

Maillage 2D 3D

Dr Wissal DRIRA

wissaldrira@yahoo.com



Plan du cours

- I. INTRODUCTION
- II. LA BIBLIOTHÈQUE GRAPHIQUE GLUT
- III. GESTION DES ÉVÈNEMENTS DU CLAVIER ET DE LA SOURIS
- IV. VISUALISATION ET CAMÉRA
- V. TEXTURE
- VI. ÉCLAIRAGE ET MATÉRIAUX

Plan du cours

- I. INTRODUCTION
- II. LA BIBLIOTHÈQUE GRAPHIQUE GLUT
- III. GESTION DES ÉVÈNEMENTS DU CLAVIER ET DE LA SOURIS
- IV. VISUALISATION ET CAMÉRA
- V. TEXTURE
- VI. ÉCLAIRAGE ET MATÉRIAUX

Introduction

- OpenGL (Open Graphic Library) est une interface de programmation graphique, contenant un ensemble normalisé de fonctions de calcul d'images 3D (et aussi 2D),
- OpenGL est lancé par Silicon Graphics, en 1992
- Une implémentation libre de droits d'OpenGL a été écrite par Brian Paul : <http://www.mesa3d.org>. Son code source est accessible.
 - ➔ Indépendante de la plate-forme matérielle, et du système d'exploitation.
 - ➔ Utilisée dans des applications variées : scientifiques, industrielles, artistiques 3D et 2D vectorielles, jeux vidéo ...

Introduction

- OpenGL est une interface de programmation graphique qui permet de visualiser une scène 3D (et aussi 2D) complexes à partir de simples primitives géométriques.
- Fréquemment abrégé "GL", Préfixe des fonctions: gl
- Elle possède des attributs graphiques : paramètres de réflexion, couleur, texture. L'éclairage de la scène est réalisé par des lumières de différents types (spot, lumière à l'infini).
- Les objets de cette scène peuvent être composés de points, de lignes, de polygones, de quadriques, de nurbs(Non Uniform Rational B-Spline)
- L'interface regroupe environ 250 fonctions différentes

Introduction

- Plusieurs bibliothèques sont développées à partir d'OpenGL afin d'apporter des fonctionnalités non-disponibles dans la bibliothèque OpenGL elle-même : GLU (OpenGL Utility Library), GLUT (OpenGL Utility Toolkit), GLUI, GLEW, GLEE, GLFW, GLM
- OpenGL ne gère pas les opérations de bas niveau
- GLU : OpenGL Utility Library contient des routines pour gérer les matrices de transformation, de projection et de visualisation, la facettisation des polygones et le rendu de surface, la gestion d'objets quadriques, des courbes et des surfaces NURBS, ...
- OpenGL ne gère pas le fenêtrage, et les entrées/sorties avec le clavier ou la souris
- GLUT: OpenGL Utility Toolkit, une bibliothèque offrant un jeu de routines pour la gestion des fenêtres OpenGL et les interactions avec le système d'exploitation (gestion clavier, souris, etc.): fonctions ayant pour préfixe glut

Variables d'état

- OpenGL est une state-machine : Des instructions existent pour placer le système dans certains états.
- Ceux-ci resteront actifs(utilisés à cette valeur) jusqu'à être changés(couleurs de tracé, matrices de transformation représentatives du(des) repère(s) courant(s), activation de l'élimination des parties cachées, caméra, lumières, matériel, texture,...).
- Chaque fois que le dessin d'un objet est réalisé(sommet, segment de droite, facette), l'ensemble de ces « variables d'états » définit la manière avec laquelle l'objet est dessiné.
- L'ensemble des variables d'états constitue l'environnement OpenGL.
- Toute variable d'état possède une valeur par défaut configurée à l'amorçage du programme utilisant OpenGL.

Fonctions de manipulation directe des variables d'état

W DRIRA

- **glEnable(GLenum pname)**
 - Activation d'une variable d'état booléenne
 - pname: variable d'état à activer
 - Exemple
 - glEnable(GL_DEPTH_TEST)-> activation de l'élimination des parties cachées
- **glDisable(GLenum pname)**
 - Inhibition d'une variable d'état booléenne
 - pname: variable d'état à désactiver
- **GLboolean glIsEnabled(GLenum pname)**
 - Consultation de variables d'état booléennes
 - pname: variable d'état consultée
 - Exemple
 - glEnable(GL_DEPTH_TEST);
 - GLboolean v = glIsEnabled(GL_DEPTH_TEST);

Elimination des surfaces cachées

W DRIRA

- OpenGL utilise la technique du Z buffer (ou buffer de profondeur) pour éviter l'affichage des surfaces cachées. On ne voit d'un objet que les parties qui sont devant et pas celles qui se trouvent derrière.
- Pour chaque élément de la scène la contribution qu'il aurait à l'image s'il était visible est calculée, et est stockée dans le Z buffer avec la distance de cet élément à la caméra (c'est cette distance qui est la profondeur).
- Chaque pixel de l'image garde donc la couleur de l'élément qui est le plus proche de la caméra. Dans le cas plus complexe d'un objet transparent, il y a une combinaison des couleurs de plusieurs éléments.

