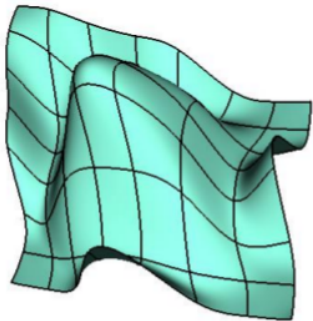
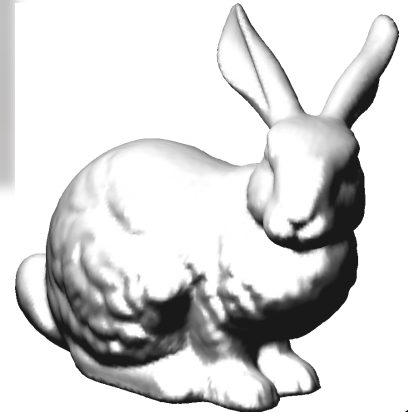


# Synthèse d'images *Computer Graphics*



CPEyon



# Synthèse d'images: C'est quoi ?

Modèle 3D

Image

*synthèse*

ex 1.

$$x^2 + y^2 + z^2 = 0$$



ex 2.



```
106659 23.9057 27.2467
10718 23.9456 27.2816
932381 23.9284 27.2213
163475 23.8595 27.1998
227447 23.9242 27.2155
v 0.227848 23.962 27.2504
v 0.164118 23.911 27.2041
v 0.308776 23.8559 27.1688
v 0.388002 23.9374 27.1731
v 0.389372 23.975 27.208
v 0.308692 23.9072 27.1733
v 0.418315 23.8898 27.1214
v 0.559994 23.9844 27.0037
v 0.421987 23.9386 27.1236
v 1.03058 23.6931 26.6911
```



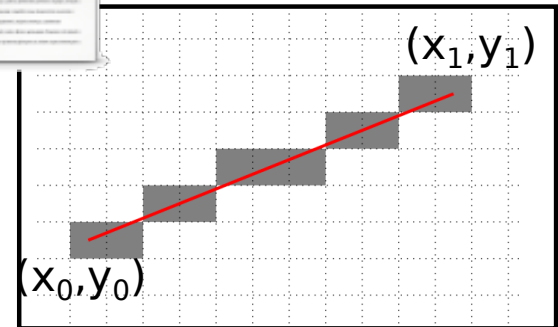
# Partie 1: Tracé de segments discrets

[Cours/TP]

[mon\_image.jpg]

$(x_0, y_0) ; (x_1, y_1)$

??



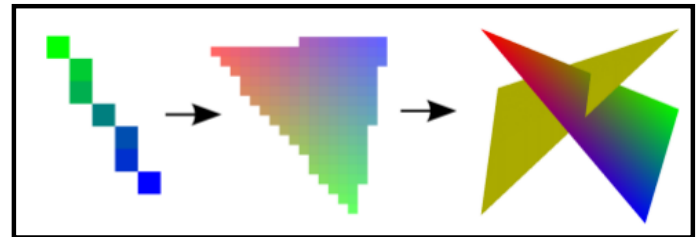
```
void image_io::export_ppm(const std::string& filename, const image&
{
    std::ofstream stream(filename.c_str(), std::ofstream::out);
    if(!stream.good())
        throw exception_image("Error in opening file "+filename+");

    unsigned int Nx=pic.get_Nx();
    unsigned int Ny=pic.get_Ny();

    //magic number
    stream<<"P3 \n";
    //size
    stream<<Nx<<" "<<Ny<<" \n";
    //color number
    stream<<"255"<<std::endl;

    std::stringstream str;
    for(unsigned int k2=0; k2<Nx; ++k2)
        for(unsigned int k1=0; k1<Ny; ++k1)
            str<<pic(p2d(k1, k2))<<std::endl;

    stream<<str.str();
    stream.close();
}
```

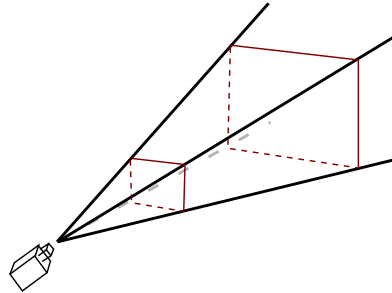


Code [C++]

