Maîtrise d'Informatique

Examen d'Infographie II
11 juin 2001 - 2h

Documents autorisés : 1 feuille A4 de notes personnelles.
Lisez l'ensemble de l'énoncé. Soyez clair, précis, concis. Justifiez vos réponses. Relisez-vous.

Exercice A - Simplification de maillage (8 points)

On considère un maillage de polygones dont toutes les faces sont des triangles. Le maillage est fermé, c'est-à-dire que toute arête a exactement deux faces adjacentes. On cherche à simplifier ce maillage, c'est-à-dire à retirer des sommets, des arêtes et des faces de façon à alterer le moins possible l'aspect du maillage. Cette opération est appelée *décimation*.

On utilise une représentation du maillage par la structure "half-edge" vue en cours et rappelée ci-dessous :

```
class Sommet {
    real x, y, z;      // les coordonnées du sommet
    DemiArete a;       // une demi-arete qui lui est adjacente
};

class DemiArete {
    Sommet s;          // le sommet origine de la demi-arête
    DemiArete autre;   // l'arête "sœur" de cette arete
    Face f;            // la face du maillage à laquelle appartient l'arête
    DemiArete suiv;    // l'arête suivante de la face dans l'ordre de parcours
    DemiArete prec;    // l'arête précédente de la face dans l'ordre de parcours
};

class Face {
    DemiArete a;       // une demi-arête qui appartient à la face
    real nx, ny, nz;   // coordonnées de la normale de la face
    real d;            // ordonnée à l'origine du plan support de la face
    // l'equation du plan supportant la face est : nx.x + ny.y + nz.z + d = 0
};

class Maillage {
    listof Sommet sommets;  // La liste des sommets
    listof Arete aretes;    // La liste des arêtes
    listof Face faces;      // La liste des faces
}
```

Pour faciliter l'écriture, on introduit le type "`listof X`" qui permet de manipuler des listes d'objets avec les opérations d'insertion, de destruction et de parcours suivantes :

```
lst.insert (e);       // insérer l'élément e en tête de la liste lst
lst.remove (e);       // retirer l'élément e de la liste lst
foreach (e, lst) {    // parcourir les éléments de la liste lst
    // traiter l'élément e
}
```

1. L'opération de base de la décimation consiste à retirer une arête en fusionnant ses deux sommets. Cette fusion consiste à supprimer l'un des sommets et à garder l'autre (figure 1). Cette opération retire 2 faces, 3 arêtes et 1 sommet.

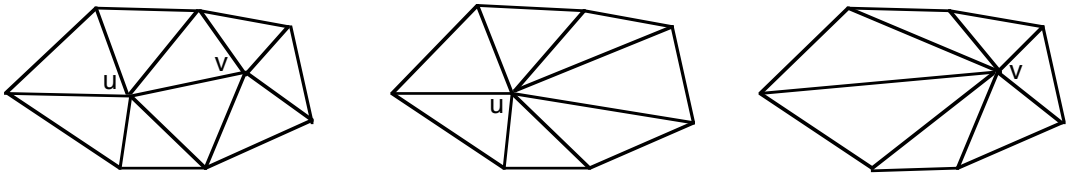

Figure 1- suppression de l'arête uv : à partir du maillage initial (à gauche), suppression du sommet v (au centre) ou du sommet u (à droite).

Ecrire l'algorithme de la fonction `Supprimer(Maillage m, DemiArete a)` Si `a` est la demi-arête `u->v`, cette fonction supprime le sommet `u` et l'arête `u-v`. Cette fonction doit mettre à jour le maillage, y compris les normales des faces affectées.

2. Dans certains cas, la décimation d'une arête n'est pas possible car elle change l'orientation d'une face du maillage (figure 2). Ecrire une fonction qui teste si la décimation d'une arête est possible.

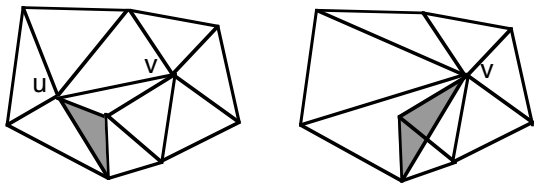

Figure 2 - suppression de l'arête uv qui conduit à l'inversion d'une face.

