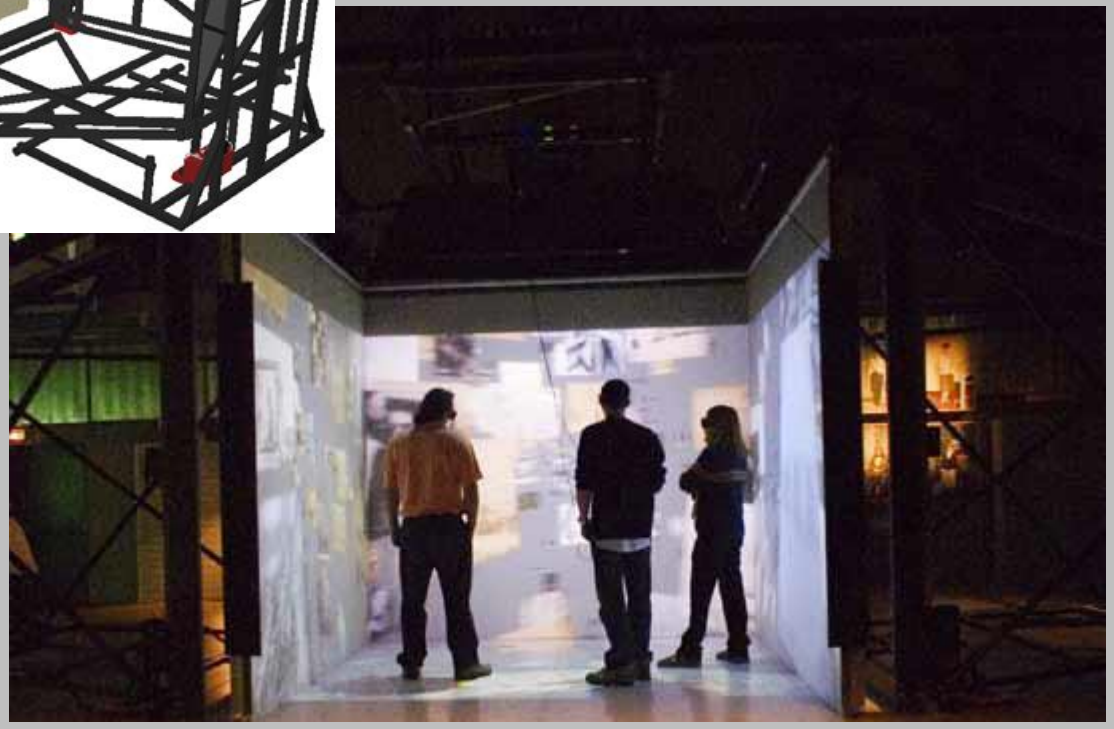
- 1984 – Radiosité - Goral



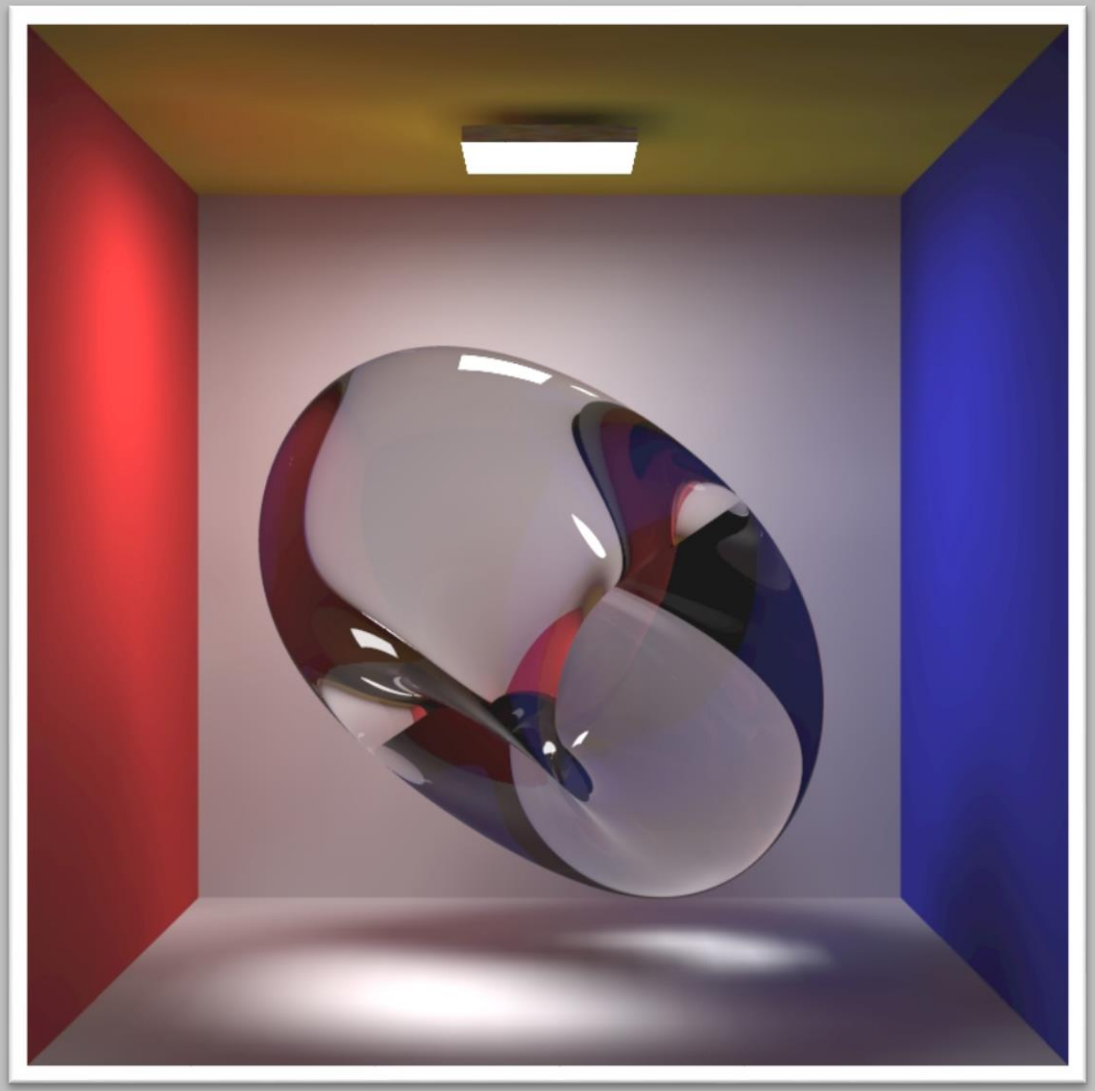
- 1984 – Motion blur

- 1992 – CAVE





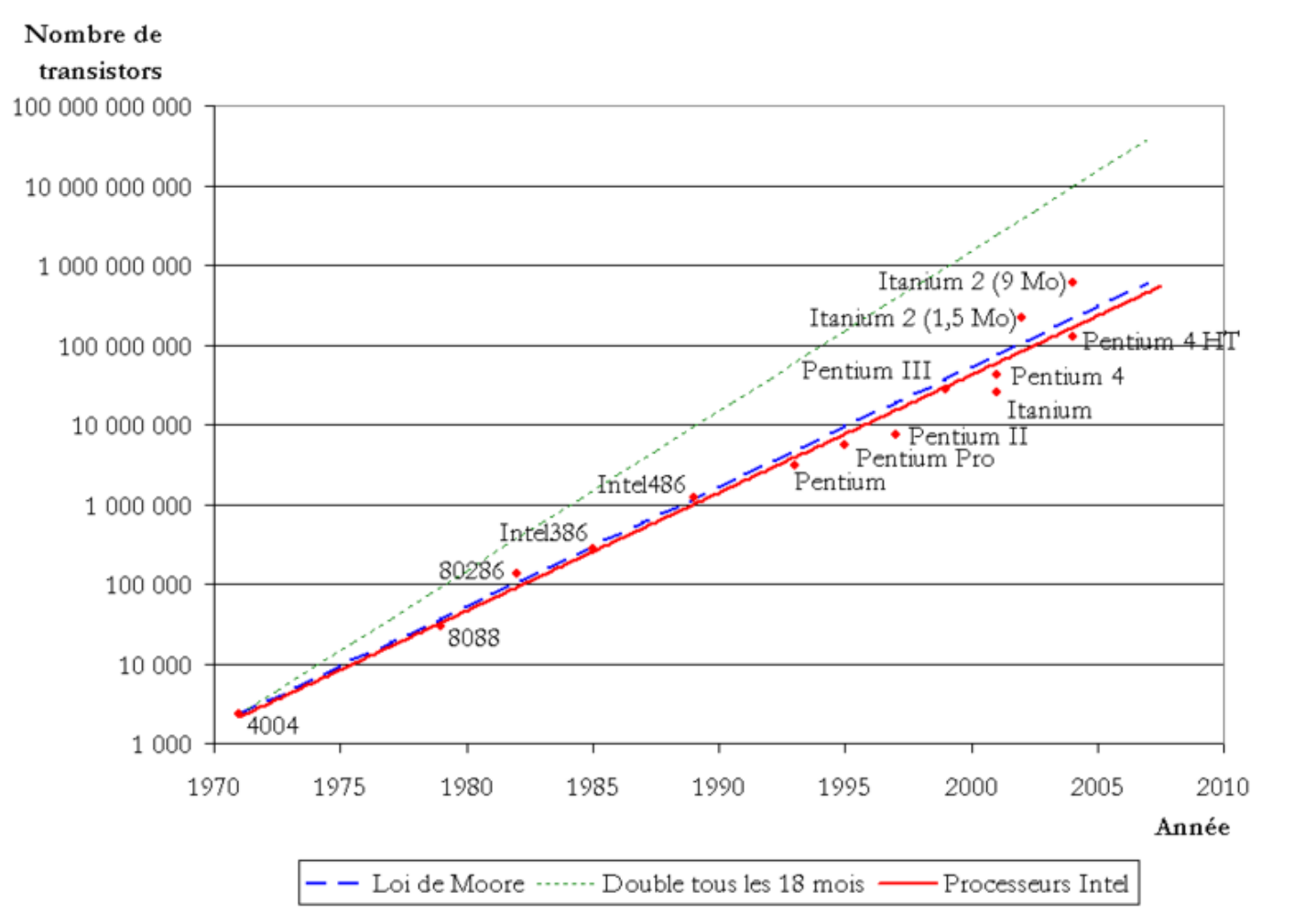
- 1996 – Photon mapping – E.W. Jensen

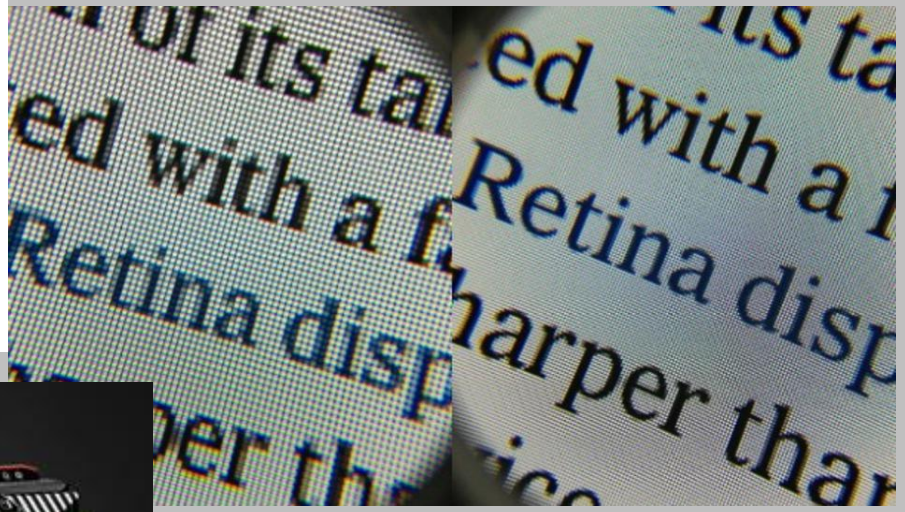


- 2001 – Subsurface Scattering – E.W. Jensen

Materiel

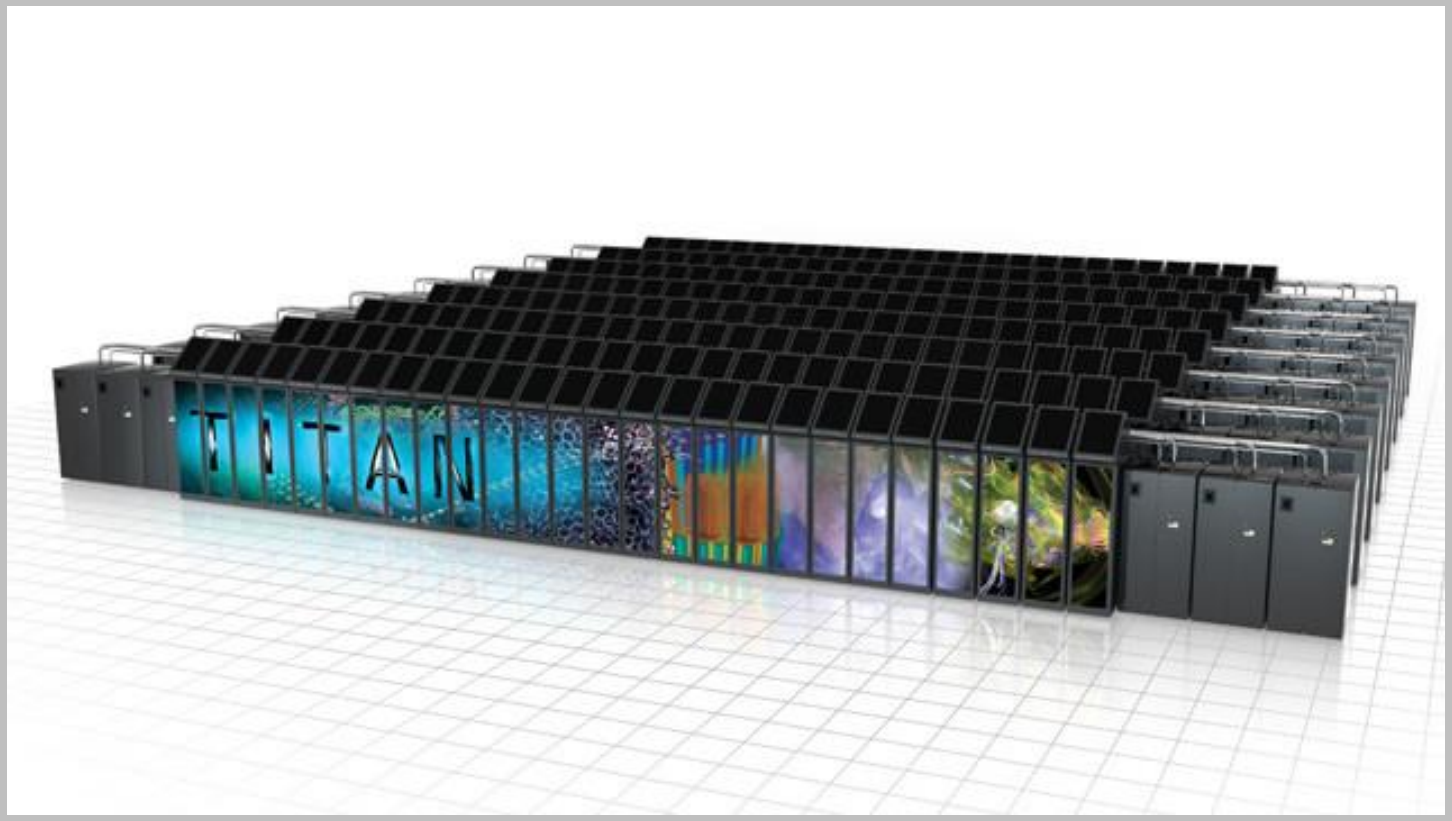
- Loie de Moore




of its ta
ed with a
Retina disp
per th
ed with a
Retina disp
harper thar


Oculus
Development Kit

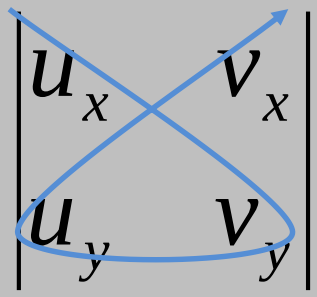
- Super calculateurs



Mathématiques

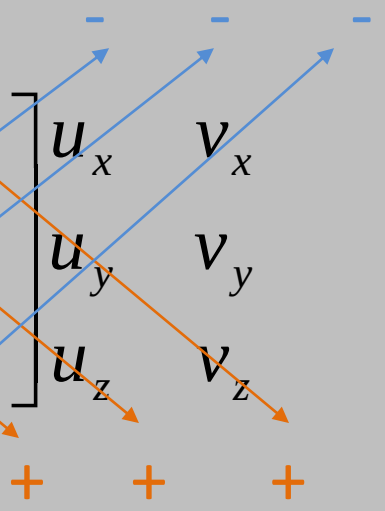
- Déterminants 2x2

$$\det(\vec{u},\vec{v}) = \begin{vmatrix} u_x & v_x \\ u_y & v_y \end{vmatrix} = u_x v_y - u_y v_x$$



- Déterminants 3x3

$$\det(\vec{u},\vec{v},\vec{w}) = \begin{vmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{vmatrix} = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \begin{matrix} u_x & v_x \\ u_y & v_y \\ u_z & v_z \end{matrix}$$



$$= u_x v_y w_z + v_x w_y u_z + w_x u_y v_z - w_x v_y u_z - u_x w_y v_z - v_x u_y w_z$$

- Produit Scalaire

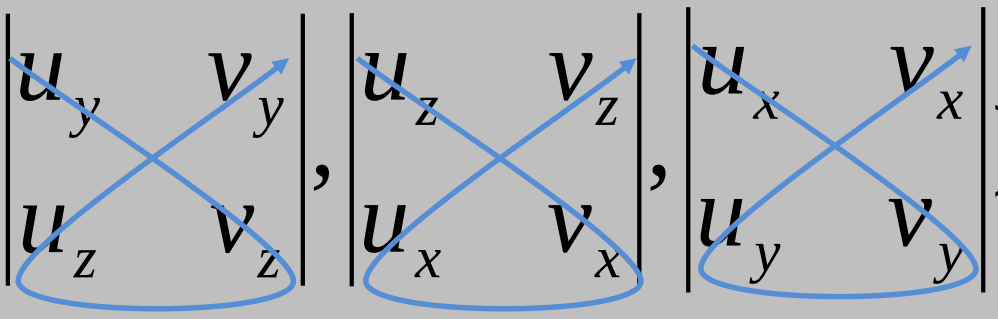
$$\vec{u}.\vec{v} = |\vec{u}||\vec{v}|\cos(\theta)$$

$$\vec{u}.\vec{v} = u_x v_x + u_y v_y + u_z v_z$$

- Produit Vectoriel

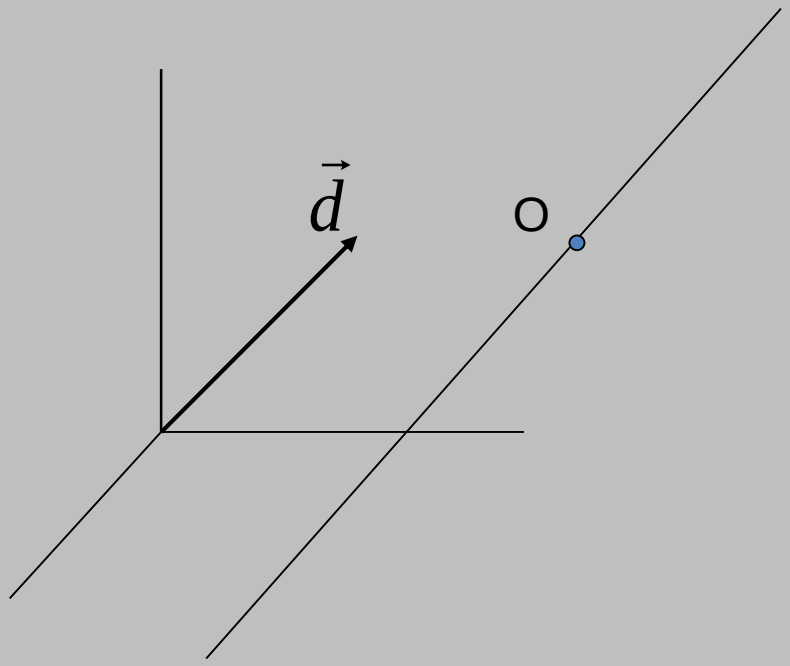
$$|\vec{u} \wedge \vec{v}| = |\vec{u}|\,|\vec{v}|\sin(\theta)$$

$$\vec{u} \wedge \vec{v} = \left(\begin{vmatrix} u_y & v_y \\ u_z & v_z \end{vmatrix}, \begin{vmatrix} u_z & v_z \\ u_x & v_x \end{vmatrix}, \begin{vmatrix} u_x & v_x \\ u_y & v_y \end{vmatrix} \right)$$



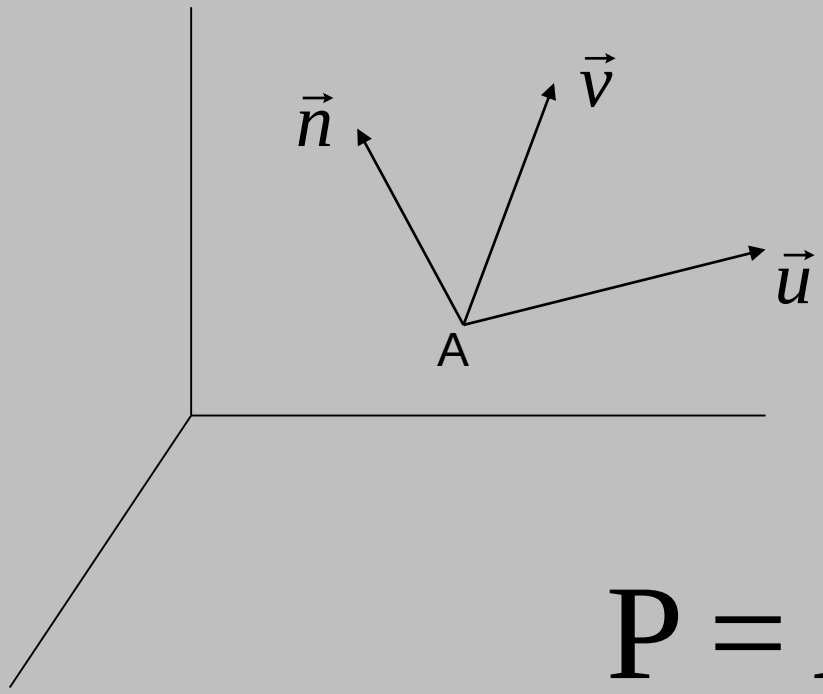
$$\vec{u} \wedge \vec{v} = (u_y v_z - u_z v_y, u_z v_x - u_x v_z, u_x v_y - u_y v_x)$$

- Droite et Plan



$$P = O + t\vec{d}$$

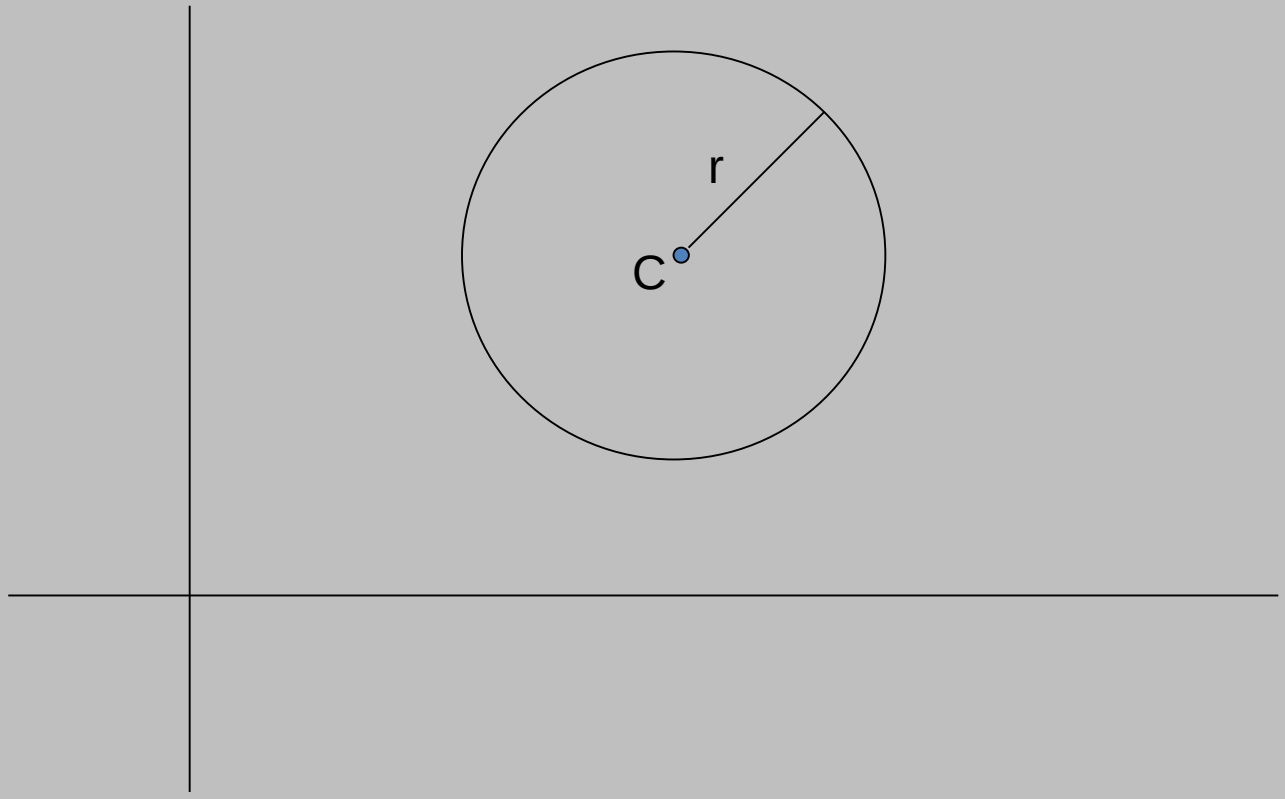
- Droite et Plan



$$P = A + \alpha\vec{u} + \beta\vec{v}$$

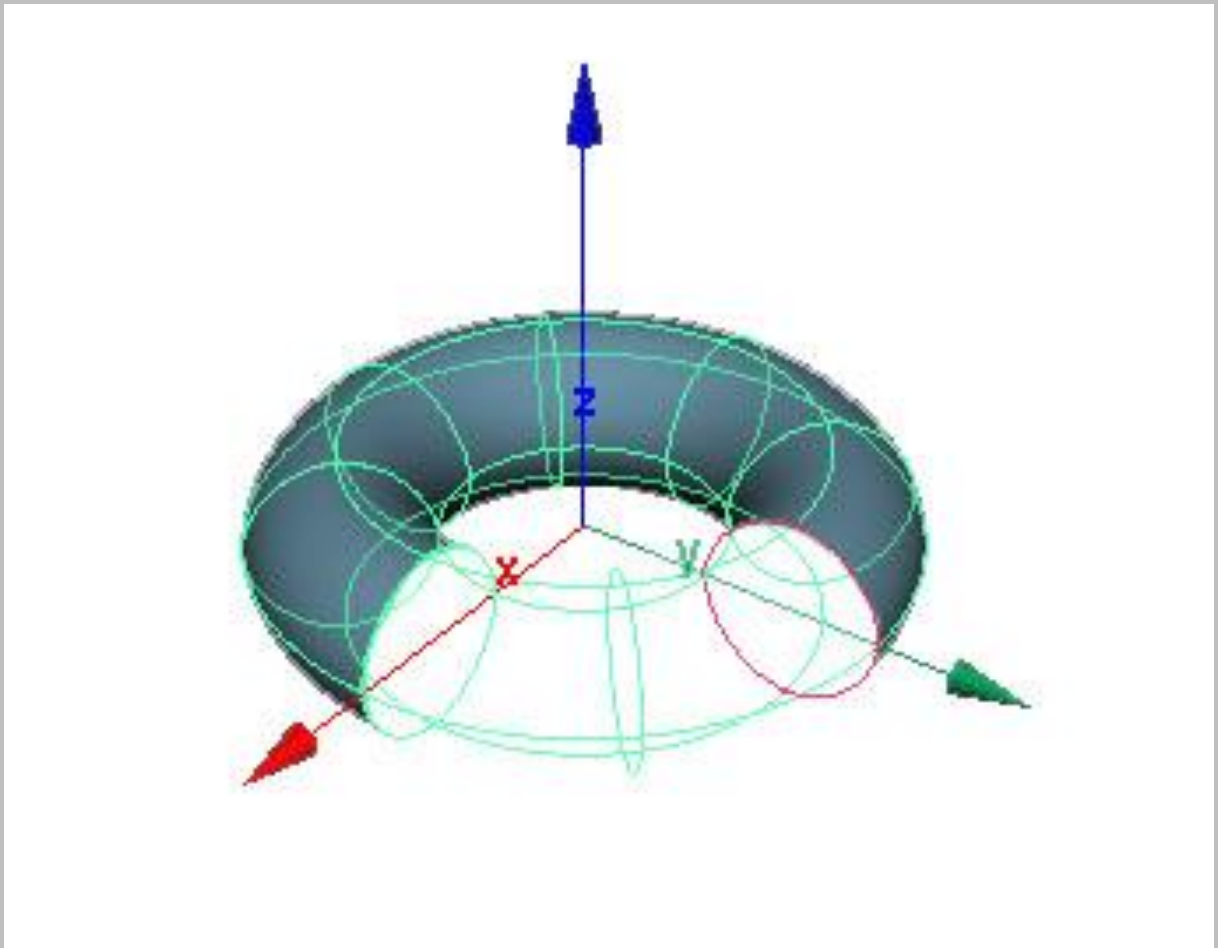
$$(P - A).\vec{n} = 0$$

- Sphère

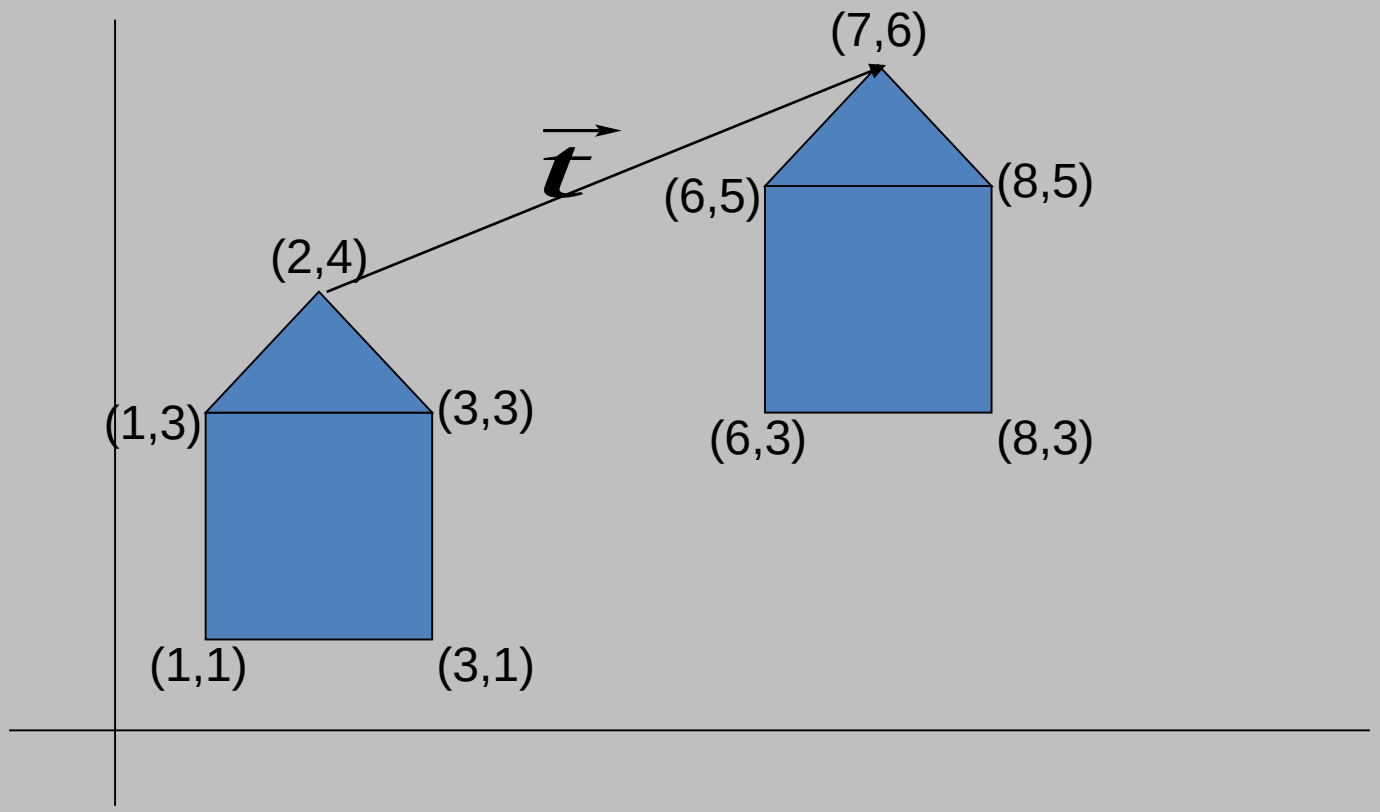


$$(P - C)^2 - r^2 = 0$$

- Tore



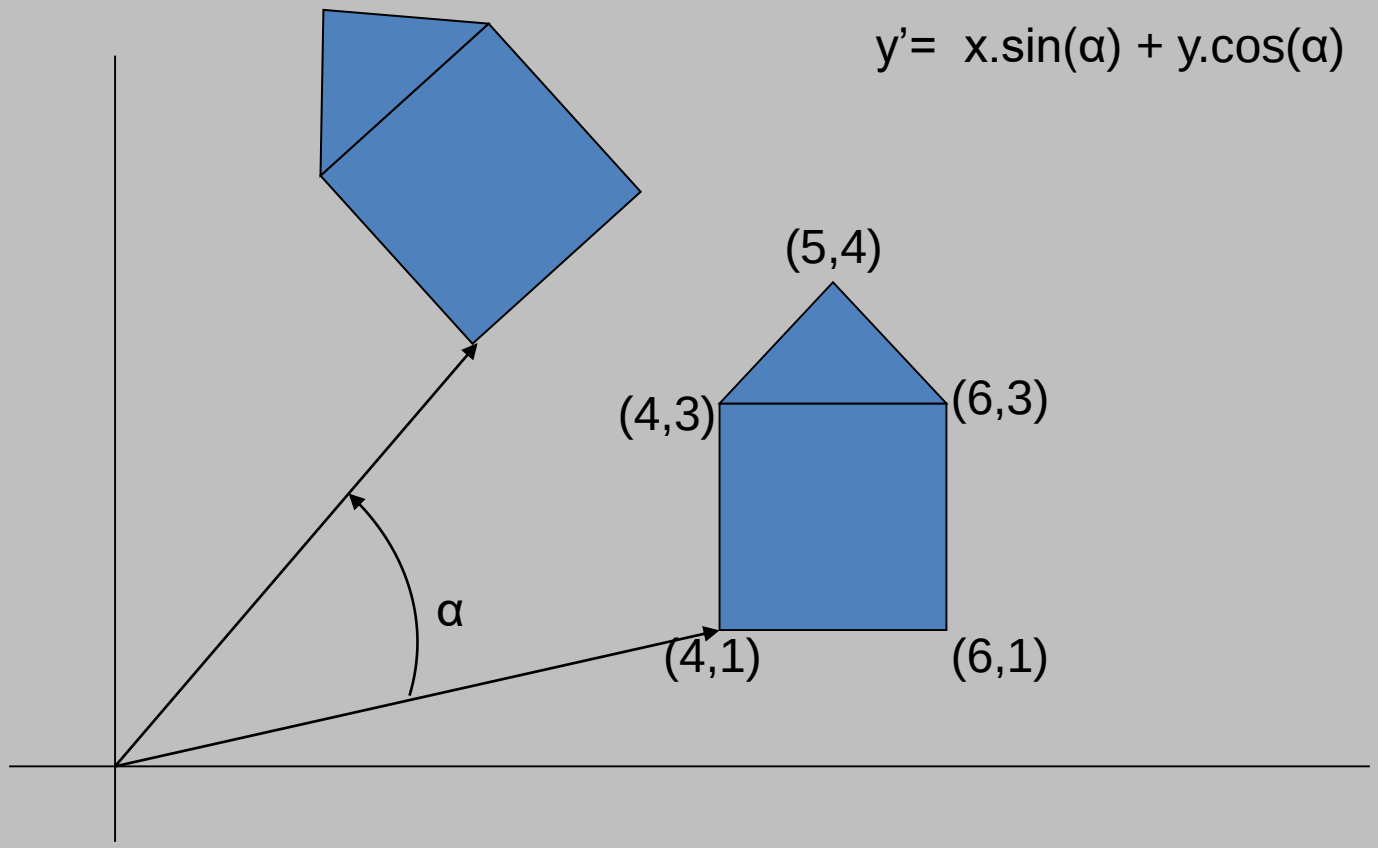
- Translation



- Rotation

$x' = x.\cos(\alpha) - y.\sin(\alpha)$

$y' = x.\sin(\alpha) + y.\cos(\alpha)$



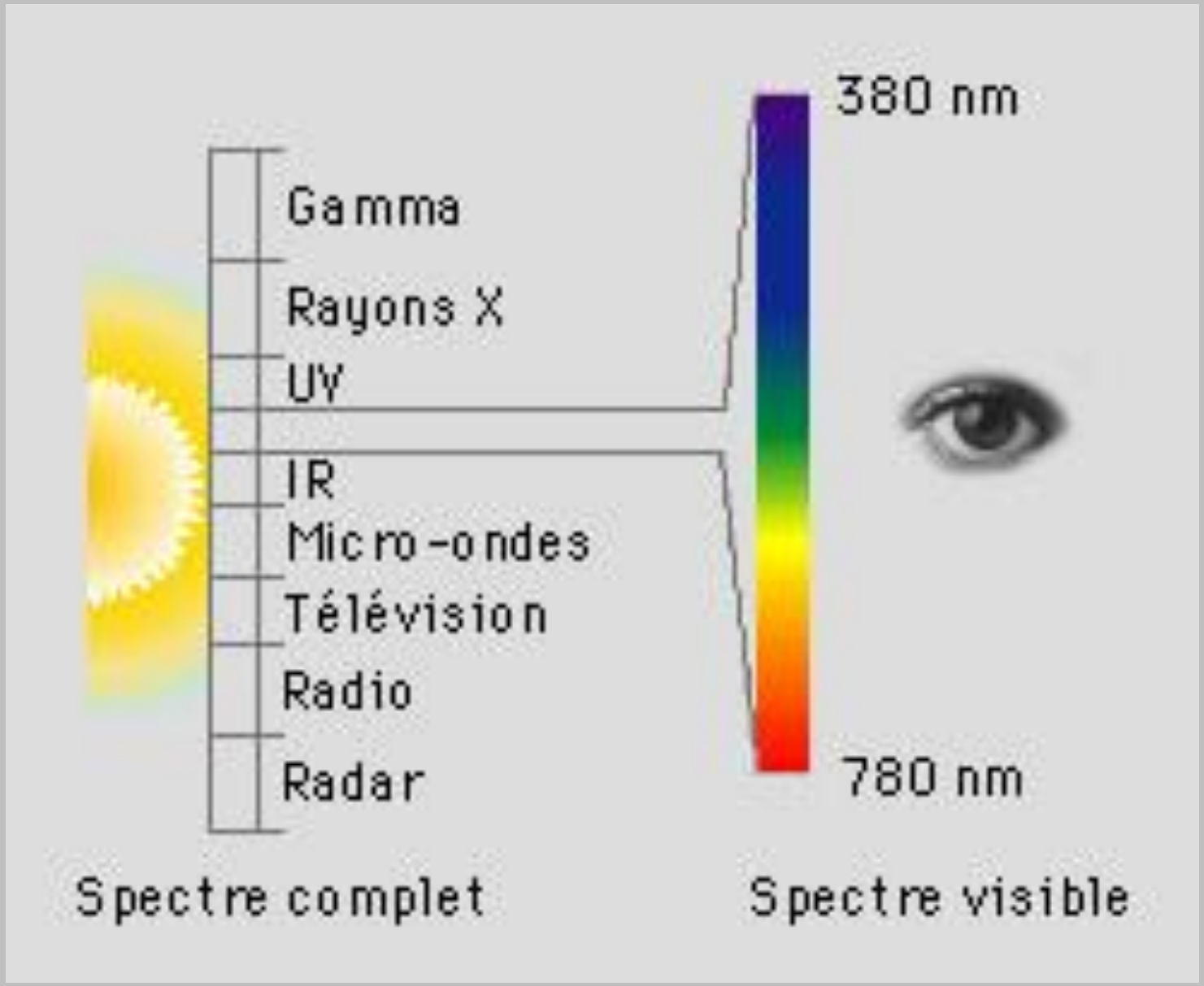
- N équations à N inconnues

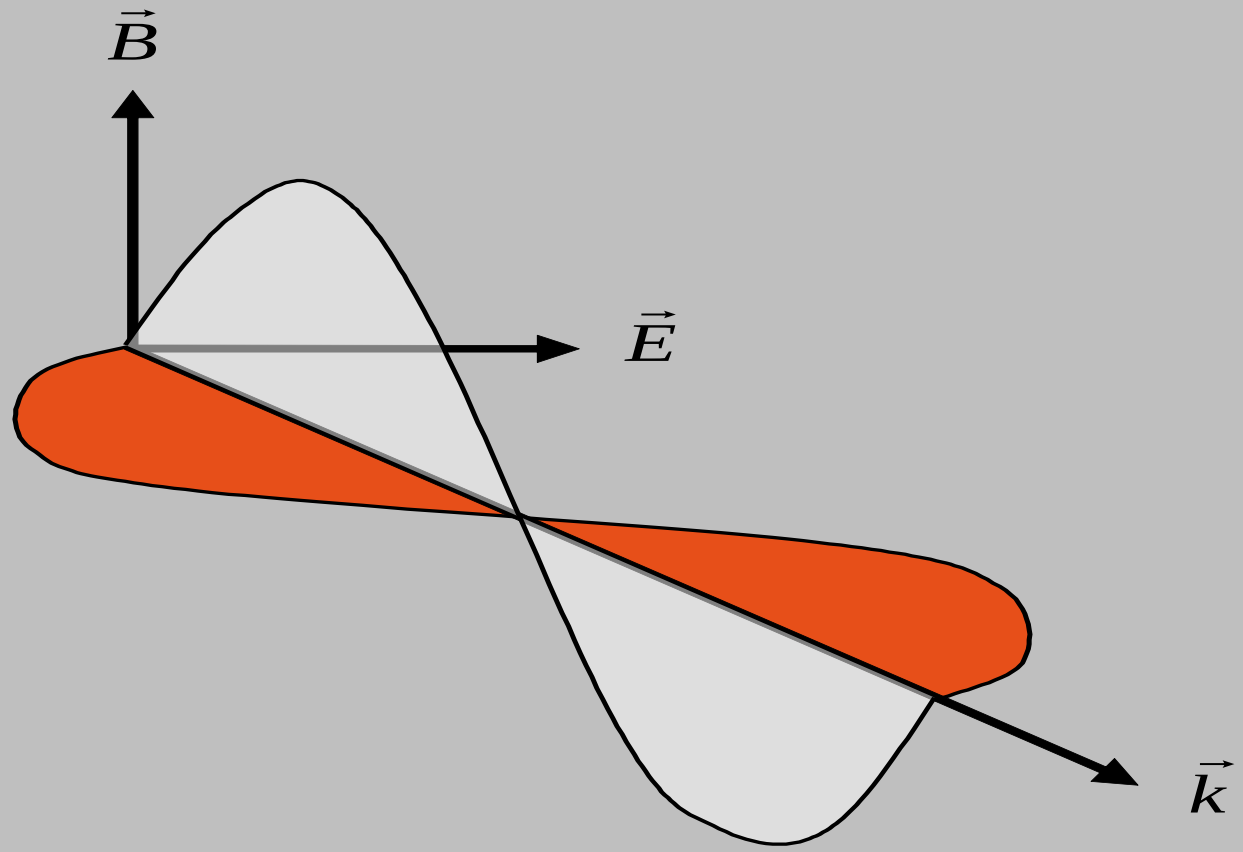
$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = \lambda_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = \lambda_2 \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = \lambda_n \end{cases}$$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix} \Leftrightarrow A \cdot X = \Lambda$$

$$x_k = \frac{\det(A_k)}{\det(A)}$$

Lumière


380 nm
Gamma
Rayons X
UV
IR
Micro-ondes
Télévision
Radio
Radar
780 nm
Spectre complet
Spectre visible