Elimination des surfaces cachées

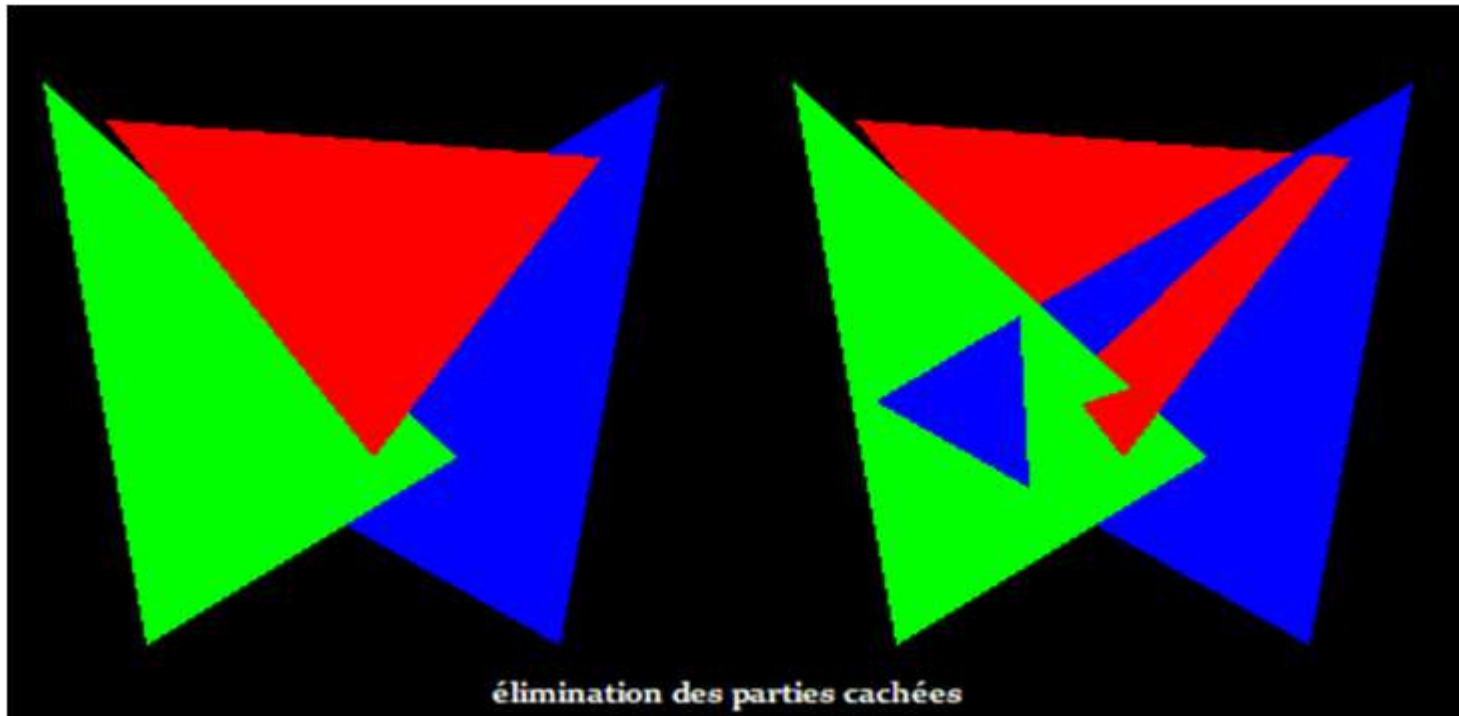
W DRIRA

- Fonctions utilisées:

glutInitDisplayMode(GLUT_DEPTH | ...) // Initialiser les buffers

glEnable(GL_DEPTH_TEST) // Activer les buffers

glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT) // Effacer les buffers



Plan du cours

- I. INTRODUCTION
- II. LA BIBLIOTHÈQUE GRAPHIQUE GLUT**
- III. GESTION DES ÉVÈNEMENTS DU CLAVIER ET DE LA SOURIS
- IV. VISUALISATION ET CAMÉRA
- V. TEXTURE
- VI. ÉCLAIRAGE ET MATÉRIAUX

Utilisation de GLUT

W DRIRA

- Installation CodeBlocks
- Configuration de la bibliothèque GLUT : Suivez les étapes détaillées dans le fichier *config.txt*
- Génération du programme de base GLUT : Lisez-le attentivement, et comprenez la définition et les effets de chacune des fonctions OpenGL et GLUT qui le composent: Pour construire des scènes complexes, il est important que les bases soient solides.

Fonctionnalités principales

W DRIRA

Les fonctions de gestion d'une fenêtre

- glutInit(int * argc, char **argv) initialise GLUT et traite les arguments de la ligne de commande. (Voir [Exp 1](#))
- glutInitDisplayMode(unsigned int mode) définit le mode de fonctionnement de la bibliothèque et initialise les buffers utilisés, (simple/double buffering, codage des couleurs couleurs RVB / couleurs indexées). Exp: `glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);`
- glutInitWindowPosition(int x, int y) spécifie la localisation dans l'écran du coin haut gauche de la fenêtre. (Voir [Exp 2](#))
- glutInitWindowSize(int width, int size) spécifie la taille en pixels de la fenêtre. (Voir [Exp 2](#))
- glutCreateWindow(char * string) crée une fenêtre contenant un contexte OpenGL et renvoie l'identificateur de la fenêtre. La fenêtre ne sera affichée que lors du premier glutMainLoop() (Voir [Exp 1](#))

Fonctionnalités principales

W DRIRA

Lancement de la boucle d'événements

- Attente d'un événement
 - Traitement de l'événement reçu
 - Retour en 1.
- glutMainLoop(void) lance la boucle principale qui tourne pendant tout le programme, attend et appelle les fonctions spécifiques pour traiter les événements.

