OpenGL 3.0

Chapitre 2

Boucle évènementielle
Transformations

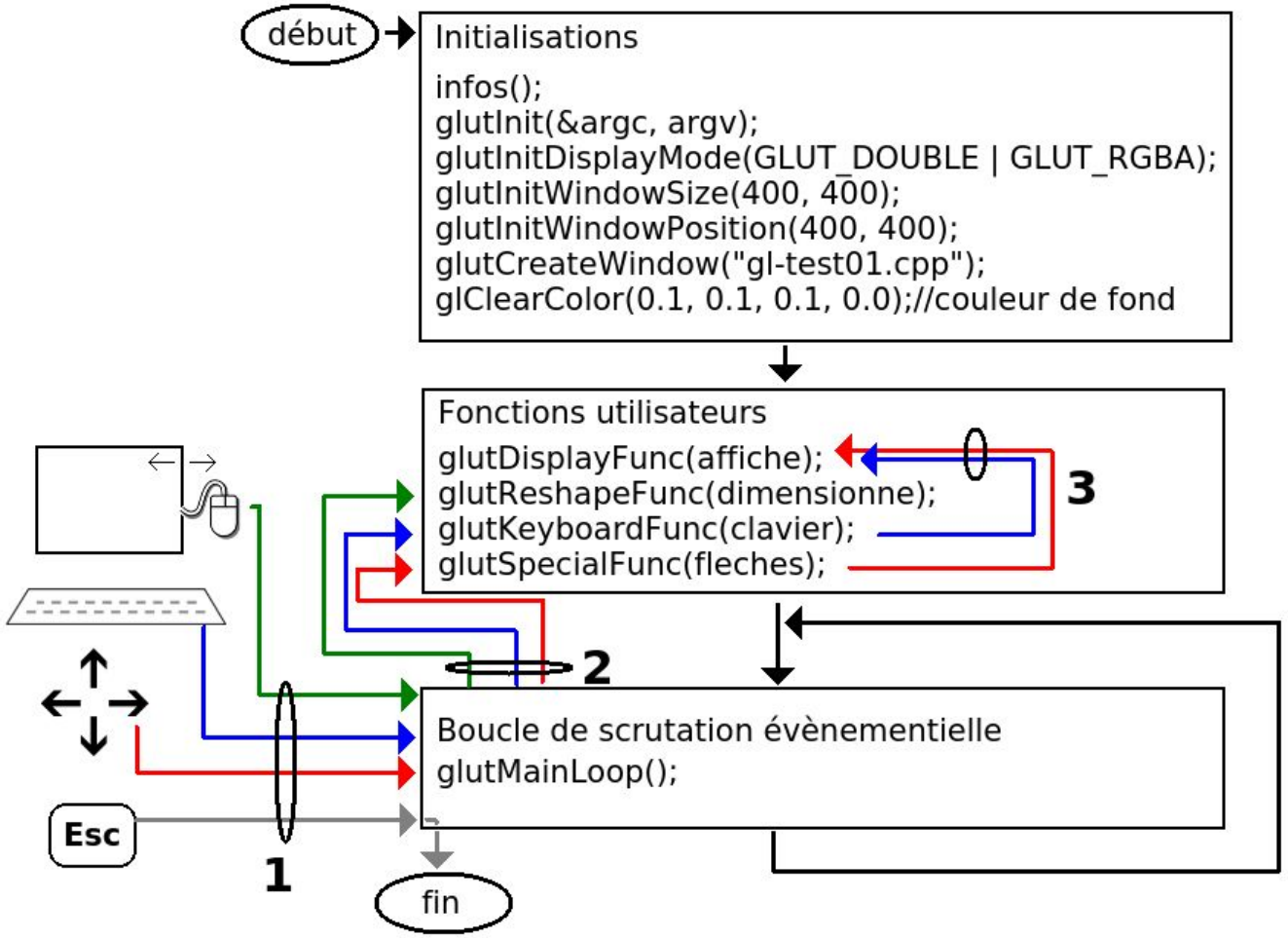
gl-test02.cpp

C.Turrier
15 décembre 2020

Sommaire

1) Boucle évènementielle

Le squelette d'un programme OpenGL est illustré par le schéma suivant. qui s'articule autour de la fonction boucle évènementielle **glutMainLoop()**. Il fonctionne de la façon suivante :
OpenGL possède diverses fonctions qui finissent par le mot "**Func**", (fonctions **glutdisplayFunc**, **glutReshapeFunc** et **glutKeyboardFunc**). Ces fonctions **Func** prenent en paramètre le nom d'une fonction de rappel écrite par le programmeur (affiche, dimensionne, clavier, fleches par exemple). La fonction **glutMainLoop()** assure une scrutation en boucle de la survenue d'un évènement (clic de souris, appui d'une touche clavier...). Si un évènement survient, la fonction **Func** correspondante est automatiquement appelée par **glutMainLoop()**, ce qui déclenche l'exécutiont de la fonction de rappel correspondante.

