



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

TP n  11 : Mod les d'illumination

Objectifs :

Le but de ce TP est d'ajouter   l'algorithme du z -buffer impl ment  au pr c dent TP des calculs d'illumination pour obtenir un affichage plus "r aliste" de poly dres 3D, notamment pour les poly dres approximant des surfaces lisses.

Il est n cessaire d'avoir effectu  le TP sur l'algorithme du z -buffer 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.

Lors de l'algorithme du z -buffer, lorsqu'on parcourt la projection d'une facette dans le plan de l' cran, on calcule pour chaque pixel le point 3D M de la facette qui se projette sur ce pixel. Dans le pr sent TP, on calculera la couleur du point M en fonction des mod les d' clairage suivants :

1. Lumi re ambiante ;
2. R flexion diffuse ;
3. R flexion sp culaire ;
4. Mod le de Phong ;

1 O  trouver les donn es

Pour les trois premiers points, il suffit d'utiliser les formules vues en cours, en prenant pour vecteur normal la normale   la facette (en principe d j   calcul e pour le z -buffer), et en utilisant la donn e membre *material* de la classe *Objet3D* de type *MaterialData* (voir les fichiers *Objet3D.h* et *Couleur.h* :

```
class MaterialData { public:
    Couleur couleur; // coefficients RGB de la couleur (entre 0 et 1)
    double coef_rd; // coefficient de r flexion diffuse
    double coef_ambient; // coefficient de r flexion de la lumi re ambiante
    double coef_rs; // coefficient de r flexion sp culaire
    double exp_rs; // exposant sp culaire
```

```

    bool phongenable; // true si Phong ou Gouraud est possible pour cet objet, false sinon
    ETC...
};

```

La classe `Couleur` contient trois champs `R`, `G` et `B`.

Dans la classe `Scene3D`, on trouvera l'information concernant la lumière ambiante et les sources lumineuses ponctuelles non directionnelles :

```

    Sourcelum * tablum; // tableau des sources lumineuses
    int nblum; // nombre de sources lumineuses insérées par l'utilisateur
    double lumambientR, lumambientG, lumambientB; // intensité de la lumière ambiante

```

2 Comment procéder

Dans la fonction `Scene3D::parcouspolyperspect` implémentée au précédent TP, avant la mise à jour des buffers, on appellera la fonction suivante, qui retourne la couleur à mettre dans les buffers (après multiplication par 255) :

```

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

```

Cette fonction, dans un premier temps, se contentera d'appeler la fonction :

```

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

```

qui calcule la couleur d'un point `p3d` d'un objet `objet` de couleur `couleurobjet`, connaissant la normale `normale` à l'objet en ce point.

Cette dernière fonction, que l'on implémentera, retournera l'intensité lumineuse (dans les trois composantes RGB) émise par le point `p3d`, avec un vecteur normal `normale` à l'objet `objet`, de couleur `couleurobjet`. On trouvera les autres données nécessaires (coefficients de réflexion diffuse, spéculaires,...) dans la donnée membre `material` de l'objet 3D. Les termes dus à toutes les sources lumineuses doivent être additionnés.