La fonction d'affichage

- glutDisplayFunc(void (*func)(void)) spécifie la fonction à appeler pour l'affichage
- glutPostRedisplay(void) permet de forcer le réaffichage de la fenêtre (En plaçant en file d'attente d'événements un ordre de rafraîchissement de la fenêtre d'affichage active)

Fonctionnalités principales

Traitement des événements

- glutReshapeFunc(void (*func)(int w, int h)) spécifie la fonction à appeler lorsque la fenêtre est retaillée.
- glutKeyboardFunc(void (*func)(unsigned char key, int x, int y)) spécifie la fonction à appeler lorsqu'une touche ASCII du clavier est pressée
- glutSpecialFunc(void (*func)(int key, int x, int y)) spécifie la fonction à appeler lorsqu'une touche spéciale (caractère non-ASCII) est pressée : les flèches, les touches de fonction, etc...
- glutMouseFunc(void (*func)(int button, int state, int x, int y)) spécifie la fonction à appeler lorsqu'un bouton de la souris est pressé.
- glutMotionFunc(void (*func)(int x, int y)) spécifie la fonction à appeler lorsque la souris est déplacée tout en gardant un bouton appuyé.

Fonctionnalités principales

W DRIRA

Exécution en tâche de fond

- glutIdleFunc(void (*func)(void)) spécifie la fonction à appeler lorsqu'il n'y a pas d'autre événement: Exécution d'actions en tâche de fond
➔ Cette fonction est fréquemment utilisée pour programmer des animations

Animation

Une animation peut être implantée de diverses manières :

- glFlush Animation avec tampon simple: forcer l'exécution des commandes GL en temps fini
- glutSwapBuffers Animation avec tampon double: assurer l'échange entre les deux buffers
- glutTimerFunc(unsigned int msec, void (*func)(int value), value) enregistre la fonction de rappel associée au timer à déclencher en un nombre spécifié de millisecondes.

glutTimerFunc(100, MyTimer, 1); Après 100ms exécuter la fonction MyTimer de valeur 1

Fonctionnalités principales

W DRIRA

Dessin d'objets Tridimensionnels

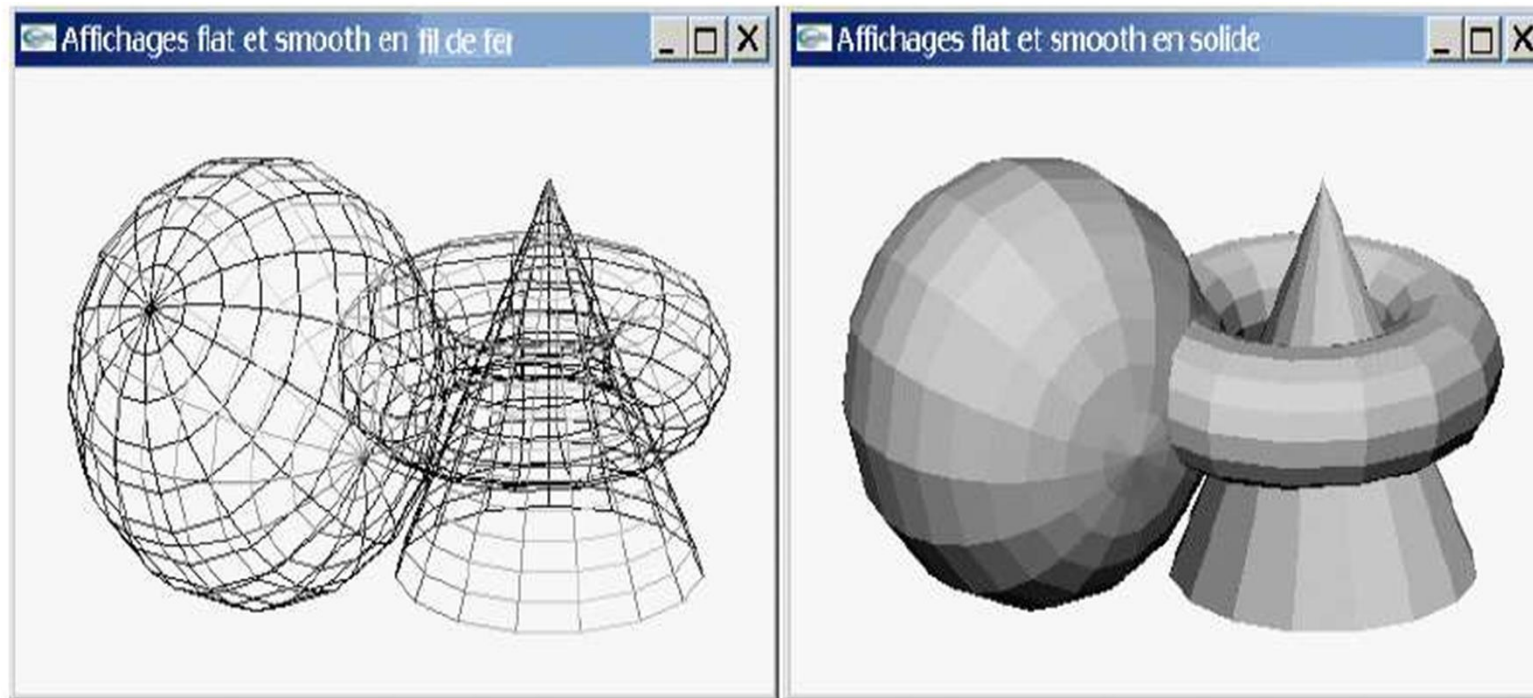
- GLUT possède des routines pour afficher les objets suivants :
 - Cube : [glutSolidCube](#)(Gldouble size) ;
[glutWireCube](#)(Gldouble size) ; (Voir [Exp 4](#))
 - Tétraèdre
 - Octaèdre
 - Icosaèdre
 - Dodécaèdre
 - Sphère
 - Tore
 - Cone
 - Teapot