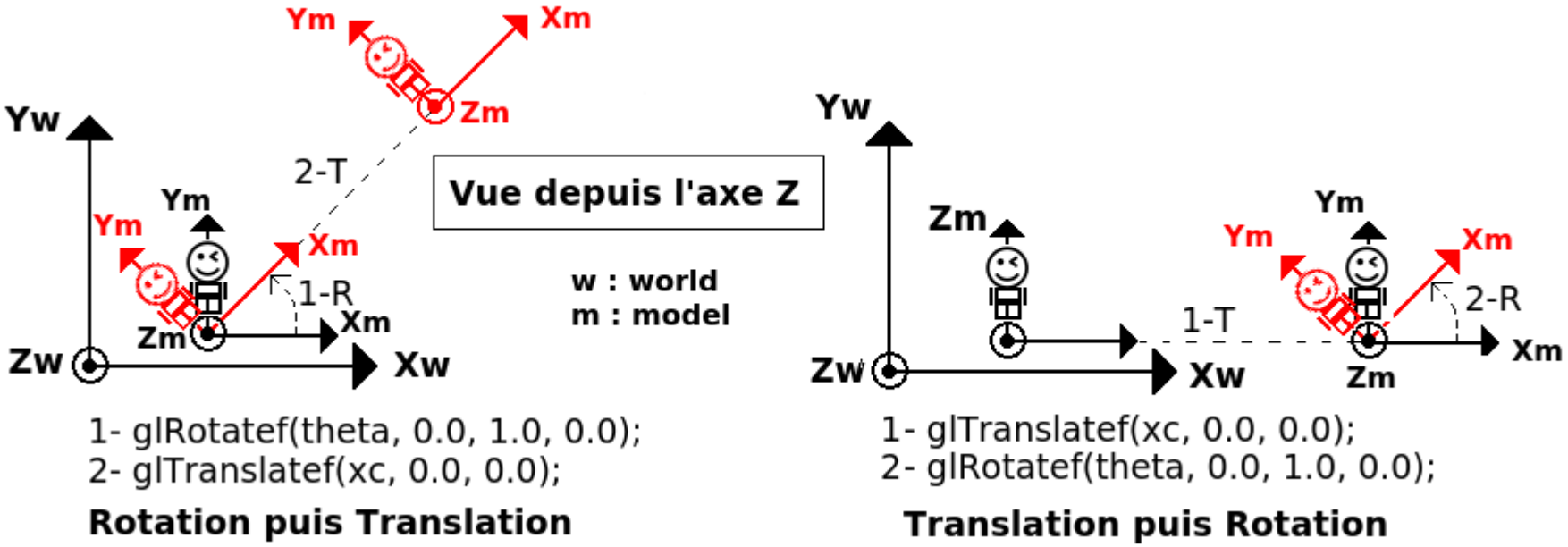


2) Transformations

➡ Quand on utilise **glTranslatef()**, on déplace l'objet 3D dans les directions des ases Xm, Ym et Zm de l'espace Om, Xm, Ym, Zm attaché au modèle.

⟲ Quand on utilise **glRotate()**, on fait pivoter l'objet 3D autour des axes xm, ym et zm de l'espace Om, Xm, Ym, Zm attaché au modèle.

La multiplication des matrices de transformation n'est pas commutative. Si on applique à un objet 3D une rotation glRotate suivi d'une translation glTranslate, le résultat n'est pas le même que si on lui applique une translation glTranslate suivie d'une rotation glRotate. Le schéma ci-après illustre cette non commutativité des matrices de transformation : $M_R * M_T \neq M_T * M_R$



1- glRotatef(theta, 0.0, 1.0, 0.0);
2- glTranslatef(xc, 0.0, 0.0);

Rotation puis Translation

1- glTranslatef(xc, 0.0, 0.0);
2- glRotatef(theta, 0.0, 1.0, 0.0);