$\vec{B}$
$\vec{E}$
$\vec{k}$

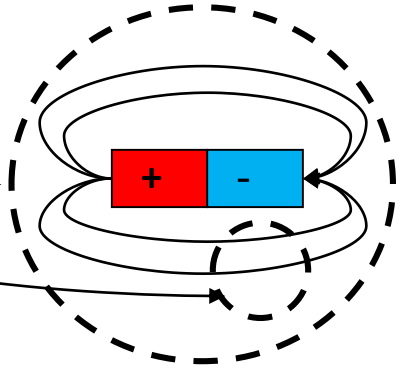
$$div\,\vec{B} = 0$$

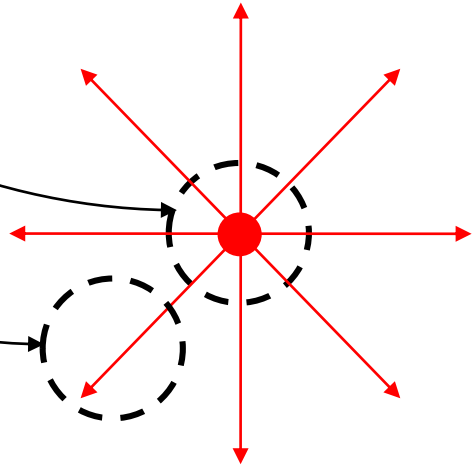
$$div\,\vec{E} = \frac{\rho}{\varepsilon_0}$$

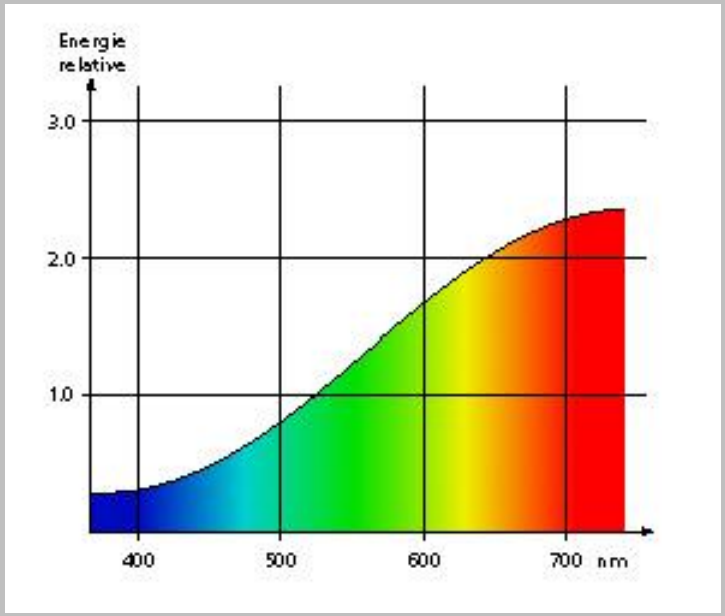
$$= 0$$

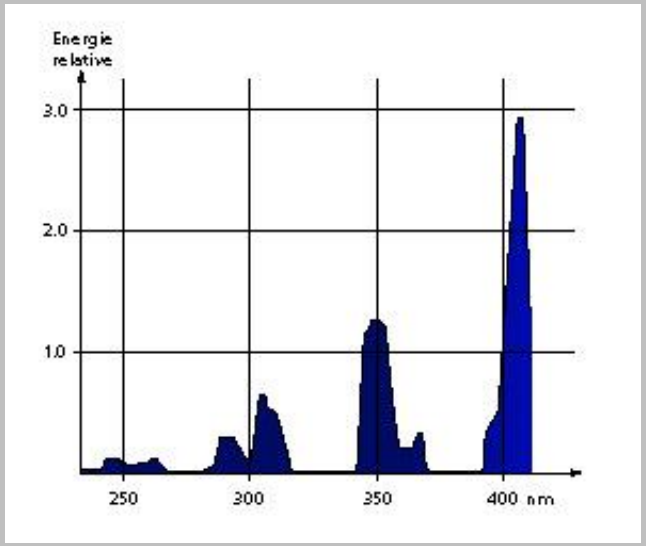
$$rot\,\vec{B} = \mu_0 \varepsilon_0 \frac{\partial E}{\partial t}$$

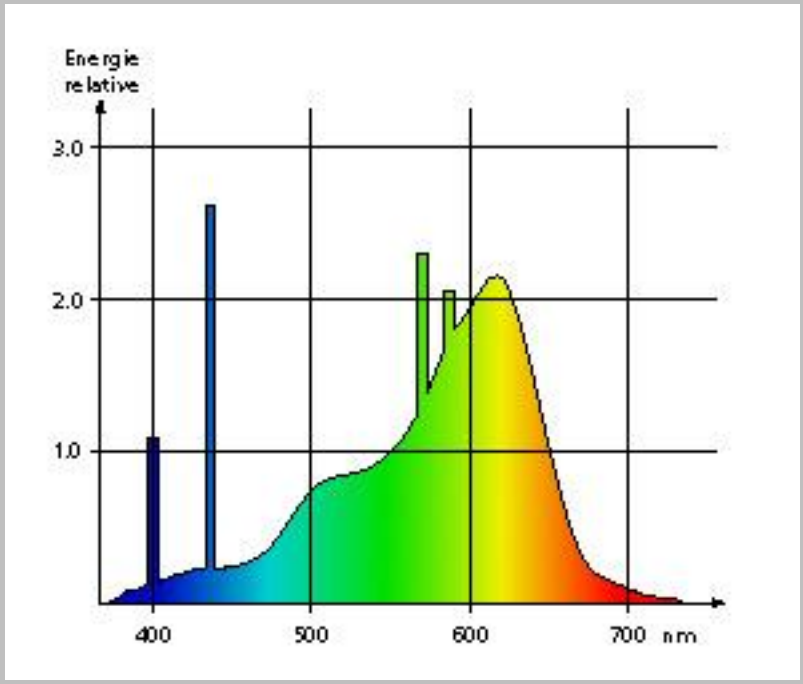
$$rot\,\vec{E} = \frac{\partial B}{\partial t}$$

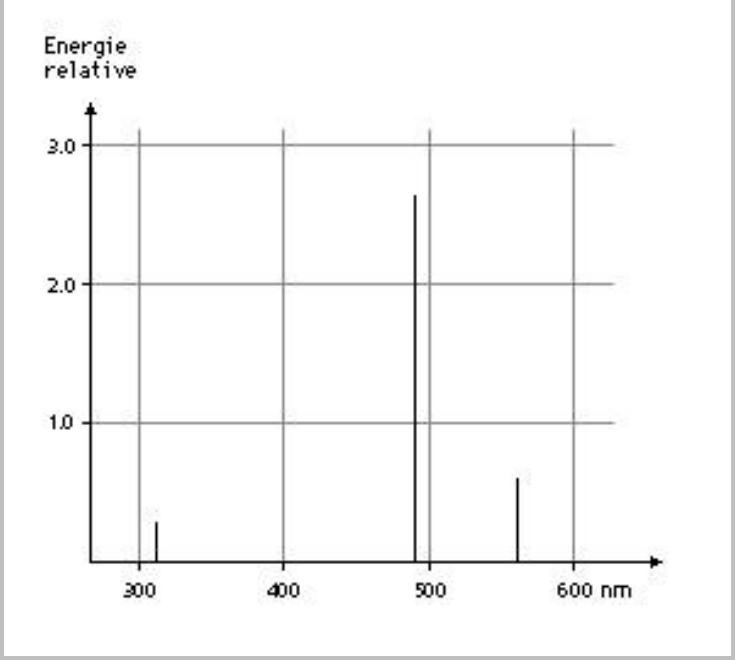


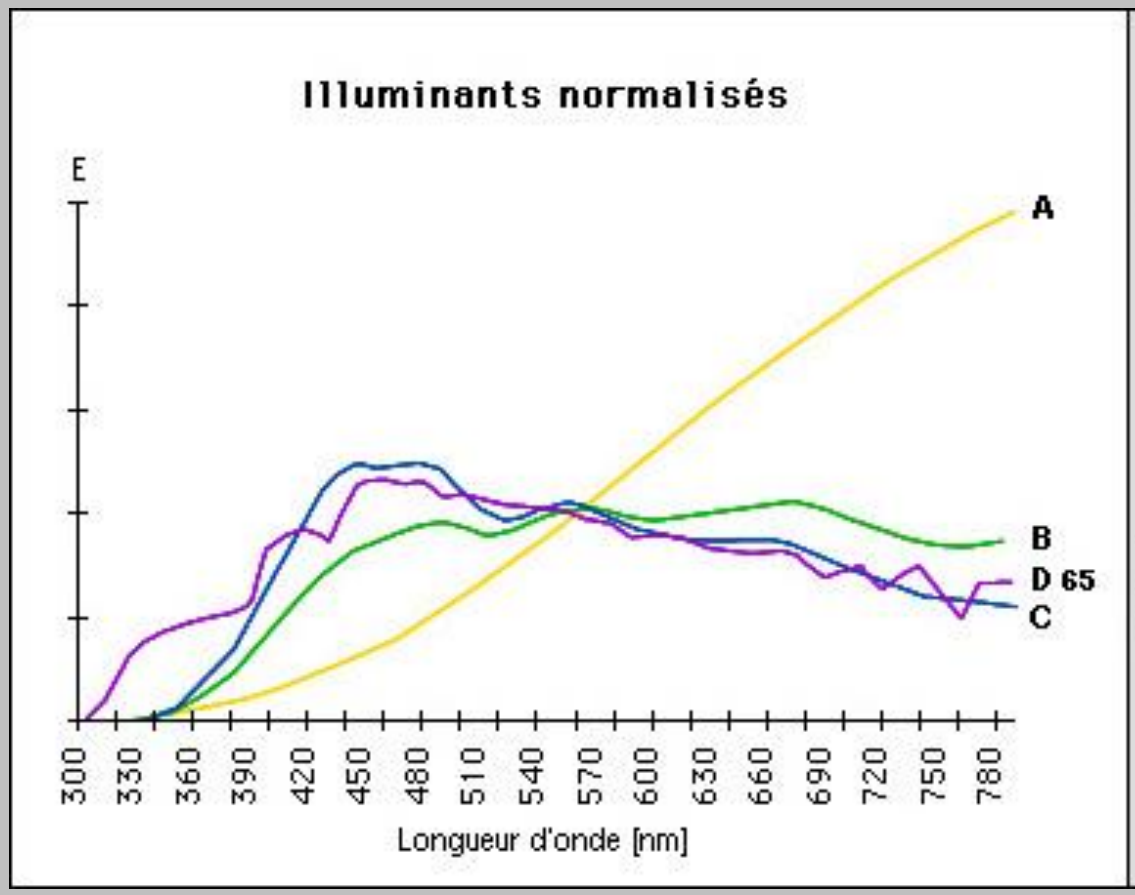



Energie
relative
3.0
2.0
1.0
400
500
600
700 nm


Energie
relative
3.0
2.0
1.0
250
300
350
400 nm


Energie
relative
3.0
2.0
1.0
400
500
600
700 nm


Energie
relative
3.0
2.0
1.0
300
400
500
600 nm



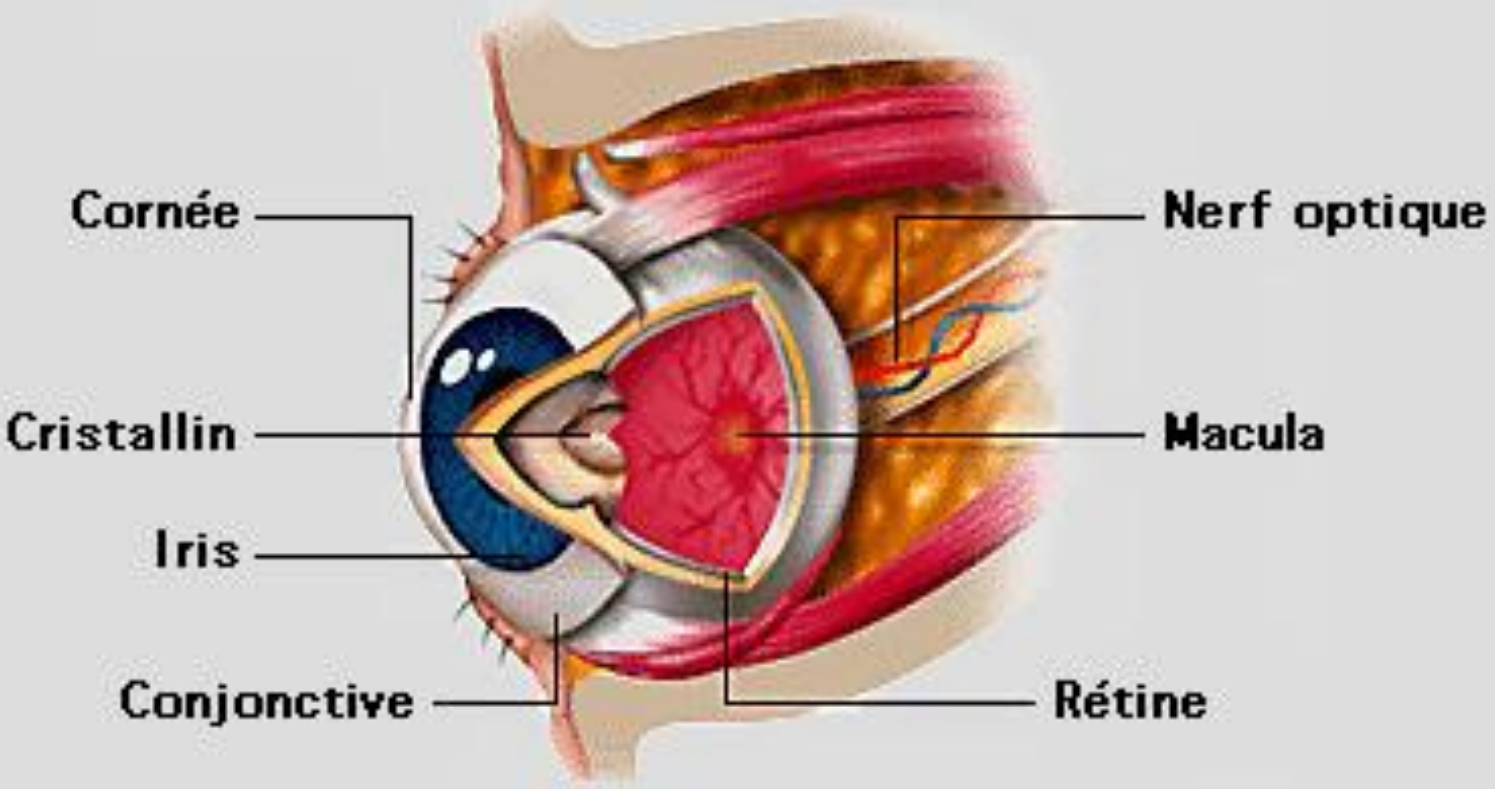
A — Incandescence (2856 K)

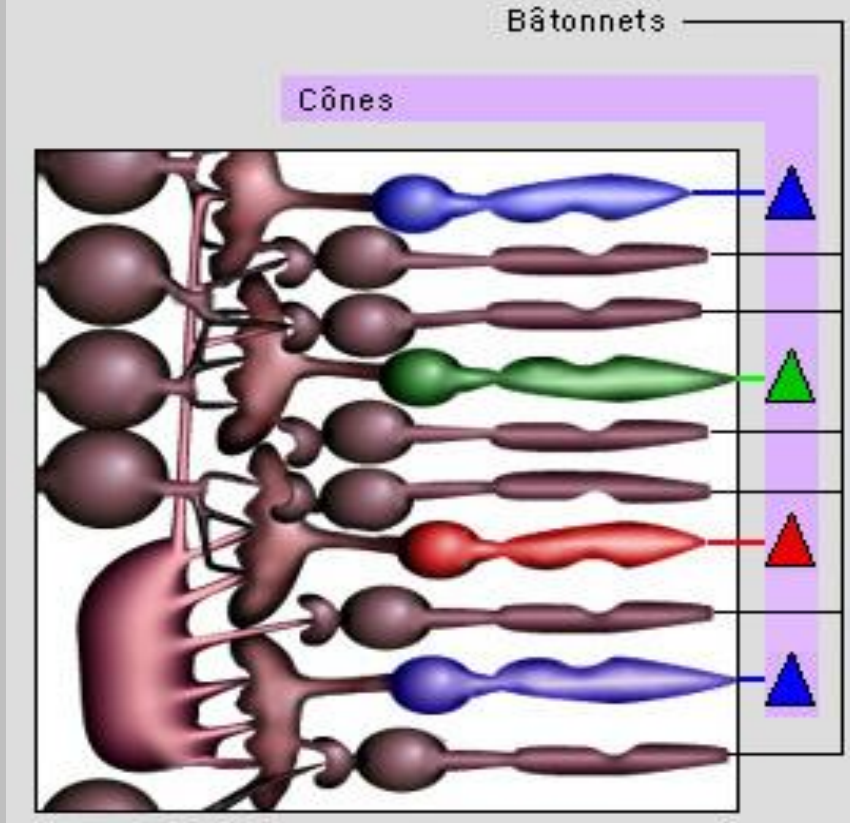
B — Soleil direct (4874 K)

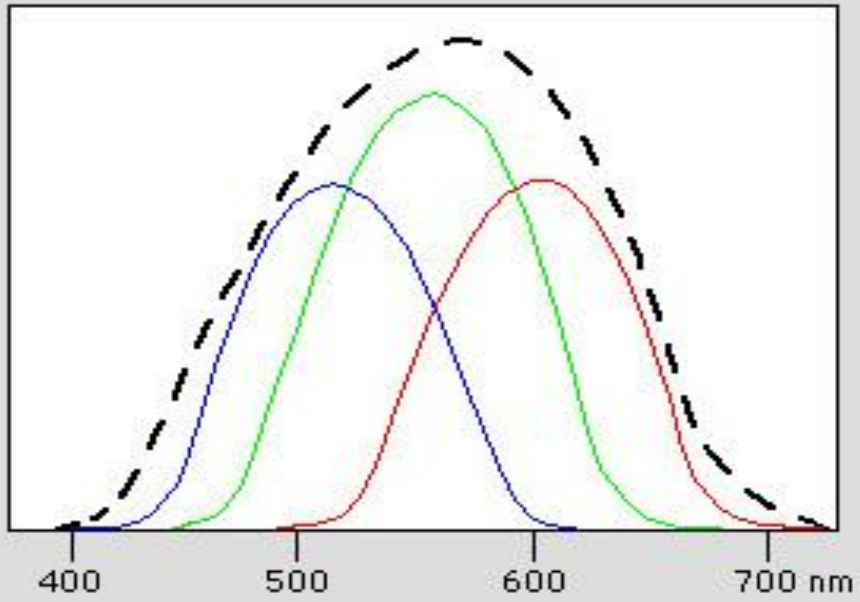
D 65 — Jour moyen, sans UV (6774 K)

C — Jour moyen, avec UV (6504 K)

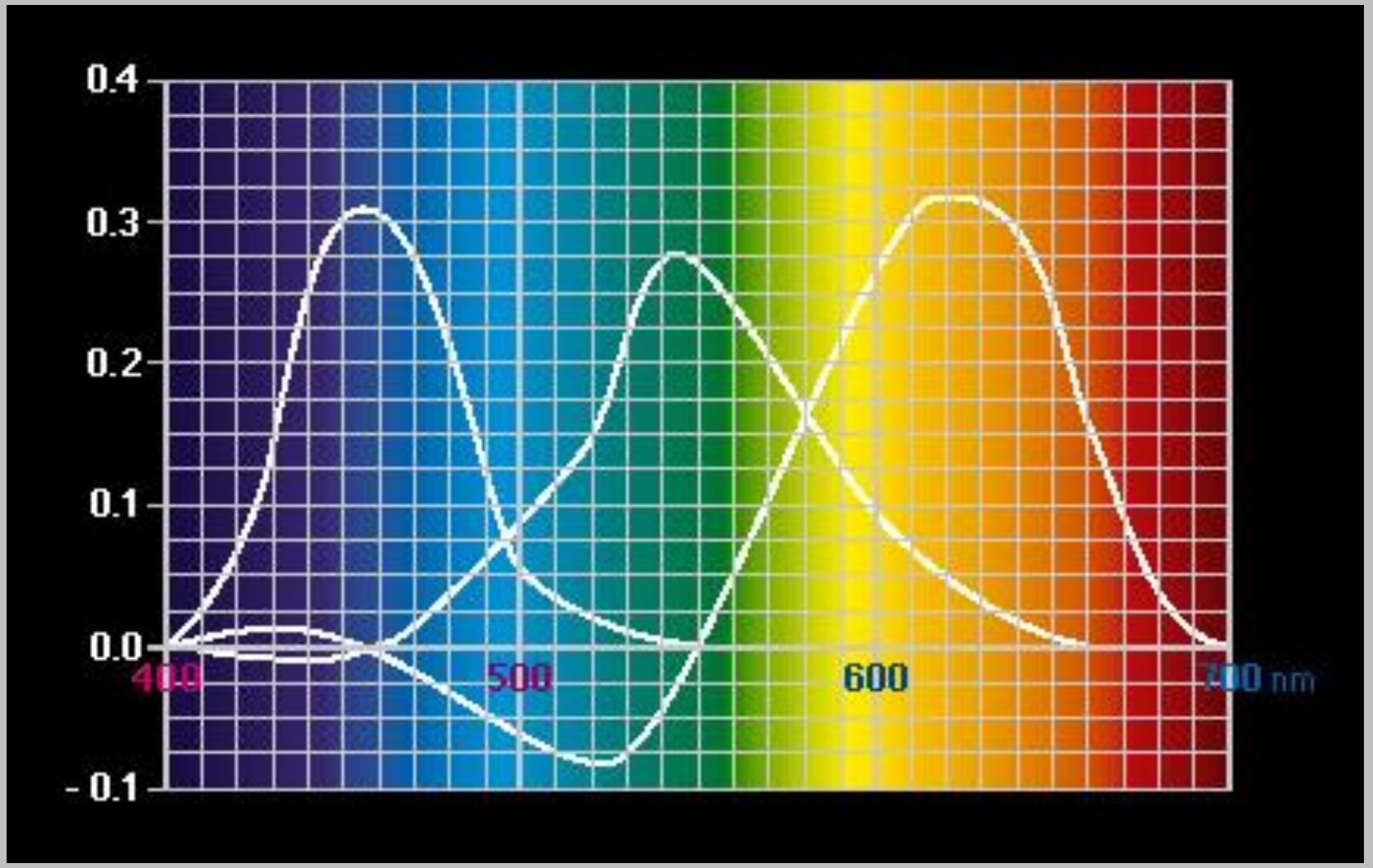
Couleur

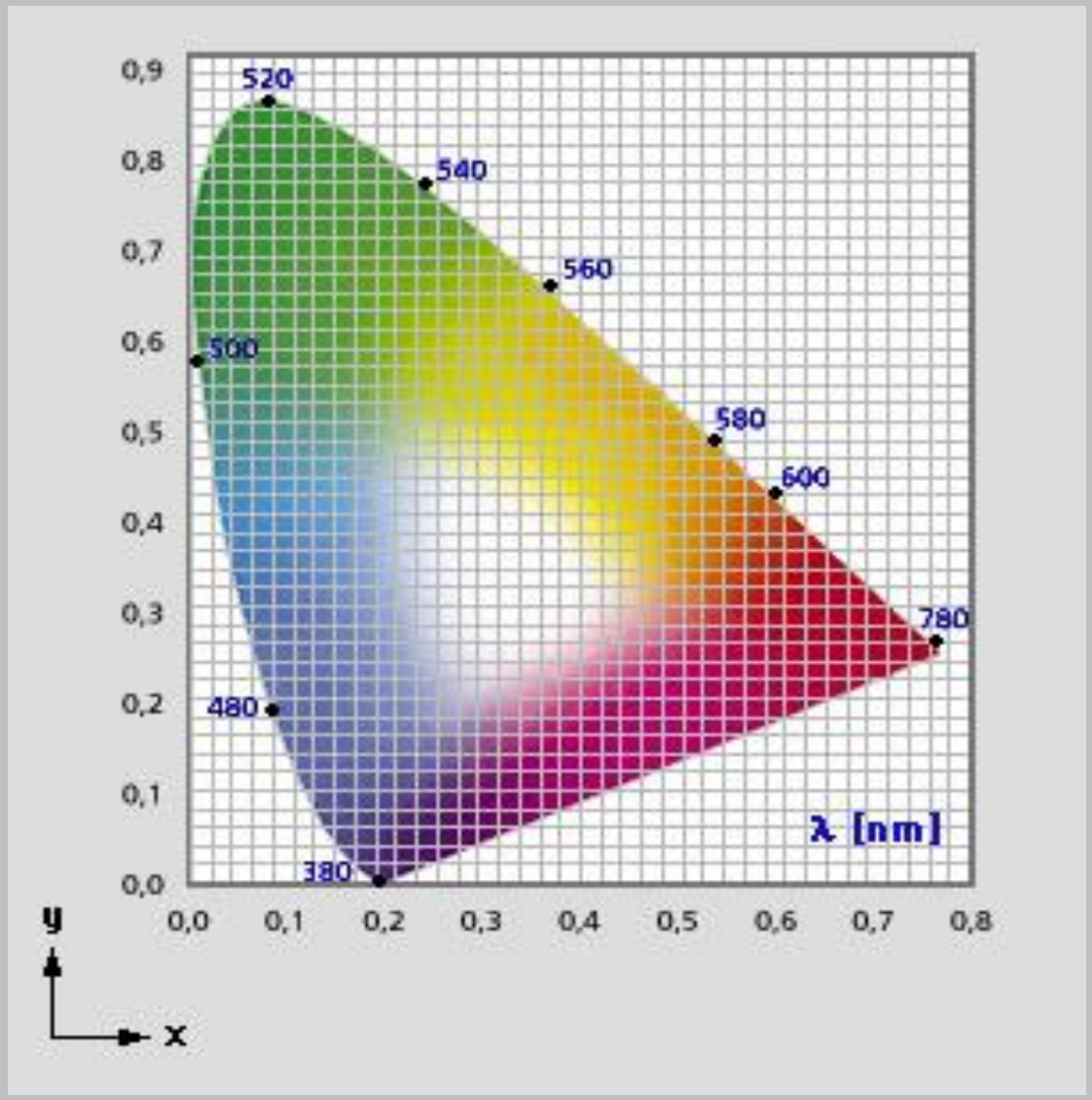

Cornée
Cristallin
Iris
Conjonctive
Nerf optique
Macula
Rétine

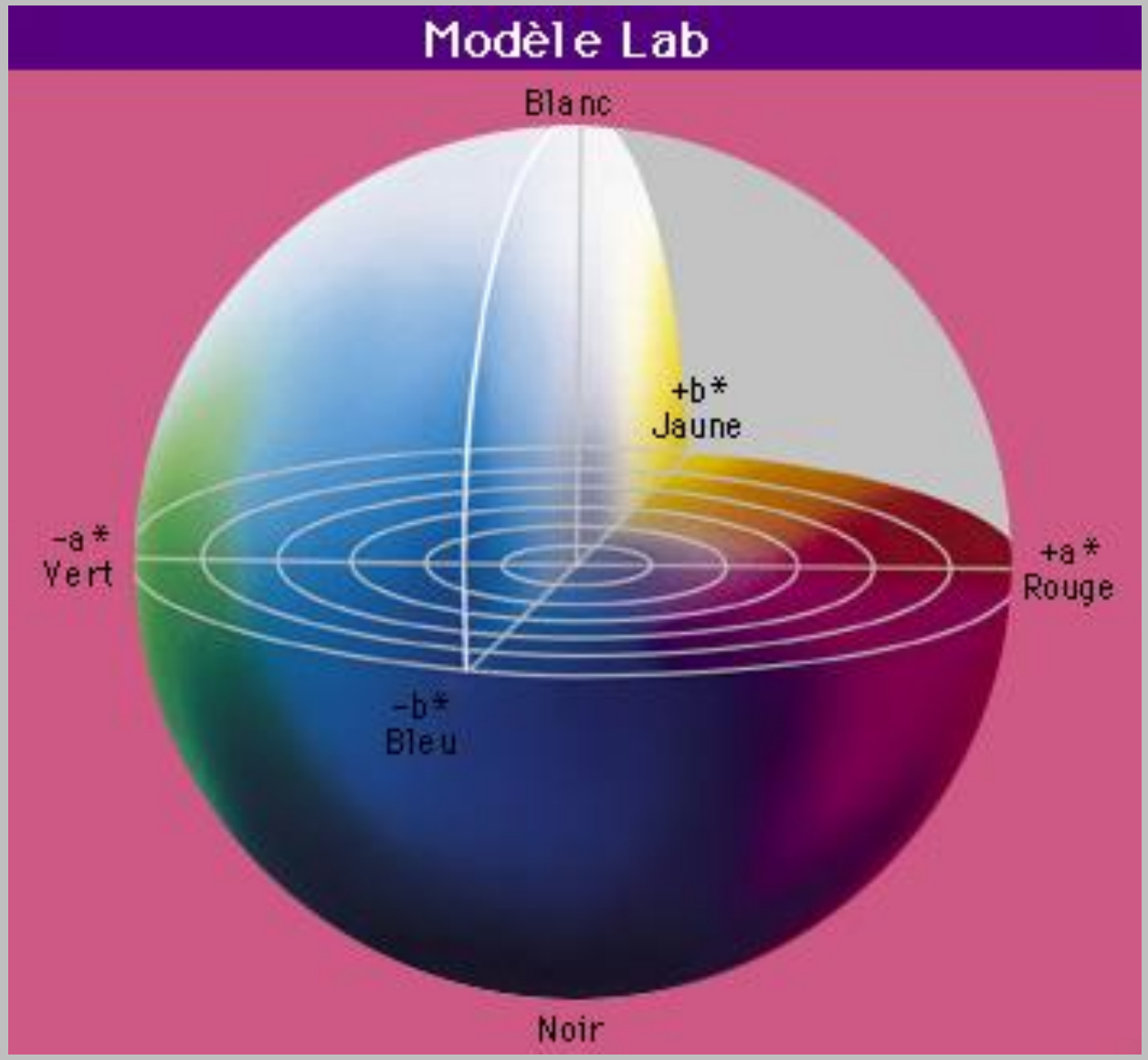


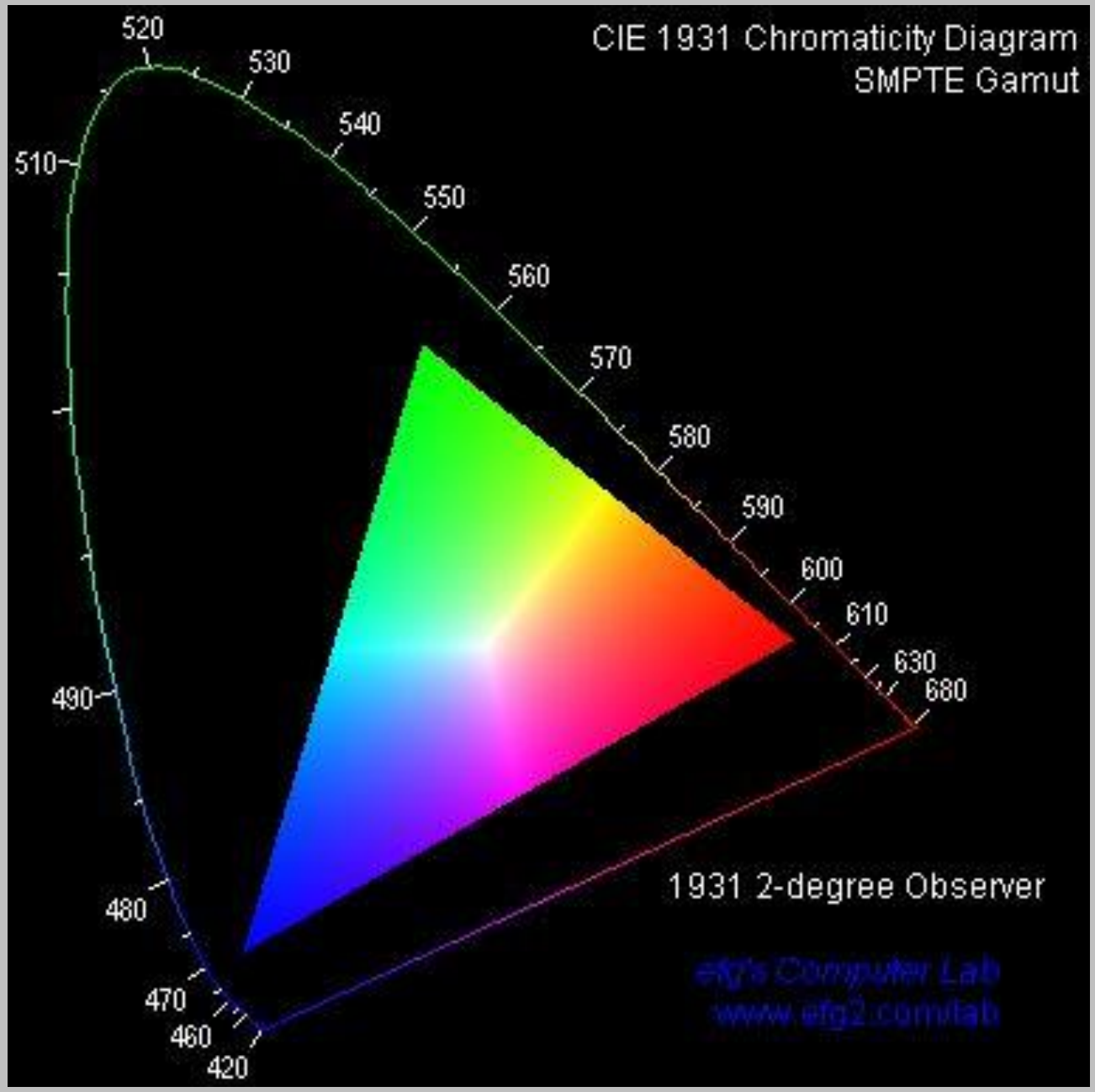


Sensibilité spectrale des 3 sortes de cônes et des bâtonnets


0.4
0.3
0.2
0.1
0.0
- 0.1
400
500
600
700 nm

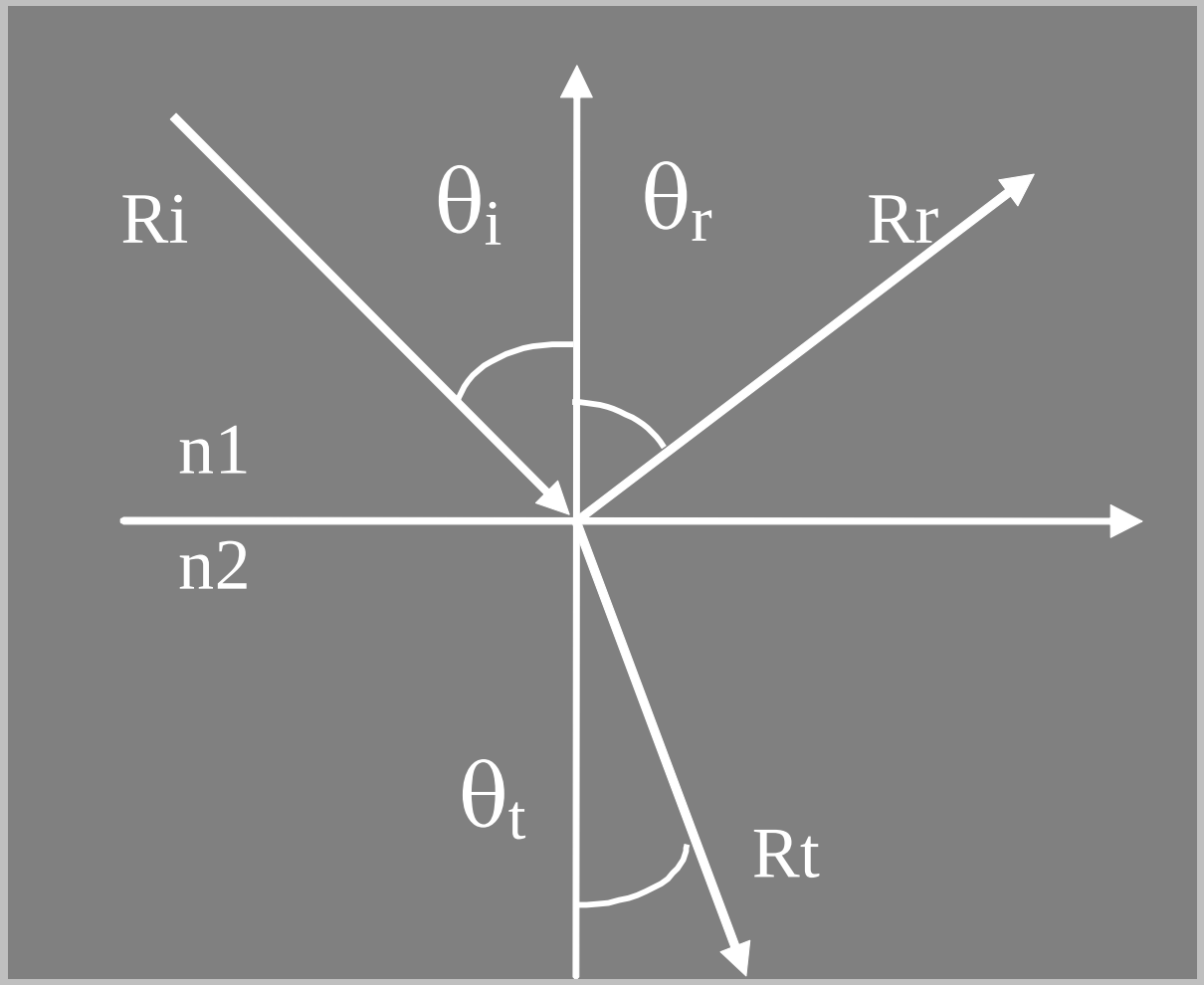

0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
520
540
560
500
580
600
780
480
380
λ [nm]
0,0
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
y
x