# Partie 2: Rendu projectif

[Cours/TP]

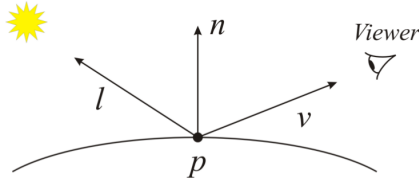
Projection:



$$P = \left( \begin{array}{ccc|c} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ \hline p_{30} & p_{31} & p_{32} & p_{33} \end{array} \right)$$

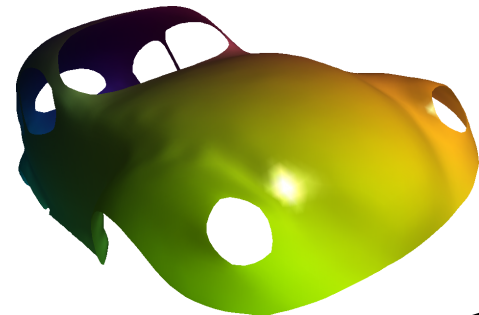
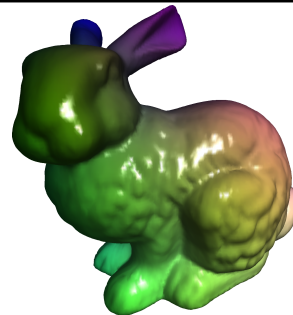
Illumination:

Light source



$$a_a \text{ (red sphere)} + a_d \text{ (shaded sphere)} + a_s \text{ (black sphere)} = \text{ (red sphere with highlight)}$$