3. Pour déterminer quelle arête enlever, on introduit une fonction de coût associée à chaque demi-arête. Soit `s1` l'origine de la demi-arête (c'est-à-dire le sommet qui serait supprimé si l'on enlevait cette arête) et `s2` son autre extrémité. La fonction de coût est la somme des carrés des distances `d(s2, pi)` du sommet `s2` au plan `pi` contenant chaque face `fi` adjacente au sommet `s1`.

Ecrire un algorithme de décimation `Decimer (Maillage m, real e)` qui réduit le maillage autant que possible tant que le coût de décimation d'une arête est inférieur à `e`.

Exercice B - Splines (6 points)

Pour spécifier la trajectoire de la caméra dans une scène 3D, on utilise une courbe B-spline uniforme :

$Q_i(t) = T_i \,.\, M_B \,.\, G_{Bi}$ avec :

$T_i = [(t - t_i)^3 \; (t - t_i)^2 \; (t - t_i) \; 1]$

$$M_B = 1/6 \begin{vmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{vmatrix} \qquad GB_i = \begin{vmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{vmatrix}$$

La position C de la caméra suit la courbe, son orientation est déterminée en calculant un trièdre orthonormé Cx'y'z' de la façon suivante (figure 3) :

L'axe Cz' est la tangente à la courbe ;
L'axe Cy' est orthogonal à Cz' et aussi vertical que possible ;
L'axe Cx' est orthogonal à Cy' et Cz'

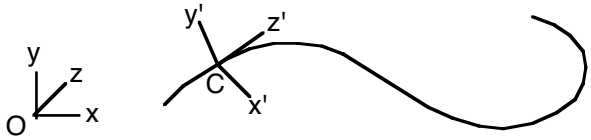

Figure 3 - Repère du modèle, trajectoire de la caméra, repère de la caméra.

1. En supposant que la courbe est définie pour t = [0, k] et que l'on souhaite parcourir cette trajectoire en S secondes à raison de 10 images par seconde, calculer le repère Cx'y'z' en fonction du repère Oxyz et de l'instant s_i correspondant à chaque image. Dans quelle condition ne peut-on pas calculer ce repère ?

2. Afin de contrôler la vitesse de déplacement de la caméra le long de la trajectoire, on se donne un segment de courbe de Bezier 2D qui donne la valeur du paramètre t de la trajectoire en fonction du temps s (figure 4). Pour cela, les points P1 et P4 du segment sont fixés : P1= (0, 0) et P4 = (k, S).

Figure 4 - Deux exemples de courbes décrivant la vitesse de déplacement de la caméra :
déplacement lent au début et à la fin (à gauche),
déplacement rapide au début et lent à la fin (à droite).

Calculer les coordonnées de la caméra sur sa trajectoire en fonction de la courbe de vitesse.

Exercice C - Traitement d'image (4 points)

On considère une image *F* formée d'un fond, de couleur unie *C*, et d'une image *I* inconnue composée sur ce fond. On cherche à extraire l'image *I* de l'image *F*, c'est-à-dire à calculer la couleur et la valeur-alpha de chaque pixel de l'image *I*.

On rappelle que la composition d'un pixel $i = (ir, ig, ib)$ sur un fond $c = (cr, cg, cb)$ avec une valeur-alpha a produit le pixel $p = (pr, pg, pb)$ tel que

$pr = a\ ir + (1-a)\ cr$; $pg = a\ ig + (1-a)\ cg$; $pb = a\ ib + (1-a)\ cb$;

1. Montrer que l'on ne peut pas espérer récuperer exactement *I* en ne connaissant que *F* et *C*.

2. Montrer comment récupérer une image *I'* à partir de *F* et *C* en faisant l'hypothèse que la valeur alpha de chaque pixel est maximale. On utilisera le fait que les valeurs-alpha et les composantes de couleurs sont toutes comprises entre 0 et 1.

Exercice D - Question de cours (2 points)

Expliquer le principe de l'algorithme de Weiman d'expansion et de contraction d'image.