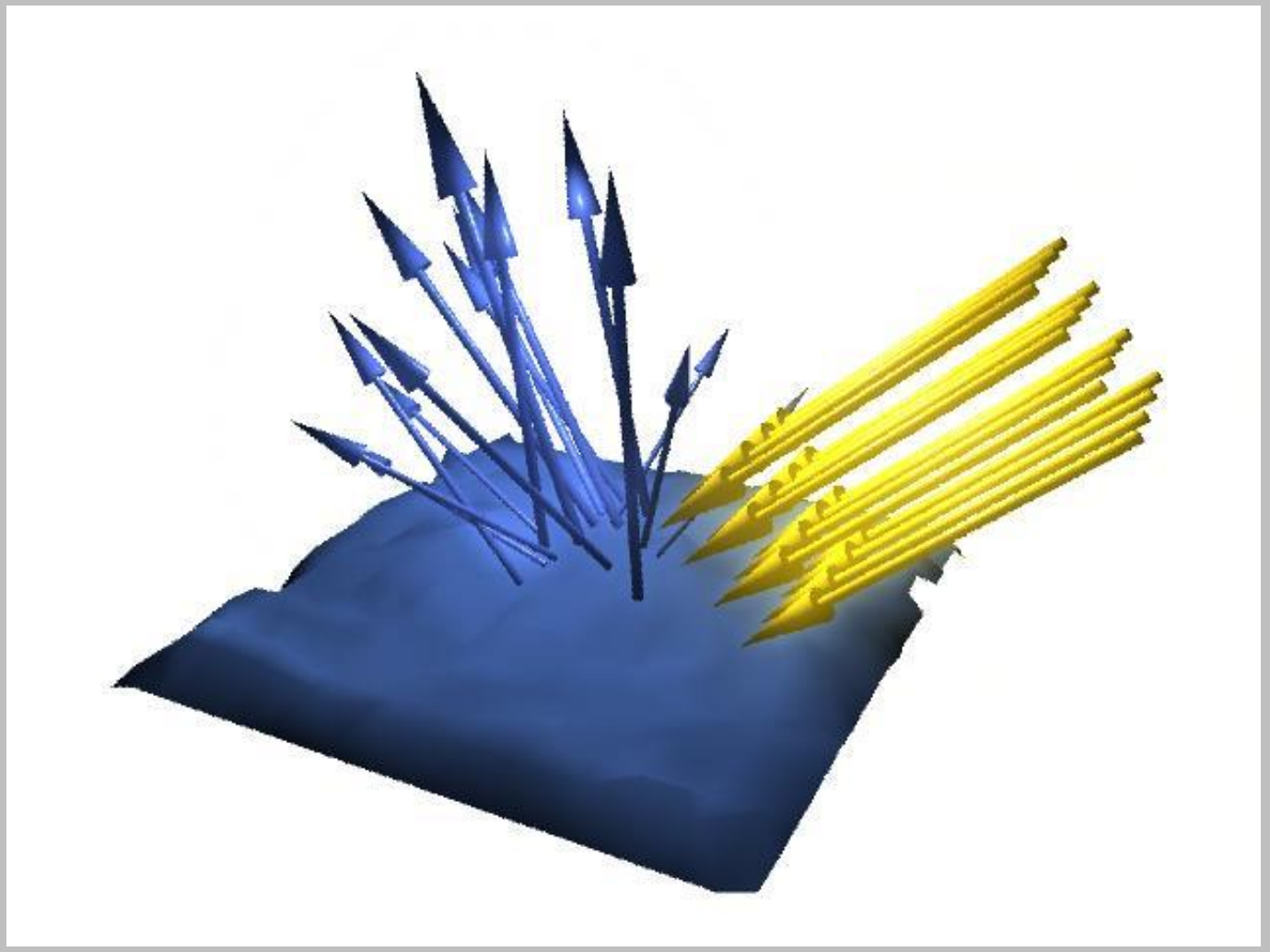

Modèle Lab
Blanc
+b*
Jaune
-a*
Vert
+a*
Rouge
-b*
Bleu
Noir

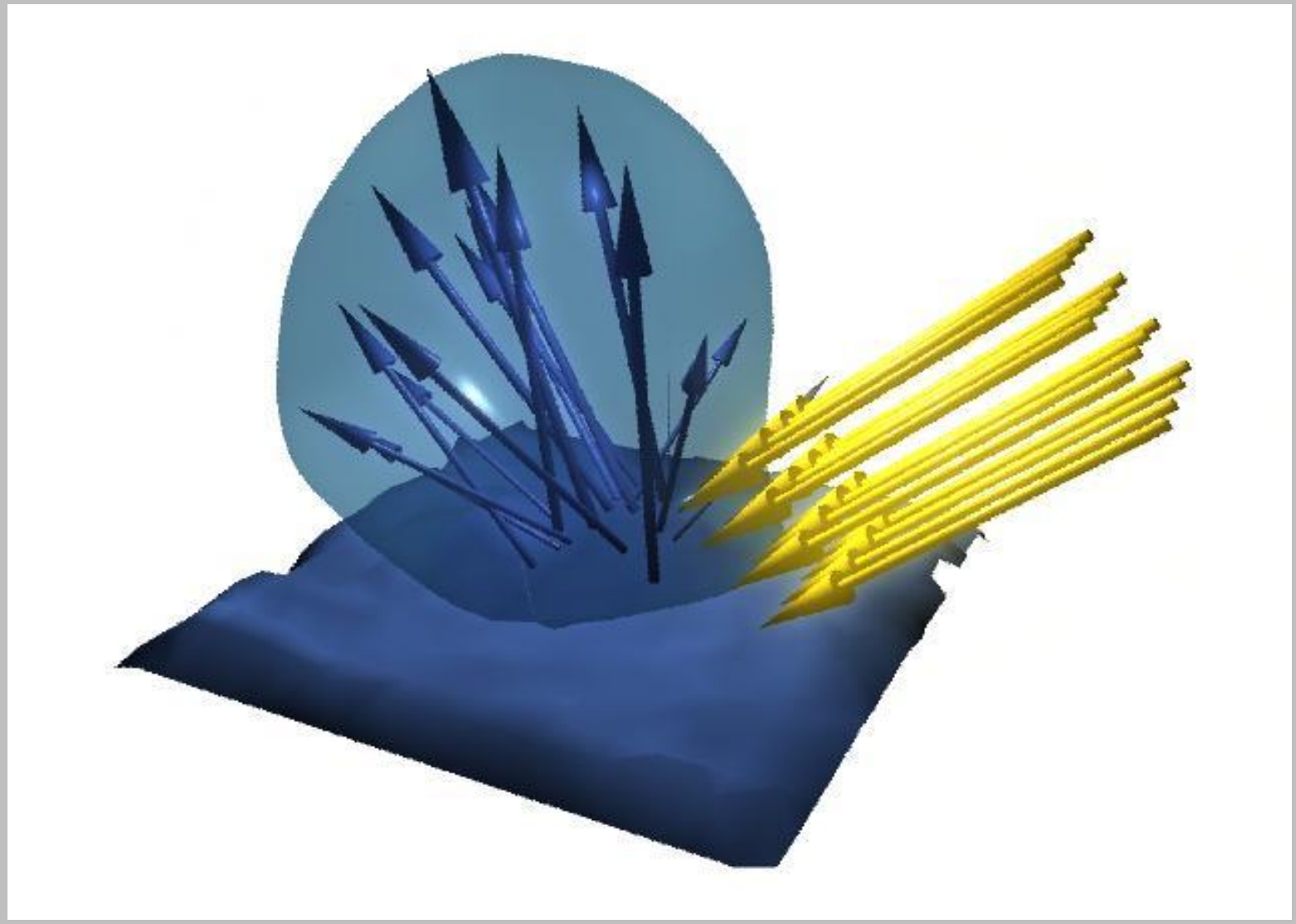

CIE 1931 Chromaticity Diagram
SMPTE Gamut
520
530
540
510
550
560
570
580
590
600
610
630
680
490
480
470
460
420
1931 2-degree Observer
efg's Computer Lab
www.efg2.com/Lab

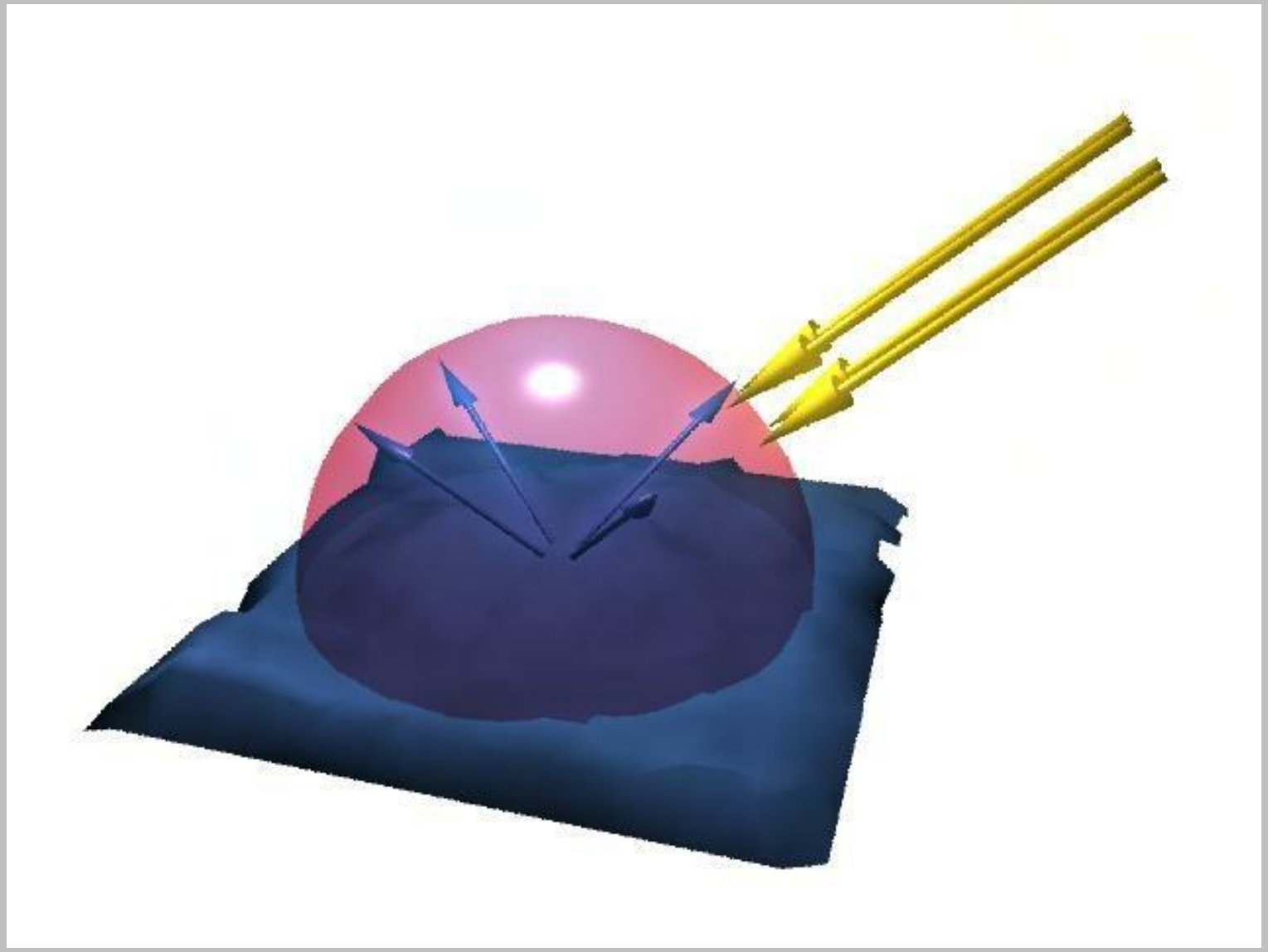
Matière

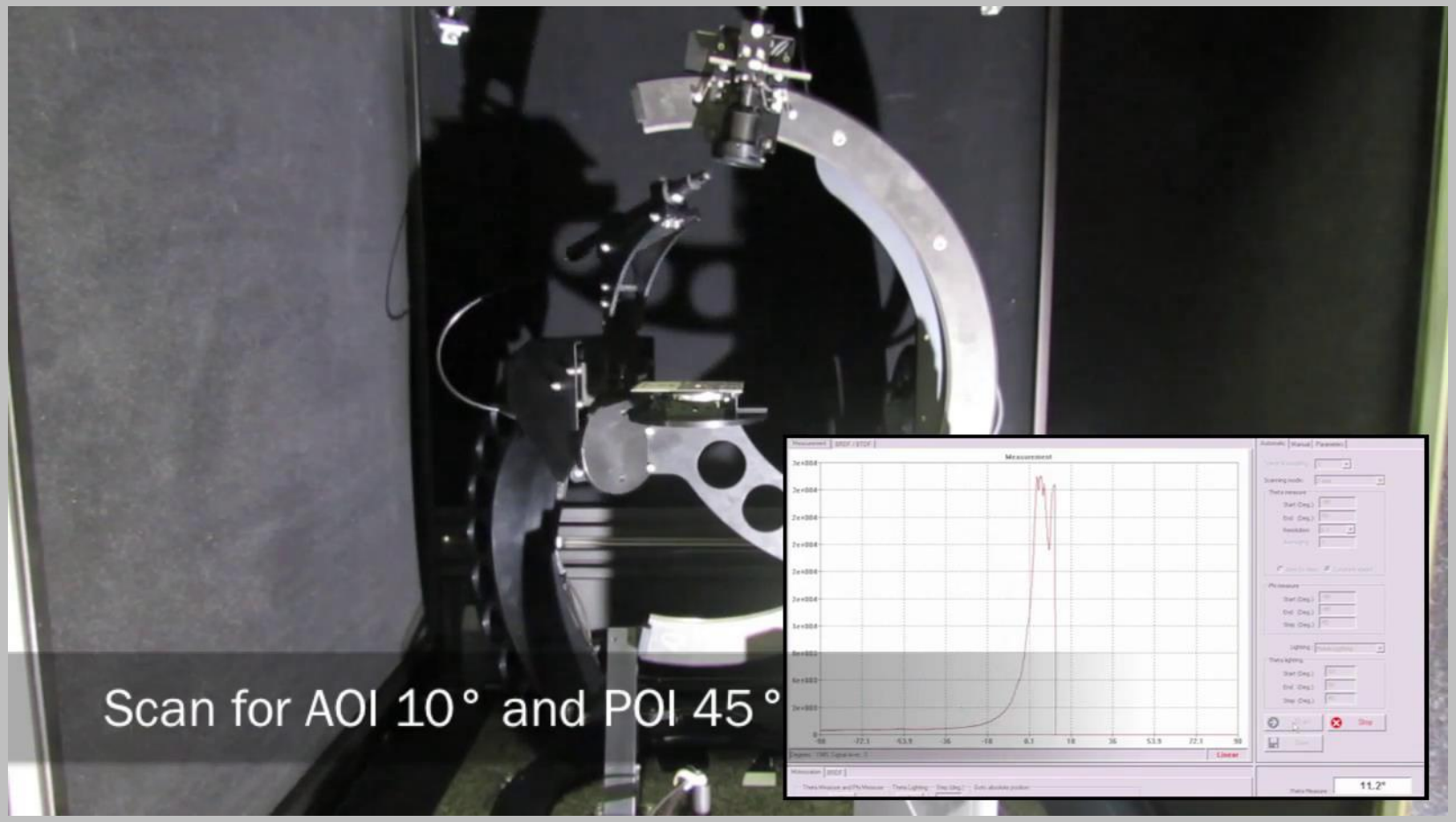
- Lois de Descartes pour la réflexion / transmission de la lumière

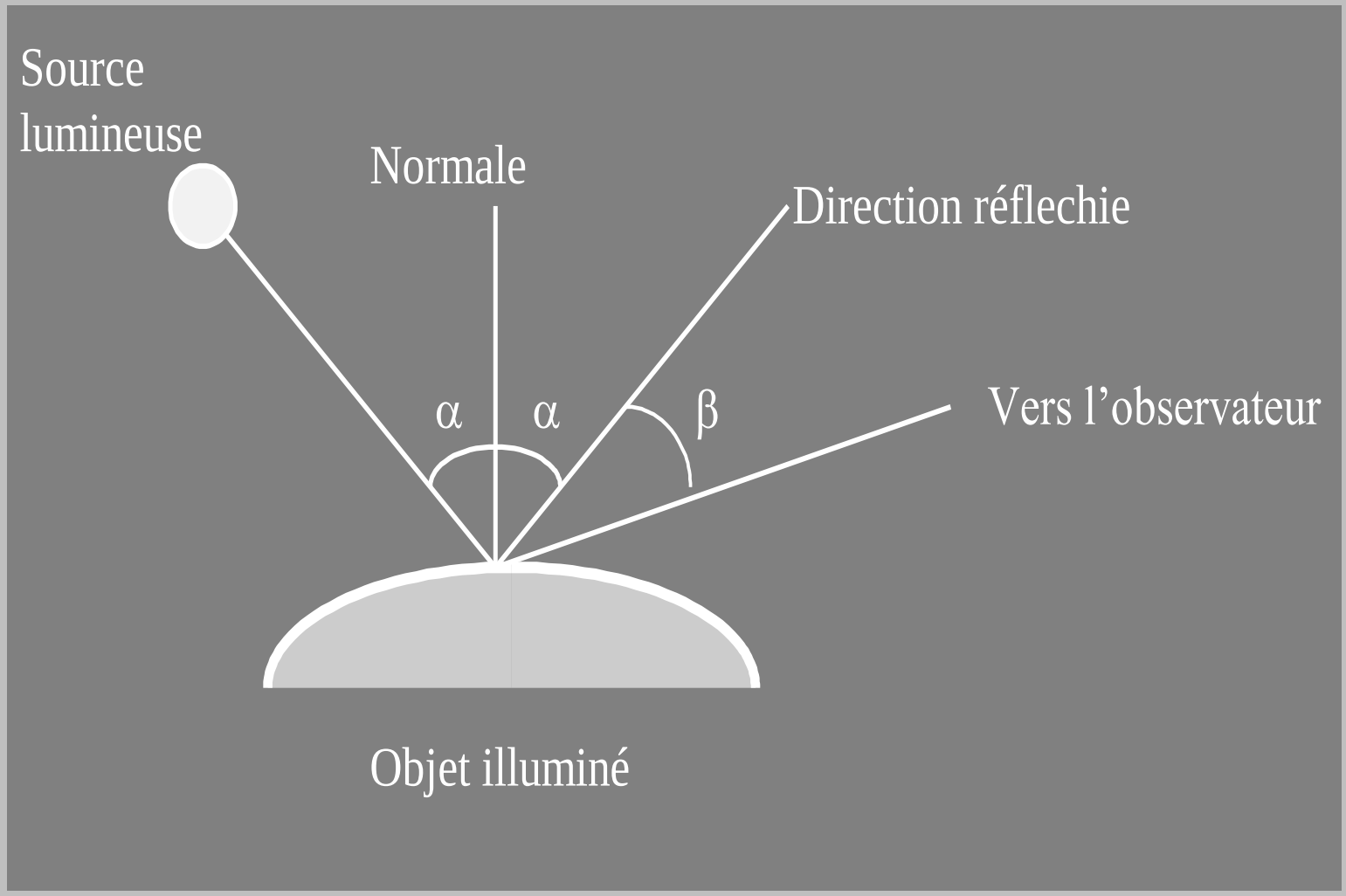




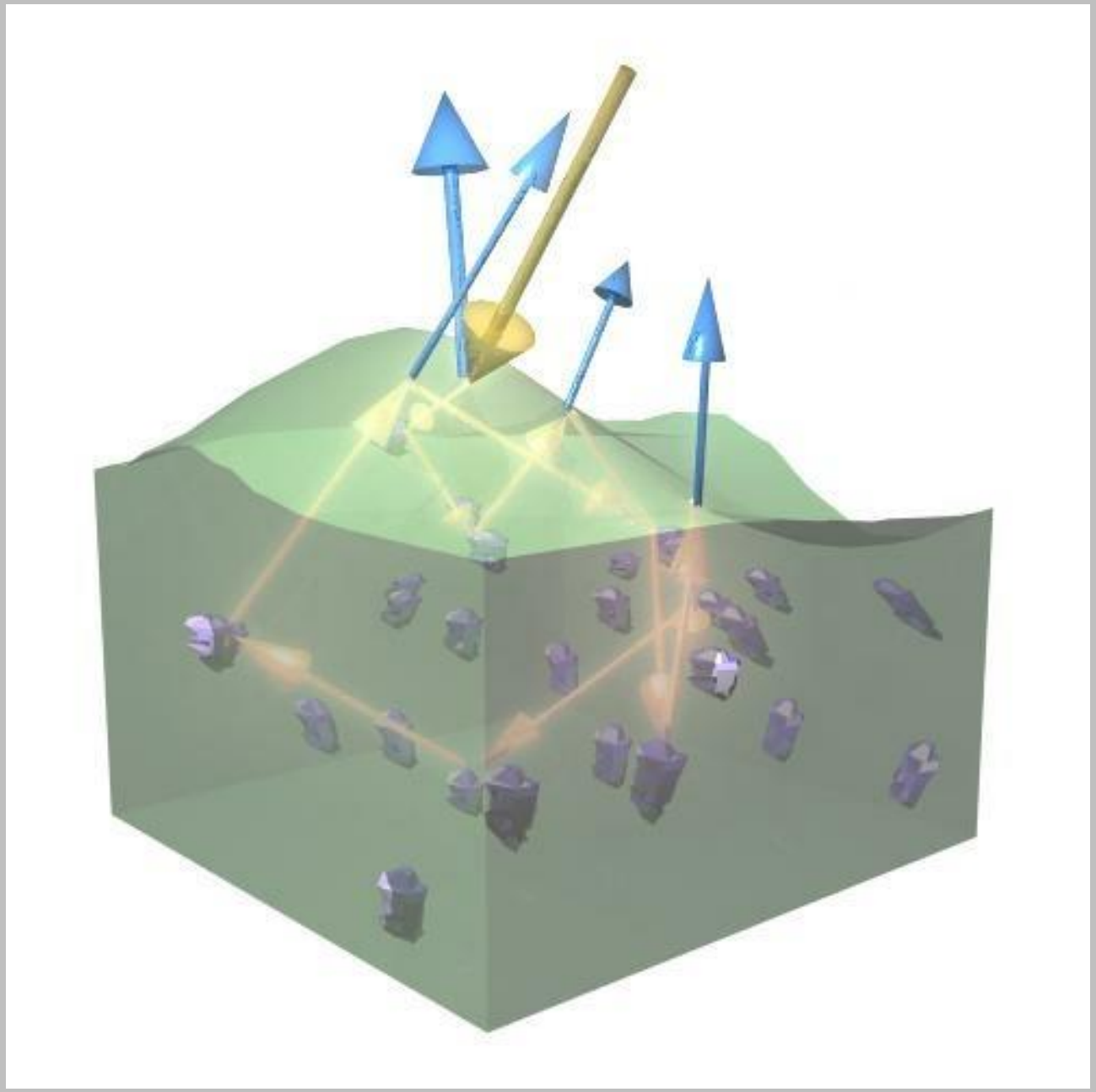


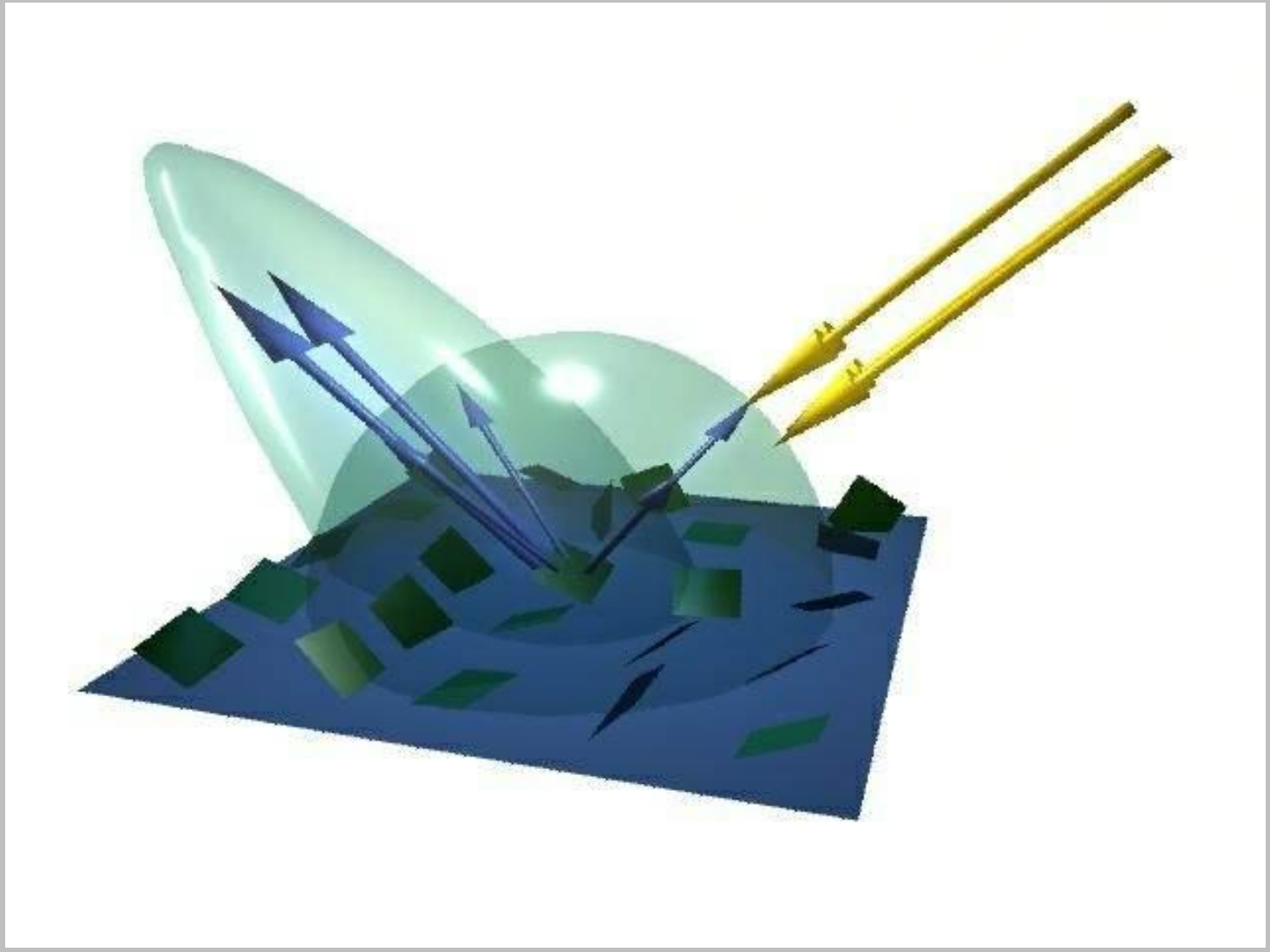



Scan for AOI 10° and POI 45°
Measurement
11.2°



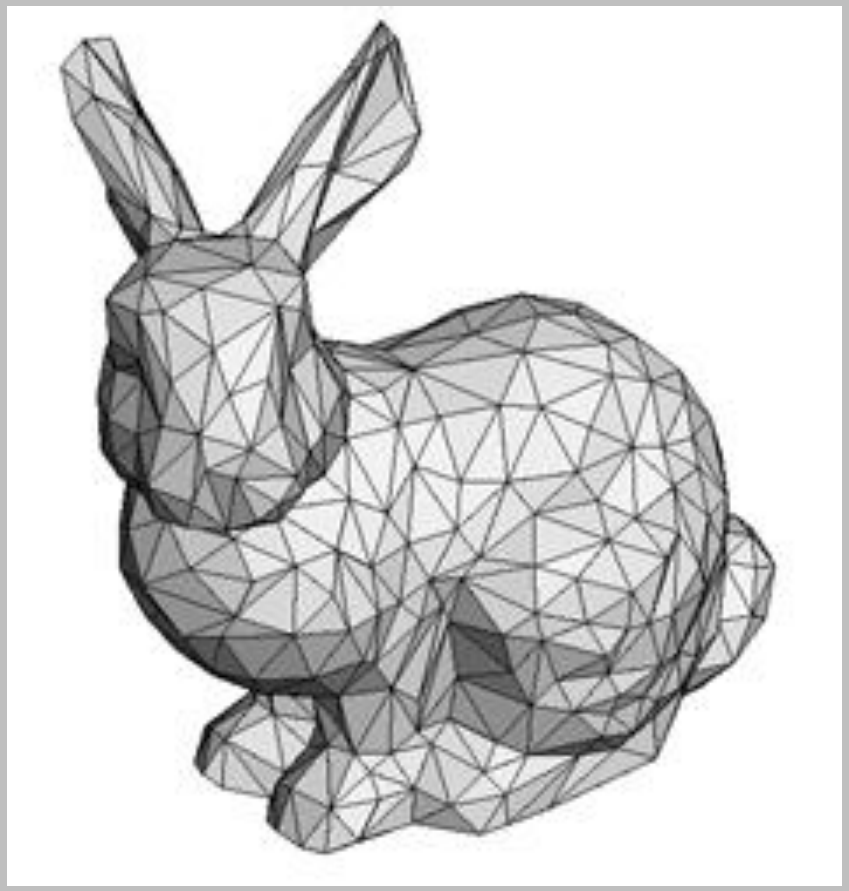
$$I(P) = k_d I_{ambiant} + \sum_{i \le N} k_d . \cos \alpha_i . I_i + \sum_{i \le N} k_s (\alpha) . \cos^n \beta_i . I_i$$