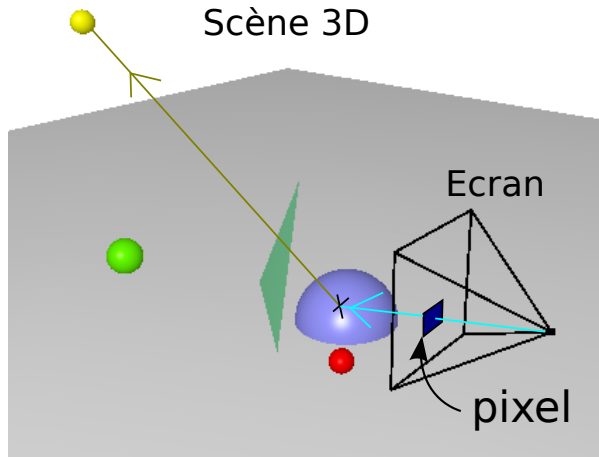
=> Maillages 3D



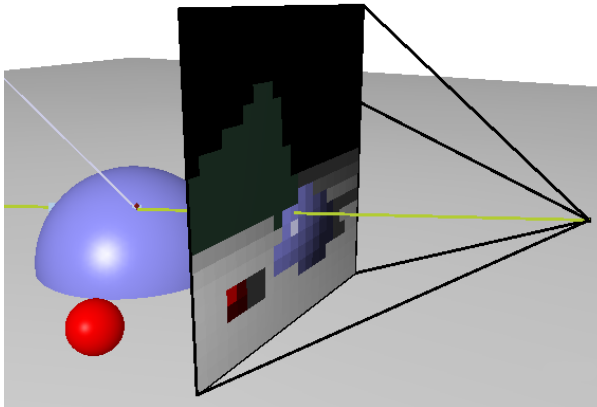


# Partie 3: Lancé de rayons

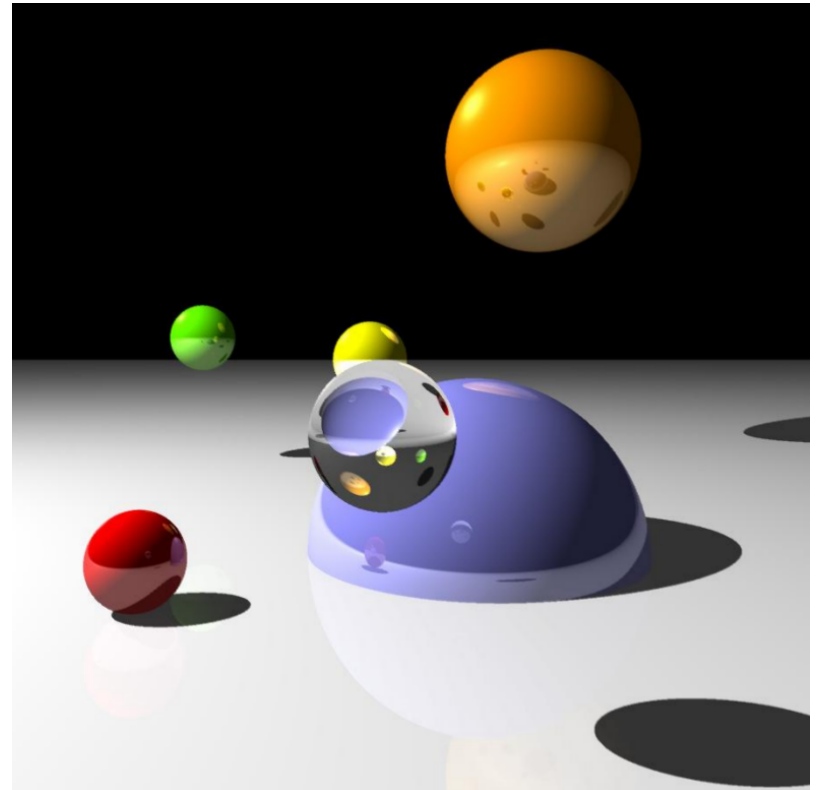
[Cours/TP]



lancé de rayons



Résultat année dernière



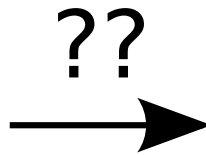
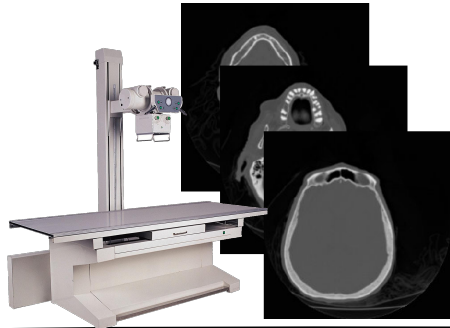
[4ETI 2012, Vaccalut/Pagliardini]

# Partie 4: Rendu volumique

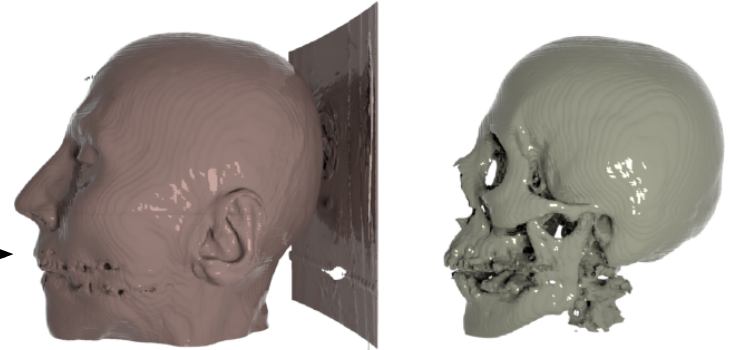
[Cours/TP]

