Algorithmique et Programmation

—

Examen sur machine

—

G1: keriven*(at)*certis.enpc.fr G2: patrick.labatut*(at)*ens.fr
G3: taron*(at)*certis.enpc.fr G4: etyngier*(at)*certis.enpc.fr
G5: segonne*(at)*certis.enpc.fr G6: chariot*(at)*certis.enpc.fr

—

17/02/06

—

1 Enoncé

Le but de ce problème est de créer et d'animer un modèle 3D et d'afficher un mouvement de cet objet vu par une caméra. Pour cela, nous procédons par étapes.

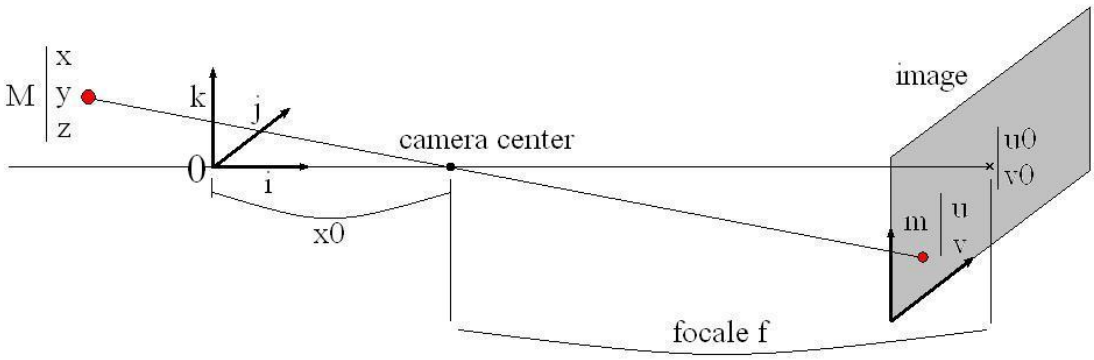
1.1 Construction du Modèle 3D

1. Travailler en local sous `D:\Info\` : créer un dossier `nom_prenom` dans `D:\Info\` (remplacer `nom_prenom` par son nom et son prénom !).
2. Créer une solution `Exam` dans `D:\Info\nom_prenom\` et ajouter un projet "Winlib" (`Winlib5 Projet`) de nom `logo` à la solution `Exam`.
3. Ajouter au projet deux fichiers `vect.cpp` et `vect.h` dans lesquels vous déclarerez et définirez à l'endroit approprié :
   (a) une structure `vect` représentant un vecteur de $\mathbf{R}^3$ (de composantes `double x,y,z` dans la base canonique (O,i,j,k) de $\mathbf{R}^3$),
   (b) un opérateur `+` entre deux `vect`.
4. On souhaite construire une structure représentant un tétraèdre. Ajouter au projet deux fichiers `tetra.cpp` et `tetra.h` et :
   (a) définir une structure `tetra` contenant 4 sommets (`vect M[4]`) et une couleur `Color c`,
   (b) déclarer et définir une fonction `regulier` ne prenant aucun argument et retournant un tétraèdre régulier de couleur rouge. Pour information, les quatre points suivant forment un tétraèdre régulier :
   $$\{1,0,0\}, \{-\tfrac{1}{2}, \tfrac{\sqrt{3}}{2}, 0\}, \{-\tfrac{1}{2}, -\tfrac{\sqrt{3}}{2}, 0\}, \{0,0,\sqrt{2}\}.$$
   On rappelle que la fonction `sqrt` est définie dans `#include <cmath>`.

1.2 Projection : du 3D au 2D

Nous nous préoccupons maintenant de la projection 2D de ces représentations 3D. Ajouter au projet deux fichiers `camera.cpp` et `camera.h` dans lesquels vous déclarerez et définirez :

1. une structure `camera` contenant les variables suivantes : `int u0,v0` (le centre de l'image), `double x0` (l'éloignement), et `double f` (la focale). La figure ci-après schématise la représentation d'une caméra.



2. une fonction `projette` qui projette un point 3D (`vect M`) par l'intermédiaire d'une caméra (`camera c`). La fonction `projette` retourne un point 2D de type `Pixel`. Nous rappelons que le type `Pixel` est défini dans la WinLib :

```
struct Pixel{
    int x,y;
};
```

Par conséquent, ne pas oublier d'ajouter les lignes `#include <win>` et `using namespace Win` dans le fichier `camera.h`. Les coordonnées `(u,v)` du projeté d'un point 3D de coordonnées `(x,y,z)` sont :

$$\text{u} = \text{c.u0} + \frac{\text{c.f} * \text{y}}{\text{c.x0-x}}$$
$$\text{v} = \text{c.v0} - \frac{\text{c.f} * \text{z}}{\text{c.x0-x}}.$$

