



R my Malgouyres,
Algorithmes pour la synth se d'images
et l'animation 3D,
DUNOD, 2002

TP n  12 : Interpolation bilin aire, lissage, textures

Objectifs :

Le but de ce TP est d'ajouter   l'algorithme du z -buffer et aux mod les d' clairnement impl ment s aux TPs pr c dents le lissage de Phong, et le plaquage de textures.

Il est n cessaire d'avoir effectu  le TP sur l'algorithme du z -buffer et le TP sur les mod les d'illumination pour aborder ce TP. Si l'on n'a pas impl ment  de constructeur d'objets 3D, on n'ins rera que des cylindres de r volution.

Pour visualiser par votre algorithme un objet dont vous n'avez pas impl ment  le constructeur, vous pouvez cr er cet objet avec l'ex cutable *French3D.exe* fourni, exporter cet objet au format *polytext*, puis importer le fichier *polytext* dans votre propre programme.

1 Mod le de Phong

1.1 Normales aux sommets

Pour le mod le de Phong, il faut d'abord calculer, si ce n'est d j  fait, un vecteur normal unitaire pour chaque sommet du poly dre, et le m moriser dans la donn e `normale` de chaque sommet. Ce travail est fait lors de la construction des objets pour les diff rents objets :

- Boîtes;
- Sph res;
- C ne de r volution;
- Surfaces cylindriques (extrusion simple);
- Surfaces de r volution sph riques.

1.2 Interpolation bilin aire

Ensuite, lors de l'utilisation du vecteur normal   l'objet pour le calcul d' clairnement pr alablement ins r  dans l'algorithme du z -buffer (fonctions `calculcouleur`), on commence par calculer le vecteur normal par interpolation.

La fonction (voir le fichier *Interpol.cpp*)

```
Sommet interpolosommettriang1(Point3D p1, Point3D p2, Point3D p3,  
                             Sommet n1, Sommet n2, Sommet n3,  
                             Point3D point)
```

calcule un sommet interpolé en un point `point` d'un triangle (`p1`, `p2`, `p3`) de sommets respectifs (`n1,n2,n3`). Ceci permet d'interpoler simultanément toutes les données de la classe `sommet` (normale, coordonnées de textures, etc...).

Pour l'implémentation de la fonction `interpolesommettriang1`, on utilisera abondamment les opérateurs (addition, multiplication par un nombre, etc...) de la classe `sommet` pour alléger les calculs.

Pour le cas d'une facette à 3 sommets, on implémentera et on utilisera la fonction `Objet3D : :interpolesommettriang`, qui appellera la fonction `interpolesommettriang1` avec pour paramètres les données aux sommets de la facette.

Pour le cas d'une facette à 4 sommets, on implémentera et on appellera la fonction `Objet3D : :interpolesommetcarre`, qui déterminera dans quel triangle se trouve le point `point` et appellera la fonction `interpolesommettriang1` avec pour paramètres les données des trois sommets appropriés.

2 Textures

2.1 Coordonnées de textures aux sommets

Dans les constructeurs des objets dont les classes sont définies dans le fichier *Objetderiv.h*, calculera les coordonnées de texture (dans $[0, 1]$) de chaque sommet, et on les mémorisera dans les données membres `coordtextureX` et `coordtextureY` de la classe `Sommet`.

Pour les quadriques (sphères, cylindres, etc...), les angles de coordonnées sphériques ou la hauteur doivent être transformés pour obtenir une coordonnée de texture, qui leur est proportionnelle.

Pour les surfaces définies à partir d'une courbe spline, on doit utiliser la technique du paramétrage pour trouver la coordonnée de texture, qui est proportionnelle au paramètre s ou t .

2.2 Plaquage de la texture lors de l'affichage

Dans l'algorithme du z -buffer, il faut apporter des modifications. Lors du calcul des coefficients RGB de la couleur de l'objet, il faut éventuellement remplacer les données membres `material.couleur.R`, `material.couleur.G` et `material.couleur.B` de la classe `Objet3D` par la couleur de l'image de texture au bon pixel.

Pour cela, dans la fonction

```
Couleur Scene3D::calculcouleur(Objet3D & objet, const Facette & face,
                               Point3D p3d, Point3D normale,
                               bool ...);
```

il faut modifier l'appel de la fonction

```
Scene3D : :calculcouleur(Couleur coupleurobj, Point3D p3d, Point3D normale,
const Objet3D & objet, bool ...)
```

en initialisant `couleurobj` à partir de la couleur de texture.

Si le booléen `objet.textures.withtexture[face.notexture]` est à vrai, on calcule la couleur de l'image de texture (`objet.textures.imtexture[face.notexture]` de type `imcouleur`) au pixel approprié. Pour cela, on procède en deux étapes :

1. Appel de la fonction d'interpolation bilinéaire des sommets, si ce n'est pas déjà fait (voir la partie sur le lissage de Phong) ;
2. multiplication des coordonnées de textures au sommet obtenu par la largeur et la hauteur de l'image `objet.textures.imtexture[face.notexture]` (utiliser les accesseurs `largeur()` et `hauteur()`). On obtient ainsi les coordonnées du pixel de l'image de texture à prendre en compte.
3. Calcul de la couleur du pixel de cette image de texture, on pourra utiliser la fonction suivante :

```
void imcouleur::getpixel(int x, int y, double& r, double& g, double& b)
```

qui retourne les coefficient RGB entre 0 et 1 d'un pixel (x, y) .