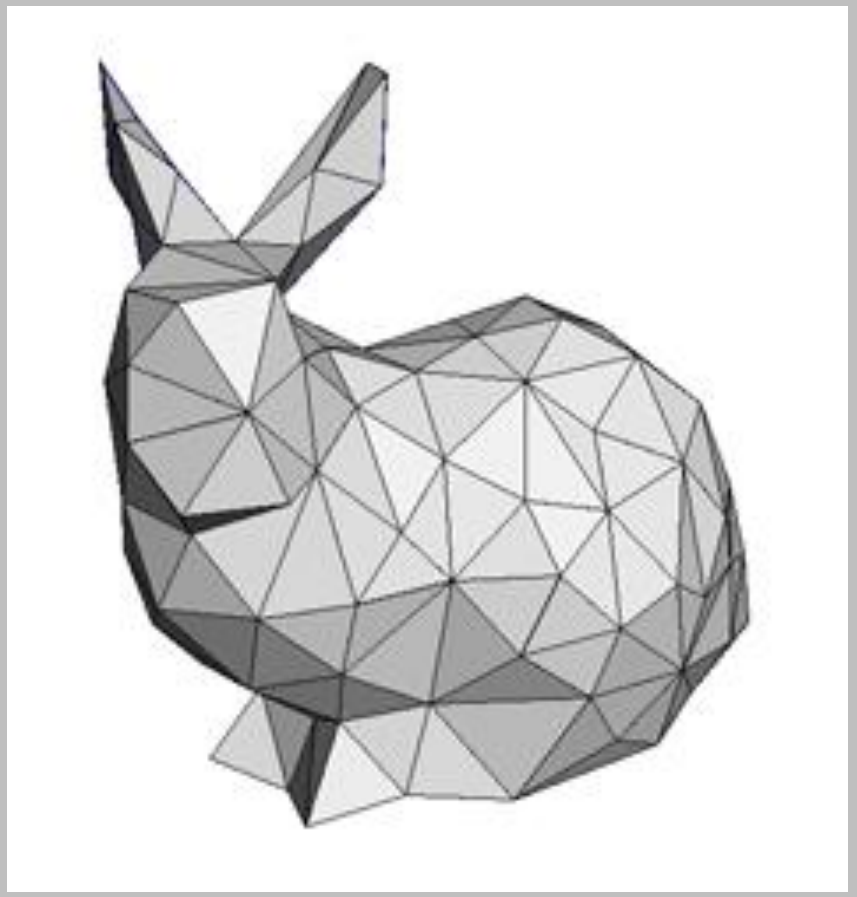




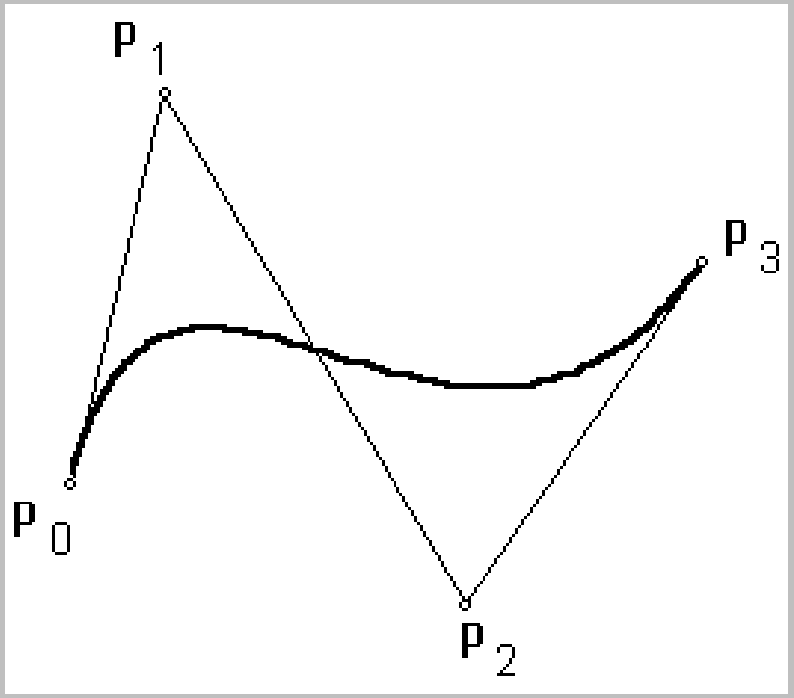
Modélisation

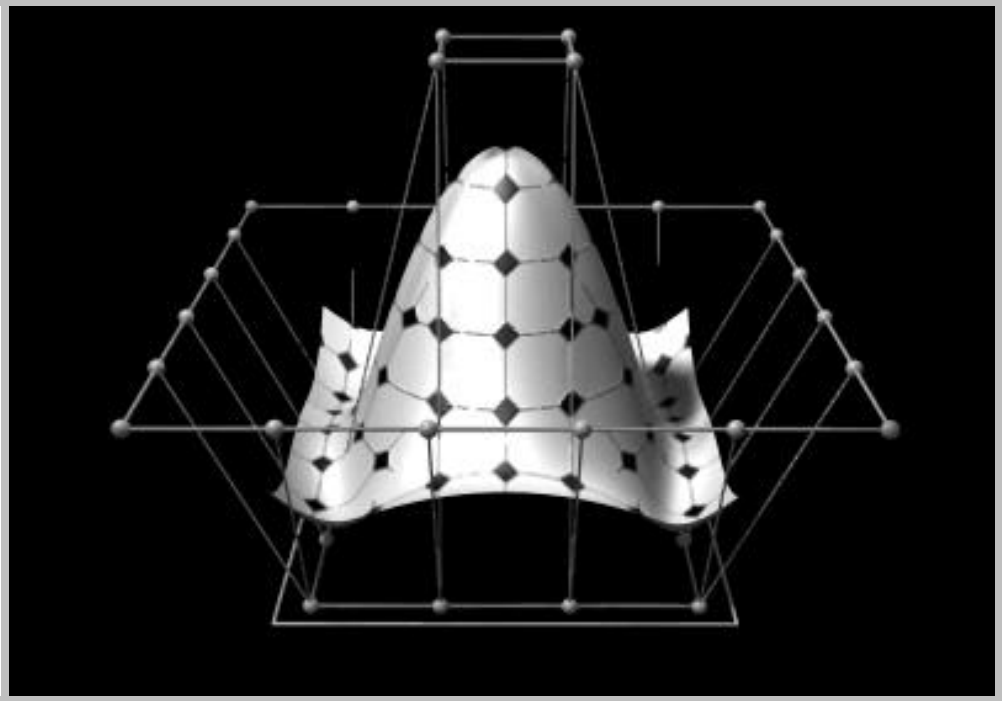
- Modélisation surfacique : polygones



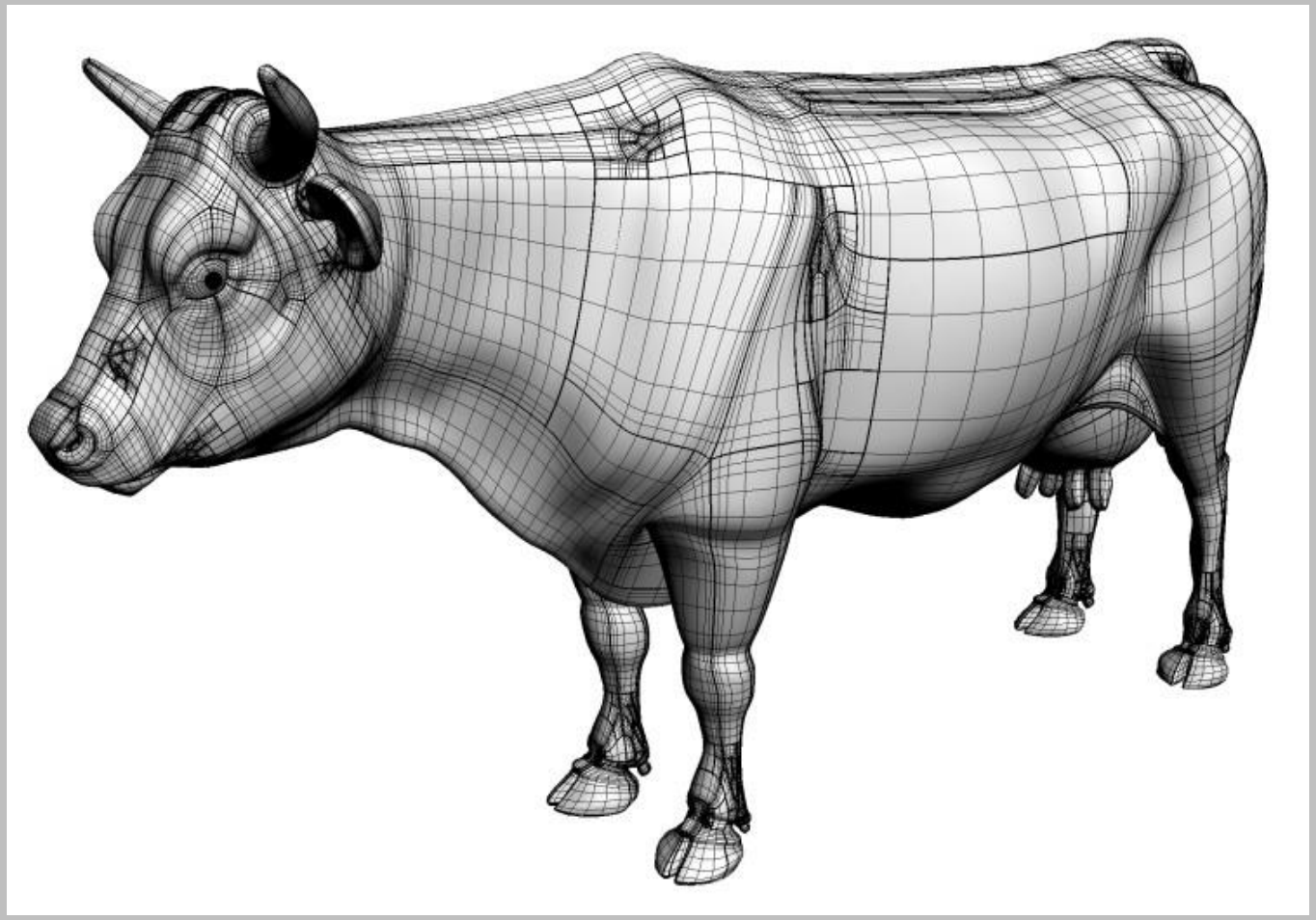


- Modélisation surfacique : Courbes et Carreaux paramétrés

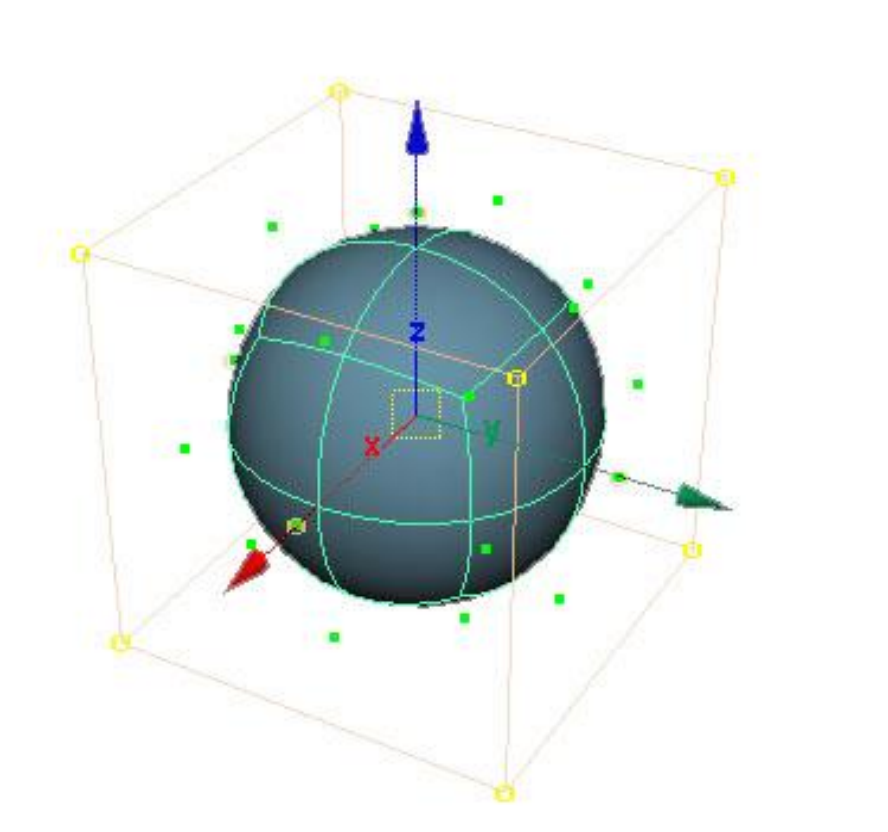


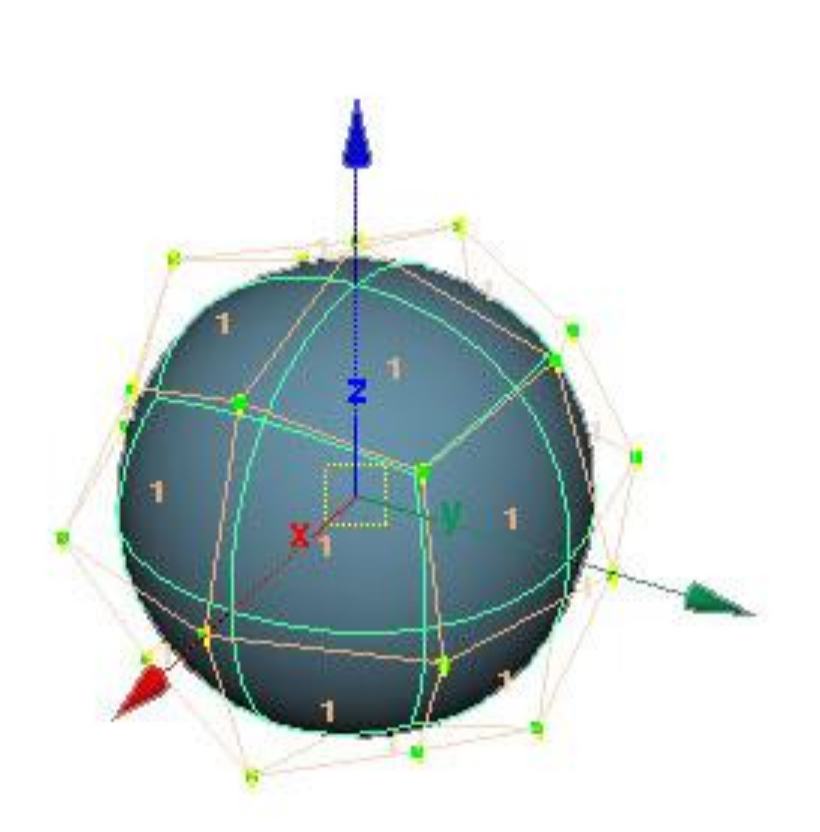


- Modélisation surfacique : Courbes et Carreaux paramétrés

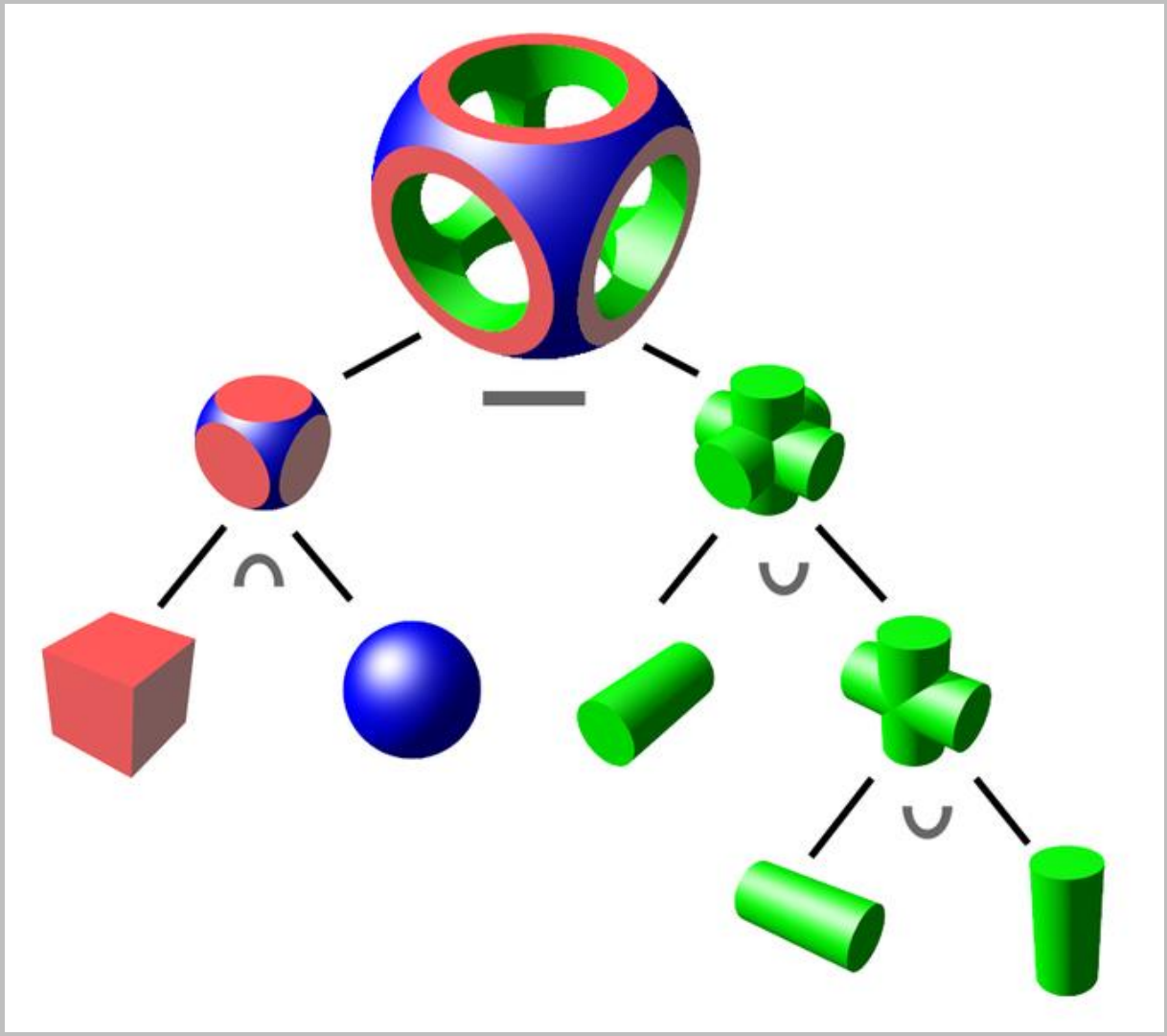


- Surfaces de subdivision





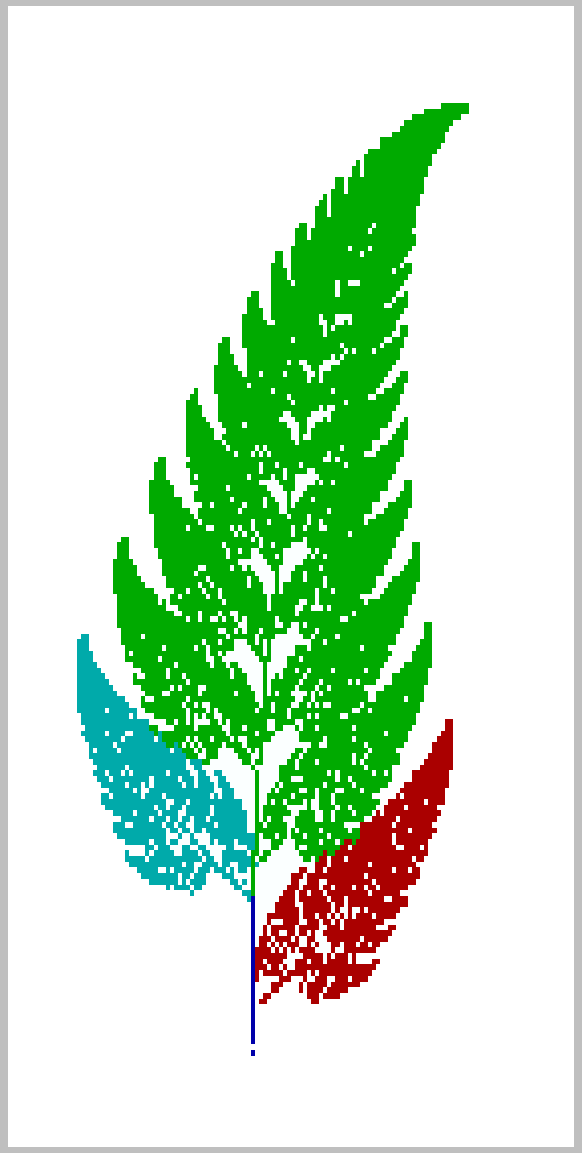
- CSG



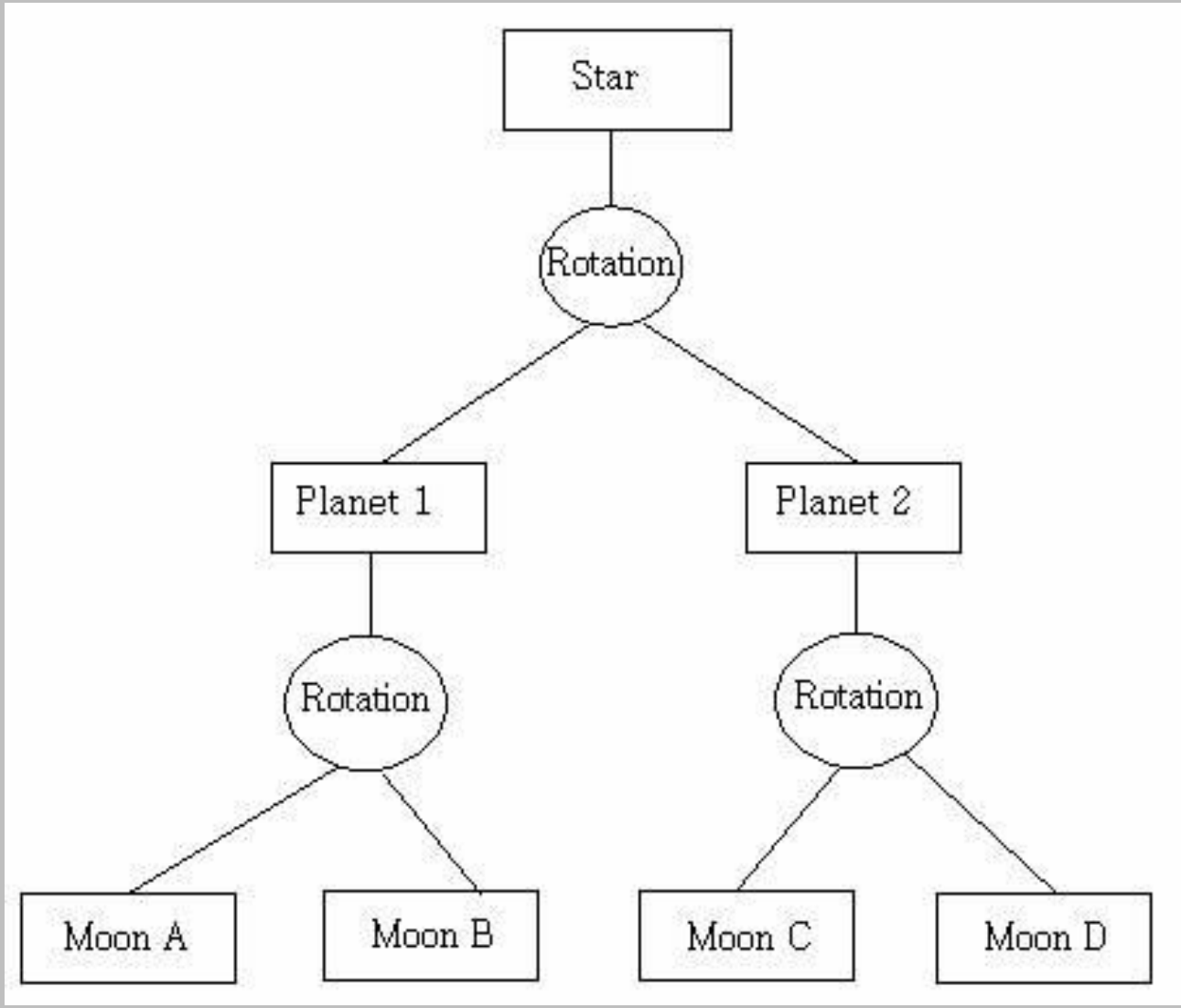
- Modélisation volumique

- Modélisation volumique

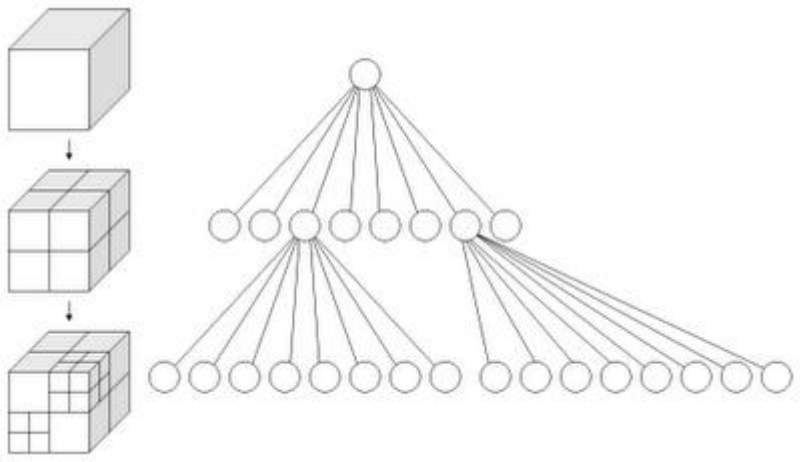
- Fractales

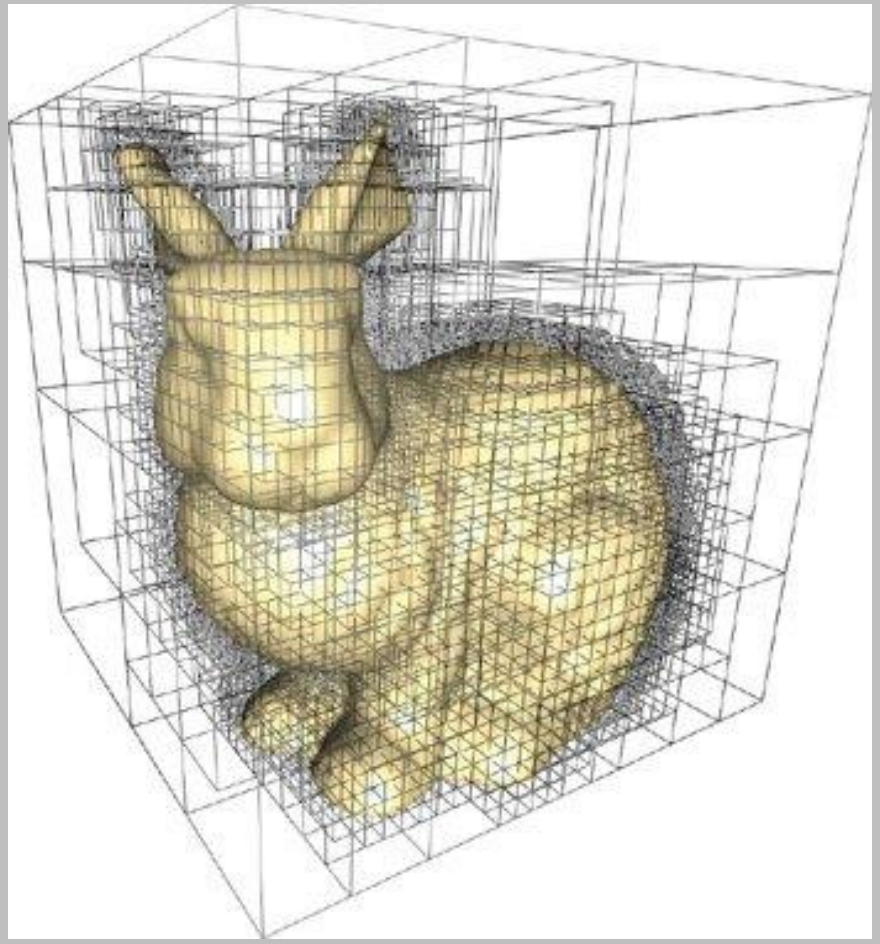


- Graphe de scène

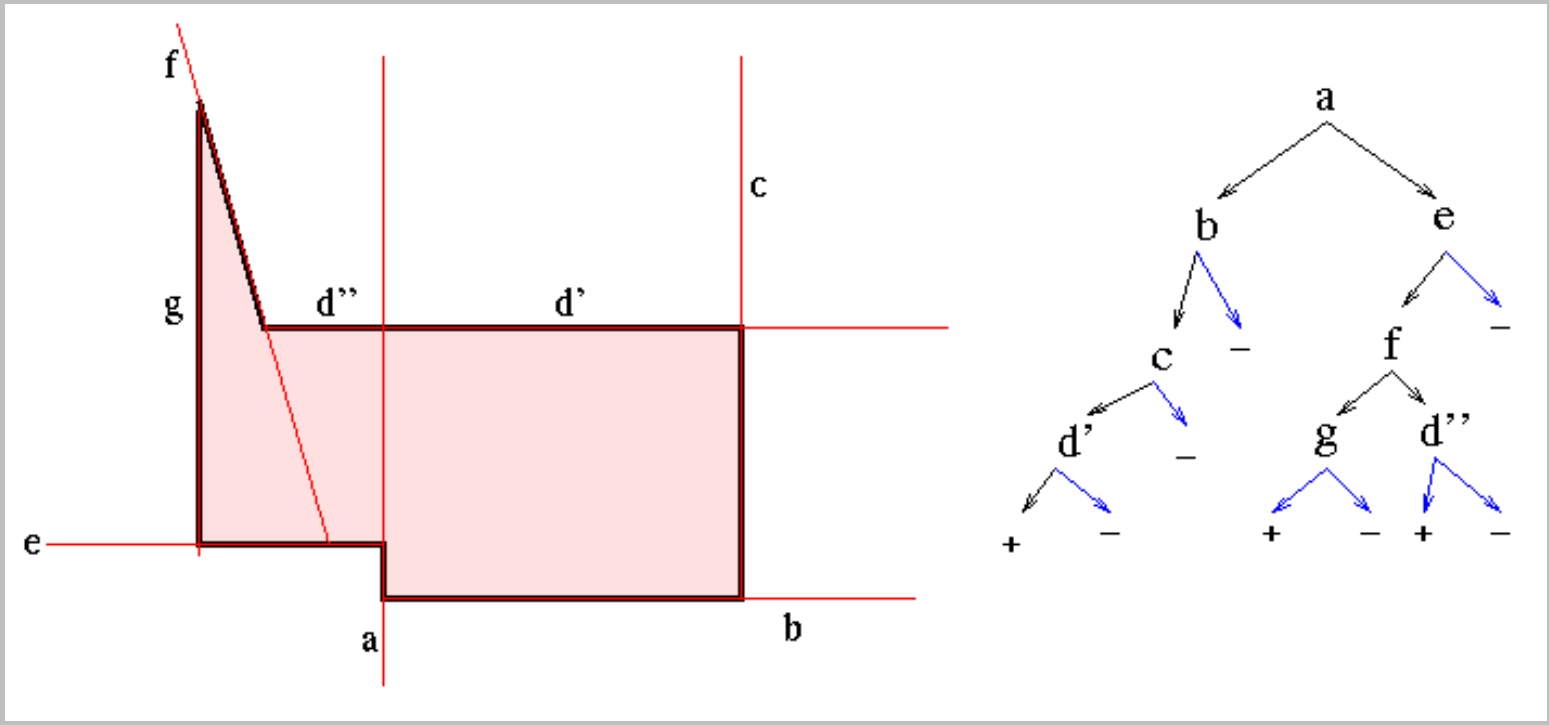


- Octrees

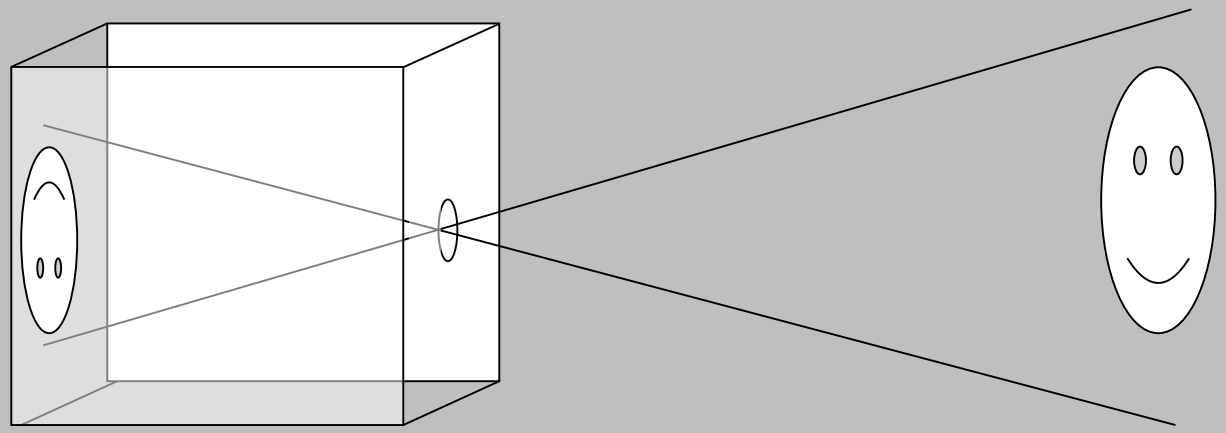


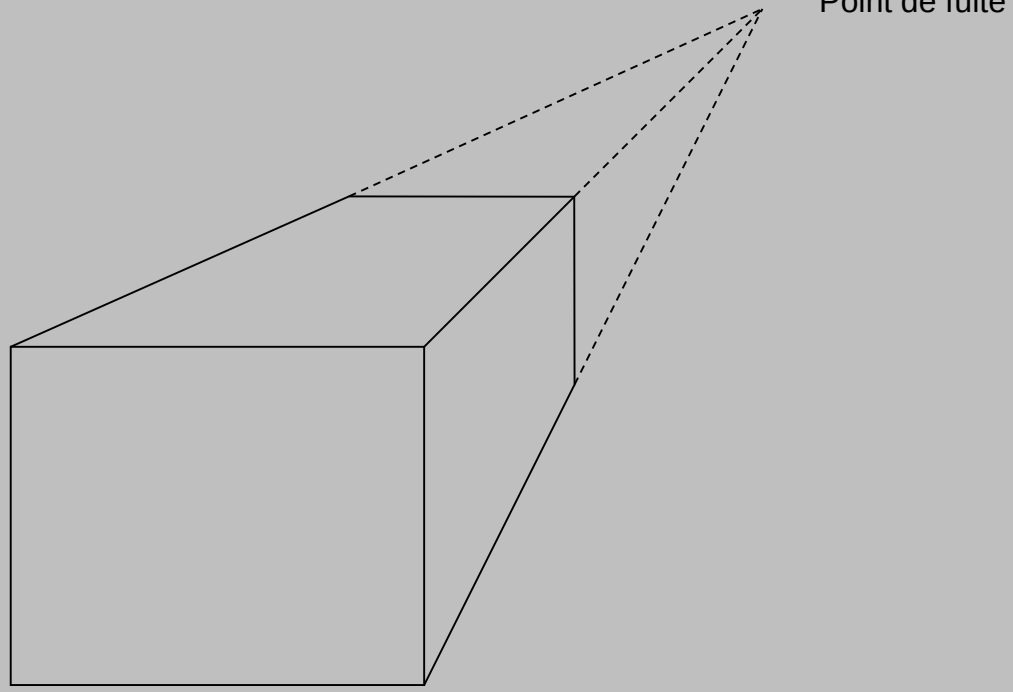


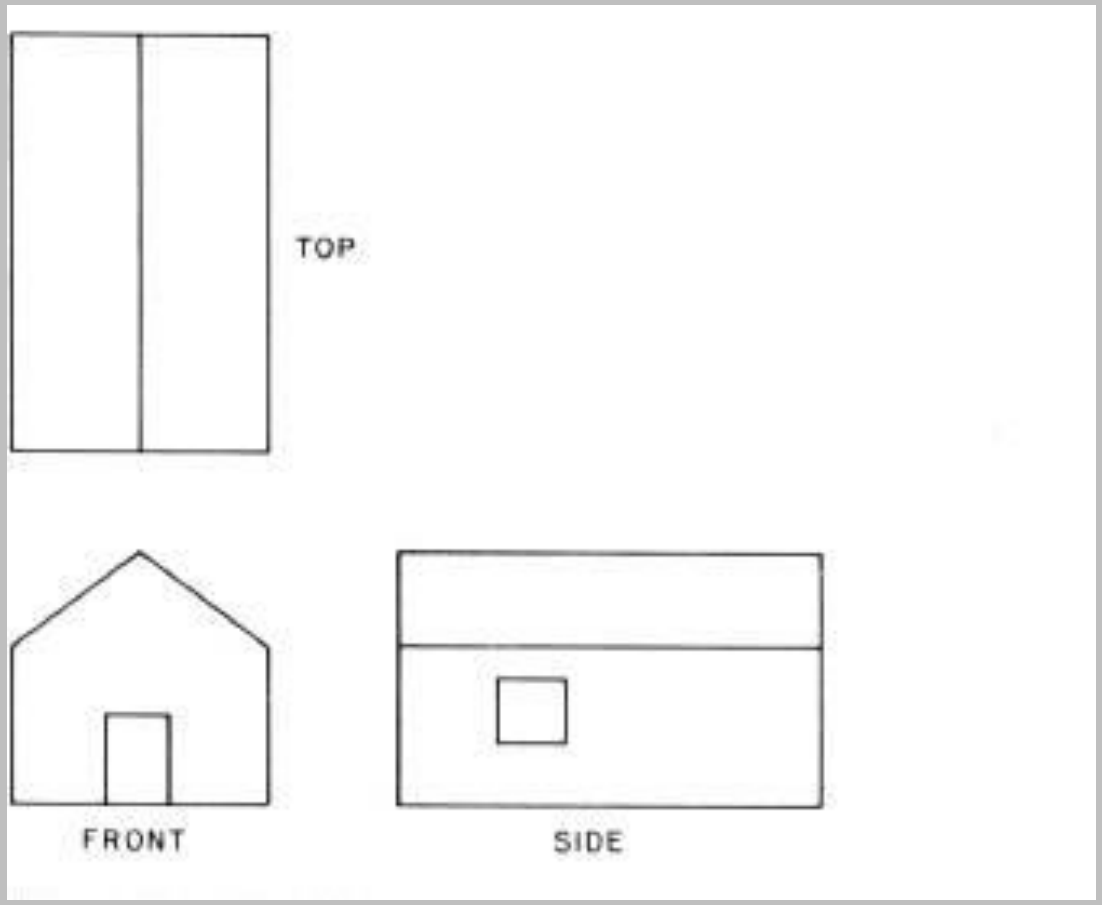
- BSP

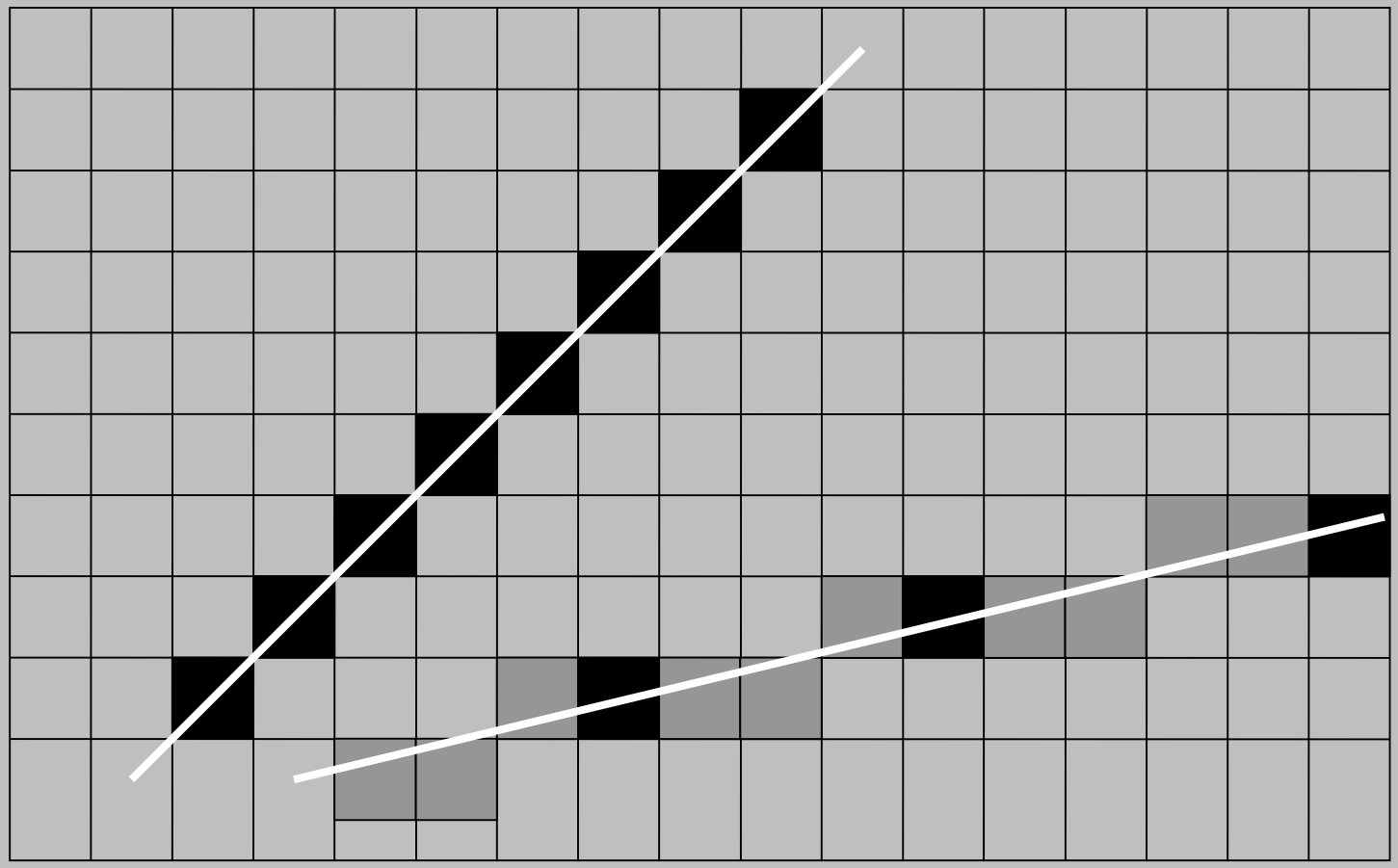


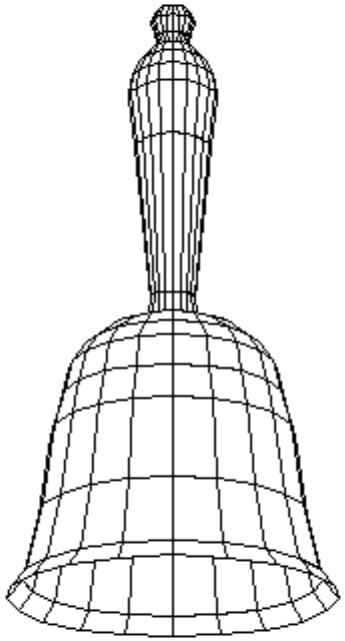
Prise de vue

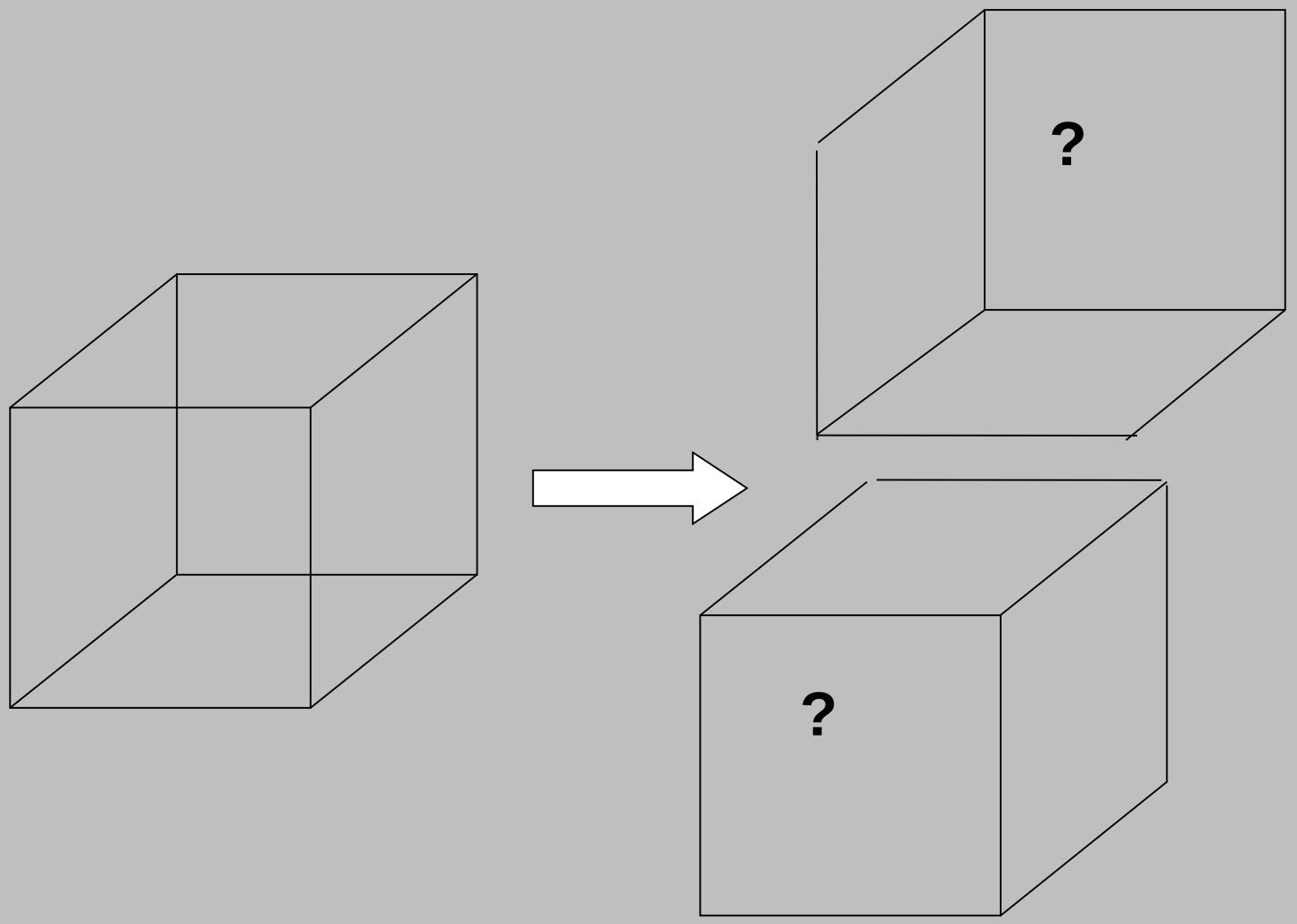


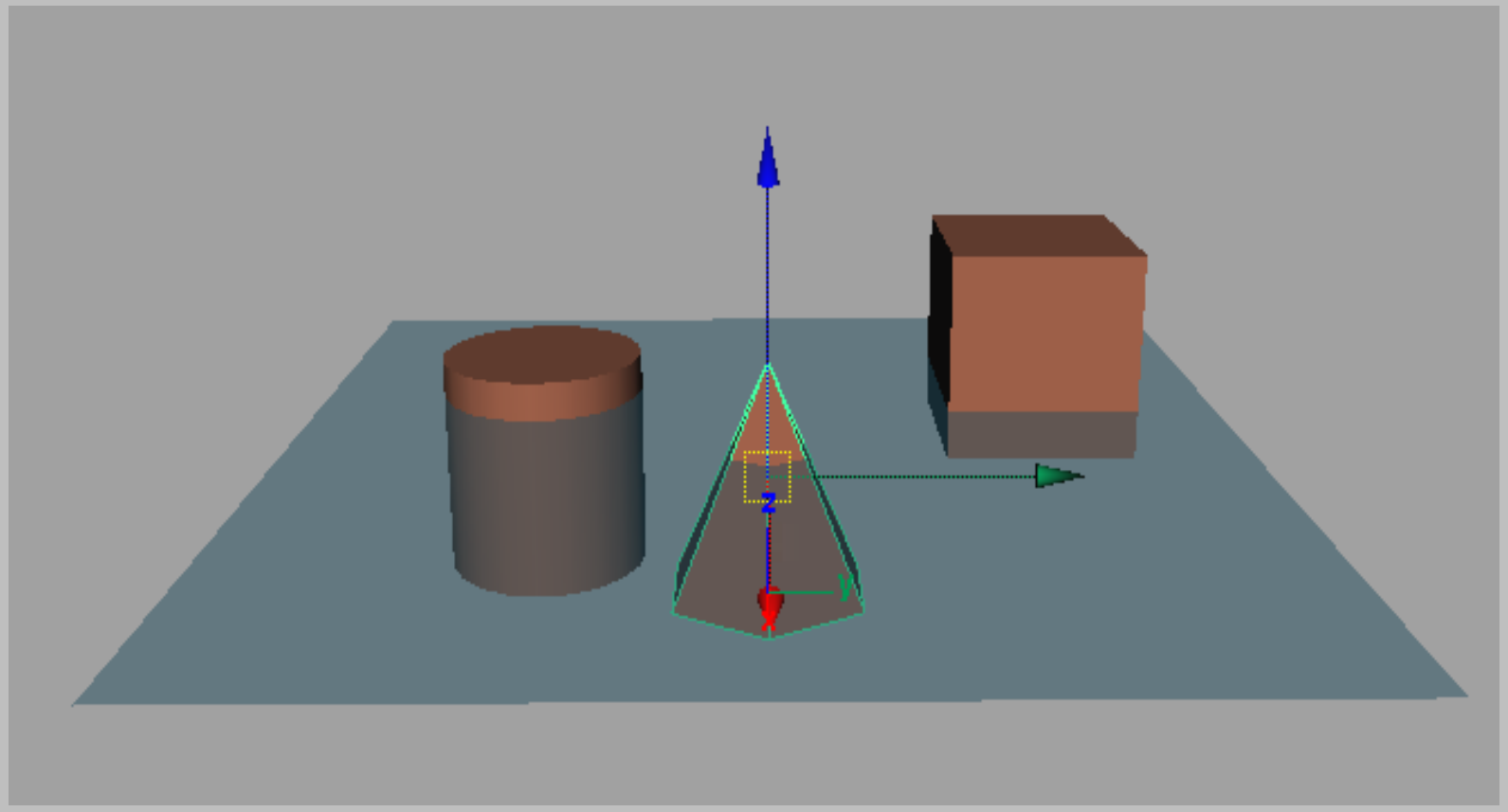