## *Imagerie médicale*

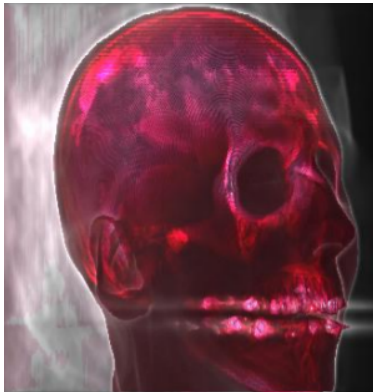
Données scanners



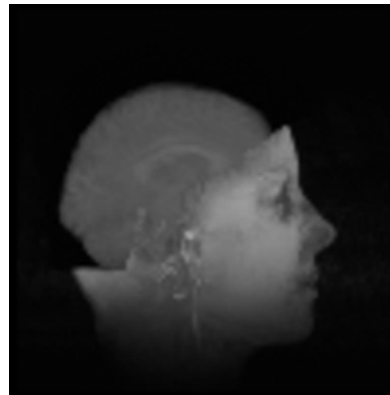
visualisation



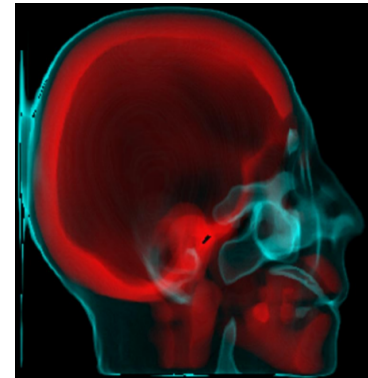
Résultats année dernière



[4ETI 2012 Pagliardini/Vaccalut]



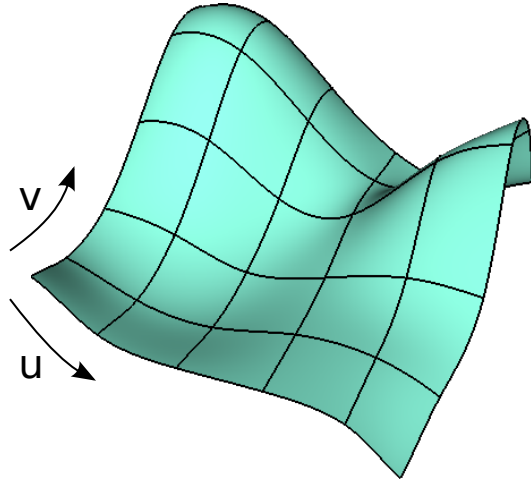
[4ETI 2012 Hauss/Frament]



[4ETI 2012 Kaiser/Rage]

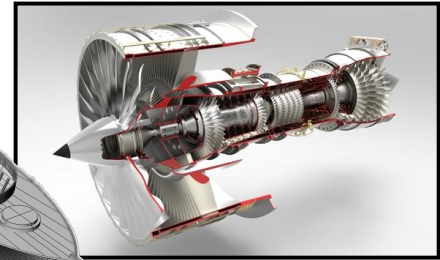
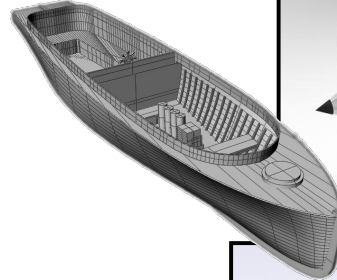
# Partie 5: Modélisation paramétrique

[Cours/TP]



$$f(u, v) = \mathbf{u} \mathbf{M} \mathbf{P} \mathbf{M}^T \mathbf{v}^T$$

Rhino 3D  
<http://www.viribusunitis.ca/jalbum/>



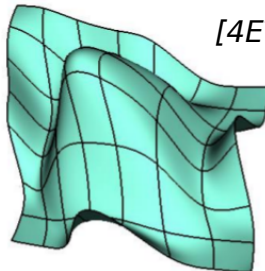
<http://grabcad.com/vasileios.thalassinos-1>



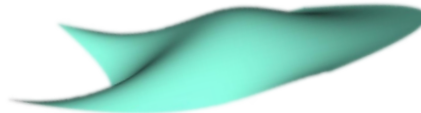
<http://www.fallingpixel.com/3d-artists/bobrock>

## Applications: CAO

Résultats année dernière:



[4ETI 2012, Salla/Rousset]