Translation puis Rotation

3) Exemple - gl-test02.cpp

3.1) Code source

Dans le programme source suivant, on applique, à l'aide du clavier, 1-une transformation la rotation puis 2-une transformation de translation à un cube et à un triangle.

```
/*************************************************************************
* gl-test02.cpp  Test de OpenGL
* Déplacement dans le plan x,y et rotation d'un cube et d'un triangle
* en mode filaire (wireframe) - Touches flèches, espace et r (reset)
* compilation :
* g++ gl-test02.cpp -o test -lglut -lGLU -lGL
* gcc gl-test02.cpp -o test -lglut -lGLU -lGL
* Fonctionnement:
* OpenGL possède diverses fonctions qui finissent par le mot "func",
* (fonctions glutdisplayfunc, glutReshapeFunc et glutKeyboardFunc).
* Ces fonctions func prenent en paramètre le nom d'une fonction de rappel écrite par le
* programmeur(affiche, dimensionne, clavier, fleches)
* La fonction glutMainLoop() assure une scrutation en boucle de la survenue d'un évènement
* (clic de souris, appui d'une touche clavier...).
* Si un évènement survient, la fonction func correspondante est automatiquement appelée
* par  glutMainLoop(), ce qui déclenche l'exécutiont de la fonction de rappel correspondante
*************************************************************************/
#include <GL/freeglut.h>
#include <stdio.h>
float xc=0.0, yc=0.0; // abscisse et ordonnée du cube
float theta=0.0; // angle de rotation du cube
void affiche(void)
```

```
{
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();
glRotatef(theta, 0.0, 1.0, 0.0);// 1- rotation
glTranslatef(xc, yc, 0.0);// 2- Translation
//cube rouge
glColor3f(1.0f, 0.0f, 0.0f);
glutWireCube(1.0);
//triangle vert
GLfloat x0=0.5;GLfloat y0=0; GLfloat z0=0;
GLfloat x1=x0+1.0; GLfloat y1=y0+1.0;
glColor3f(0.0, 1.0, 0.0);
glBegin(GL_LINES);
  glVertex3f(x0, y0, z0); glVertex3f(x1, y1, z0);
  glVertex3f(x1, y1, z0); glVertex3f(x1, y0, z0);
  glVertex3f(x1, y0, z0);glVertex3f(x0, y0, z0);
glEnd();

//glFlush();
glutSwapBuffers();
}

void dimensionne(int w, int h)
{
glViewport(0, 0, w, h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-1.5, 1.5, -1.5, 1.5, 1.5, 20.0);
gluLookAt(0.0, 0.0, -3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
glMatrixMode(GL_MODELVIEW);
}
```

```
void clavier(unsigned char touche, int x, int y)
{
switch (touche)
{
case 'i':
  xc=0.0; yc=0.0; theta=0.0;
  glutPostRedisplay();
break;
case ' ':
  theta = theta+10.0;
  glutPostRedisplay();
break;
case 27:
  exit(0);
break;
default:
break;
}
}

void fleches(int touche, int x, int y)
{
if (touche == GLUT_KEY_UP) yc= yc+ 0.1;
if (touche == GLUT_KEY_DOWN) yc= yc- 0.1;
if (touche == GLUT_KEY_LEFT) xc= xc- 0.1;
if (touche == GLUT_KEY_RIGHT) xc= xc+ 0.1;
glutPostRedisplay();
}

void infos(void)
```

```
{
printf("Touche espace: rotation du cube autour de l'axe des y\n");
printf("Touches flèches: translation du cube sur les axes x et y \n");
printf("Touches i: retour du cube en position initiale\n");
}

int main(int argc, char **argv)
{
infos();
glutInit(&argc, argv);
//glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
glutInitWindowSize(300, 300);
glutInitWindowPosition(100, 100);
glutCreateWindow("gl-test02.cpp");
glClearColor(0.1, 0.1, 0.1, 0.0);//couleur de fond
glutDisplayFunc(affiche);
glutReshapeFunc(dimensionne);
glutKeyboardFunc(clavier);
glutSpecialFunc(fleches);
glutMainLoop();
}
```

3.2) Commentaires

● #include <GL/freeglut.h>
pour pouvoir utiliser OpenGL

● #include <stdio.h>
pour pouvoir utiliser l'instruction printf

● float xc=0.0, yc=0.0; // abscisse et ordonnée du cube
abscisse xc et ordonnée yc du cube, dans le plan z=0 de la scène 3D (world space)

● float theta=0.0; // angle de rotation du cube
angle de rotation du cube autour de l'axe y de la scène 3D

