UNIVERSITE DE BOURGOGNE
U.F.R. Sciences et Techniques
Filière : *Licence 3 Informatique*

2017-2018

Session 2

EPREUVE :

Examen Synthèse d'Image juin 2018

Durée : 1h30

*Seul document autorisé : une feuille A4 recto-verso **manuscrite**.*
Les exercices peuvent être traités indépendamment les uns des autres.
Le barème est donné à titre indicatif.

N° d'anonymat :

Partie 1 : *Cours (environ 6 points)*
Écrire la réponse dans les cadres.

Question 1 :
A quelle transformation correspond la matrice ci-contre. Préciser les paramètres de la transformation.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

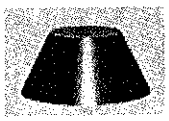
Question 2 :
A quelle transformation correspond la matrice ci-contre. Préciser les paramètres de la transformation.

$$\begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

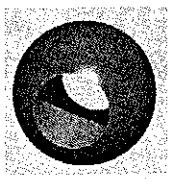
Question 3 :
A quelle transformation correspond la matrice ci-contre. Préciser les paramètres de la transformation.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Question 4 :
A partir d'un cône et d'un cube, quelle opération booléenne permet d'obtenir l'objet ci-contre.



Question 5 :
A partir d'une sphère et d'un cylindre, quelle opération booléenne permet d'obtenir l'objet ci-contre.



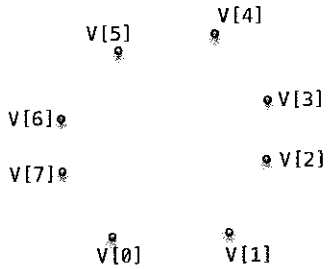
Question 6 :

Compléter l'affichage obtenu en exécutant le code suivant.

Code	*Affichage*

```
class Point{
public:
    double x,y,z;
};

void text()
{
    Point V[8];
    glColor3f(0.0,0.0,0.0);
    glBegin(GL_TRIANGLES);
    {
        for(int i=0;i<8;i++)
            glVertex3f(V[i].x,V[i].y,V[i].z);
    }
    glEnd();
}
```

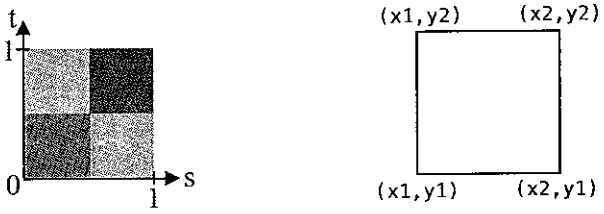


Question 7 :

Compléter l'affichage obtenu en exécutant le code suivant.

Code	*Affichage*

```
glEnable(GL_TEXTURE_2D);
glBegin(GL_QUADS);
    glTexCoord2f(0,0);
    glVertex2f(x1,y1);
    glTexCoord2f(1,0);
    glVertex2f(x2,y1);
    glTexCoord2f(1,3/4);
    glVertex2f(x2,y2);
    glTexCoord2f(0,3/4);
    glVertex2f(x1,y2);
glEnd();
```



Partie 2 : *Transformations (environ 7 points)*

Soit une transformation M composée d'une translation T de vecteur (1,0,0) suivie d'une rotation R d'axe z et d'angle 90°.

1. Donner l'expression de cette translation et de cette rotation sous la forme de matrices homogènes T et R.

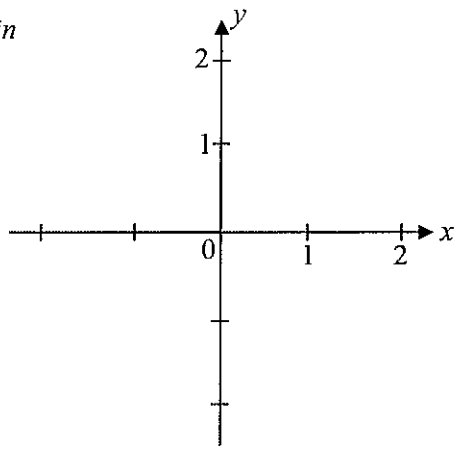
2. Calculer T et R.

3. Donner l'expression de cette transformation sous la forme d'une matrice homogène M en fonction des matrices T et R.

4. Calculer M.

5. Soit P le point de coordonnées (1,0,0,1). Donner les coordonnées du point P' image de P par la transformation M (toujours en coordonnées homogènes).