Fonctionnalités principales

W DRIRA

Affichages OpenGL en fil de fer et en surfacique

- Dans le mode de dessin en fil de fer (Wireframe), les objets sont représentés par dessin de segments de droite. Exp `glutWireCube(Gldouble size)` ;
- Dans le mode de dessin en surfacique (Solid), les objets sont représentés par dessin de surfaces. Exp `glutSolidCube(Gldouble size)` ;



Fonctionnalités principales

W DRIRA

- `glutWireSphere(GLdouble radius, GLint slices, GLint stacks);`
- `glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);`
- `glutWireCone(GLdouble base, GLdouble height, GLint slices, GLint stacks);`
- `glutSolidCone(GLdouble base, GLdouble height, GLint slices, GLint stacks);`
- `glutWireCube(GLdouble size);`
- `glutSolidCube(GLdouble size);`
- `glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint slices, GLint stacks);`
- `glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint slices, GLint stacks);`

Fonctionnalités principales

W DRIRA

- `glutWireDodecahedron();`
- `glutSolidDodecahedron();`
- `glutWireTeapot(GLdouble size);`
- `glutSolidTeapot(GLdouble size);`
- `glutWireOctahedron();`
- `glutSolidOctahedron();`
- `glutWireTetrahedron();`
- `glutSolidTetrahedron();`
- `glutWireIcosahedron();`
- `glutSolidIcosahedron();`

Fonctionnalités principales

W DRIRA

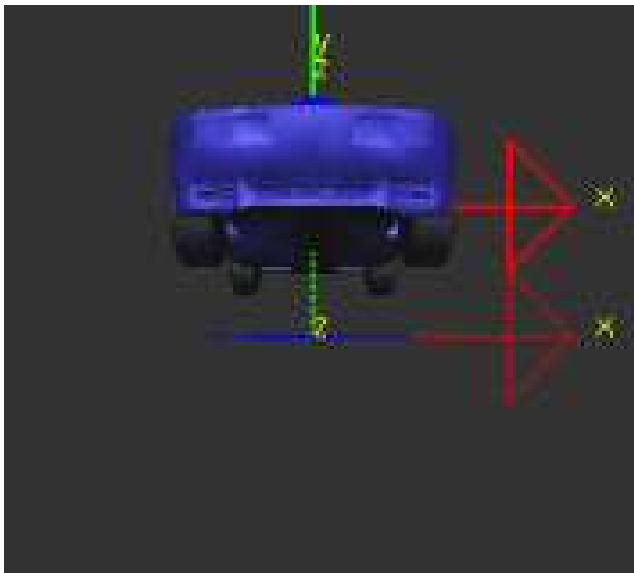
Les couleurs

- Les couleurs sont définies en OpenGL de deux manières :
 - Couleurs indexées : 256 couleurs sont choisies, et on se réfère au numéro de la couleur (son index): C'est un mode qui était intéressant lorsque les écrans d'ordinateurs ne savaient afficher que 256 couleurs simultanées.
 - Couleurs RVBA : une couleur est définie par son intensité sur 3 composantes: Rouge, Vert et Bleu. La quatrième composante est appelée canalAlpha, et code l'opacité.
- La couleur d'un objet est spécifiée par l'appel à glColor(...). (Voir [Exp 4](#))
 - **void glColor3{b s i f d ub us ui} (TYPE r,TYPE v,TYPE b);**
 - **void glColor4{b s i f d ub us ui} (TYPE r,TYPE v,TYPE b,TYPE alpha);**
- r, v, b, alpha: composantes colorées élémentaires de la couleur de tracé
- La couleur choisie est active pour tous les objets créés après spécification, jusqu'à nouvelle spécification.