● void affiche(void)
Fonction utilisateur qui est appelée automatiquement (à travers la fonction glutDisplayFunc) lors du lancement du programme ou lors d'un appel de la fonction glutPostRedisplay();

● glClear(GL_COLOR_BUFFER_BIT);
La fonction glClear() met à zéro les buffers utilisés par OpenGL
Le paramètre GL_COLOR_BUFFER_BIT indique que les buffers courants sont activés pour le dessin des couleurs ;

● glLoadIdentity();
La fonction glLoadIdentity () est appelée, avant d'effectuer une transformation (glTranslate, glRotate…), lorsqu'on souhaite réinitialiser la matrice de transformation à son état d'origine (matrice identité).

On fait cet appel lorsqu'on souhaite qu'une transformation soit effectuée à l'aide de la matrice de transformation dans son état d'origine (et non pas l'aide de la matrice de transformation dans son état courant) .

● glTranslatef(xc, yc, -5.0);
La fonction glTranslatef() produit une translation dans la direction x, y,z.
En terme de calculs, la matrice de transformation courante reinitialisée par glLoadIdentity est multipliée par le vecteur de translation (x,y,z,1) pour donner la matrice de transformation suivante associée à glTranslate.
1 0 0 x
0 1 0 y
0 0 1 z
0 0 0 1
Si le mode de matrice courrant est GL_MODELVIEW ou GL_PROJECTION tous les objets dessinés après l'appel de glTranslate sont translatés.

● glRotatef(theta, 0.0, 1.0, 0.0);
La fonction glRotatef(theta, 0.0, 1.0, 0.0); produit une rotation d'un angle theta autour de l'axe y

● glColor3f(1.0, 0.0, 0.0);
fixe la couleur courante à la valeur rouge

● glutWireCube(1.0);
Dessine un cube filaire rouge de coté 1.0 (en unité relative)

● //triangle vert
● GLfloat x0=0.5;GLfloat y0=0; GLfloat z0=0;
● GLfloat x1=x0+1.0; GLfloat y1=y0+1.0;
● glColor3f(0.0, 1.0, 0.0);
● glBegin(GL_LINES);
  ● glVertex3f(x0, y0, z0); glVertex3f(x1, y1, z0);
  ● glVertex3f(x1, y1, z0); glVertex3f(x1, y0, z0);

● glVertex3f(x1, y0, z0);glVertex3f(x0, y0, z0);
● glEnd();
Dessine un triangle vert de coté 1 et (x0,y0)=(0.5,0)

● glutSwapBuffers();
Bascule le contenu du buffer arrière ou back buffer (situé en mémoire) dans le buffer avant ou front buffer (écran). A l'issue de l'opération, le back buffer est vide

● void dimensionne(int w, int h)
fonction utilisateur qui est appelée automatiquement (à travers la fonction glutReshapeFunc) lors du lancement du programme ou lors d'un redimensionnement de la fenêtre du progamme

● glViewport(0, 0, w, h);
La fonction glViewport(x, y, w, h) définie la fenêtre de vue (viewport) à l'écran.

- x, y : coin inférieur gauche du rectangle de la fenêtre de vue, en pixels (la valeur initiale est (0,0)) ;
- w,h : largeur et hauteur de la fenêtre de vue. Lorsqu'un contexte GL est d'abord attaché à une fenêtre, la largeur et la hauteur sont définies sur les dimensions de cette fenêtre.

● glMatrixMode(GL_PROJECTION);
Indique que les opérations matricielles suivantes vont s'appliquer à la pile de matrices de projection.
En effet,
glMatrixMode spécifie à quelle type de matrice s'appliquent les opérations qui vont suivre
GL_MODELVIEW
Applique les opérations de matrice suivantes à la pile de matrices modelview.
GL_PROJECTION
Applique les opérations matricielles suivantes à la pile de matrices de projection.
GL_TEXTURE
Applique les opérations de matrice suivantes à la pile de matrices de texture.
GL_COLOR
Applique les opérations matricielles suivantes à la pile de matrices couleur.

● glLoadIdentity();
Réinitialise la matrice de transformation à son état d'origine (matrice identité)

● glFrustum(-1.5, 1.5, -1.5, 1.5, 1.5, 20.0);
La fonction glFrustum(left, right, bottom, top, near, far) effectue une transformation de projection de la scène 3D

- left, right : spécifie les coordonnées des plans de coupure (clipping) verticaux gauche et droit ;
- bottom, top : spécifie les coordonnées des plans de coupre horizontaux inférieur et supérieur ;
- near, far : spécifie les distances par rapport aux plans de coupre de profondeur proche et éloignée. Les deux distances doivent être positives.

