Rendu

Travaux pratiques

Au début de ce TP/TD, vous recevrez une archive zip contenant une base de code. Ce code permet d'afficher un *maillage triangulaire* à l'aide d'openGL.

1. Nous commencerons par l'analyser ensemble pour vous familiariser avec.
2. Vous devez faire évoluer ce code au fur et à mesure du TP, pour répondre aux questions.

1 Base de code

Installation

Téléchargez l'archive sur le moodle `https://moodle.umontpellier.fr/mod/resource/view.php?id=558869`. Pour compiler le code et l'exécuter :

```
$ make
$ ./tp ./data/sphere.off
```

Interactions utilisateur

```
void key (unsigned char keyPressed, int x, int y)
```

La fonction `key` permet de d'interpréter les entrées clavier utilisateur. Les options de visualisation activées par des touches sont les suivantes, en appuyant sur la touche :

— `w` : changement du mode d'affichage (fil de fer/éclairé).

Vous pouvez interagir avec le modèle avec la souris :

— Bouton du milieu appuyé : zoomer ou reculer la caméra,

— Clic gauche appuyé : faire tourner le modèle.

Rendu de maillages

Vous constaterez que les méthodes de dessin de maillage du fichier `tp.cpp` ont changées. Nous utilisaons maintenant des shaders pour l'affichage.

Nous allons implémenter l'éclairage utilisant le modèle de Phong vu en cours. Pour cela nous initialisons les matériaux ambiants, diffus et spéculaire de l'objet (r,g,b,a pour chaque, correspondant aux ka, kd, ks du cours) :

```
void setDefaultMaterial () {
    GLfloat material_color[4] = {1.0, 1.0, 1., 1.0f };
    GLfloat material_specular[4] = {0.5, 0.5, 0.5, 1.0 };
    GLfloat material_ambient[4] = {1.0, 0.0, 0.0, 1.0};

    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, material_specular);
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, material_color);
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, material_ambient);
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 128);
}
```

Nous initialisons maintenant les propriétés des lumières (positions, directions, couleurs ambiante, diffuse et spéculaire) :

```
void initLights () {

    GLfloat light_position_0[4] = {42, 374, 161, 0};
    GLfloat light_position_1[4] = {473, -351, -259, 0};
    GLfloat light_position_2[4] = {-438, 167, -48, 0};

    GLfloat direction_0[3] = {-42, -374, -161,};
    GLfloat direction_1[3] = {-473, 351, 259};
    GLfloat direction_2[3] = {438, -167, 48};

    GLfloat diffuse_color_0[4] = {1.0, 1.0, 1.0, 1};
    GLfloat diffuse_color_1[4] = {0.28, 0.39, 1.0, 1};
    GLfloat diffuse_color_2[4] = {1.0, 0.69, 0.23, 1};
```

```
14
    GLfloat specular_color_0[4] = {0.8, 0.0, 0.0, 1};
    GLfloat specular_color_1[4] = {0.0, 0.8, 0.0, 1};
    GLfloat specular_color_2[4] = {0.0, 0.0, 0.8, 1};

19  GLfloat ambient[4] = {0.3f, 0.3f, 0.3f, 0.5f};

    glLightfv (GL_LIGHT0, GL_POSITION, light_position_0);
    glLightfv (GL_LIGHT0, GL_SPOT_DIRECTION, direction_0);
    glLightfv (GL_LIGHT0, GL_DIFFUSE, diffuse_color_0);
24  glLightfv (GL_LIGHT0, GL_SPECULAR, specular_color_0);

    glLightfv (GL_LIGHT1, GL_POSITION, light_position_1);
    glLightfv (GL_LIGHT1, GL_SPOT_DIRECTION, direction_1);
    glLightfv (GL_LIGHT1, GL_DIFFUSE, diffuse_color_1);
29  glLightfv (GL_LIGHT1, GL_SPECULAR, specular_color_1);

    glLightfv (GL_LIGHT2, GL_POSITION, light_position_2);
    glLightfv (GL_LIGHT2, GL_SPOT_DIRECTION, direction_2);
    glLightfv (GL_LIGHT2, GL_DIFFUSE, diffuse_color_2);
34  glLightfv (GL_LIGHT2, GL_SPECULAR, specular_color_2);

    glLightModelfv (GL_LIGHT_MODEL_AMBIENT, ambient);

}
```

Notez que la couleur ambiante est la même pour toutes les lumières.
Les variables suivantes permettent de moduler l'influence de chacune des composantes du modèle de Phong :

```
static float ambientRef = 0.1f;
2 static float diffuseRef = 0.8f;
static float specularRef = 0.5f;
static float shininess = 16.0f;
```

L'éclairage est ensuite calculé dans le fragment shader (shader.frag), notez que vous y retrouvez les mêmes variables :

```
1  uniform float ambientRef;
   uniform float diffuseRef;
   uniform float specularRef;
   uniform float shininess;

6  varying vec4 p;
   varying vec3 n;

   void main (void) {
       vec3 P = vec3 (gl_ModelViewMatrix * p);
11     vec3 N = normalize (gl_NormalMatrix * n);
       vec3 V = normalize (-P);

       vec4 Isa = gl_LightModel.ambient;
       vec4 Ka = gl_FrontMaterial.ambient;
16     vec4 Ia = Isa * Ka;

       vec4 I = ambientRef * Ia ;

       gl_FragColor = vec4 (I, 1);
21 }
```

Dans la fonction précédente la couleur affichée correspond à la composante ambiante du modèle de Phong en utilisant le matériaux de l'objet (`gl_FrontMaterial.ambient`) et la couleur ambiante de la lumière (`gl_LightModel.ambient`)

2 Exercice : Calcul de l'ombrage d'un maillage

1. Implémentez la modulation de la couleur ambiante en incrémentant/décrémentant `ambientRef` (A/a : Augmente/Diminue la reflection ambiante de 0.1).
2. Dans le fragment shader, ajouter la composante diffuse en calculant l'éclairage de la lumière 0 (`gl_LightSource[0]`) à I (slide 36). Modulez la en utilisant `diffuseRef`.
3. Ajouter la composante speculaire en calculant l'éclairage de la lumière 0 (`gl_LightSource[0]`) à I (slide 43). Modulez la en utilisant `specularRef`.
4. Implémentez les interactions claviers suivantes :
   — D/d : Augmente/Diminue la reflection diffuse (incrémentant/décrémentant `diffuseRef`, utilisez un pas de 0.1),

— S/s : Augmente/Diminue la reflection specular (incrémentant/décrémentant `specularRef`, utilisez un pas de 0.1),
— +/- : Augmente/Diminue la brillance (incrémentant/décrémentant `shininess`, utilisez un pas de 1.).

5. Itérez sur les 3 lumières définies et ajouter leurs contributions à `I`.