Point de fuite

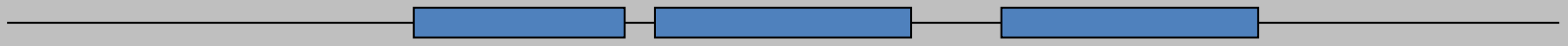

TOP
FRONT
SIDE

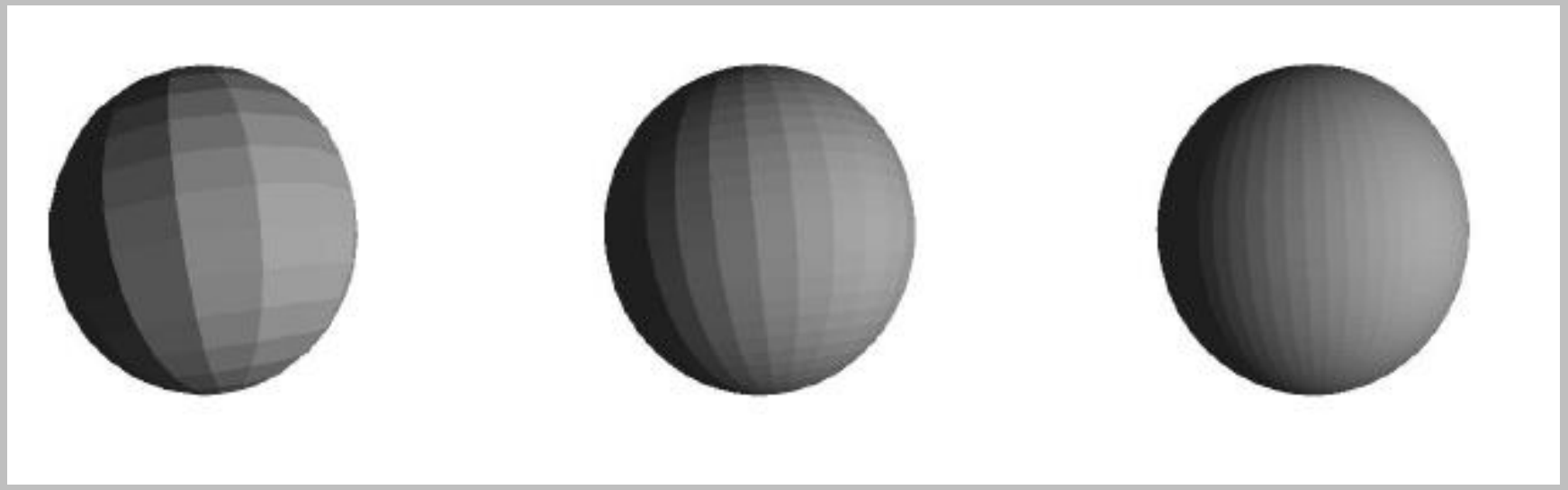


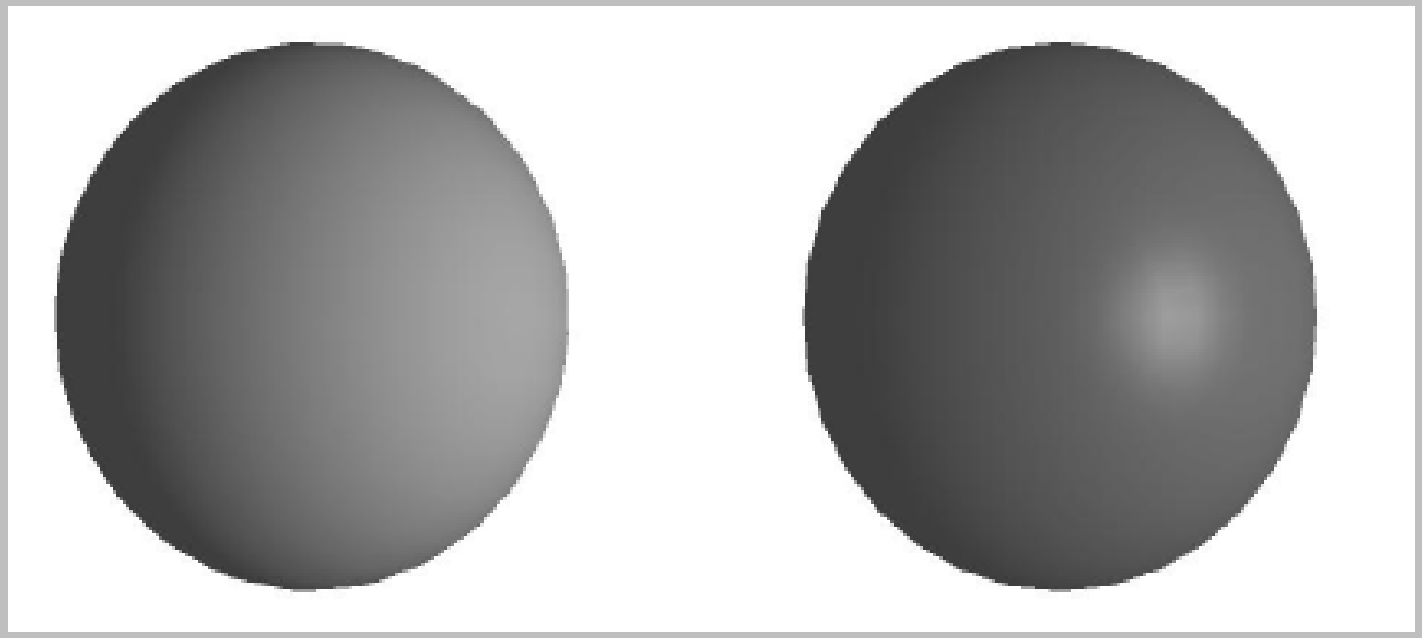


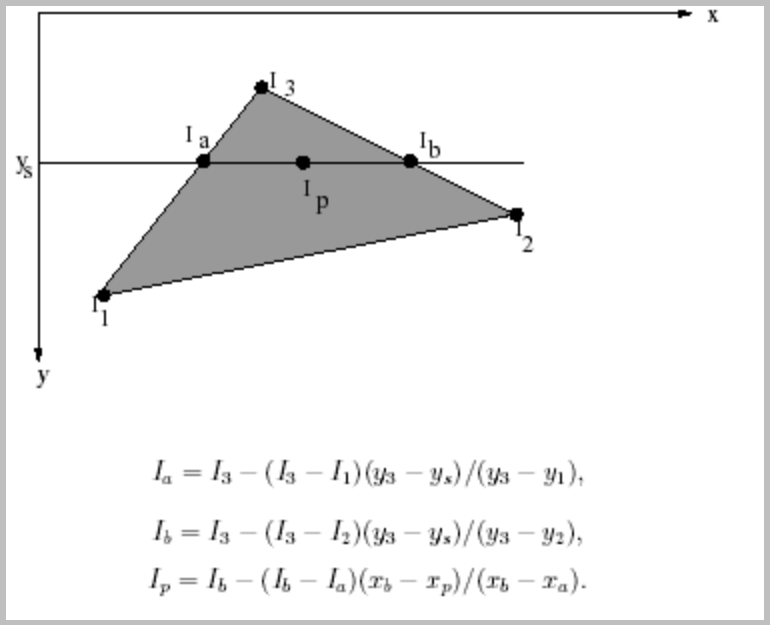



Z
Y





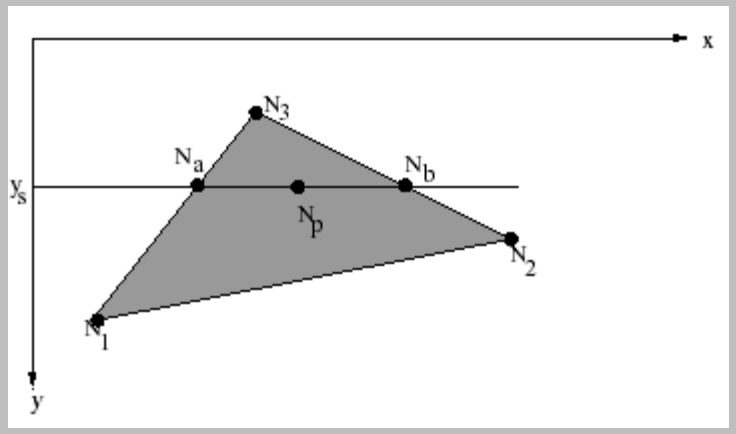


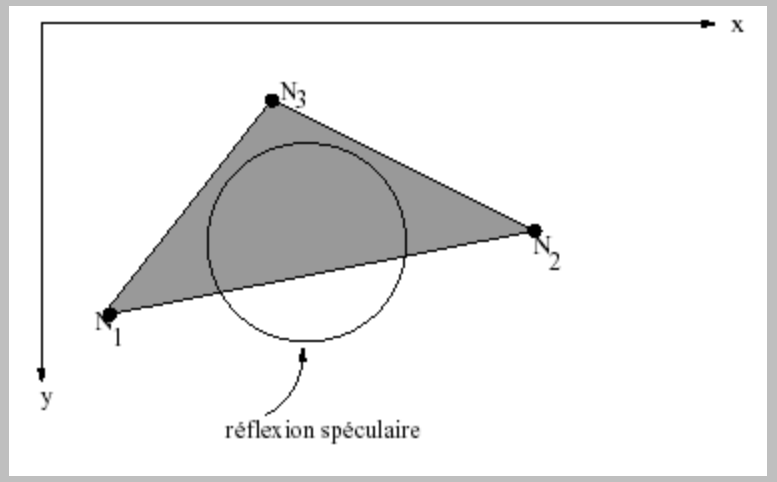


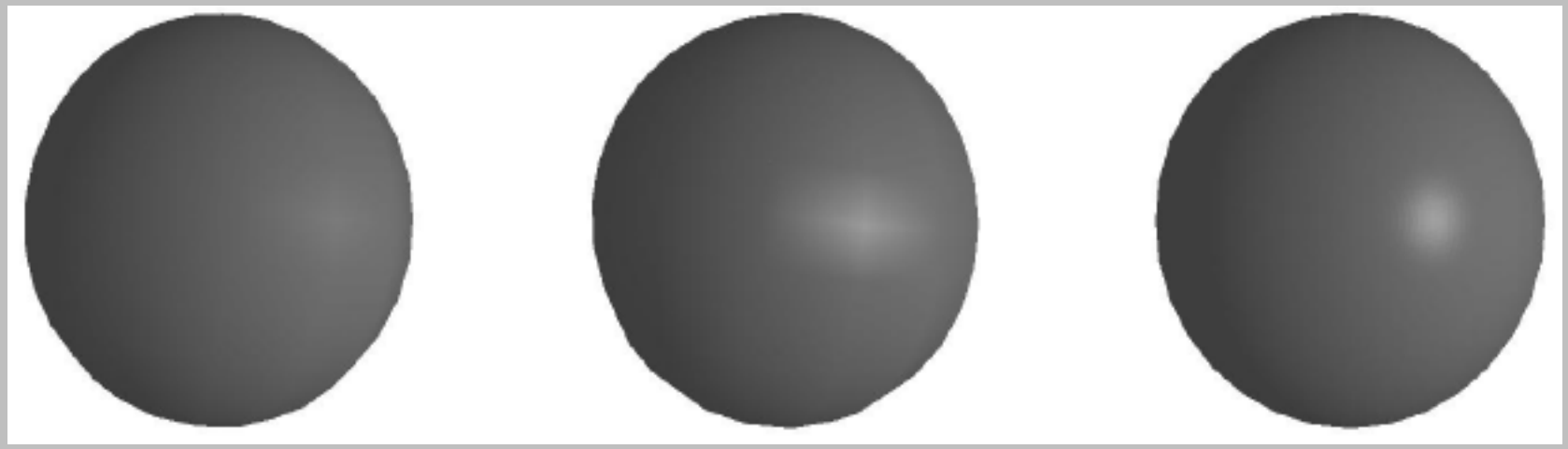
$$I_a = I_3 - (I_3 - I_1)(y_3 - y_s)/(y_3 - y_1),$$

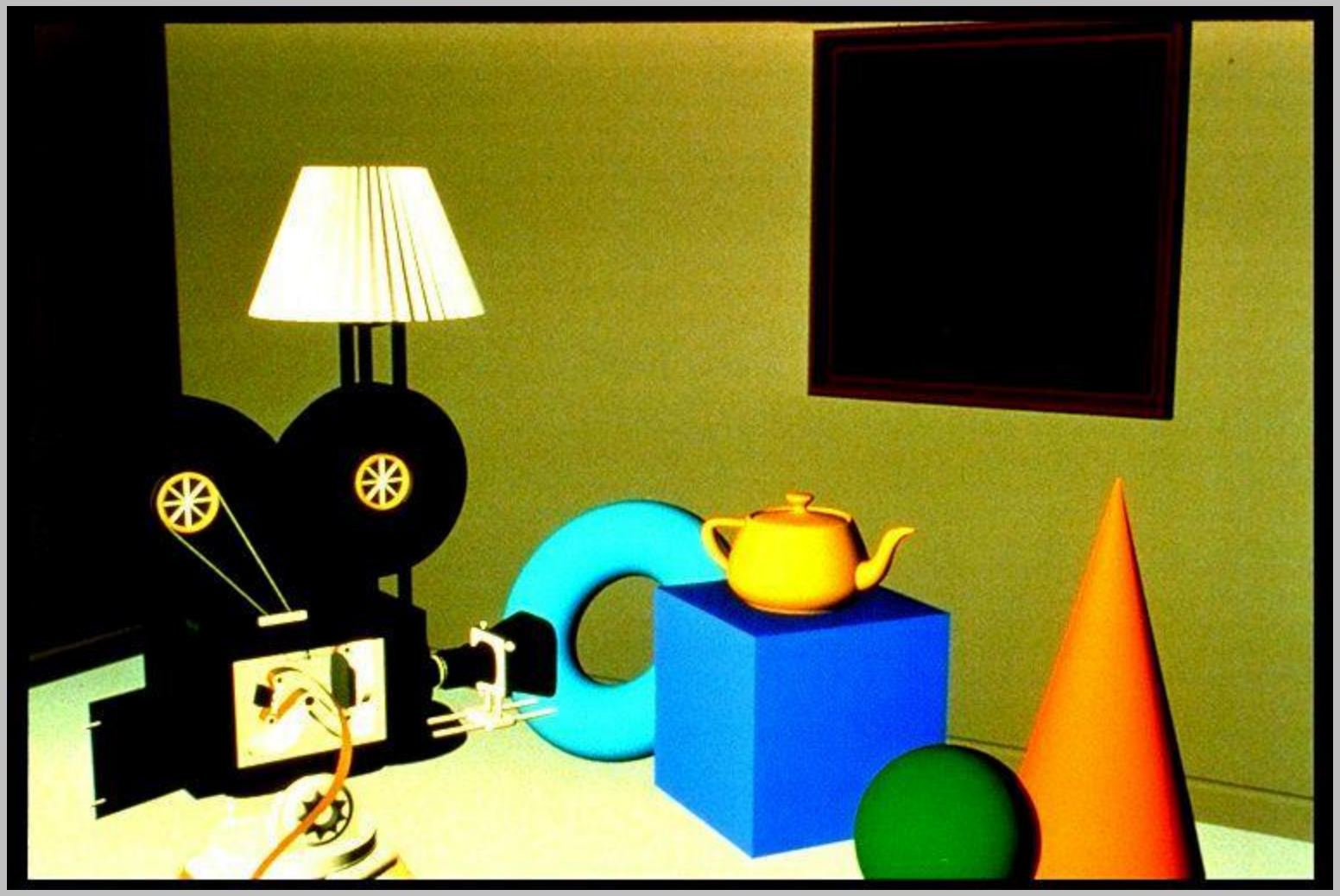
$$I_b = I_3 - (I_3 - I_2)(y_3 - y_s)/(y_3 - y_2),$$

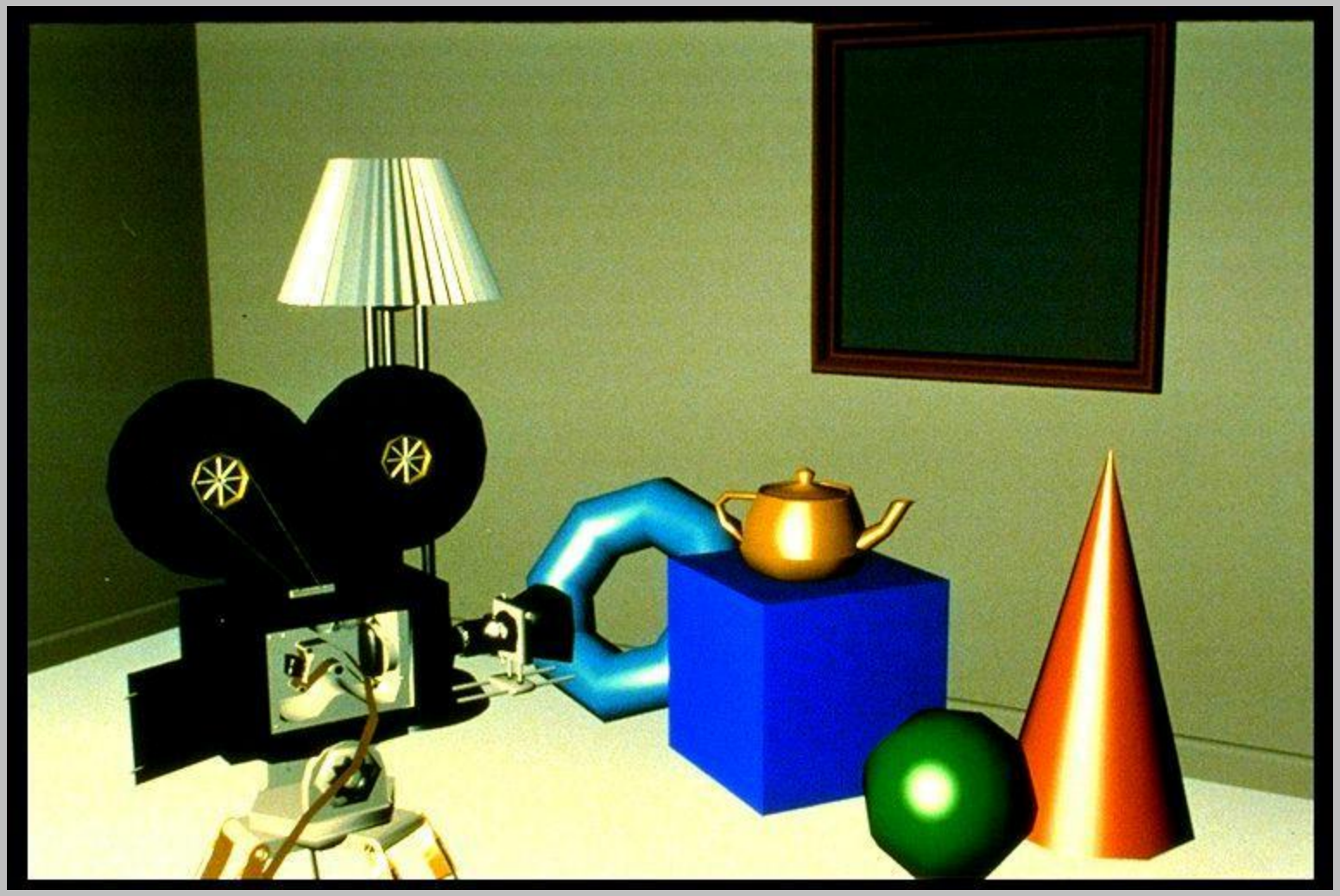
$$I_p = I_b - (I_b - I_a)(x_b - x_p)/(x_b - x_a).$$


x
y
N_3
N_a
N_b
y_s
N_p
N_2
N_1

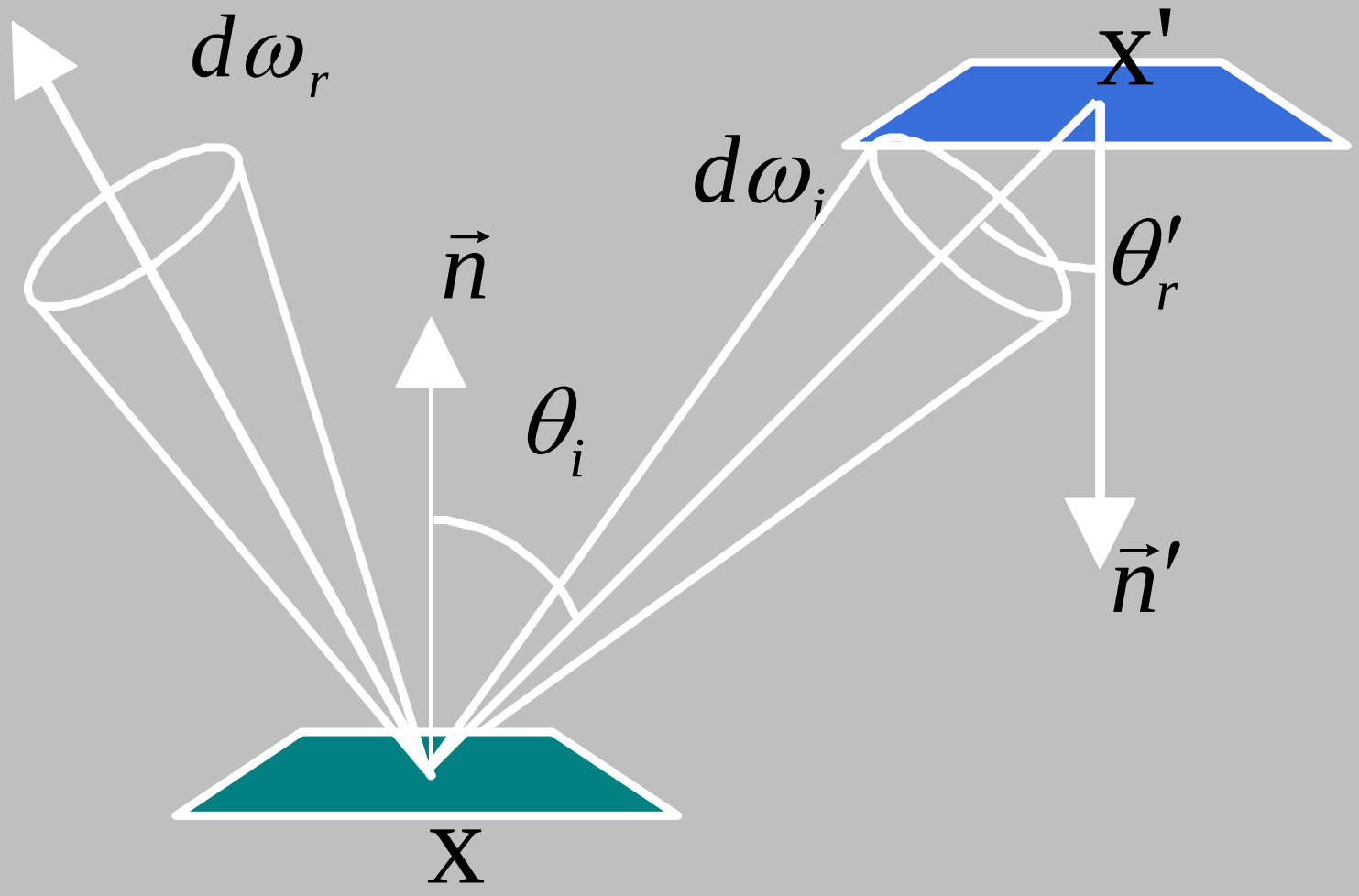

x
y
N_3
N_2
N_1
réflexion spéculaire







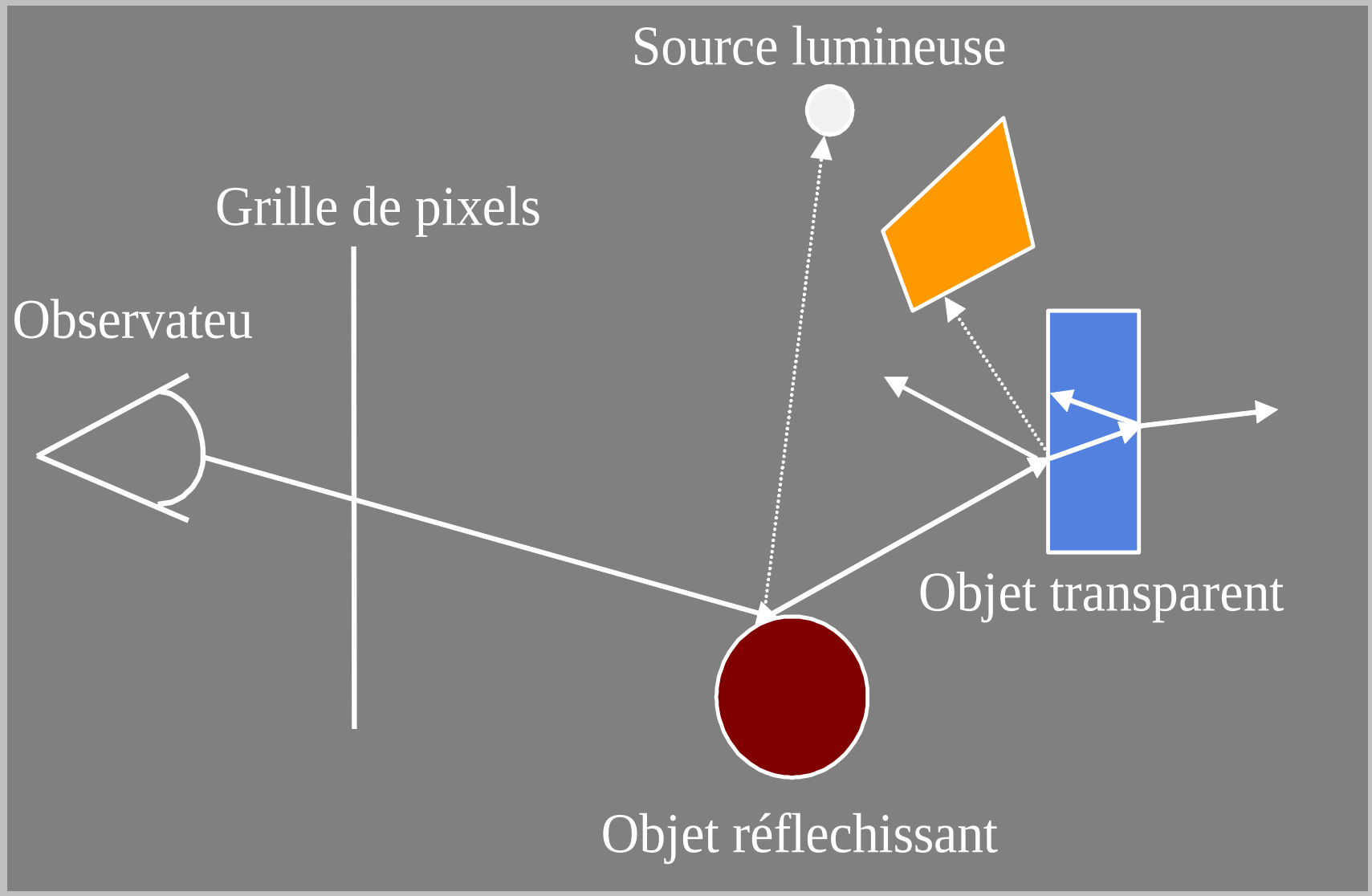
Éclairage

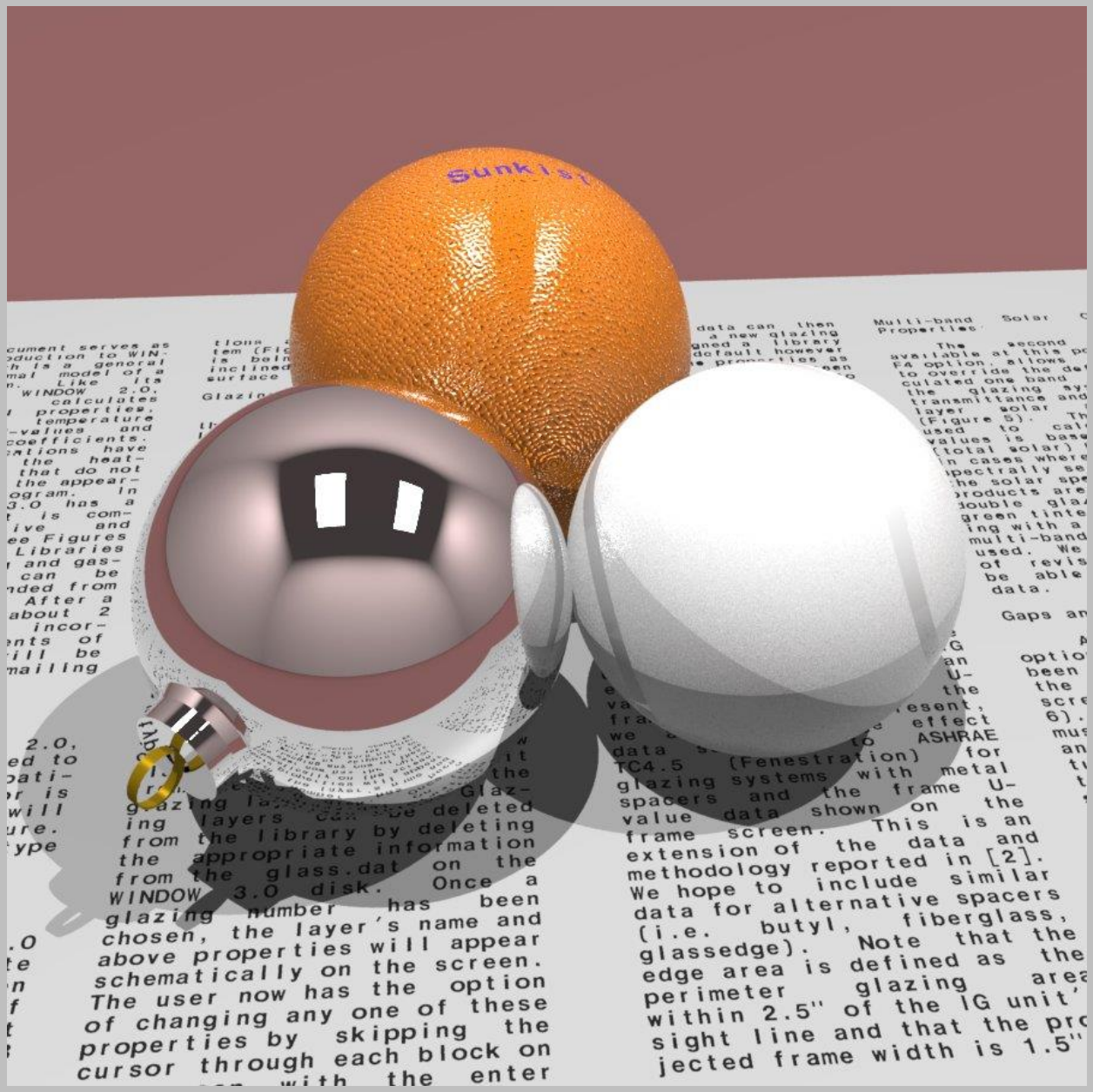

$d\omega_r$
$d\omega_i$
$\vec{n}$
θ_i
θ'_r
$\vec{n}'$
X'
X

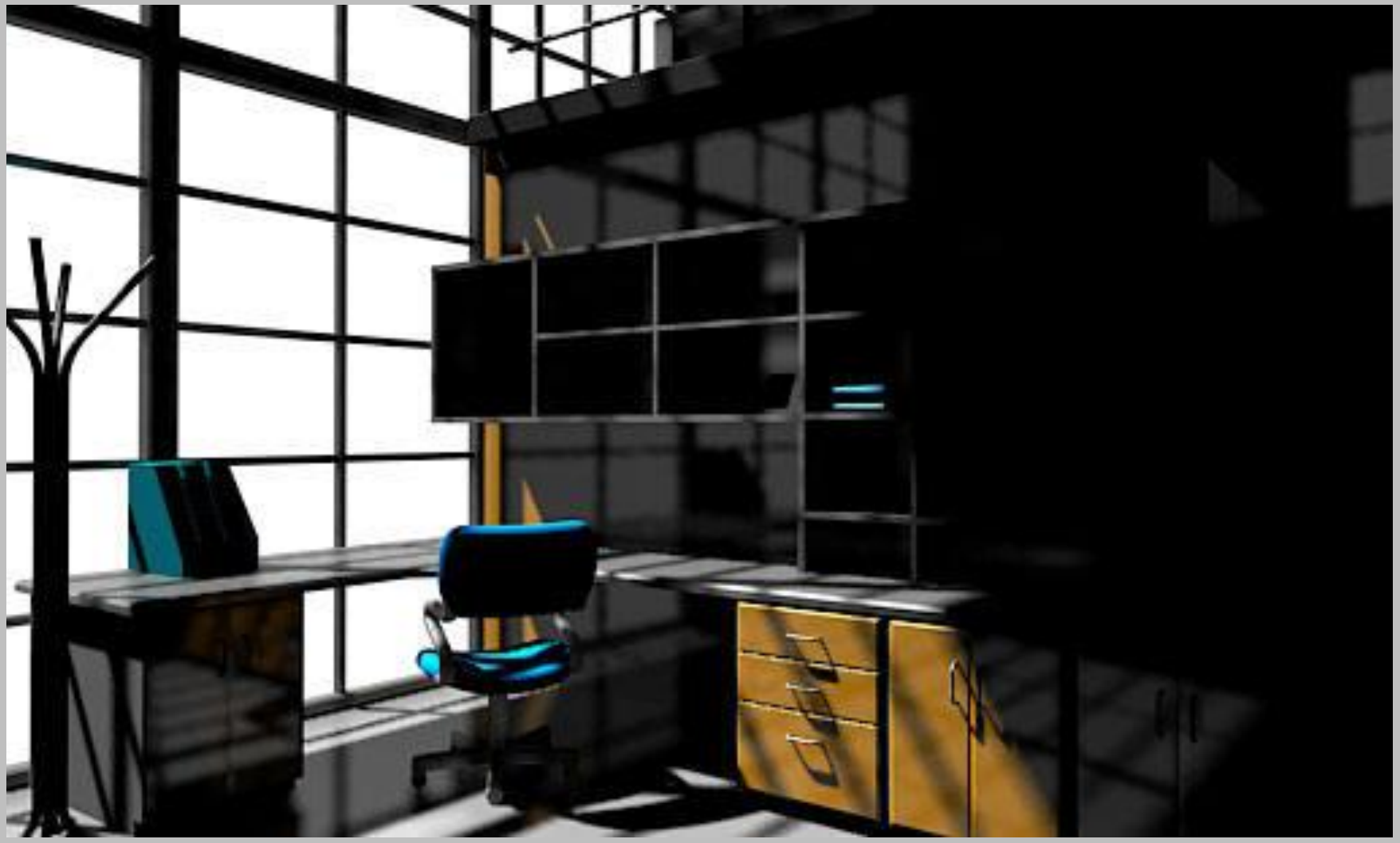
- Equation du rendu

$$L_r\left(x,\vec{\omega}_r\right)=$$
$$L_e\left(x,\vec{\omega}_r\right)+$$
$$\int_S \rho_{bd}\left(x,\vec{\omega}_I,\vec{\omega}_r\right).L_r\left(x',\vec{\omega}'_r\right).G(x,x')dxdx'$$