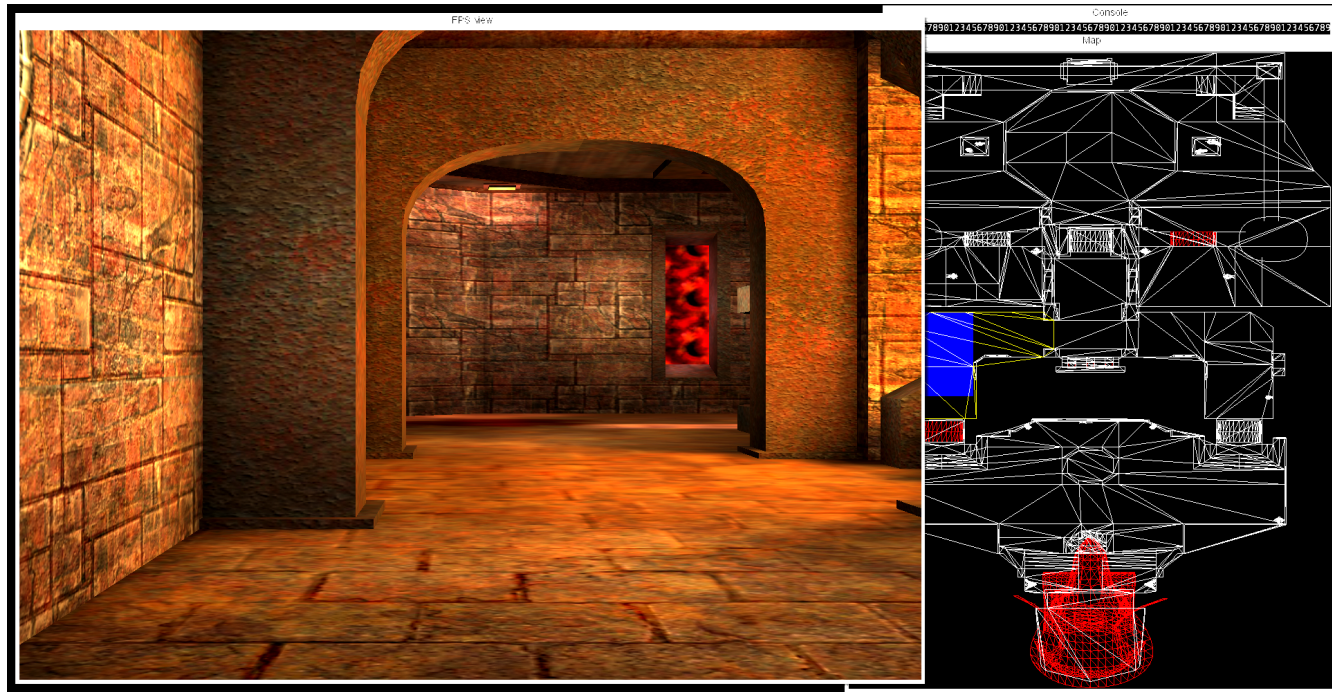


[4ETI 2012, Pagliardini/Vaccalut]



# Partie 6: Mondes virtuels

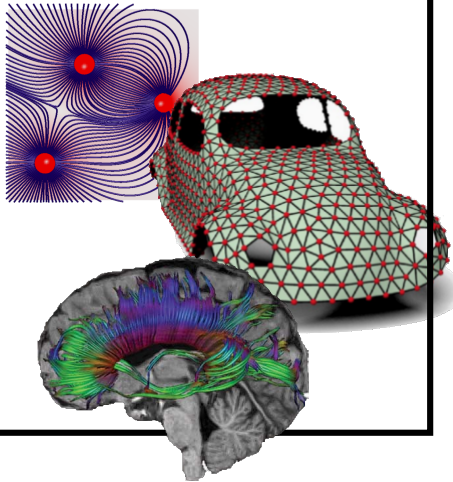
[Cours]



# Fonctionnement module

## Cours (2h)

$$\Delta' = \langle x_0 - x_s, \mathbf{u} \rangle^2 - (\|x_0 - x_s\|^2 - r^2)$$
$$t_{1/2} = - \langle x_0 - x_s, \mathbf{u} \rangle \pm \sqrt{\Delta'}$$