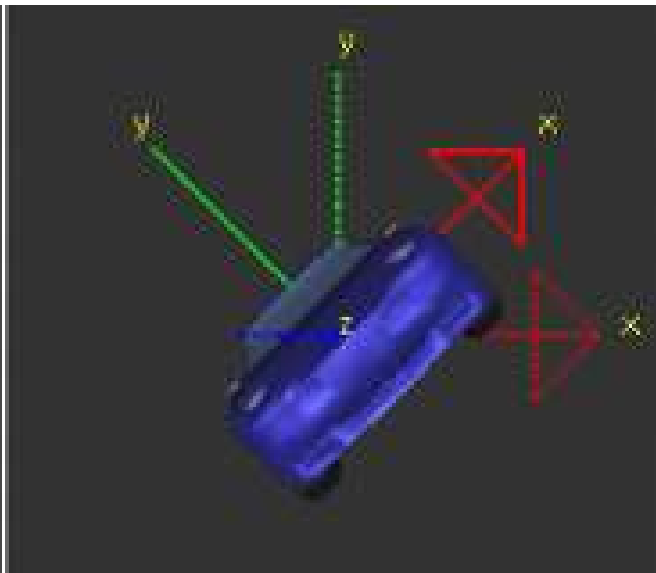
Fonctionnalités principales

W DRIRA

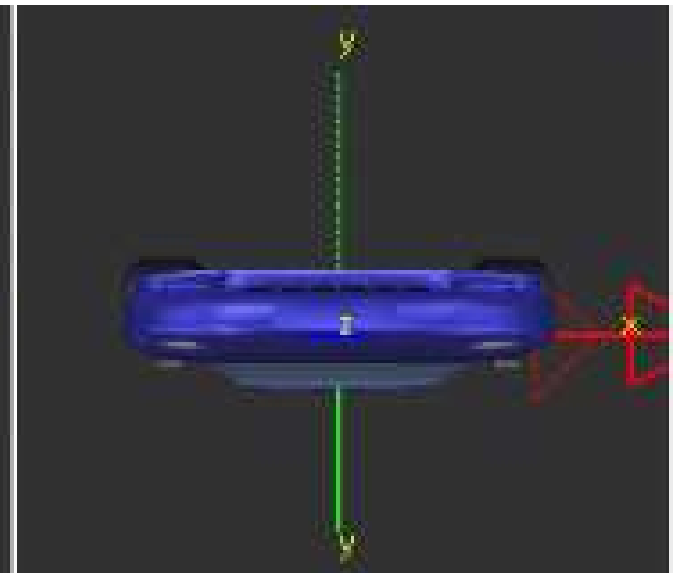
- **glTranslate*(TYPE x,TYPE y,TYPE z)** : translate le repère local de l'objet du vecteur (x,y,z).
- **glRotate*(TYPE α ,TYPE x,TYPE y,TYPE z)** : opère une rotation de l'objet d'angle α autour du vecteur (x,y,z).
- **glScale*(TYPE a,TYPE b,TYPE c)** : opère un changement d'échelle sur 3 axes: Les coordonnées en x sont multipliées par a, en y par b et en z par c



`glTranslatef(0.0, 0.5, 0.0)`



`glRotatef(45, 0.0 , 0.0 , 1.0)`



`glScalef(1.5 , -0.5 , 1.0)`

Fonctionnalités principales

W DRIRA

Pile de transformations

- OpenGL utilise les coordonnées projectives pour manipuler ses objets: Il maintient un ensemble limité de matrices pour contenir les différentes transformations:
 - ➔ Pour coder l'arbre de description de la scène, il faut utiliser la pile de transformation, en empilant la matrice de transformation courante (sauvegarde des caractéristiques du repère local associé) avant de descendre dans chaque nœud de l'arbre, et en dépilant la matrice en remontant (récupération du repère local associé).
- **glPushMatrix()** Empile la matrice courante pour sauvegarder la transformation courante.
- **glPopMatrix()** Dépille la matrice courante (La matrice du haut de la pile est supprimée de la pile, et elle devient la matrice courante)
 - Exemple: (*Voir [Exp 17](#)*)

```
glPushMatrix(); glRotated(45,1,1,0); glTranslated(-1,-1,0); glColor3d(1,1,1); glutSolidCube(1); glPopMatrix();  
glPushMatrix(); glTranslated(0.7,0.7,-0.4); glColor3d(0,1,1); glutSolidSphere(0.5,20,20); glPopMatrix();
```

Example 1: Programme Glut basique

W DRIRA

```
#include <stdio.h>
#include <GL/glut.h>

void Display(void) {

}

int main(int argc, char *argv[])
{
    glutInit(&argc, argv);

    glutCreateWindow("This is the window title"); // Create main Window

    glutDisplayFunc(Display); // Set CallBack Window

    glutMainLoop(); // Window Message Loop

    return 0;
}
```



Example 2 : Changer la taille et la position du fenêtre

W DRIRA

```
#include <GL/glut.h>

void Display(void) {

}

int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitWindowSize(300,400);
    glutInitWindowPosition(100,100);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glutMainLoop();
    return 0;
}
```



Example 3 : Changer la couleur de la fenêtre en rouge

W DRIRA

```
#include <GL/glut.h>

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
}

int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(1,0,0,0);
    glutMainLoop();
    return 0;
}
```



Example 4 : Dessiner un cube vert

W DRIRA

```
#include <GL/glut.h>

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3d(0,1,0);    // glColor3f(0.0,1.0,0.0); // glColor3f(0.0f,1.0f,0.0f);
    glutSolidCube(1);    // glRotated(45,1,1,0); glutWireCube(1);
    glFlush();
}

int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutCreateWindow("Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(1,0,0,0);
    glutMainLoop();
    return 0;
}
```



Example 5 : Définir un point Blanc/Rouge à l'origine

W DRIRA

```
#include <GL/glut.h>

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(10);
    glBegin(GL_POINTS);
    glColor3d(1,0,0);
    glVertex3f(0.5, 0, 0.5); // glVertex3f(0,0,0); // au centre du fenêtre
    glEnd();
    glFlush();
}

int main(int argc, char *argv[]){
    glutInit(&argc, argv);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(0,0,1,0);
    glutMainLoop();
    return 0;
}
```