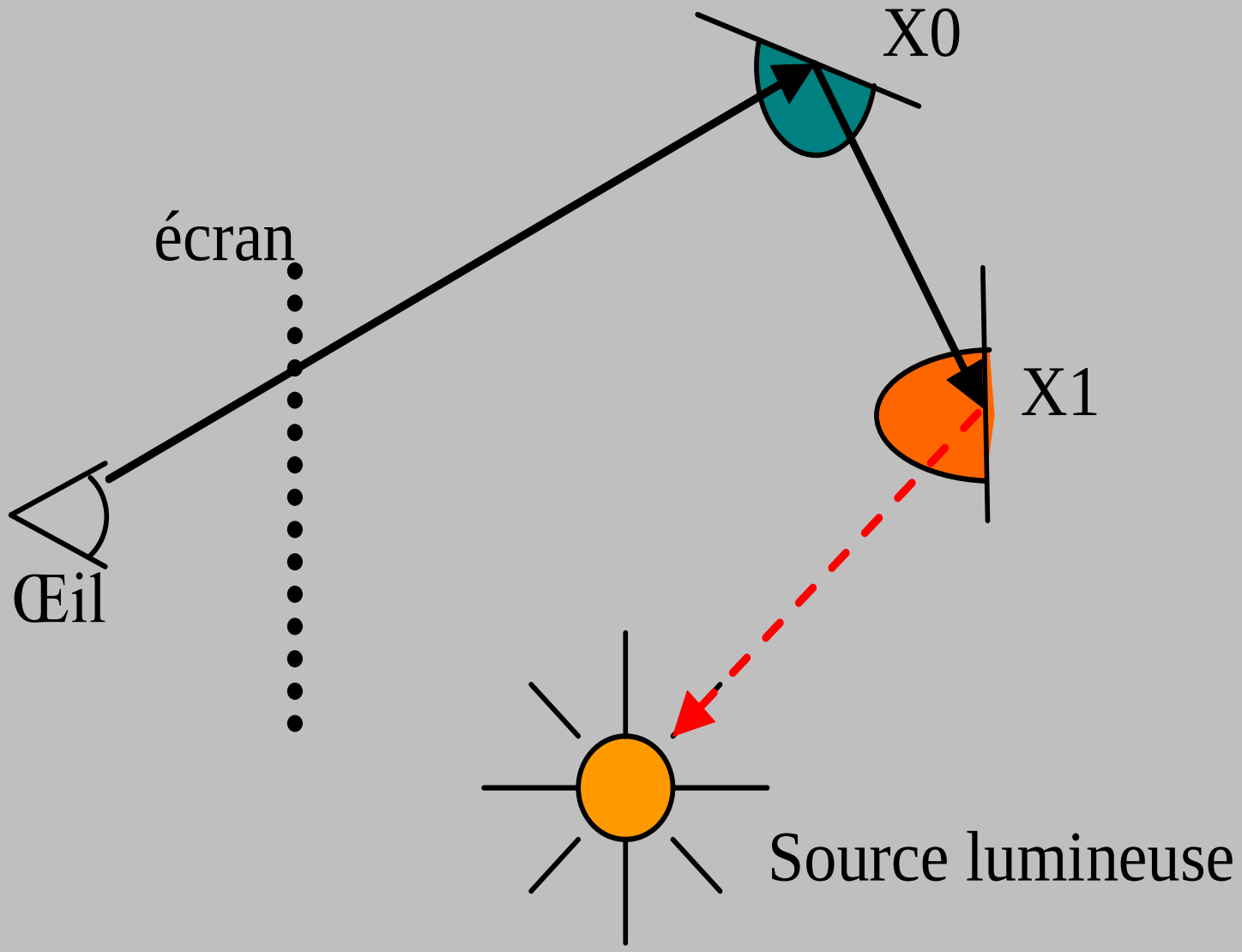
- Lancer de rayon

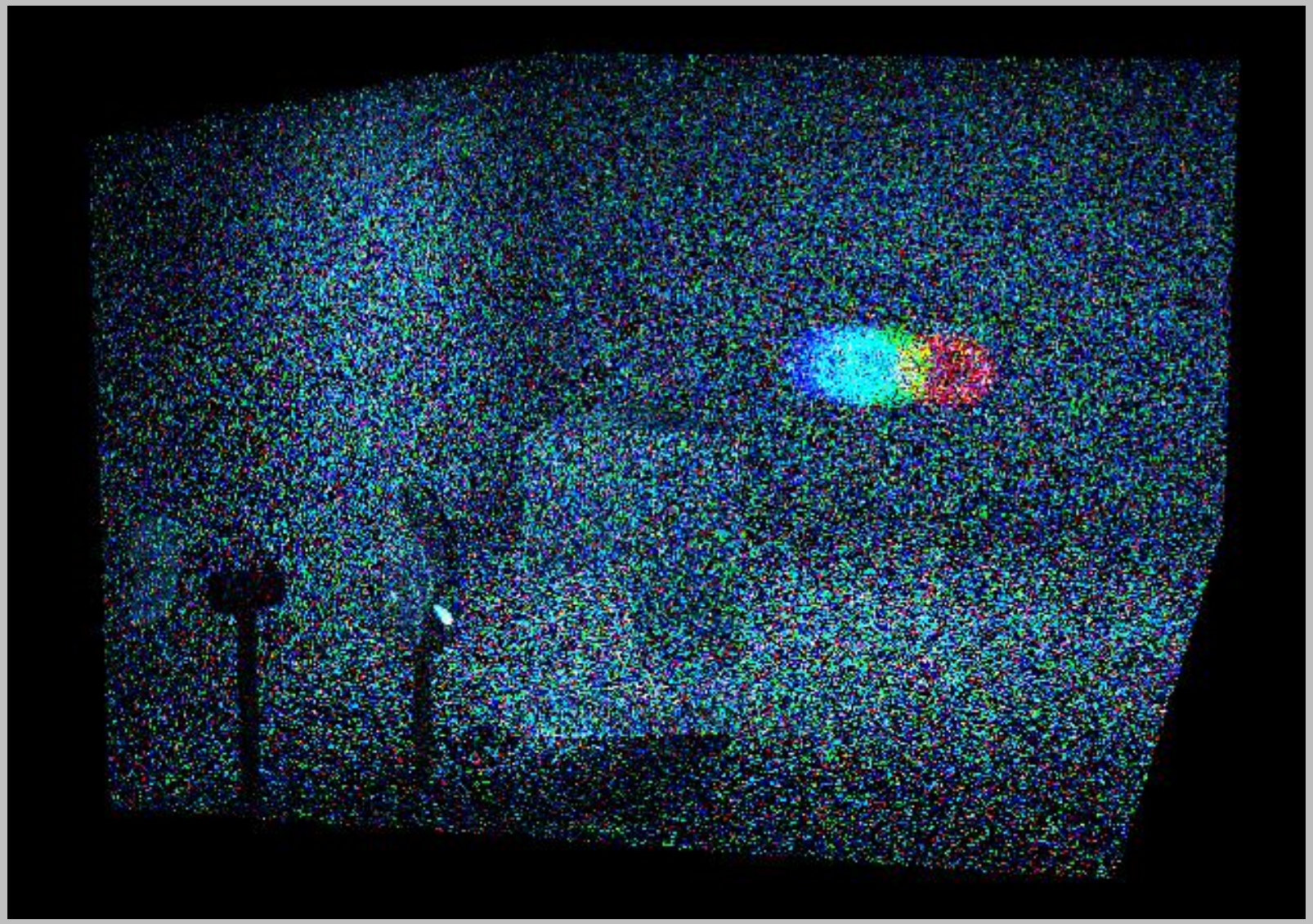


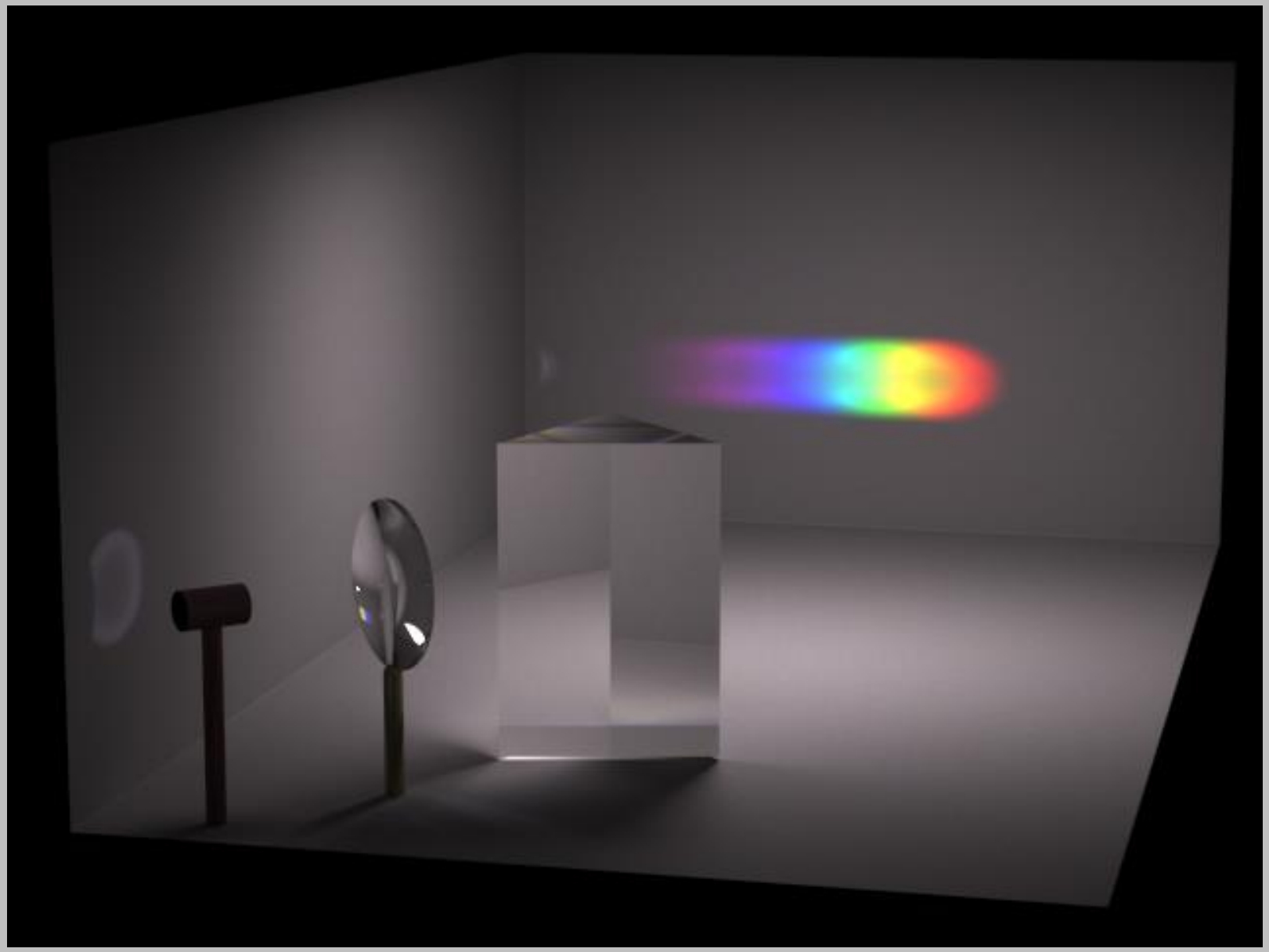

Sunkist
WINDOW 3.0 disk. Once a
glazing number has been
chosen, the layer's name and
above properties will appear
schematically on the screen.
The user now has the option
of changing any one of these
properties by skipping the
cursor through each block on
ASHRAE
TC4.5 (Fenestration)
glazing systems with metal
spacers and the frame U-
value data shown on the
frame screen. This is an
extension of the data and
methodology reported in [2].
We hope to include similar
data for alternative spacers
(i.e. butyl, fiberglass,
glassedge). Note that the
edge area is defined as the
perimeter glazing area
within 2.5" of the IG unit's
sight line and that the pro
jected frame width is 1.5"



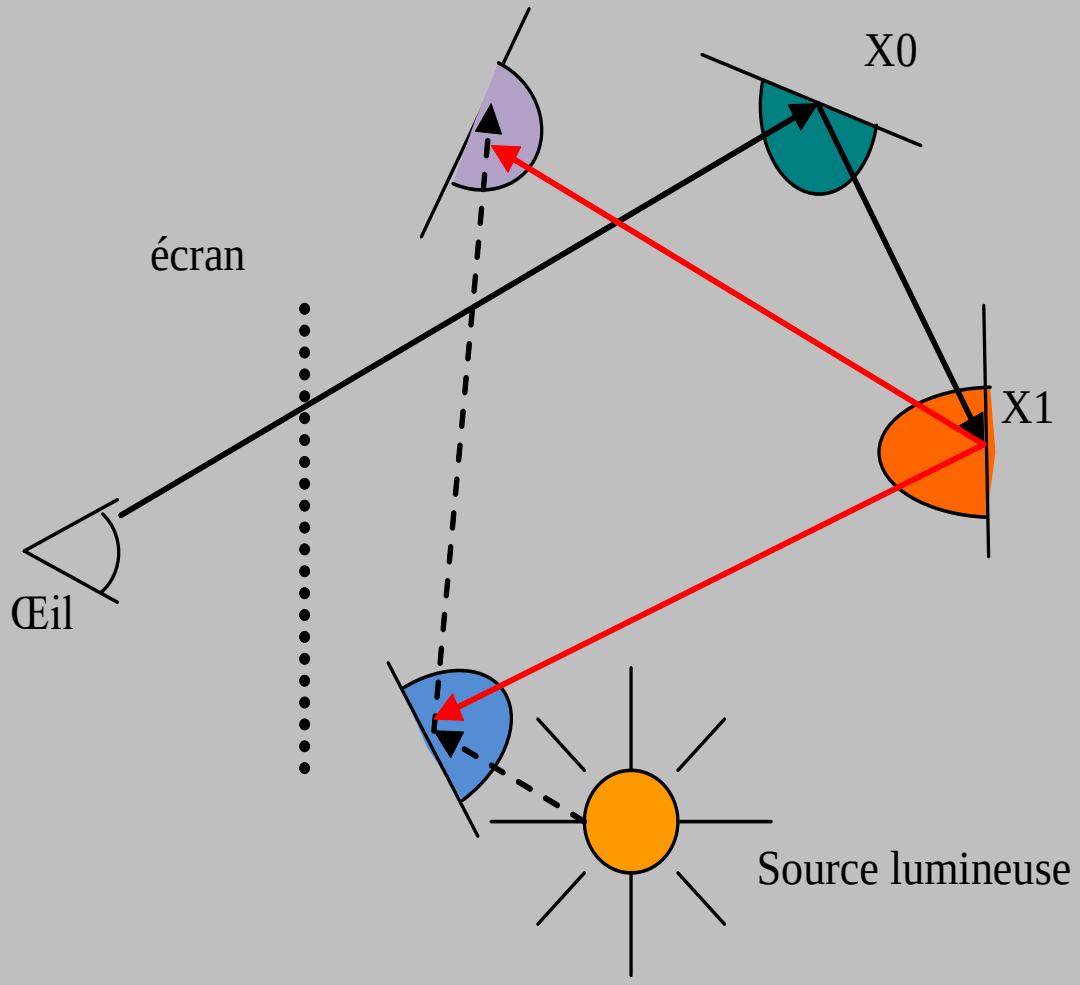
- Path tracing



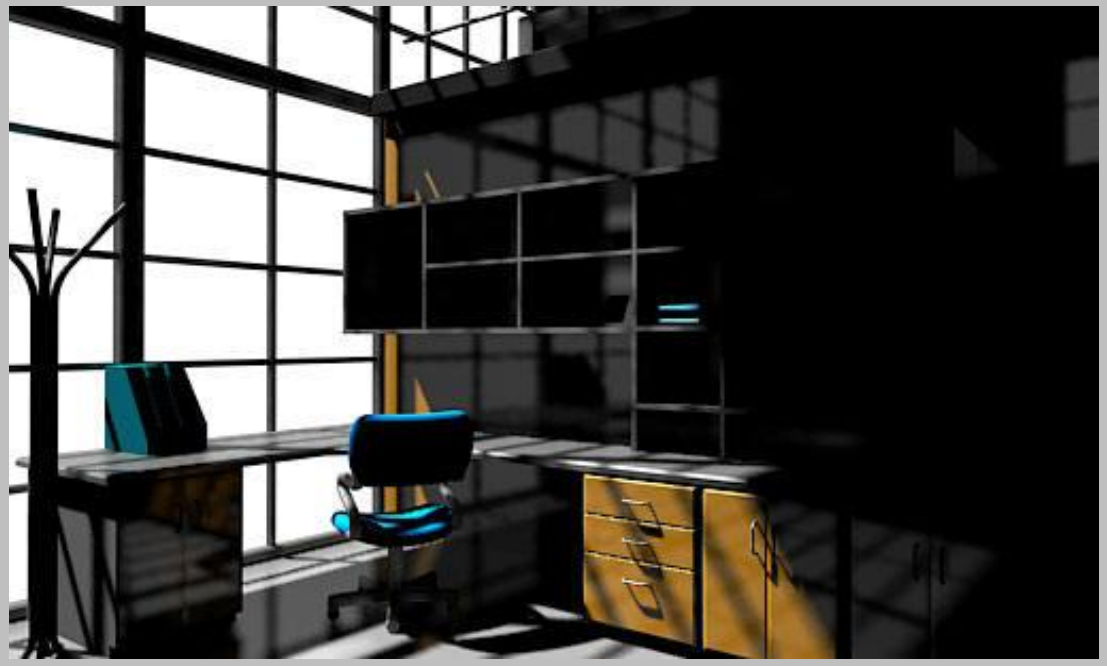




- Bidirectional path tracing



- Photon Mapping



Radiosité

- Radiosité

$$L_r\left(x,\vec{\omega}_r\right)=$$

$$L_e\left(x,\vec{\omega}_r\right)+$$

$$\rho(x)\int_S L\left(x',\vec{\omega}'_r\right)G(x,x')dxx'$$

- Radiosité

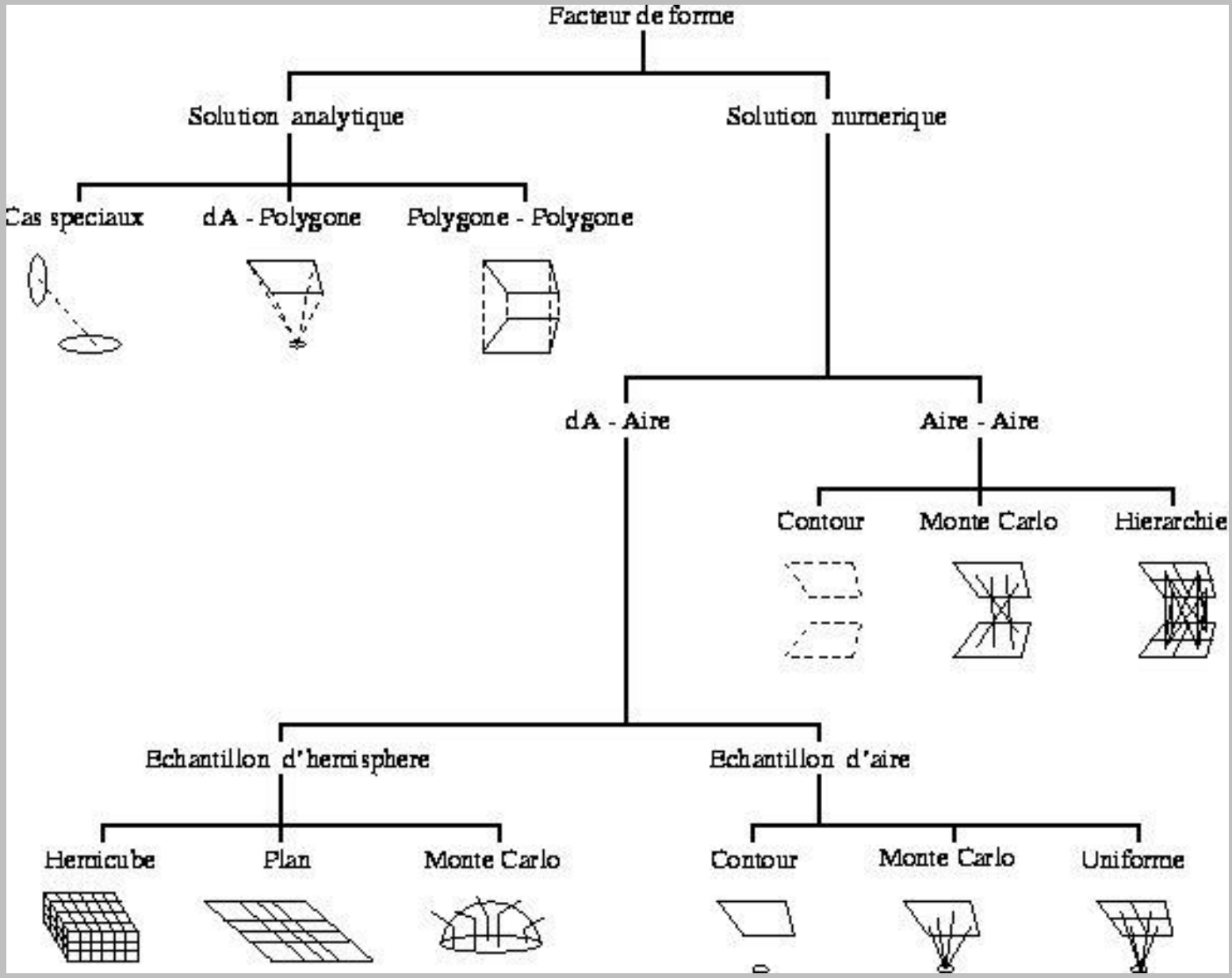
$$B_j = E_j + \rho_j \sum_{i \neq j} B_i F_{ij}$$

- Radiosité

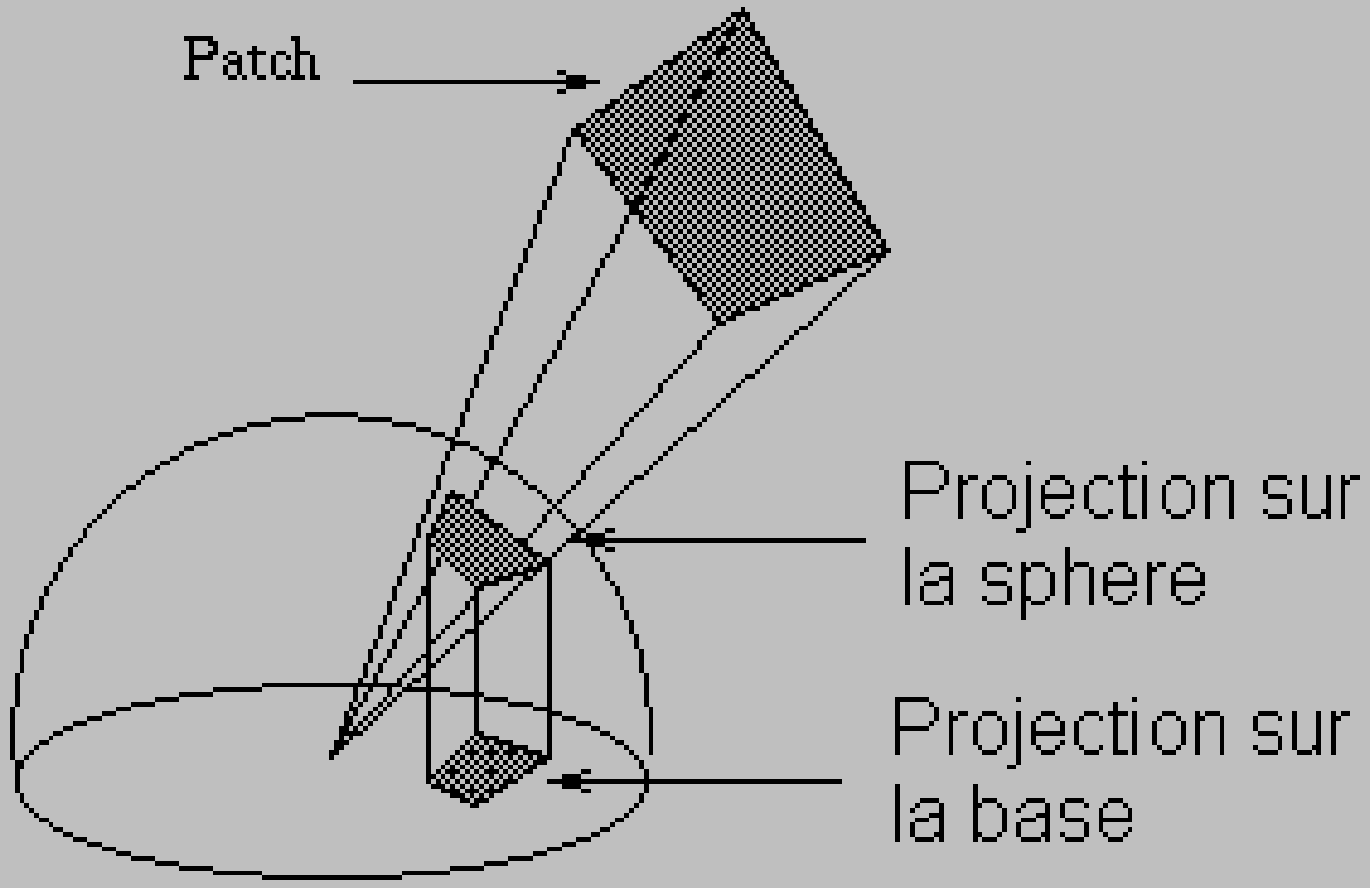
$$[M]\times[B]=[E]$$

$$\begin{bmatrix} 1 & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1 & \cdots & -\rho_2 F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_n F_{n1} & -\rho_1 F_{n2} & \cdots & 1 \end{bmatrix} \times \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix}$$

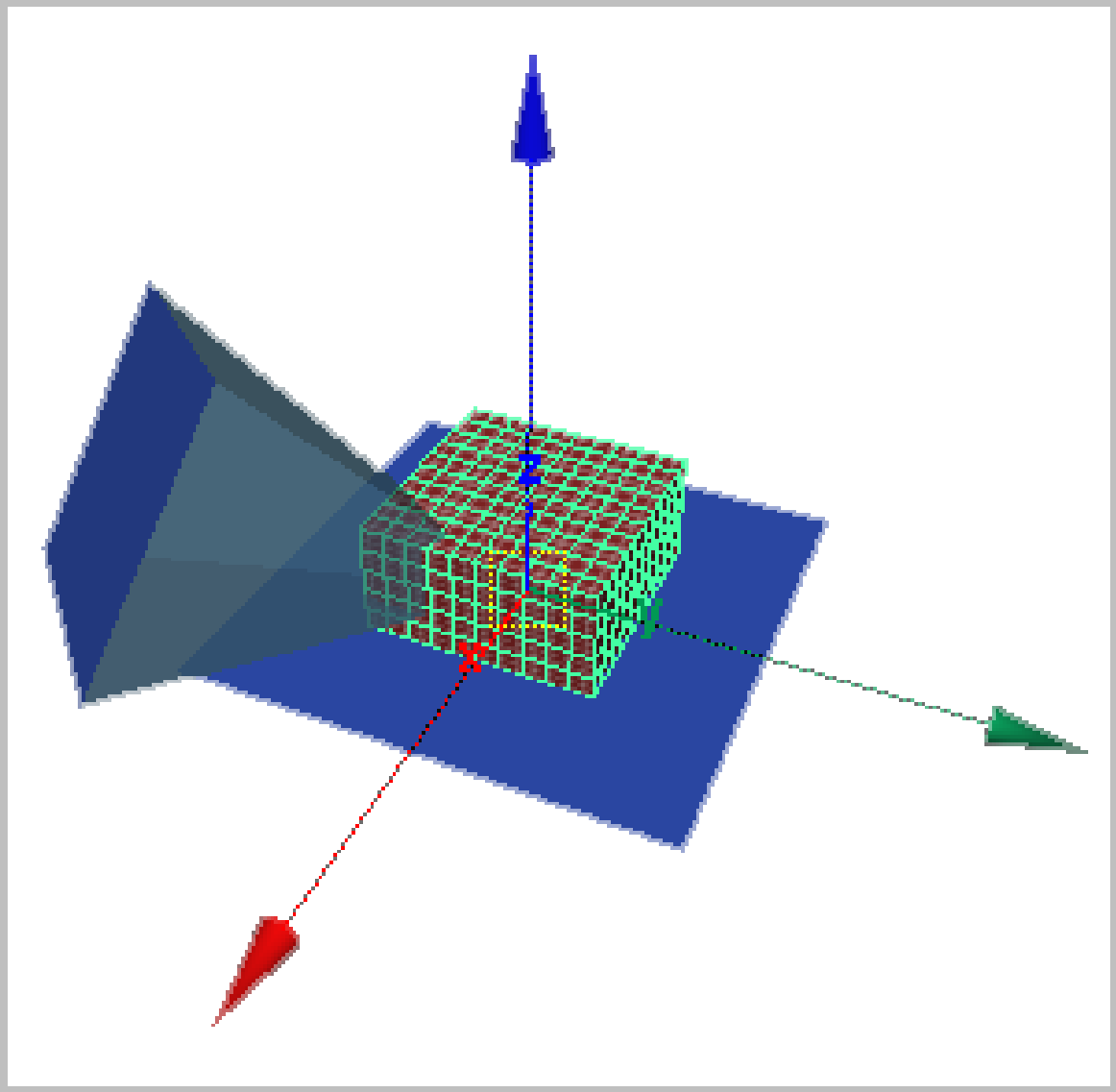
- Radiosité


Facteur de forme
Solution analytique
Solution numerique
Cas speciaux
dA - Polygone
Polygone - Polygone
dA - Aire
Aire - Aire
Contour
Monte Carlo
Hierarchie
Echantillon d'hemisphere
Echantillon d'aire
Hemicube
Plan
Monte Carlo
Contour
Monte Carlo
Uniforme

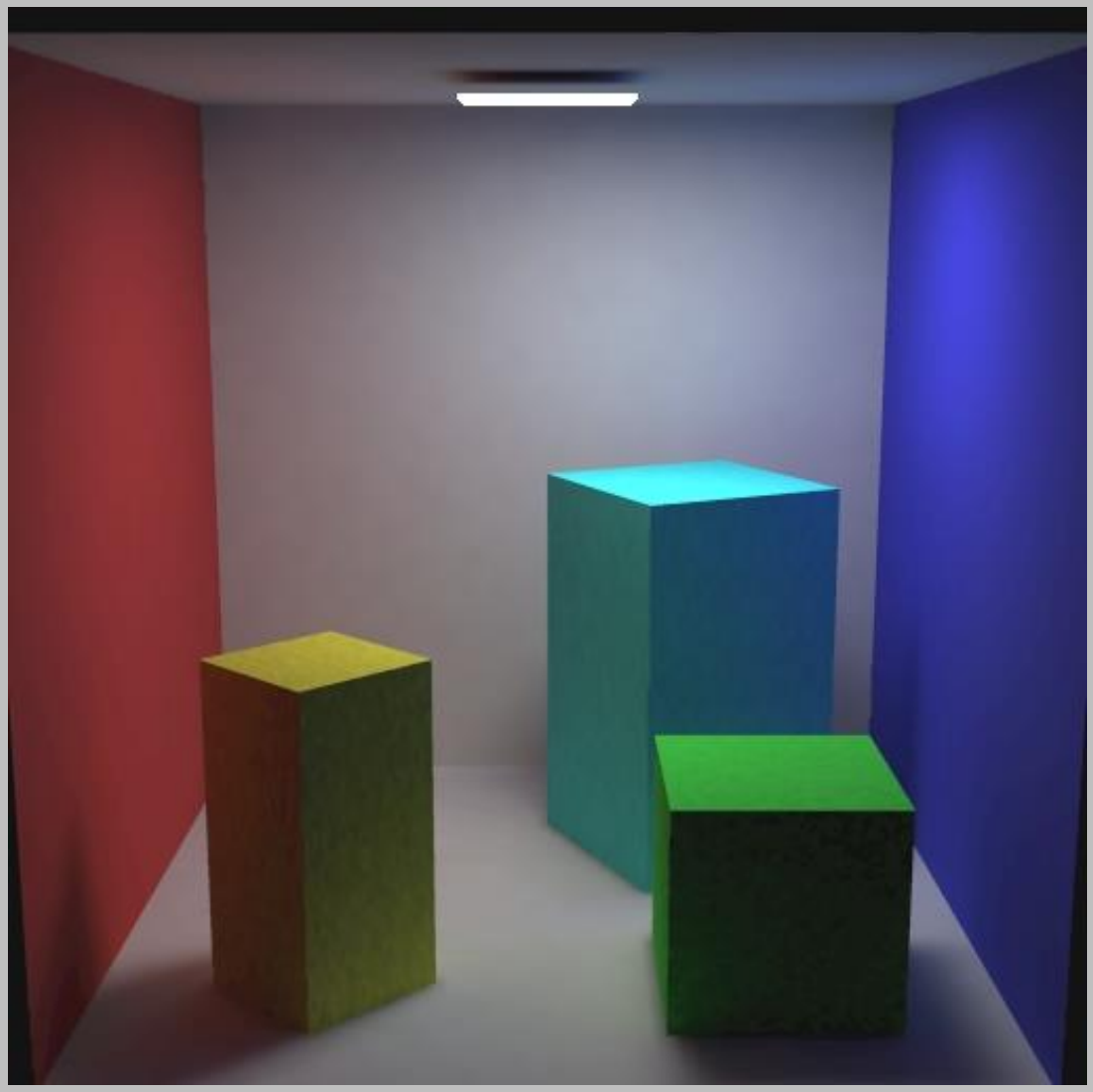
- Radiosité



- Radiosité

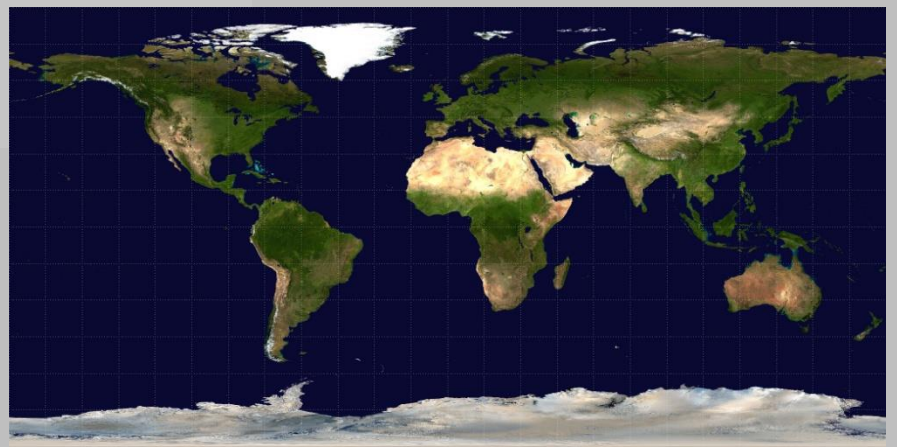


- Radiosité

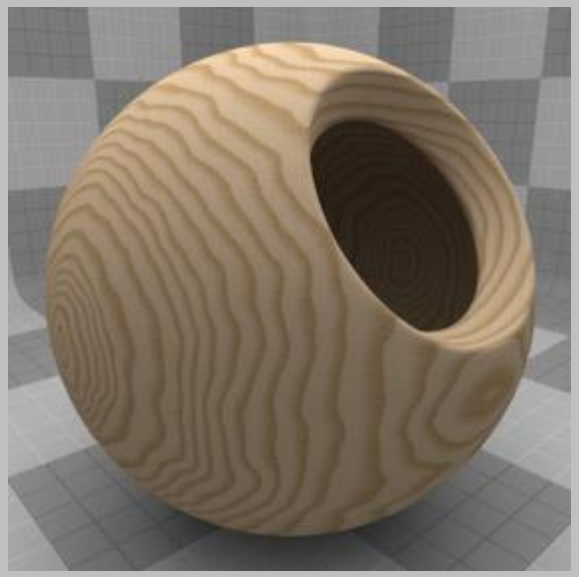


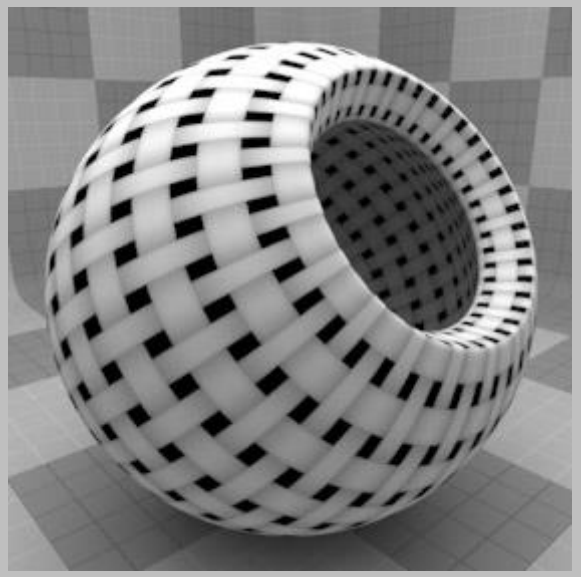
Textures

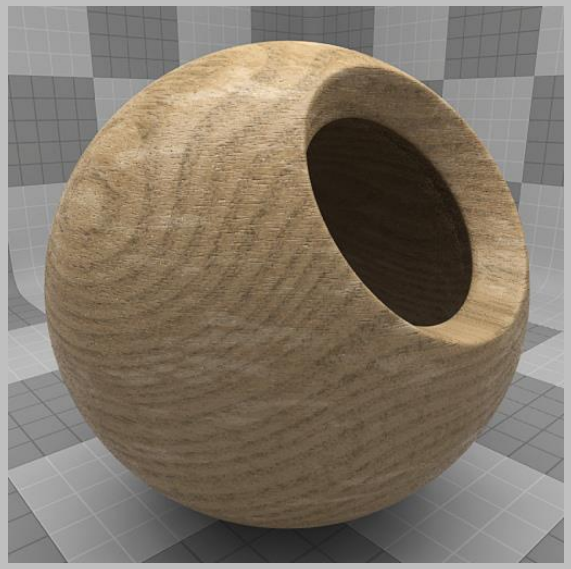
- Texture bitmap (kd)



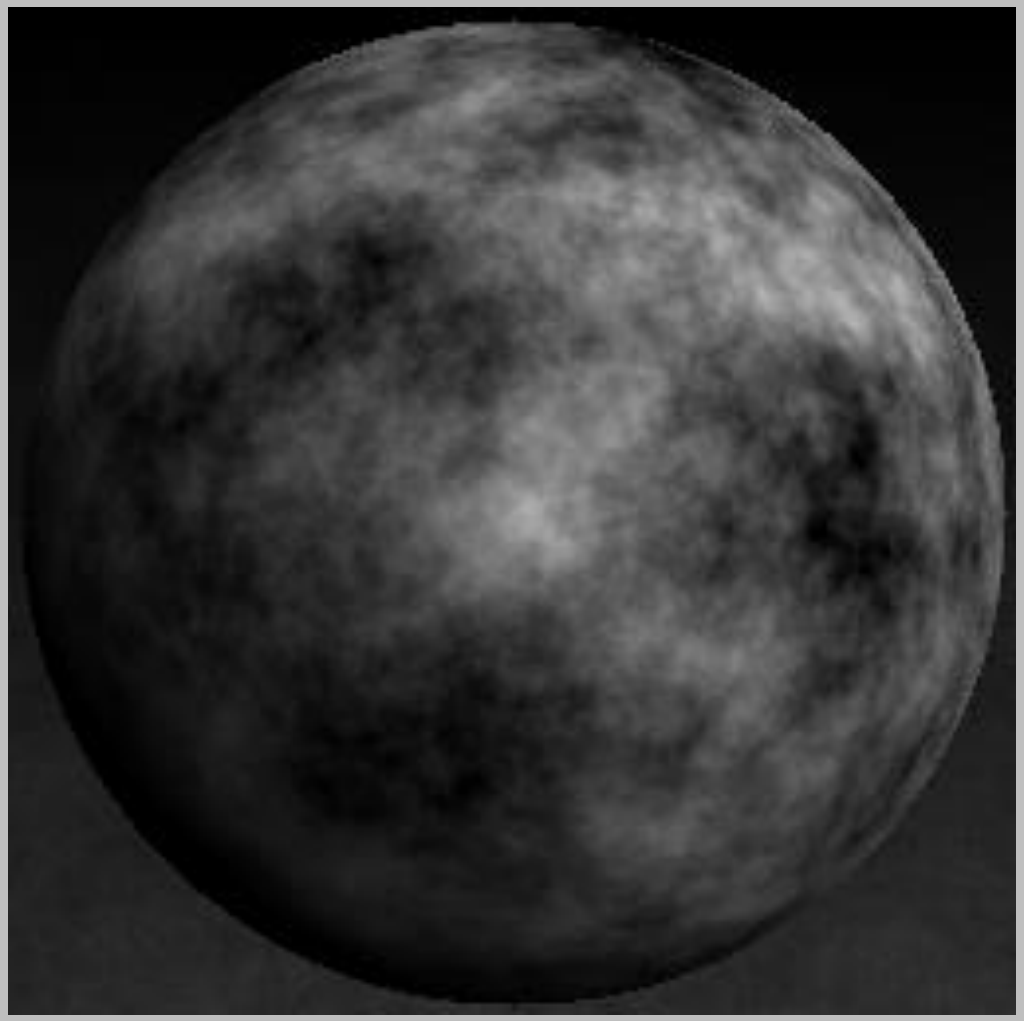
- Texture procédural (kd)