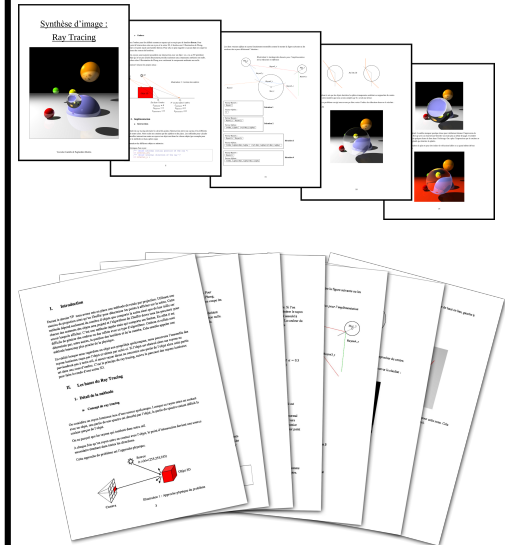
# C++

A screenshot of a C++ IDE. The left pane shows a project tree with files like 'main.cpp', 'ray.cpp', 'sphere.cpp', 'triangle.cpp', 'vector.cpp', 'raytracer.cpp', 'ray.h', 'sphere.h', 'triangle.h', 'vector.h', 'raytracer.h', 'raytracer.cpp', 'ray.h', 'sphere.h', 'triangle.h', 'vector.h'. The right pane shows C++ code for ray tracing, including comments in French and code for ray-sphere intersection and shading.

```
1 // Ray-tracer (ray-tracing)
2 //
3 // Ce programme est un exemple de rendu par ordinateur.
4 // Il est basé sur le livre "Computer Graphics: Principles and Practice" de van Dam et al.
5 //
6 // Les fichiers de ce projet sont :
7 // - main.cpp : Point d'entrée du programme.
8 // - ray.cpp : Définition de la classe Ray.
9 // - sphere.cpp : Définition de la classe Sphere.
10 // - triangle.cpp : Définition de la classe Triangle.
11 // - vector.cpp : Définition de la classe Vector.
12 // - raytracer.cpp : Définition de la classe RayTracer.
13 // - ray.h : Déclaration de la classe Ray.
14 // - sphere.h : Déclaration de la classe Sphere.
15 // - triangle.h : Déclaration de la classe Triangle.
16 // - vector.h : Déclaration de la classe Vector.
17 // - raytracer.h : Déclaration de la classe RayTracer.
18 //
19 // Le rendu est effectué en utilisant OpenGL.
20 //
21 // Les paramètres de rendu sont :
22 // - resolution : Résolution du rendu (par défaut : 1024x768).
23 // - fov : Angle de champ de vision (par défaut : 45 degrés).
24 // - near : Plan de coupe avant (par défaut : 0.1).
25 // - far : Plan de coupe arrière (par défaut : 1000).
26 // - gamma : Correction gamma (par défaut : 2.2).
27 // - verbose : Activer le mode verbeux (par défaut : false).
28 //
29 // Les options de ligne de commande sont :
30 // - -h : Afficher l'aide.
31 // - -r <resolution> : Définir la résolution du rendu.
32 // - -f <fov> : Définir l'angle de champ de vision.
33 // - -n <near> : Définir le plan de coupe avant.
34 // - -f <far> : Définir le plan de coupe arrière.
35 // - -g <gamma> : Définir la correction gamma.
36 // - -v : Activer le mode verbeux.
37 //
38 // Exemple d'utilisation :
39 // ./raytracer -r 1024 -f 45 -n 0.1 -f 1000 -g 2.2 -v
40 //
41 // Auteur : Damien Rohmer & David Odin
```

## TP (4h)

## Rapports



**Intervenants:** Damien Rohmer & David Odin



# Débouchés Synthèse d'Images

## Intitulés métiers:

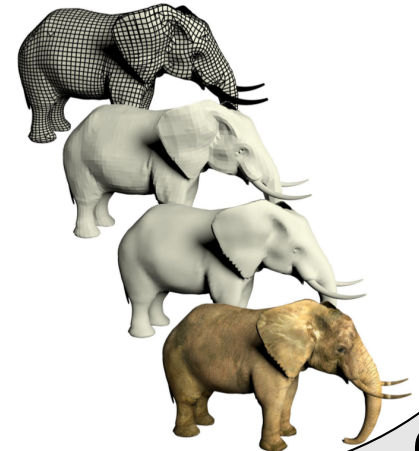
Développeurs logiciels  
/ Software engineer  
Analyste programmeur  
Ingénieur développement  
Ingénieur recherche  
Ingénieur d'études  
Editeur logiciels

## Mots clés:

Jeux Vidéos, serious game  
CAO  
Imagerie médicale  
Simulation, moteur physique  
Calcul numérique  
Graphique 3D, cinéma  
Reconstruction 3D  
...



$$\frac{d}{dt} \int_{\Omega} f(\mathbf{x}, t) dV = \dots$$



Recherche

Development



GE Health