6. Placer P et P' dans le repère ci-contre :



Partie 3 : *OpenGl (environ 7 points)*

On considère les primitives unitaires suivantes :

- Un cube centré en (0,0,0) d'arête 1
- Une sphère centrée en (0,0,0) de rayon 1
- Un cylindre centré en (0,0,0) de rayon 1 et de longueur 1

Soit le code suivant :

```
/*********************************************/
/*              toupie.cpp                   */
/*********************************************/
/*       Affiche a l'ecran une toupie en 3D    */
/*********************************************/

#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cmath>

#define largimg 256
#define hautimg 256
#define PI 3.14159
#define n 6
#define m 10
using namespace std;

char presse;
int anglex=30,angley=20,x,y,xold,yold;
int MODE = 0;
int alpha=0,theta=0;

void affichage();
void clavier(unsigned char touche,int x,int y);
void loadJpegImage(char *fichier);
void anim();
void Mode(int mo);

int main(int argc,char **argv)

{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE |
GLUT_DEPTH);
    glutInitWindowSize(400,400);
    glutCreateWindow("toupie");

    glClearColor(1.,1.,1.,0.0);
    glShadeModel(GL_FLAT);
    glEnable(GL_DEPTH_TEST);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-4, 4, -4, 4, -4, 4);
    glMatrixMode(GL_MODELVIEW);

    glutDisplayFunc(affichage);
    glutKeyboardFunc(clavier);
    glutMouseFunc(souris);
    glutMotionFunc(sourismouv);
    glutReshapeFunc(redim);
    glutIdleFunc(anim);

    glutMainLoop();
    return 0;
}
```

```
void Mode(int mo)
{
      switch (mo) {
        case 0:
glPolygonMode(GL_FRONT_AND_BACK,GL_FILL); break;
        case 1:
glPolygonMode(GL_FRONT_AND_BACK,GL_LINE); break;
        case 2:
glPolygonMode(GL_FRONT_AND_BACK,GL_POINT);break;
      }
}

void toupie(int M)
{
    if (M==0)
        glColor3f(0.7,0.8,1.0);
    else
        glColor3f(0.0,0.0,0.0);
    glutSolidSphere(1,20,10);

    glPushMatrix();
    glTranslatef(0,1.5,0);
    glRotatef(90,1,0,0);
    if (M==0)
        glColor3f(0.8,0.7,1.0);
    else
        glColor3f(0.0,0.0,0.0);
    glutSolidCylinder(0.3,3,10,10);
    glPopMatrix();

    if (M==0)
        glColor3f(1.0,0.8,1.0);
    else
        glColor3f(0.0,0.0,0.0);
    glPushMatrix();
    glScalef(1,0.5,1);
    glutSolidCube(1.8);
    glPopMatrix();

    glRotatef(30, 0,1,0);
    glPushMatrix();
    glScalef(1,0.5,1);
    glutSolidCube(1.8);
    glPopMatrix();

    glRotatef(30, 0,1,0);
    glPushMatrix();
    glScalef(1,0.5,1);
    glutSolidCube(1.8);
    glPopMatrix();
}
```

```
void affichage()
{
    glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    //gluLookAt(-
4.0,0.0,0.5,0.0,0.0,0.0,0.0,0.0,1.0,0.0);
    glRotatef(angley,1.0,0.0,0.0);
    glRotatef(anglex,0.0,1.0,0.0);
    glRotatef(alpha,0.0,1.0,0.0);
    glTranslatef(2,0,0);
    glRotatef(theta,0.0,1.0,0.0);
    Mode(MODE);
    toupie(MODE);
   // Mode(1);
  //  toupie(1);
    glutSwapBuffers();
}

void anim(){
    theta++;
    if (theta == 360)
        theta = 0;
    alpha+=2;
    if (alpha == 360)
        alpha = 0;
    glutPostRedisplay();
}
```

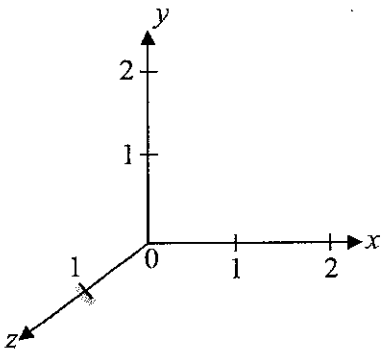
```
void clavier(unsigned char touche,int x,int y)
{
  switch (touche)
    {
    case 'p':
      MODE = 0;
      glutPostRedisplay();
      break;
    case 'f':
      MODE = 1;
        glColor3f(0.0,0.0,0.0);
      glutPostRedisplay();
      break;
    case 's' :
        MODE = 2;
      glutPostRedisplay();
      break;
    case 'q' : /*la touche 'q' permet de quitter
le programme */
      exit(0);
    }
}
```

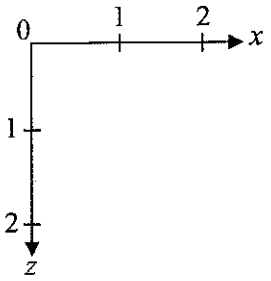
1. A partir des primitives unitaires données et de ce code, décrire la toupie à l'aide d'un modèle hiérarchique. L'arbre choisi pour la toupie contiendra les précisions suivantes : primitives, couleur, transformations (simplifier les transformations lorsque deux transformations de même type se suivent : par exemple, deux mises à l'échelle de suite)…

2. Au lancement du programme, donner le mode de représentation de la toupie (sommet, fil de fer, plein…).

3. Qu'en est-il si on enlève les commentaires devant : `Mode(1);toupie(1);`.

4. Représenter la toupie dans les repères suivants :





5. Représenter les trajectoires de la toupie dans le repère suivant en précisant le nom des axes. Préciser pour chaque trajectoire à quelle commande elle correspond.