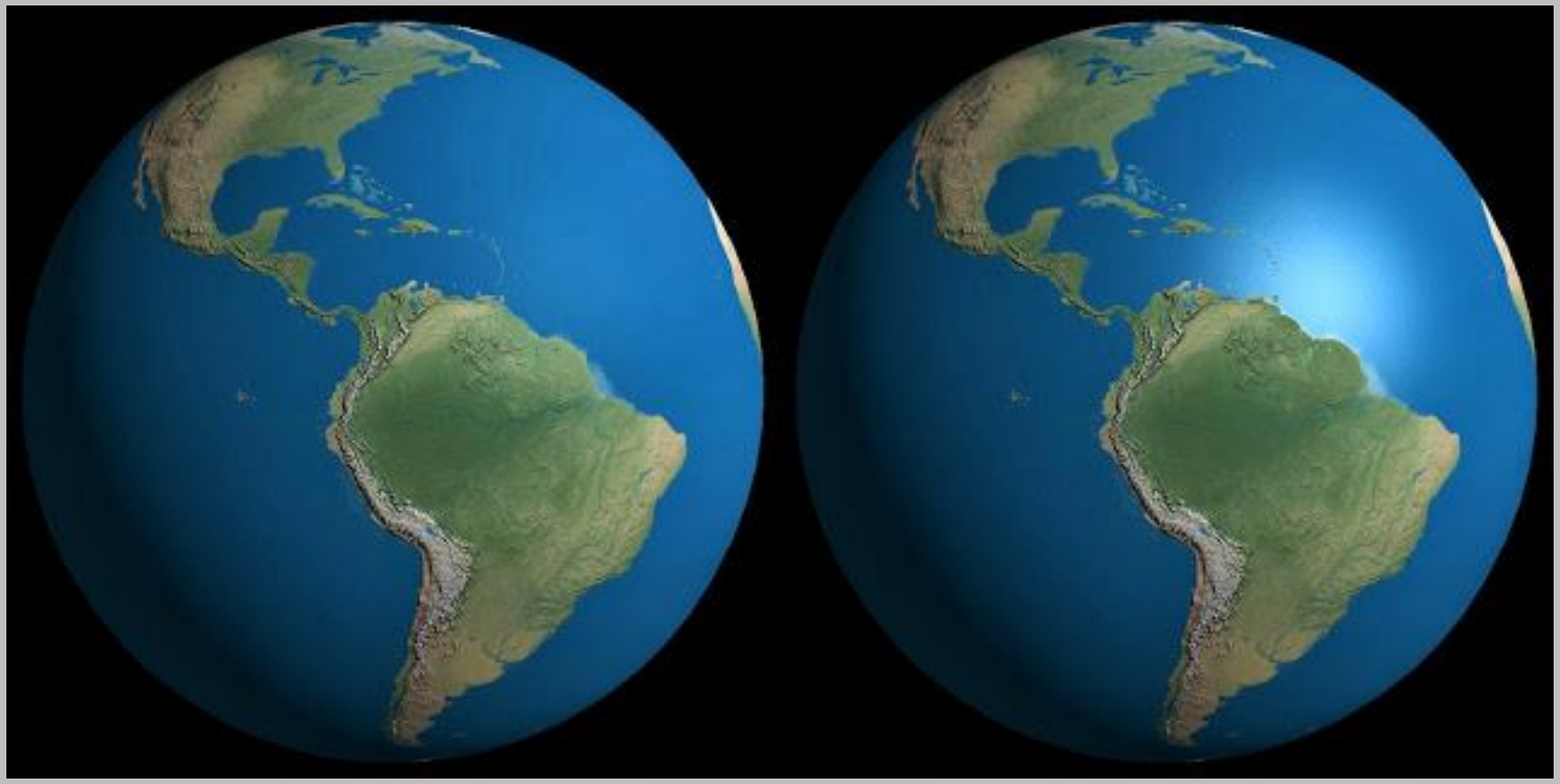




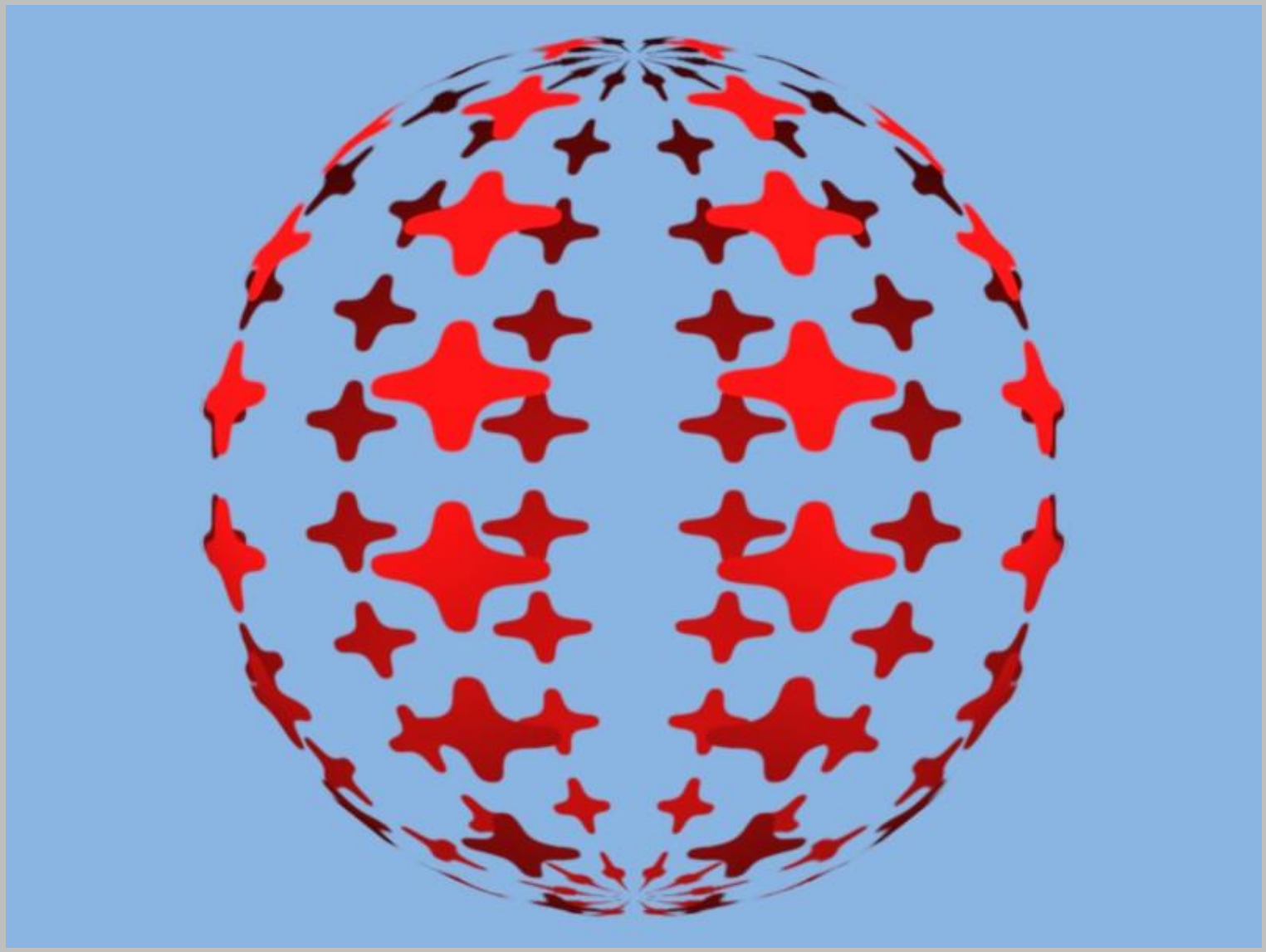
- Texture noise (Kd)



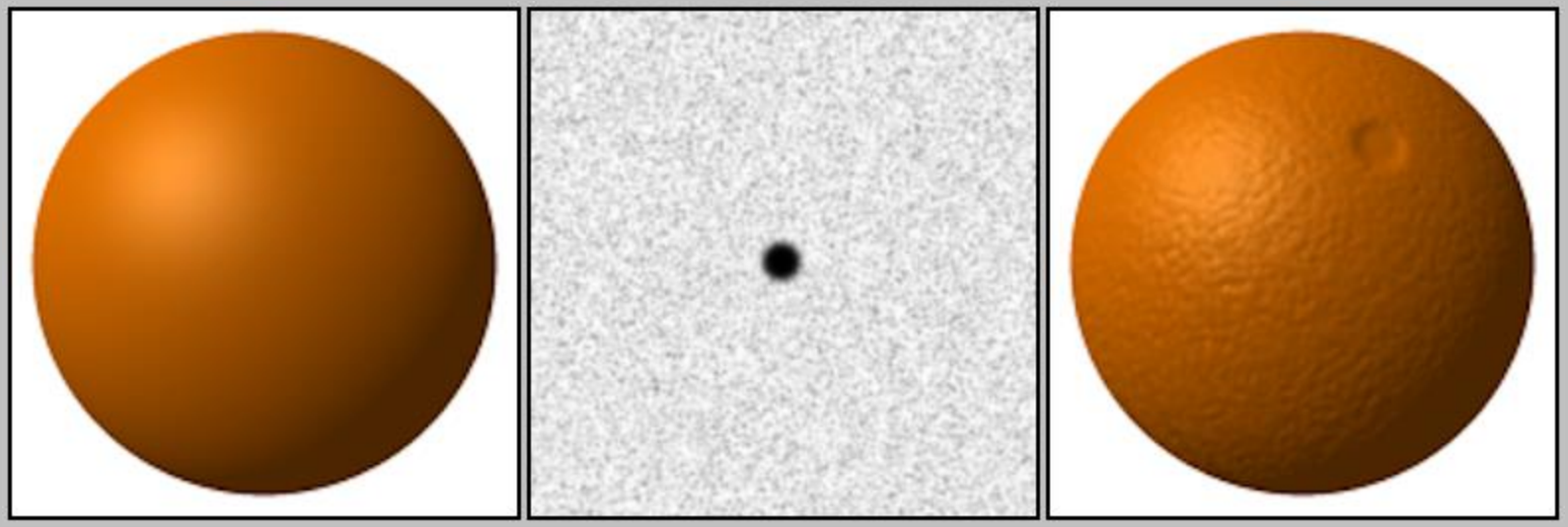
- Texture (ks)



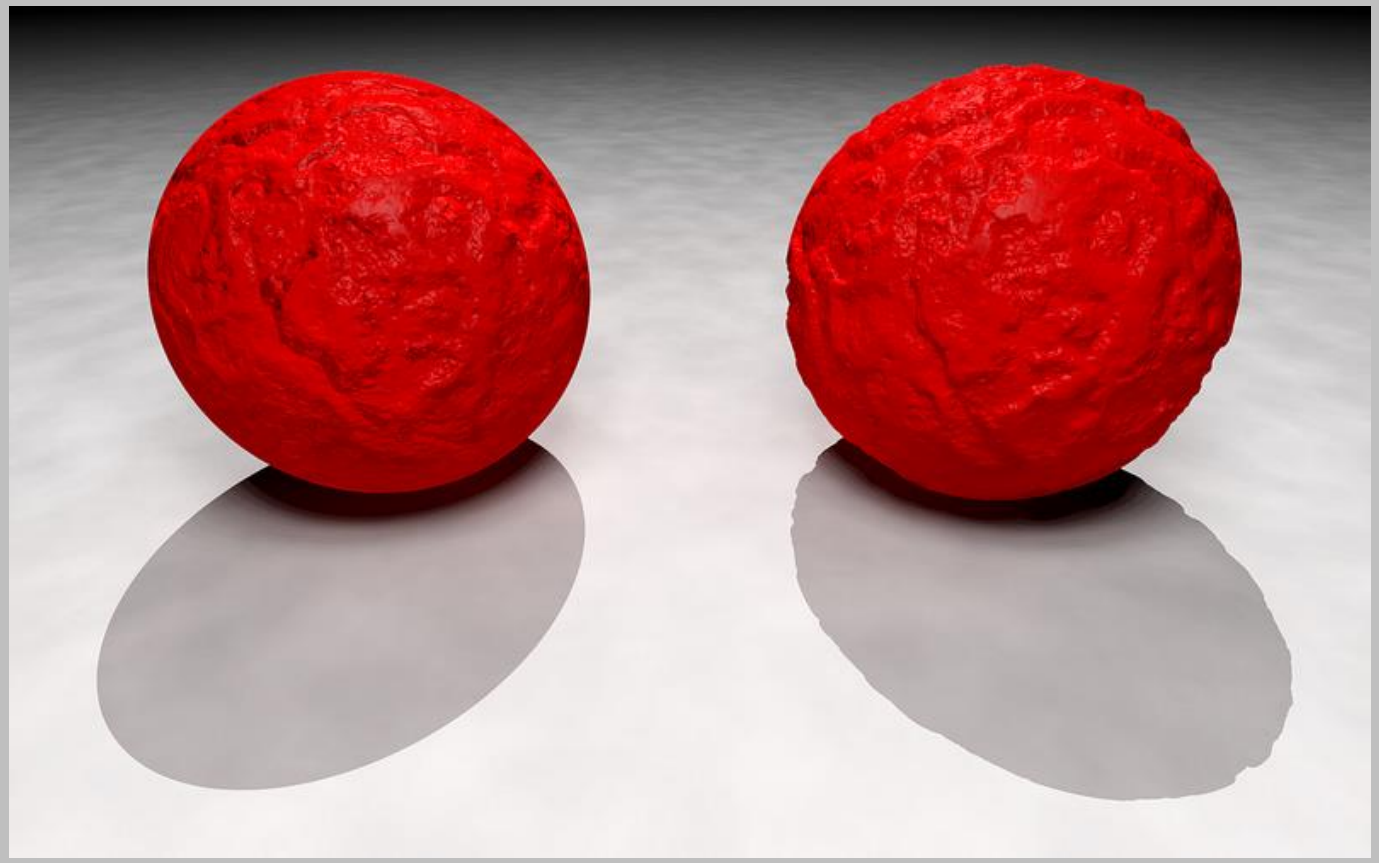
- Texture (alpha)



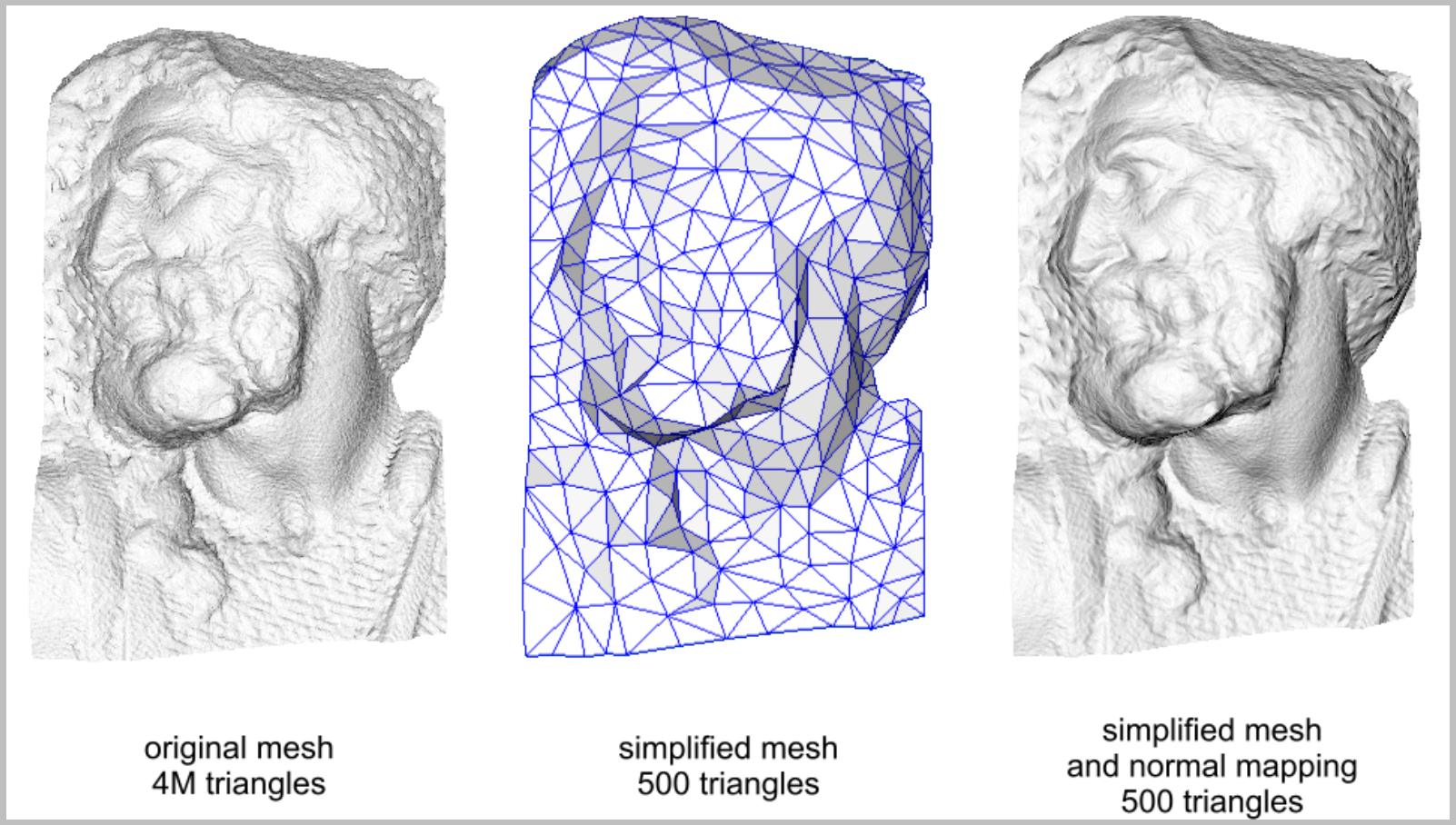
- Bump mapping



- Bump mapping vs Sub Polygon Displacement

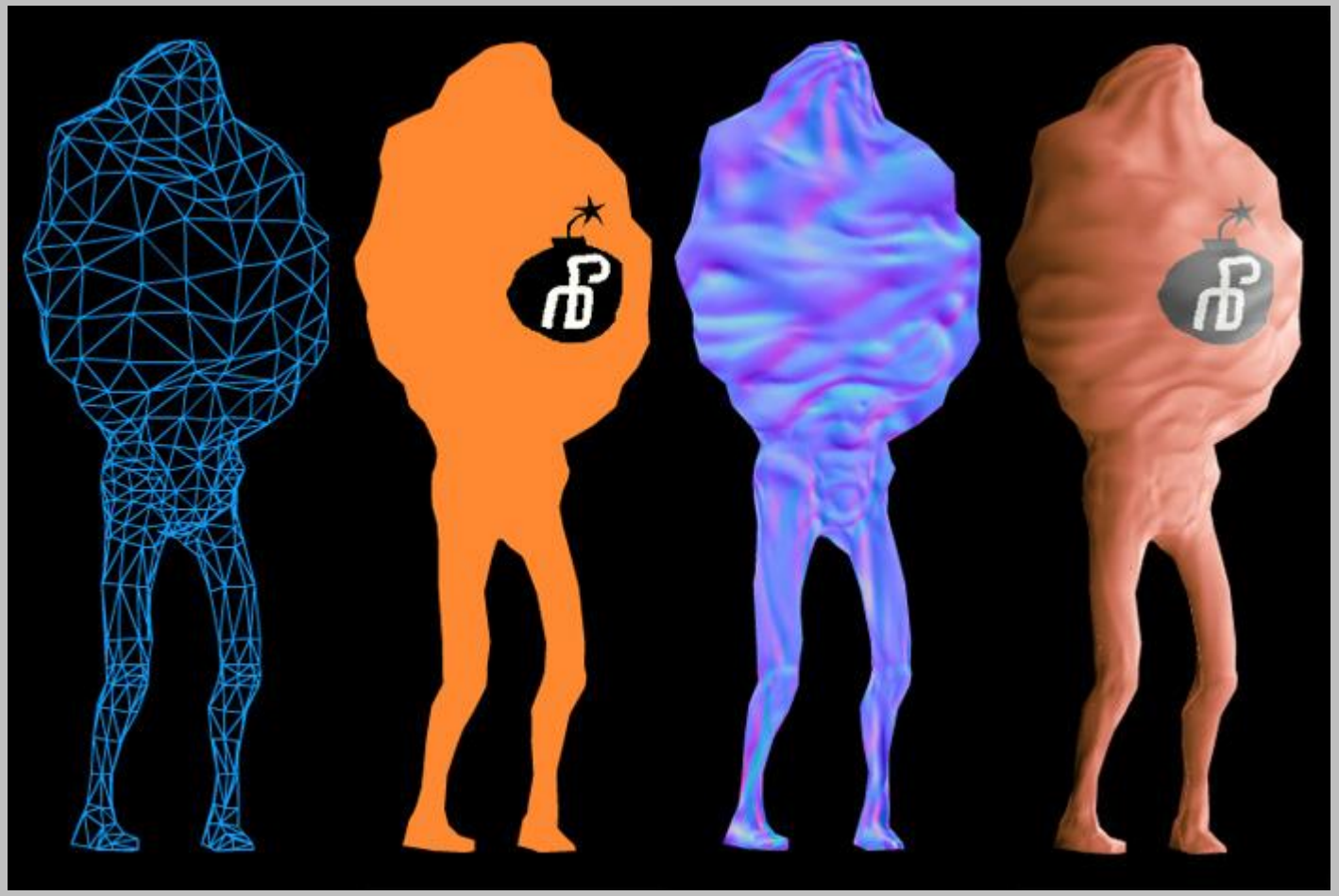


- Normal Mapping

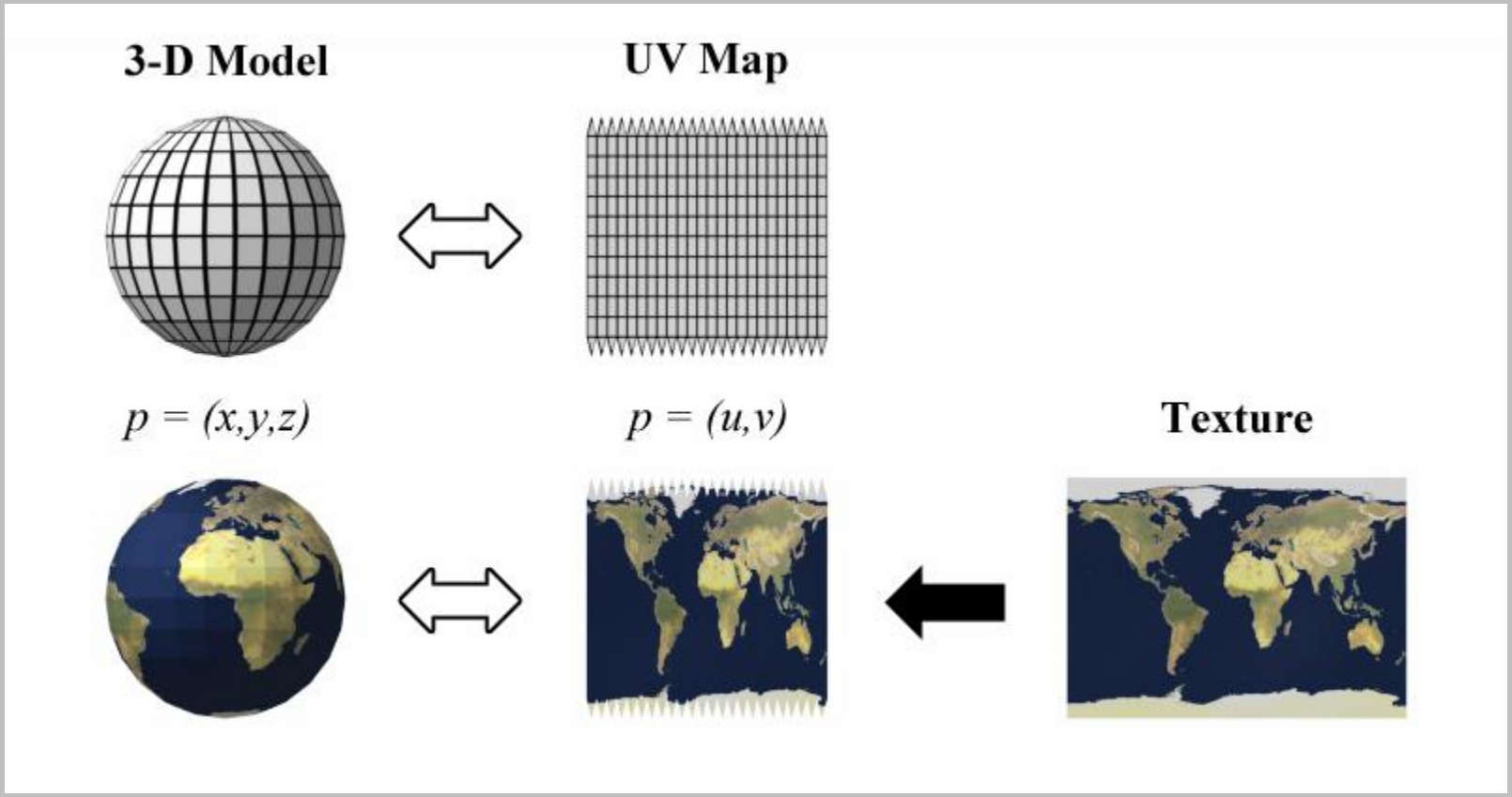


- Normal Mapping

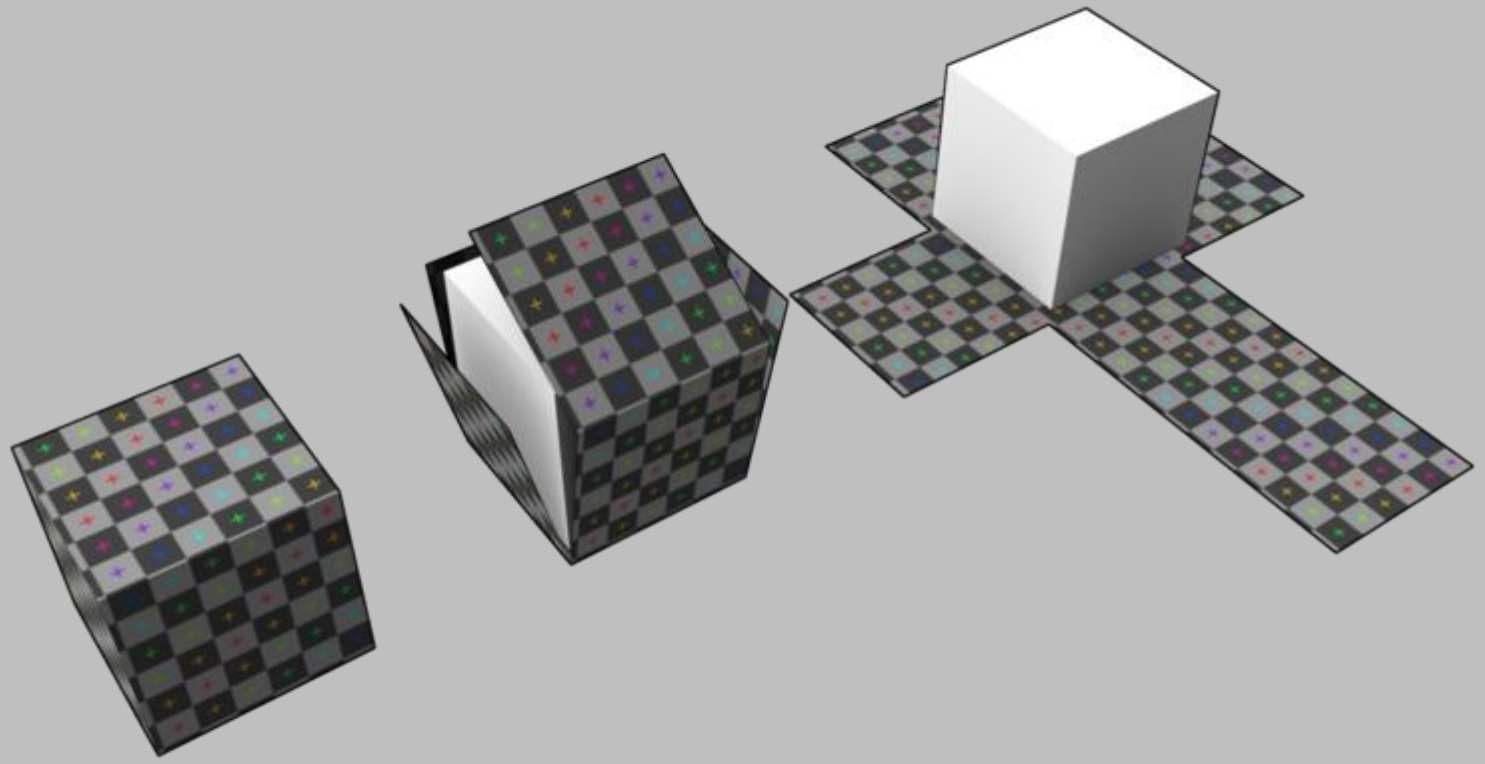
- Normal Mapping



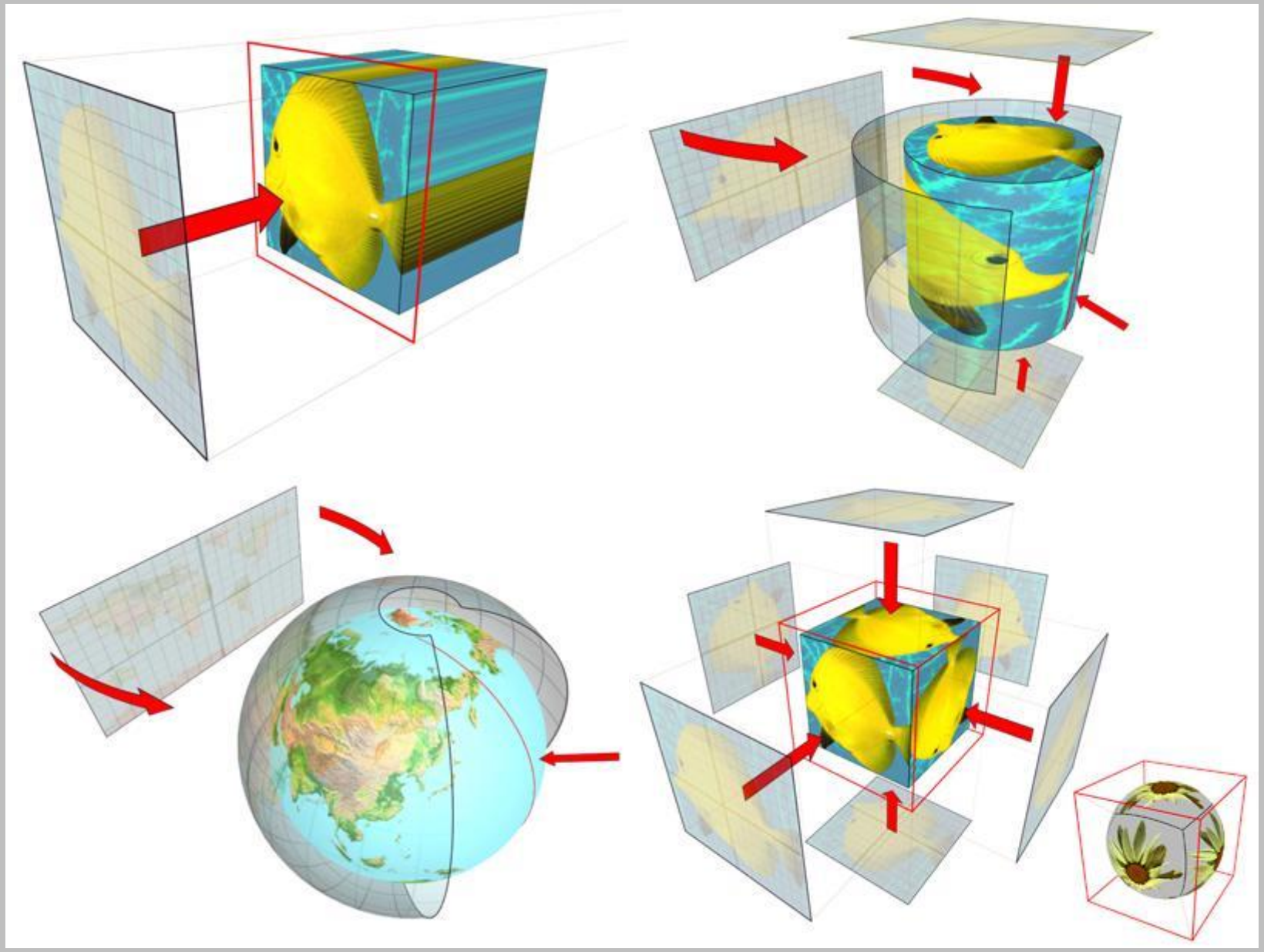
- UV Mapping



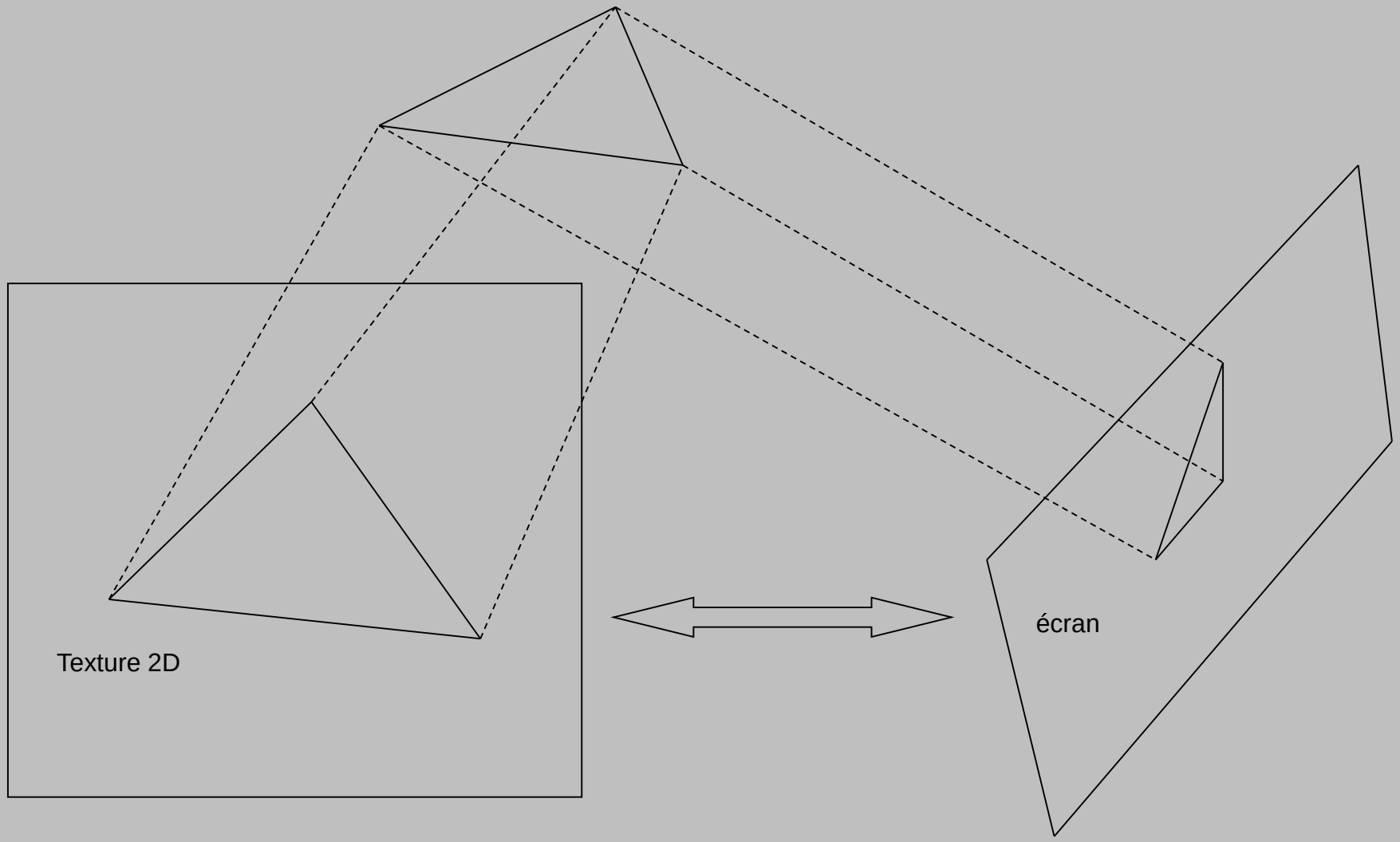
- UV Mapping



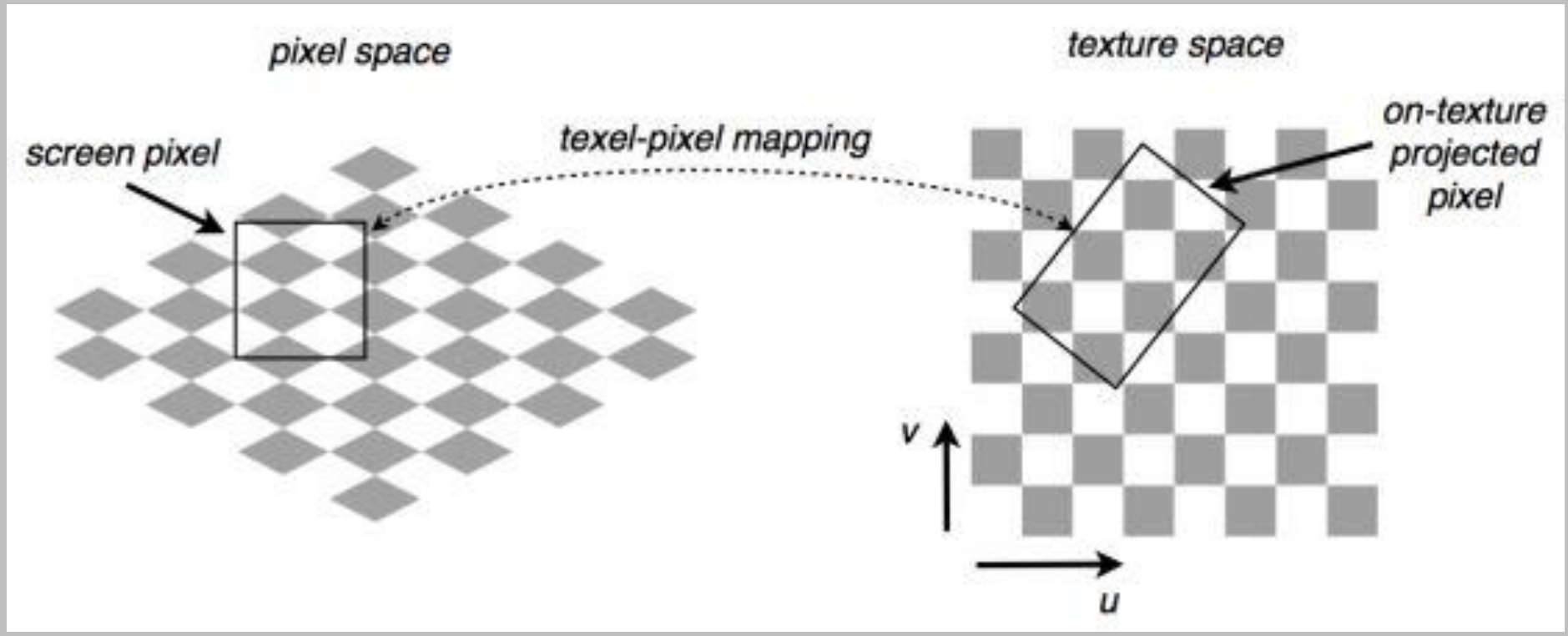
- UV Mapping



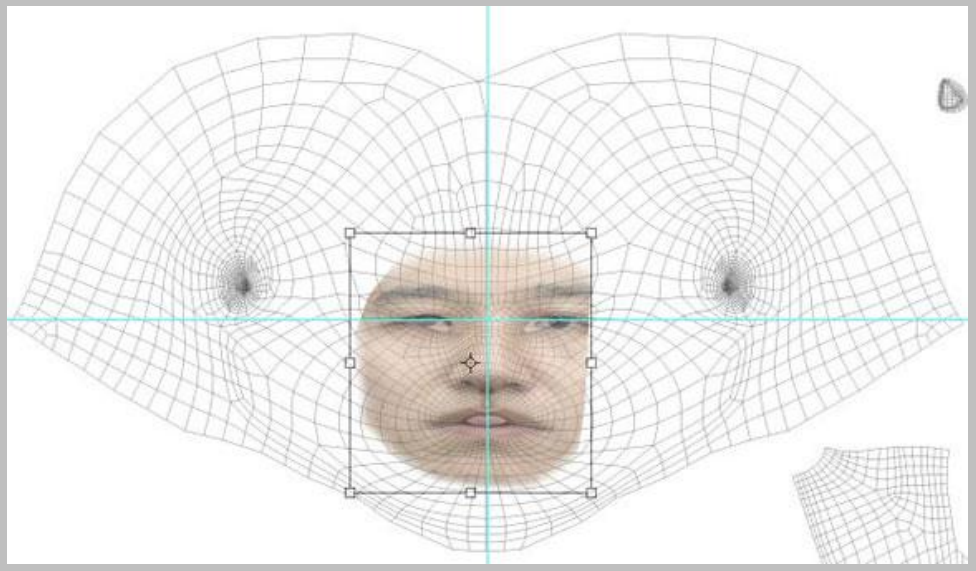
- UV Mapping



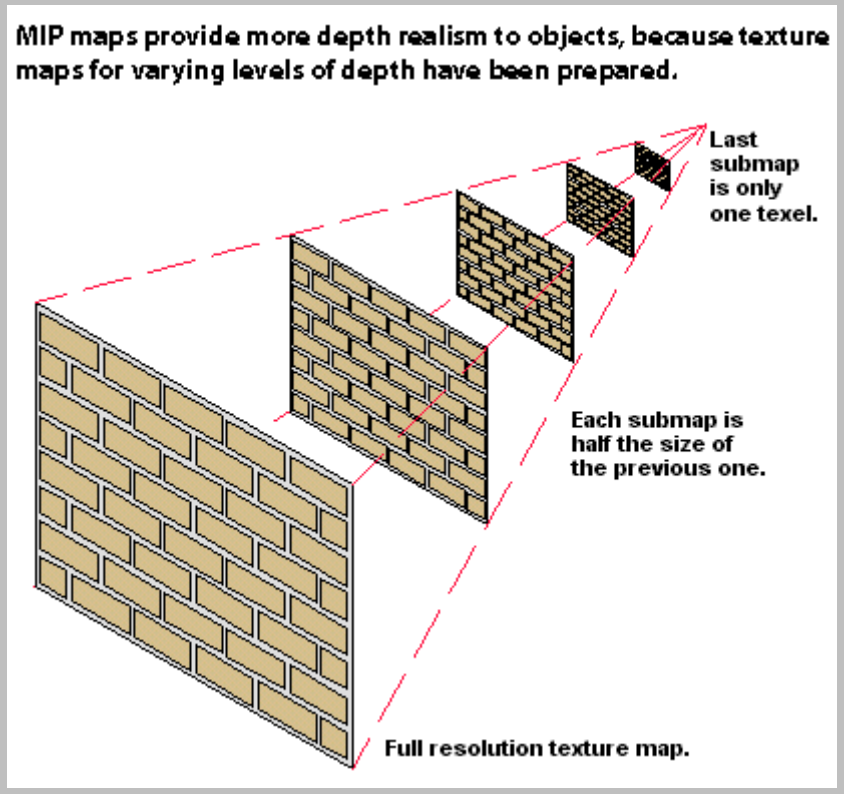
- UV Mapping

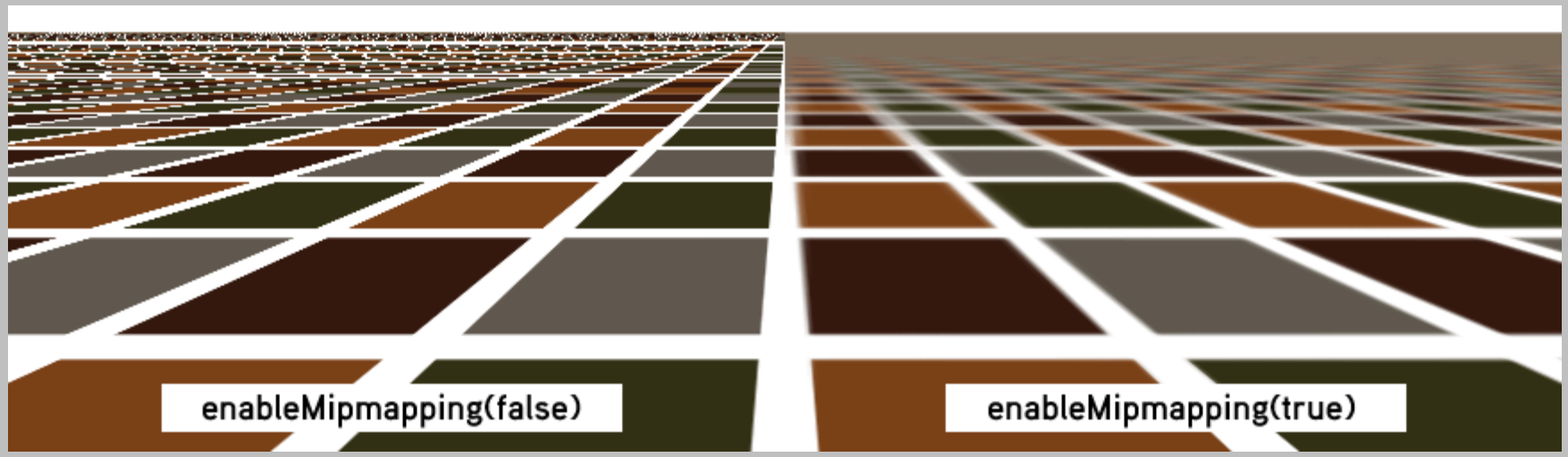

pixel space
texture space
screen pixel
texel-pixel mapping
on-texture
projected
pixel
v
u

- UV Mapping



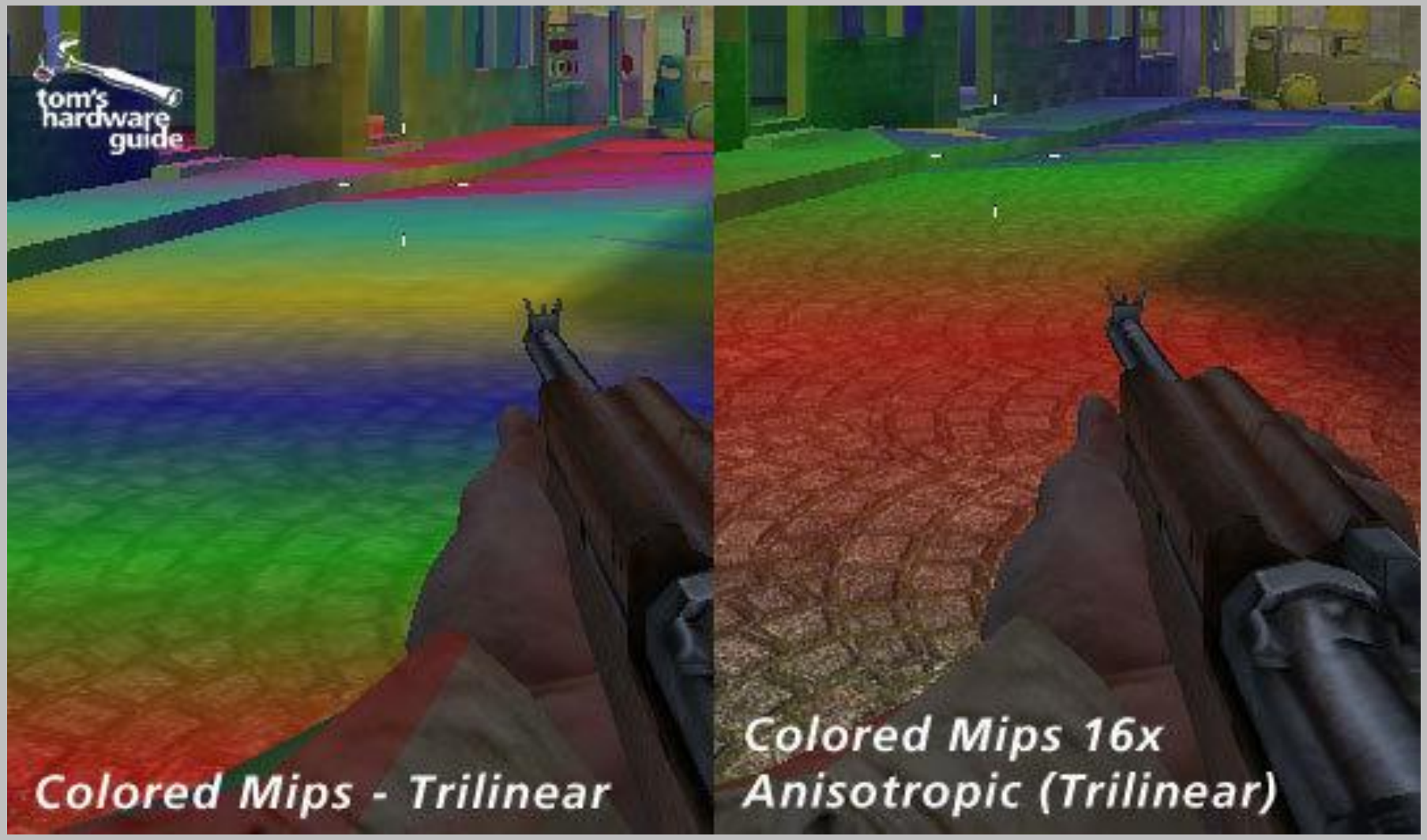
- Mipmap


MIP maps provide more depth realism to objects, because texture maps for varying levels of depth have been prepared.
Last submap is only one texel.
Each submap is half the size of the previous one.
Full resolution texture map.