Exemple 6 : Définir plusieurs points aux coins de l'écran

W DRIRA

```
#include <GL/glut.h>

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(10);
    glBegin(GL_POINTS);
        glColor3f(1,0,0);
        glVertex3f(0,0,0);
        glVertex3f(1,1,0); //glVertex3f(0.9,0.9,0);
        glVertex3f(-1,1,0);
        glVertex3f(1,-1,0);
        glVertex3f(-1,-1,0);
    glEnd();
    glFlush();
}

int main(int argc, char *argv[]){
    glutInit(&argc, argv);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(0,0,1,0);
    glutMainLoop();
    return 0;
}
```



Example 7 : Tracer un Cercle de 10 points

W DRIRA

```
#include <GL/glut.h>
#include <math.h>

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(10);
    glBegin(GL_POINTS);
        glColor3f(1,0,0);
        glVertex3f(cos(2*3.14159*0/10),sin(2*3.14159*0/10),0);
        glVertex3f(cos(2*3.14159*1/10),sin(2*3.14159*1/10),0);
        glVertex3f(cos(2*3.14159*2/10),sin(2*3.14159*2/10),0);
        glVertex3f(cos(2*3.14159*3/10),sin(2*3.14159*3/10),0);
        glVertex3f(cos(2*3.14159*4/10),sin(2*3.14159*4/10),0);
        glVertex3f(cos(2*3.14159*5/10),sin(2*3.14159*5/10),0);
        glVertex3f(cos(2*3.14159*6/10),sin(2*3.14159*6/10),0);
        glVertex3f(cos(2*3.14159*7/10),sin(2*3.14159*7/10),0);
        glVertex3f(cos(2*3.14159*8/10),sin(2*3.14159*8/10),0);
        glVertex3f(cos(2*3.14159*9/10),sin(2*3.14159*9/10),0);
        glVertex3f(cos(2*3.14159*10/10),sin(2*3.14159*10/10),0);
    glEnd();
    glFlush();
}
```

```
int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutCreateWindow("Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(0,0,1,0);
    glutMainLoop();
    return 0;
}
```



Example 7' : Tracer un Cercle de 100 points

W DRIRA

```
#include <GL/glut.h>
#include <math.h>

void Display(void) {
    int i;
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(10);
    glBegin(GL_POINTS);
        glColor3f(1,0,0);
        for(i=0;i<100;++i){ glVertex3f(cos(2*3.14159*i/100),sin(2*3.14159*i/100),0); }
    glEnd();
    glFlush();
}

int main(int argc, char *argv[]){
    glutInit(&argc, argv);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(0,0,1,0);
    glutMainLoop();
    return 0;
}
```



Example 8 : Dessiner une ligne

W DRIRA

```
#include <GL/glut.h>

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_LINES);

        glVertex3f(-1,0,0);
        glVertex3f(1,0,0);
        //glVertex3f(0.5,-0.5,0); glVertex3f(-0.5,0.5,0);

    glEnd();
    glFlush();
}

int main(int argc, char *argv[]){
    glutInit(&argc, argv);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(0,0,1,0);
    glutMainLoop();
    return 0;
}
```



Example 9 : Dessiner une ligne Strip

W DRIRA

```
#include <GL/glut.h>

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_LINE_STRIP); //glBegin(GL_LINES);
        glVertex3f(-1,0,0);
        glVertex3f(0,0,0);
        glVertex3f(0,1,0);
        glVertex3f(1,0,0);
    glEnd();
    glFlush();
}

int main(int argc, char *argv[]){
    glutInit(&argc, argv);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(0,0,1,0);
    glutMainLoop();
    return 0;
}
```



Example 10 : Dessiner une ligne

Loop

W DRIRA

```
#include <GL/glut.h>

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_LINE_LOOP); //glBegin(GL_LINE_STRIP);
        glVertex3f(-1,0,0);
        glVertex3f(0,0,0);
        glVertex3f(0,1,0);
        //glVertex3f(0.5,0.5,0);
    glEnd();
    glFlush();
}

int main(int argc, char *argv[]){
    glutInit(&argc, argv);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(0,0,1,0);
    glutMainLoop();
    return 0;
}
```



Example 11 : Dessiner des triangles

W DRIRA

```
#include <GL/glut.h>
```

```
void Display(void) {  
    glClear(GL_COLOR_BUFFER_BIT);  
    glBegin(GL_TRIANGLES);  
        glColor3f(1,1,1);  
        glVertex3f(0,0,0);  
        glVertex3f(-1,0,0);  
        glVertex3f(0,1,0);  
  
        glColor3f(1,0,0);  
        glVertex3f(0,0,0);  
        glVertex3f(1,0,0);  
        glVertex3f(0,-1,0);  
    glEnd();  
    glFlush();  
}
```

```
int main(int argc, char *argv[])  
{  
    glutInit(&argc, argv);  
    glutCreateWindow("Ma fenetre");  
    glutDisplayFunc(Display);  
    glClearColor(0,0,1,0);  
    glutMainLoop();  
    return 0;  
}
```