En effet, la camera qui permet d'observer la scène 3D se caractérise par son angle de vision, son aspect ratio (largeur de vue / hauteur de vue) et par ses plans de coupure proche et éloignés. La transformation de projection consiste à ne conserver que les coordonnées des points (décrits dans l'espace de la caméra) qui peuvent être effectivement vues par la caméra compte tenu des particularités de cette camera (clipping). Le clipping (en français "coupure") est l'opération qui supprime toutes les parties d'une scène 3D qui se situent hors du champs de vision de la caméra (ce champs de vision se limite à un espace en forme de cône rectangulaire tronqué).

● gluLookAt(0.0, 0.0, -3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

● glMatrixMode(GL_MODELVIEW);
Indique que les opérations matricielles suivantes vont s'appliquer à la pile de matrices de vue.

● void clavier(unsigned char touche, int x, int y)
Fonction utilisateur qui est appelée automatiquement (à travers la fonction glutKeyboardFunc) lors de l'appui d'une touche du clavier
Si la touche du clavier qui est appuyée est la touche i, on remet à zéro les coordonnées xc et yc du cube dans la vue de la caméra, c'est à dire dans l'espace de la caméra ainsi que la valeur de sa rotation autour de l'axe y dans cet espace.

Si la touche du clavier qui est appuyée est la touche espace, on augmente de 10 degrés l'angle de rotation du cube autour de l'axe y, dans la vue de la caméra.

Si la touche du clavier qui est appuyée est la touche escape, on referme la fenêtre OpenGL et on met fin au programme.

● void fleches(int touche, int x, int y)
Fonction utilisateur qui est appelée automatiquement (à travers la fonction glutSpecialFunc) lors de l'appui d'une touche spéciale du clavier

Si la touche du clavier qui est appuyée est la touche flèche haute, on augmente de 0.1 (grandeur exprimée en unité relative) l'ordonnée yc du cube dans l'espace de la caméra.

Si la touche du clavier qui est appuyée est la touche flèche basse, on diminue de 0.1 (grandeur exprimée en unité relative) l'ordonnée yc du cube dans l'espace de la caméra.

Si la touche du clavier qui est appuyée est la touche flèche gauche, on diminue de 0.1 (grandeur exprimée en unité relative) l'abscisse xc du cube dans l'espace de la caméra.

Si la touche du clavier qui est appuyée est la touche flèche droite, on augmente de 0.1 (grandeur exprimée en unité relative) l'abscisse xc du cube dans l'espace de la caméra.

glutPostRedisplay();
Provoque un appel automatique de la fonction glutDisplayFunc() et par voie de conséquence de la fonction affiche() qui redessine le cube dans sa nouvelle position dans l'espace de la caméra (camera space ou view space)

● void infos(void)
Fonction uitlisateur qui affiche des informations dans la fenêtre du Terminal de commande de Linux.

● int main(int argc, char **argv)
Programme principal

Initialisations
infos();
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
glutInitWindowSize(400, 400);
glutInitWindowPosition(400, 400);
glutCreateWindow("gl-test02.cpp");
glClearColor(0.1, 0.1, 0.1, 0.0);//couleur de fond

Fonctions utilisateurs
glutDisplayFunc(affiche);
glutReshapeFunc(dimensionne);
glutKeyboardFunc(clavier);
glutSpecialFunc(fleches);

Boucle de scrutation évènementielle
glutMainLoop();

● int main(int argc, char **argv)
Programme principal

Initialisations

● infos();
● glutInit(&argc, argv);
Intialisation de OpenGL

● glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

Initialisation du mode graphique RGBA, avec double buffer

- glutInitWindowSize(300, 300);
- glutInitWindowPosition(100, 100);
- glutCreateWindow(argv[0]);
- glClearColor(0.1, 0.1, 0.1, 0.0);

Création d'une fenêtre de programme de 300x300 pixels, de fond noir et dont le coin haut gauche est placé en (100,100) sur l'écran

Fonctions utilisateurs

glutDisplayFunc(affiche);
glutReshapeFunc(dimensionne);
glutKeyboardFunc(clavier);
glutSpecialFunc(fleches)

Boucle de scrutation évènementielle
glutMainLoop();

Fin du programme
return EXIT_SUCCESS;
}