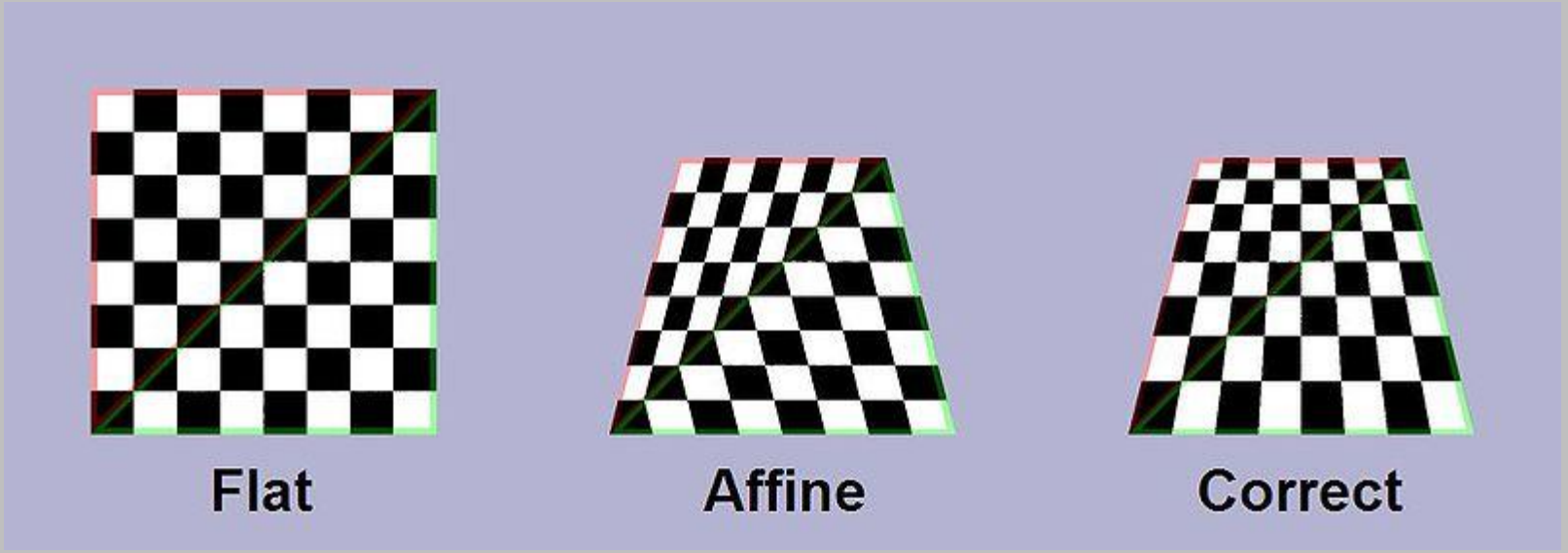

enableMipmapping(false)
enableMipmapping(true)

- Filtrage anisotrope

- Filtrage anisotrope

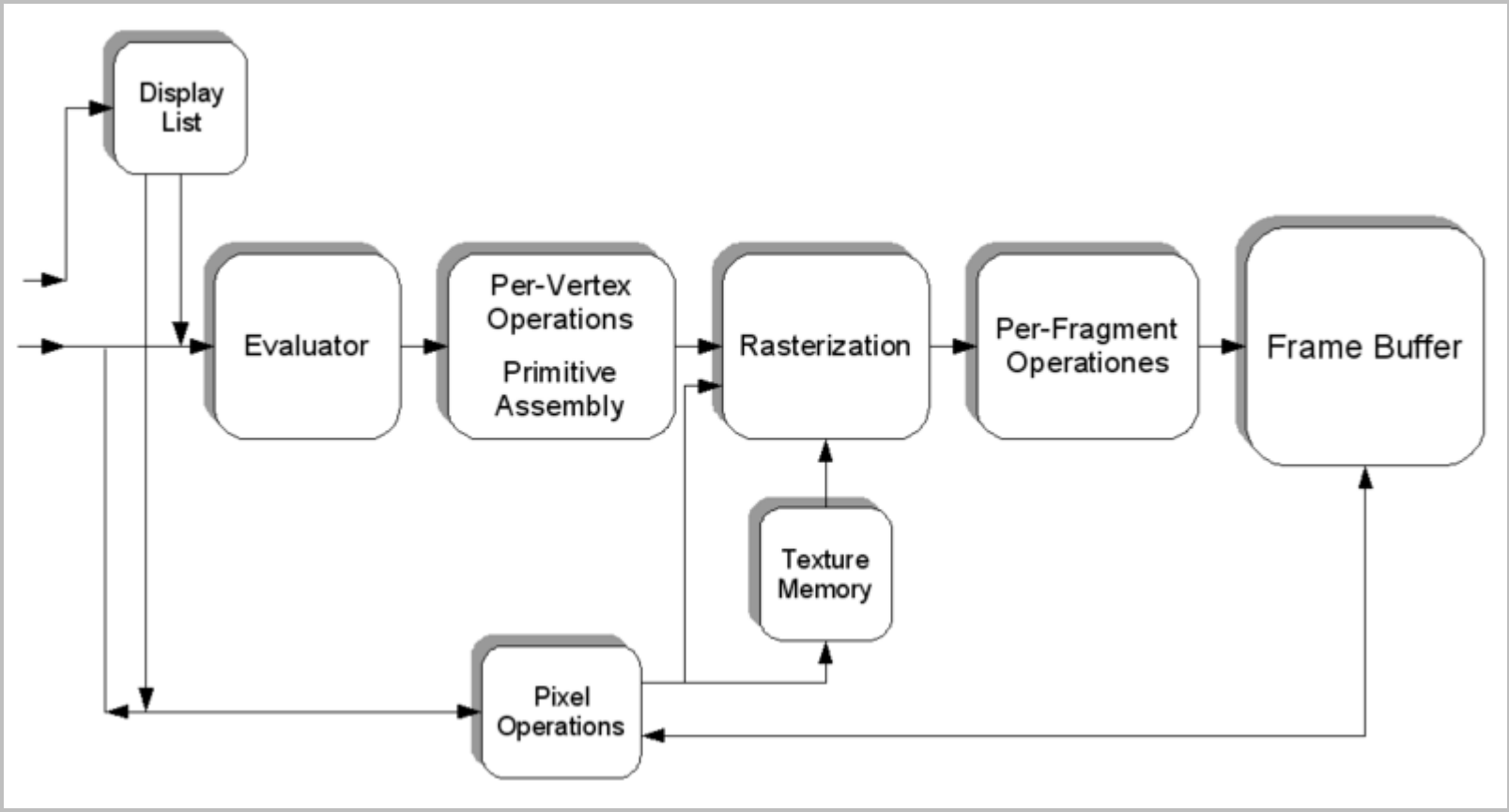


- Affine texture mapping

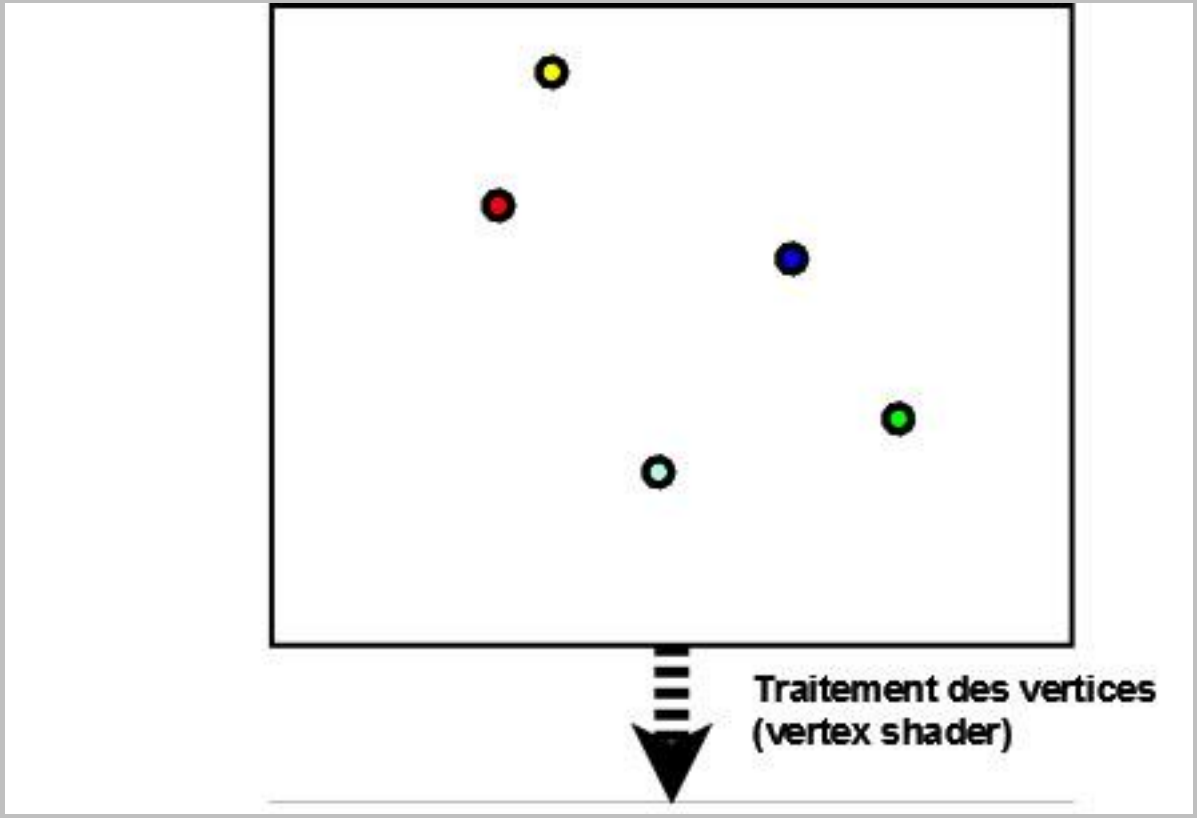


OpenGL

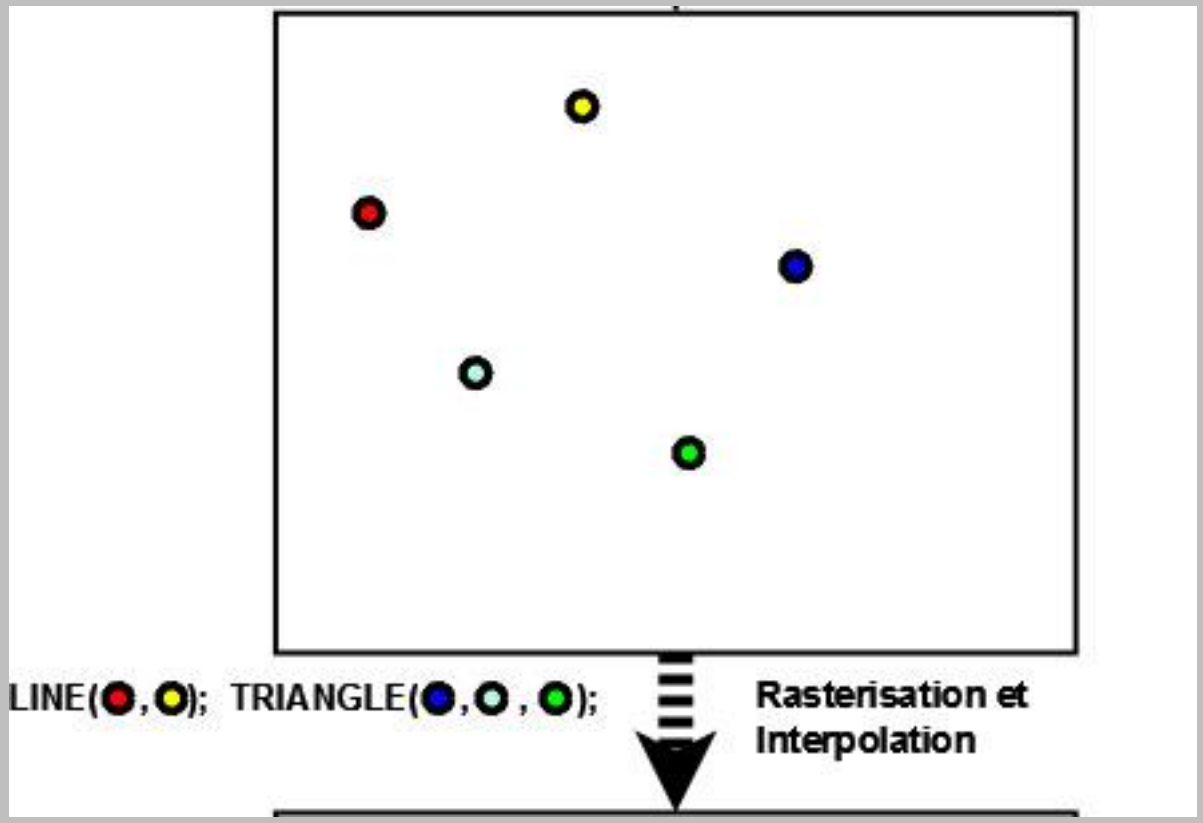
- Pipeline graphique



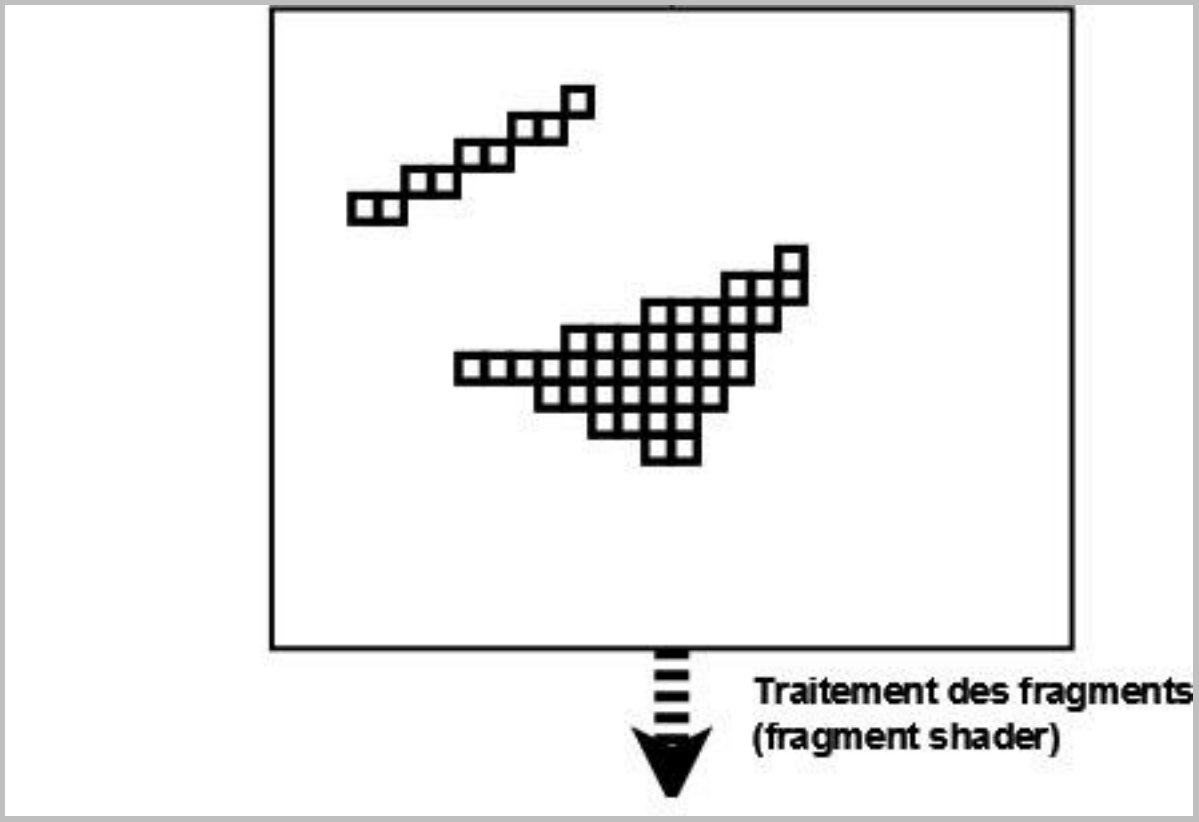
- Pipeline graphique



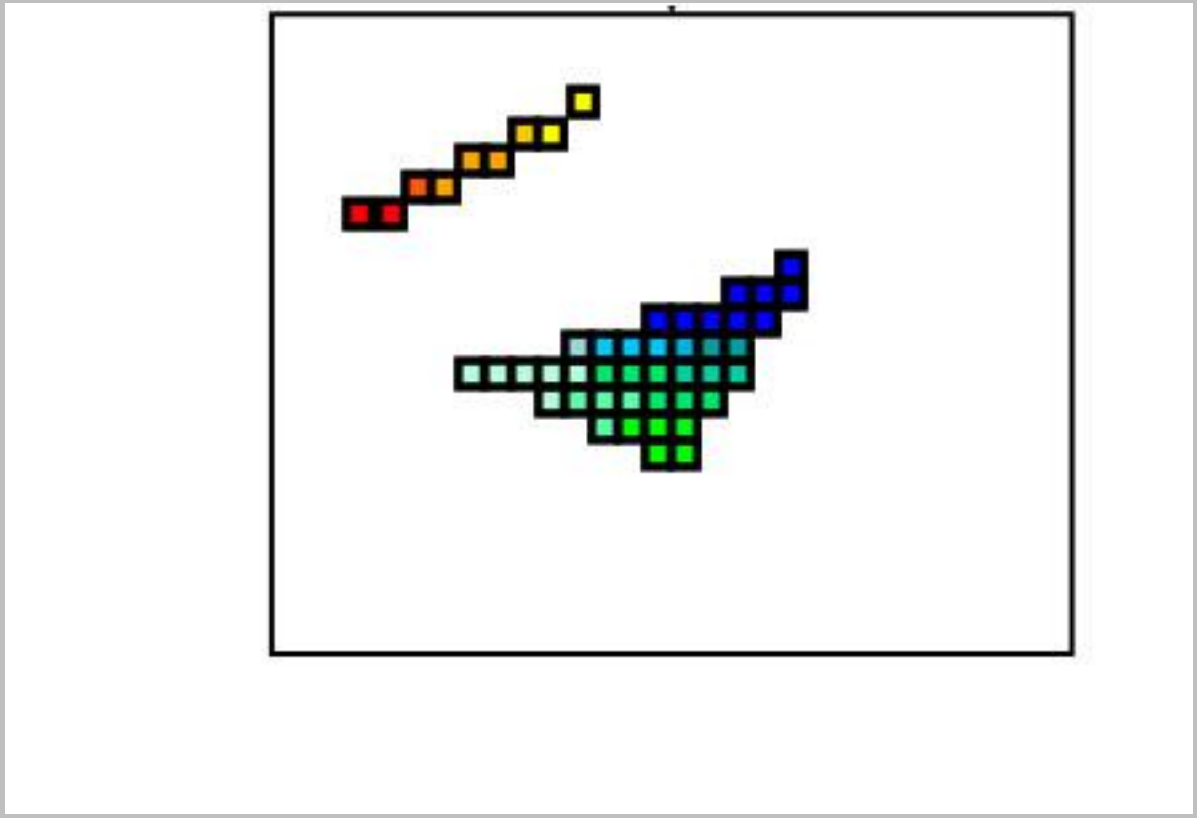
- Pipeline graphique



- Pipeline graphique



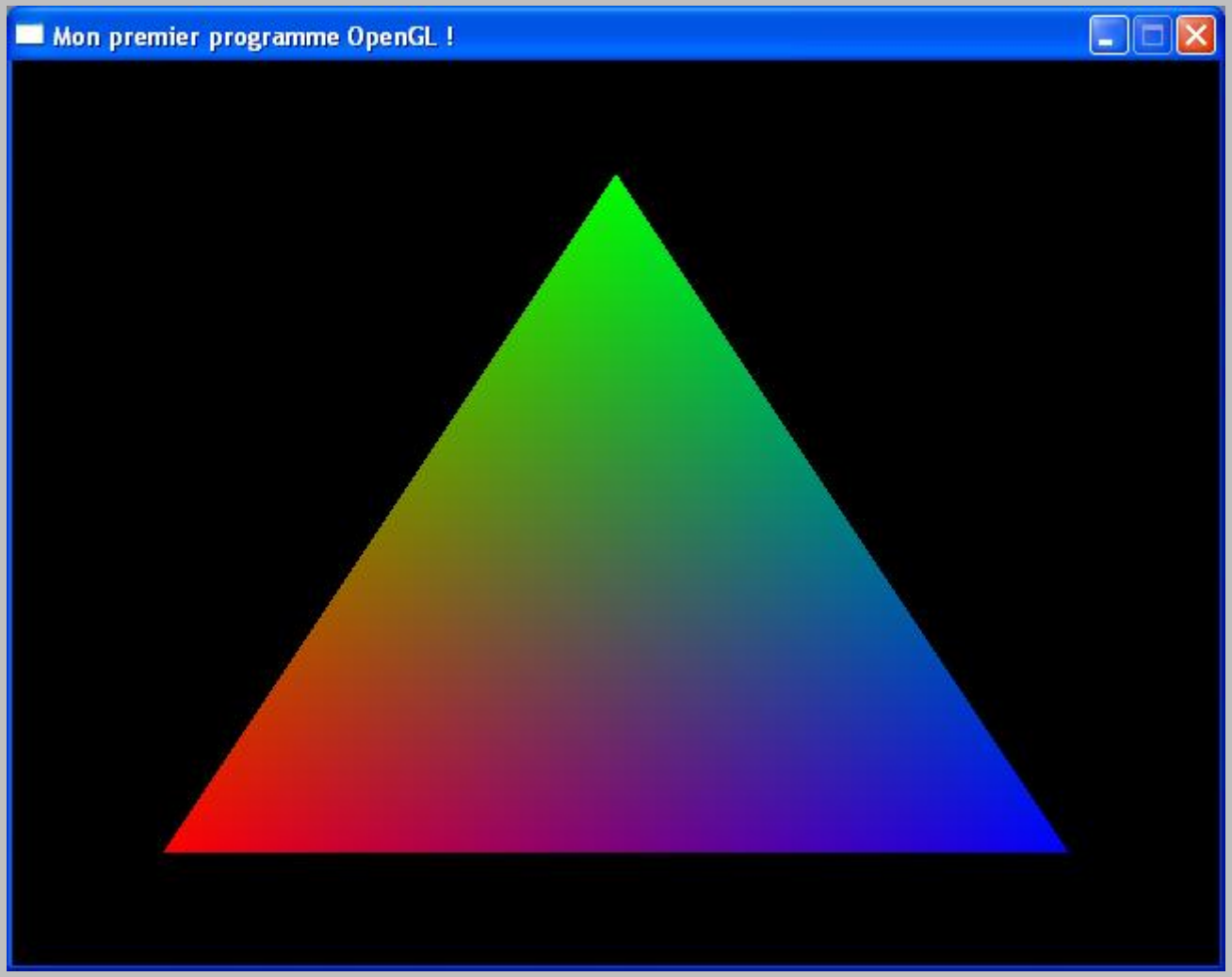
- Pipeline graphique



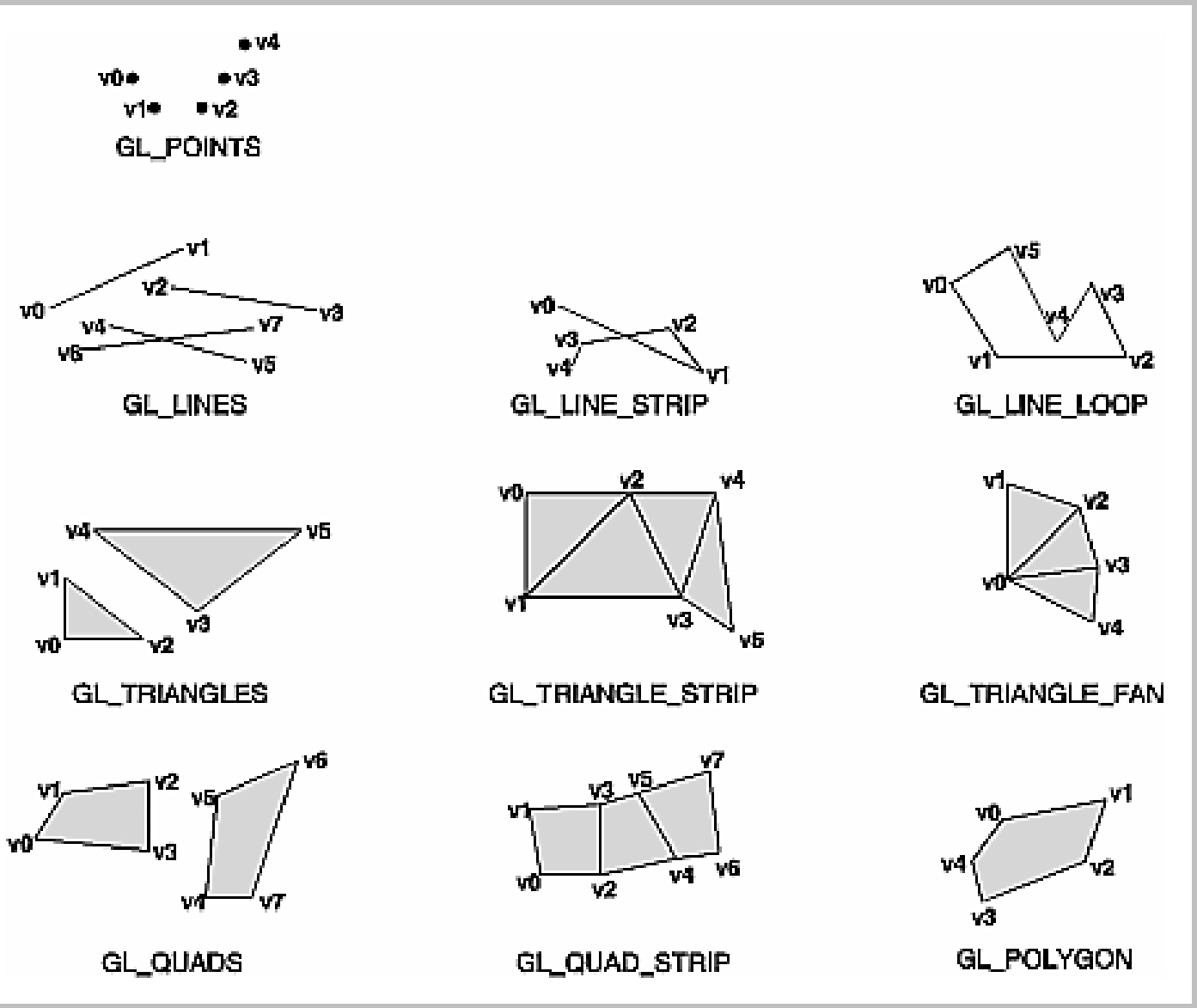
- Exemple

```
glClear(GL_COLOR_BUFFER_BIT);
        glBegin(GL_TRIANGLES);
            glColor3ub(255,0,0);    glVertex2d(-0.75,-0.75);
            glColor3ub(0,255,0);    glVertex2d(0,0.75);
            glColor3ub(0,0,255);    glVertex2d(0.75,-0.75);
        glEnd();
```

- Exemple



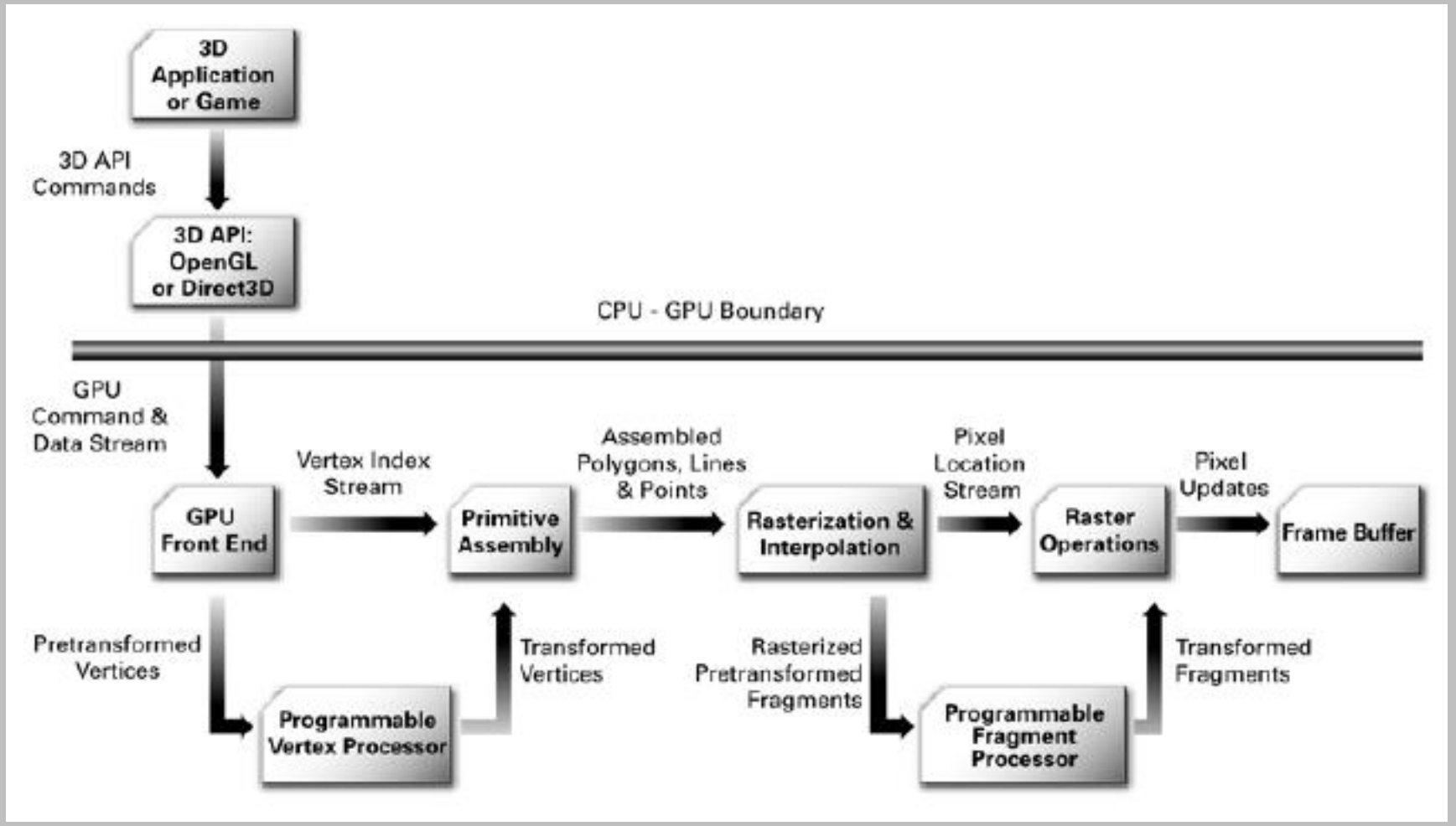
- Types de primitives



- Matrices & transformations

```
glMatrixMode(GL_PROJECTION);
gluLookAt(3,2,3,0,0,0,0,1,0);
glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );
glPushMatrix() ;
    glTranslate3f(x,y,z) ;
    glRotate3f(ax,ay,az) ;
    glBegin(…)
                …
    glEnd() ;
glPopMatrix() ;
```

- Shaders

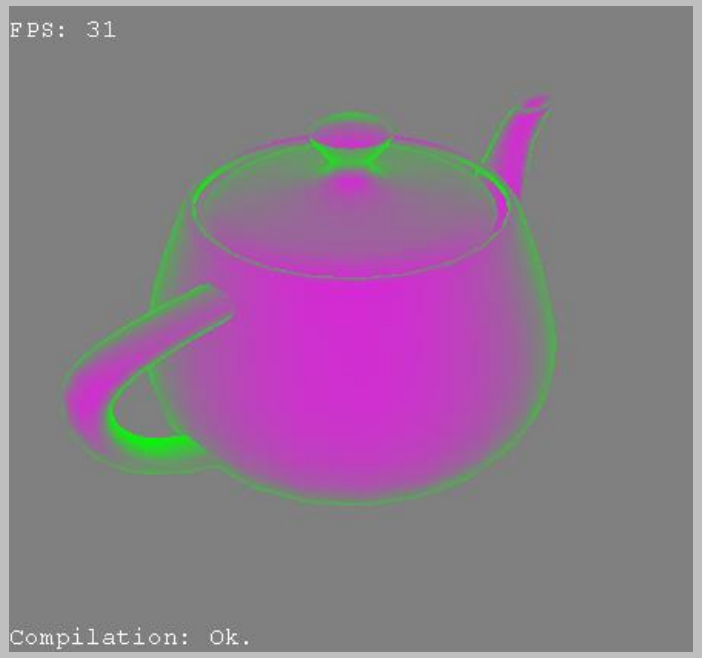

3D Application or Game
3D API Commands
3D API: OpenGL or Direct3D
CPU - GPU Boundary
GPU Command & Data Stream
GPU Front End
Vertex Index Stream
Primitive Assembly
Assembled Polygons, Lines & Points
Rasterization & Interpolation
Pixel Location Stream
Raster Operations
Pixel Updates
Frame Buffer
Pretransformed Vertices
Programmable Vertex Processor
Transformed Vertices
Rasterized Pretransformed Fragments
Programmable Fragment Processor
Transformed Fragments

- Fragment shader

```
void main()
{
  gl_FragColor = vec4(1.0, 0.6, 0.0, 1.0);
}
```

- Fragment shader

```
void main()
{
  gl_FragColor = gl_Color;
  gl_FragColor[1] = 1.0 - gl_Color[1];
}
```



- Vertex shader

```
void main()
{
  vec4 coords = gl_Vertex;
  coords.z = coords.z * 0.25;
  gl_Position = gl_ModelViewProjectionMatrix * coords;
}
```

- Vertex shader (diffus)

```
varying vec3 normale;
varying vec3 lumiere;
void main()
{
  gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
  normale = normalize(gl_NormalMatrix * gl_Normal);
  vec4 pospoint = gl_ModelViewMatrix * gl_Vertex;
  vec4 poslum = gl_LightSource[0].position;
  lumiere = vec3(poslum - pospoint);
}
```

- Fragment shader (diffus)

```
varying vec3 normale;
varying vec3 lumiere;
void main()
{
  gl_FragColor=vec3(0,0,0);
  const vec4 Kd = vec4(0.8, 0.4, 0.0, 1.0);
  float NL = dot(normalize(normale), normalize(lumiere));
  if (NL > 0) {
     gl_FragColor = Kd * NL;
  }
}
```


FPS: 31
Compilation: Ok.


Activision R&D
Property of Activision Publishing
Not actual gameplay

FIN