Example 12 : Dessiner un triangle Strip

W DRIRA

```
#include <GL/glut.h>
```

```
void Display(void) {  
    glClear(GL_COLOR_BUFFER_BIT);  
    glBegin(GL_TRIANGLE_STRIP);  
        glColor3f(0,1,0);  
        glVertex3f(-1,0,0);  
        glVertex3f(0,0,0);  
        glVertex3f(0,1,0);  
  
        glColor3f(1,0,0);  
        glVertex3f(0.5,0.5,0);  
  
        glColor3f(1,1,0);  
        glVertex3f(0.5,-0.5,0);  
    glEnd();  
    glFlush();  
}
```

```
int main(int argc, char *argv[])  
{  
    glutInit(&argc, argv);  
    glutCreateWindow("Ma fenetre");  
    glutDisplayFunc(Display);  
    glClearColor(0,0,1,0);  
    glutMainLoop();  
    return 0;  
}
```



Example 13 : Dessiner un triangle

Fan

W DRIRA

```
#include <GL/glut.h>
```

```
void Display(void) {  
    glClear(GL_COLOR_BUFFER_BIT);  
    glBegin(GL_TRIANGLE_FAN);  
        glColor3f(0,1,0);  
        glVertex3f(0,0,0);  
        glVertex3f(1,0,0);  
        glVertex3f(0,1,0);  
  
        glColor3f(0,0,0);  
        glVertex3f(-1,0,0);  
  
        glColor3f(1,0,0);  
        glVertex3f(-0.5,-0.5,0);  
    glEnd();  
    glFlush();  
}
```

```
int main(int argc, char *argv[])  
{  
    glutInit(&argc, argv);  
    glutCreateWindow("Ma fenetre");  
    glutDisplayFunc(Display);  
    glClearColor(0,0,1,0);  
    glutMainLoop();  
    return 0;  
}
```



Example 14 : Dessiner un polygone

W DRIRA

```
#include <GL/glut.h>
```

```
void Display(void) {  
    glClear(GL_COLOR_BUFFER_BIT);  
    glColor3f(0,1,0);  
    glBegin(GL_POLYGON);  
        glVertex3f(0,0,0);  
        glVertex3f(0,1,0);  
        glVertex3f(1,0,0);  
        glVertex3f(0,-1,0);  
        glVertex3f(-1,0.5,0);  
    glEnd();  
    glFlush();  
}
```

```
int main(int argc, char *argv[])  
{  
    glutInit(&argc, argv);  
    glutCreateWindow("Ma fenetre");  
    glutDisplayFunc(Display);  
    glClearColor(0,0,1,0);  
    glutMainLoop();  
    return 0;  
}
```

Example 15 : Dessiner des Quadrilatères

W DRIRA

```
#include <GL/glut.h>
```

```
void Display(void) {  
    glClear(GL_COLOR_BUFFER_BIT);  
    glColor3f(0,1,0);  
    glBegin(GL_QUADS);  
  
        glVertex3f(0,0.5,0);  
        glVertex3f(0,0,0);  
        glVertex3f(-0.5,0,0);  
        glVertex3f(-0.5,0.5,0);  
  
        glColor3f(1,0,0);  
        glVertex3f(0.5,-0.5,0);  
        glVertex3f(0.5,0.3,0);  
        glVertex3f(-1,-0.5,0);  
        glVertex3f(-0.5,-1,0);  
    glEnd();  
    glFlush();  
}
```

```
int main(int argc, char *argv[])  
{  
    glutInit(&argc, argv);  
    glutCreateWindow("Ma fenetre");  
    glutDisplayFunc(Display);  
    glClearColor(0,0,1,0);  
    glutMainLoop();  
    return 0;  
}
```

Example 16 : Dessiner des Quadrilatères Strip

W DRIRA

```
#include <GL/glut.h>
```

```
void Display(void) {  
    glClear(GL_COLOR_BUFFER_BIT);
```

```
    glBegin(GL_QUAD_STRIP);  
        glColor3f(0,1,0);  
        glVertex3f(0,0.5,0);  
        glVertex3f(0,0,0);  
        glVertex3f(0.2,0.8,0);  
        glVertex3f(0.2,0,0);
```

```
    glColor3f(1,0,0);  
        glVertex3f(0.5,0.3,0);  
        glVertex3f(0.5,-0.5,0);
```

```
    glEnd();
```

```
    glFlush();
```

```
}
```

```
int main(int argc, char *argv[])  
{  
    glutInit(&argc, argv);  
    glutCreateWindow("Ma fenetre");  
    glutDisplayFunc(Display);  
    glClearColor(0,0,1,0);  
    glutMainLoop();  
    return 0;  
}
```


Example 17 : Dessiner plusieurs objets

W DRIRA

```
#include <GL/glut.h>

void Display(void) { glClear(GL_COLOR_BUFFER_BIT) ;
    glPushMatrix();
    glTranslated(-0.5,-0.5,0);glColor3d(0,1,1); glutSolidSphere(0.5,20,20);
    glPopMatrix();
    glPointSize(10);
    glBegin(GL_POINTS);
    glColor3d(1,0,0);glVertex3f(-0.4,-0.4,-0.2);
    glColor3d(1,1,0);glVertex3f(0,0,0);
    glEnd();
    glBegin(GL_TRIANGLES);
    glColor3d(0,1,0);
    glVertex3f(1,0,0);
    glVertex3f(0,0,0);
    glVertex3f(0,1,0);
    glEnd();
    glPushMatrix();
    glRotated(45,1,1,0);glTranslated(0.5,-0.5,0.5);glColor3d(0,0,0); glutSolidCube(0.5);
    glPopMatrix();
    glFlush();}
```

```
int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutCreateWindow("Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(1,1,1,0);
    glutMainLoop();
    return 0;
}
```

