Corrigé

23 juin 2008

1 Couleur et illumination

Question 1

Figure 1.

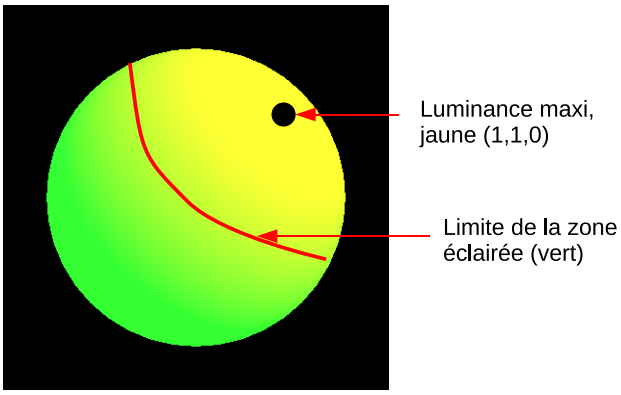

FIG. 1 – La sphère illuminée.

2 Maillages et textures

Question 2

Penser à ordonner les points de chaque face dans le sens trigonométrique vu de l'extérieur de l'objet.

coordonnées :

(0,0,1)	(1,0,1)	(1,1,1)	(0,1,1)	*z=1*
(1,0,1)	(1,0,0)	(1,1,0)	(1,1,1)	*x=1*
(1,0,0)	(0,0,0)	(0,1,0)	(1,1,0)	*z=0*
(0,0,0)	(0,0,1)	(0,1,1)	(0,1,0)	*x=0*
(0,1,1)	(1,1,1)	(1,1,0)	(0,1,0)	*y=1*
(0,0,0)	(1,0,0)	(1,0,1)	(0,0,1)	*y=0*

Question 3

On ne précisait pas si les normales étaient données par face ou par sommet. Par face, on obtient le tableau suivant. Par sommet, le même avec les normales apparaissant 4 fois pour chaque face.

normales :

(0,0,1)	*z=1*
(1,0,0)	*x=1*
(0,0,-1)	*z=0*
(-1,0,0)	*x=0*
(0,1,0)	*y=1*
(0,-1,0)	*y=0*

Question 4

corrdonnées de texture :

(0,0.33)	(0.25,0.33)	(0.25,0.67)	(0,0.67)	*z=1*
(0.25,0.33)	(0.5,0.33)	(0.5,0.67)	(0.25,0.67)	*x=1*
(0.5,0.33)	(0.75,0.33)	(0.75,0.67)	0.5,0.67)	*z=0*
(0.75,0.33)	(1,0.33)	(1,0.67)	(0.75,0.67)	*x=0*
(0.5,0.67)	(0.75,0.67)	(0.75,1)	(0.5,1)	*y=1*
(0.5,0)	(0.75,0)	(0.75,0.33)	(0.5,0.33)	*y=0*

Question 5

Voici une solution avec des `quad_strip`*. D'autres étaient envisageables, par exemple avec de* `triangle_strip`*.*

```
void torus(double R, double r, int n, int m)
{
   int i, j, k;
   double s, t, x, y, z, twopi;

   twopi = 2 * M_PI;
   for (i = 0; i < n; i++) {
      glBegin(GL_QUAD_STRIP);
      for (j = 0; j <= m; j++) {
         for (k = 1; k >= 0; k--) {
            s = (i + k) % n + 0.5;
            t = j % m;

            x = (R+r*cos(s*twopi/n))*cos(t*twopi/m);
            y = (R+r*cos(s*twopi/n))*sin(t*twopi/m);
            z = r * sin(s * twopi / n);
            glVertex3f(x, y, z);
         }
      }
      glEnd();
   }
}
```

3 Positionnement

Question 6

Dans la solution suivante, on utilise `glScalef` *pour obtenir les cubes de dimensions voulues, encadré par* `glPushMatrix/glPopMatrix` *pour toujours revenir à une échelle 1 entre les tracés.*

```
void afficheMoulin(GLfloat anglePalme)
{
  // base
```

```
  glPushMatrix();
  glScalef(1,2,1);
  glutSolidCube( 1.0 );
  glPopMatrix();

  // deplacer vers l'axe de rotation palme/base, appliquer la rotation
  glTranslatef(0,0.75,0);
  glRotatef(anglePalme,0,0,1);

  // palme
  glPushMatrix();
  glScalef(0.1,1,0.1);
  glutSolidCube( 2.0 );
  glPopMatrix();

  // aller au bout de la palme
  glTranslatef(0,0.95,0);
  // revenir à l'orientation d'origine
  glRotatef(-anglePalme,0,0,1);
  // deplacer vers le centre du bras
  glTranslatef(0,-0.25,0);

  // bras
  glPushMatrix();
  glScalef(0.1,1,0.1);
  glutSolidCube( 0.5 );
  glPopMatrix();

  // deplacer en bas du bras
  glTranslatef(0,-0.35,0);

  // bac
  glPushMatrix();
  glScalef(1,1,0.1);
  glutSolidCube( 0.3 );
  glPopMatrix();
}
```

Question 7

```
Separator {
 Caisse {}
 Separator {
  Transform { translation -1.5 0 0 } # arriere
  Separator {
   Transform {            # droite
     translation 0 0 1
     rotation 1 0 0 1.5709
     }
   Roue {}
  }
  Separator {
   Transform {            # gauche
     translation 0 0 -1
     rotation 1 0 0 -1.5709  # tournee dans l'autre sens
     }
   Roue {}
  }
 }
 Separator {
  Transform { translation 1.5 0 0 } # avant
  Separator {
   Transform {            # droite
     translation 0 0 1
     rotation 1 0 0 1.5709
     }
```

```
   Transform {            # tournee
     rotation 0 0 1 -0.785
     }
   Roue {}
  }
  Separator {
   Transform {            # gauche
     translation 0 0 -1
     rotation 1 0 0 -1.5709  # tournee dans l'autre sens
     }
   Transform {            # tournee
     rotation 0 0 1 0.785
     }
   Roue {}
  }
 }
}
```