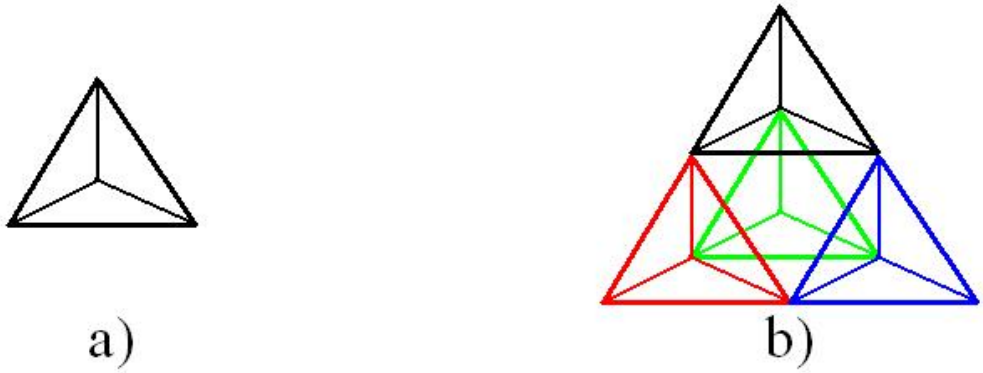
1.3 Affichage à l'écran

Nous sommes maintenant prêts pour afficher notre objet tétraèdre à l'écran.

1. Dans les fichiers `tetra.cpp` et `tetra.h`, déclarer et définir une fonction `affiche` prenant en argument un `camera c`, un `tetra T`. Cette fonction dessine un tétraèdre `tetra T` dans sa couleur, vu par la camera `camera c`. Pour cela, on utilisera la fonction `projette` (déclarée et définie dans `camera.h` et `camera.cpp`), ainsi que la fonction `DrawLine` (définie dans la WinLib) pour tracer un segment entre deux `Pixel`. La définition de la fonction `DrawLine` est la suivante :

`void DrawLine(const Pixel& p1,const Pixel& p2,const Color& col).`

Le projeté d'un tétraèdre doit ressembler à la figure ci-dessous (figure-a) :



2. Dans le fichier `main.cpp`, créer un tétraèdre régulier et l'afficher à l'écran dans une image de taille 512×512. Pour cela, créer une caméra `camera C` de paramètres :

`u0=256,v0=256`, `double x0=10`, et `double f=500`.

1.4 Animation du tétraèdre

Nous cherchons maintenant à animer notre modèle 3D.

1. Rotation d'un `vect` : dans `vect.h` et `vect.cpp`, ajouter une fonction `rotate` qui applique une rotation d'angle `double alpha` autour de l'axe Oz sur un vecteur `vect a` et qui renvoie un `vect b`. Les fonctions

`cos` et `sin` sont définies dans `#include <cmath>`.

$$\begin{aligned} \text{b.x} &= \cos(\alpha)\ \text{a.x} - \sin(\alpha)\ \text{a.y} \\ \text{b.y} &= \sin(\alpha)\ \text{a.x} + \cos(\alpha)\ \text{a.y} \\ \text{b.z} &= \text{a.z} \end{aligned}$$

2. Rotation d'un `tetra` : dans `tetra.h` et `tetra.cpp`, déclarer et définir une fonction `rotate` qui applique une rotation d'angle `double alpha` autour de l'axe Oz sur un `tetra` et qui retourne un `tetra`.
3. Première animation : dans `main.cpp`, animer le tétraèdre d'un mouvement de rotation sur place (pour l'instant sans ce soucier de l'effacer) : dans une boucle `for (int i=0;i<10000;i++)`, appliquer une rotation d'angle $\frac{i}{15}$ à chaque pas de temps. On pourra utiliser la fonction `MilliSleep(10)`.
4. Deuxième animation : maintenant, nous souhaitons afficher et effacer notre tétraèdre en mouvement :
   (a) dans les fichiers `tetra.cpp` et `tetra.h`, déclarer et définir une fonction `changeColor` prenant en argument un `tetra T` et un `Color c` et retournant une structure `tetra` (de couleur `Color c` et dont les sommets possèdent les mêmes coordonnées que celles de `tetra T`),
   (b) ajouter une fonction pour effacer un tétraèdre (utiliser les fonctions définies précédemment !).
   (c) dans `main.cpp`, animer le tétraèdre d'un mouvement de rotation en effaçant proprement à chaque pas de temps.

1.5 Un modèle plus élaboré

1. Dans le fichier `main.cpp`, créer un tableau `tetra tetras[4]` de 4 structures `tetra`. Initialiser les 4 tétraèdres du tableau de manière à produire la figure representée ci-dessus (figure-b). Pour cela, générer 4 tétraèdres "réguliers" et appliquer l'une des 4 translations suivantes sur chacun d'entre eux :

   $$\{1,0,0\}, \{-\tfrac{1}{2}, \tfrac{\sqrt{3}}{2}, 0\}, \{-\tfrac{1}{2}, -\tfrac{\sqrt{3}}{2}, 0\}, \{0,0,\sqrt{2}\}.$$

   Définir pour cela une fonction `translate` pour générer le translaté d'un tétraèdre `tetra T` par un vecteur `vect t` (utiliser l'opérateur `+` de `vect`).
2. Finalement, nous allons animer notre objet `tetra tetras[4]` d'un mouvement complexe et afficher ce dernier dans la caméra `camera C`. A chaque pas de temps dans la boucle `for (int i=0;i<10000;i++)`, appliquer une rotation d'angle $\frac{i}{15}$ suivie par une translation de vecteur :
   `vect t={-12+8*cos(i/150.),8*sin(i/150.0),-3.0}`.