Example 18 : le Z-buffer

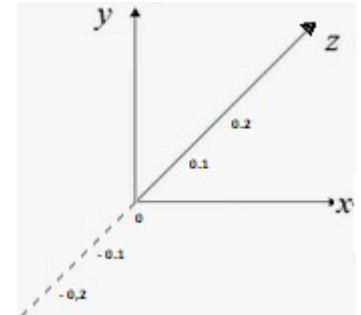
W DRIRA

```
#include <GL/glut.h>
void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

    glPointSize(10);
    glBegin(GL_POINTS);
        glColor3d(1,0,0);glVertex3f(0.4,0.4,-0.2); //glVertex3f(0.4,0.4,0.2);
        glColor3d(1,1,0);glVertex3f(0,0,-0.1);
    glEnd();

    glBegin(GL_TRIANGLES);
        glColor3d(0,1,0);

        glVertex3f(1,0,0);
        glVertex3f(0,0,0);
        glVertex3f(0,1,0);
    glEnd();
```



```
int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("Ma fenetre");
    glEnable(GL_DEPTH_TEST);
    glutDisplayFunc(Display);
    glClearColor(1,1,1,0);
    glutMainLoop();
    return 0;}
```

```
glutSwapBuffers(); // Utiliser le double buffering
}
```

Plan du cours

- I. INTRODUCTION
- II. LA BIBLIOTHÈQUE GRAPHIQUE GLUT
- III. **GESTION DES ÉVÈNEMENTS DU CLAVIER ET DE LA SOURIS**
- IV. VISUALISATION ET CAMÉRA
- V. TEXTURE
- VI. ÉCLAIRAGE ET MATÉRIAUX

Objectifs

Se familiariser avec la manipulation de :

- **Clavier**

- Gestion des événements Clavier : Touches normales
- Gestion des événements Clavier : Touches spéciales

- **Souris**

- Gestion des événements Souris

Gestion des événements Clavier : Touches normales

W DRIRA

- `void glutKeyboardFunc(void (*fonct) (unsigned char key, int x, int y))`
 - Définit la fonction **fonct** exécutée automatiquement par GLUT lorsqu'une touche ASCII du clavier est frappée. Au moment de son exécution, cette fonction recevra les paramètres:
 - key : le code ASCII de la touche de clavier tapée
 - x et y : les coordonnées de la souris (en pixels) par rapport au coin en haut à gauche de la fenêtre lors de la frappe d'une touche.
- `glutKeyboardFunc(NULL)` ; pour désactiver la fonction de rappel pour les touches normales du clavier.

Example 1: Gestion des événements

Clavier : Touches normales

W DRIRA

```
#include <stdio.h>
#include <GL/glut.h>
int WIDTH=640;
int HEIGHT=480;
int nFullScreen=0;

void tClavier(unsigned char key, int x, int y);

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity(); // Initialiser la matrice courante par la matrice d'identité
    glColor3d(0,1,0);
    glutSolidCube(0,5);
    glFlush(); }

int main(int argc, char *argv[]){
    glutInit(&argc, argv);
    glutInitWindowSize(WIDTH, HEIGHT);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(1,1,1,0);
    glutKeyboardFunc ( tClavier) ;
    glutMainLoop();
    return 0; }
```

Example 1 (Suite)

W DRIRA

```
void tClavier(unsigned char key, int x, int y)
{
    switch (key){
        case 'f' : //Mode plein écran
        case 'F' : if (nFullScreen==0){
                    glutFullScreen();
                    nFullScreen=1;    break;    }
                if (nFullScreen==1){
                    glutReshapeWindow(WIDTH,HEIGHT);
                    glutPositionWindow (100,100);
                    nFullScreen=0;    break;    }
        case 's' : //Supprimer le traitement des touches normales du clavier
        case 'S' : glutKeyboardFunc(NULL);  break;
        case 'q' : //Quitter
        case 'Q' :
        case 27 : // Touche ESC
                    exit (0);  break;
        default : //Afficher les coordonnées de la souris par rapport au coin haut
                    printf(" Touche: %c\n Souris a: %d %d \n",key,x,y);, break; }
    glutPostRedisplay();
}
```

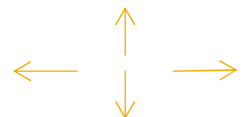
Gestion des événements Clavier : Touches spéciales

W DRIRA

- La fonction **glutSpecialFunc** gère les touches spéciales du clavier :

void glutSpecialFunc (void (*func) (int key, int x, int y));

- **func** identifie la fonction de rappel du clavier pour les caractères non-ASCII

(exemple les touches de fonction (F1 à F12) ou de direction ).

- int key, int x, int y sont les paramètres de cette fonction **func**
- key est une constante correspondant à une touche spéciale (GLUT_KEY_*)
- x et y les coordonnées de la souris (en pixels) par rapport au coin en haut à gauche de la fenêtre lors de la frappe d'une touche.

- *glutSpecialFunc(NULL);* pour désactiver la fonction de rappel pour les touches spéciales du clavier.

Gestion des événements Clavier:

Touches spéciales

W DRIRA

Touche	Valeur
F1	GLUT_KEY_F1
F2	GLUT_KEY_F2
F3	GLUT_KEY_F3
F4	GLUT_KEY_F4
F5	GLUT_KEY_F5
F6	GLUT_KEY_F6
F7	GLUT_KEY_F7
F8	GLUT_KEY_F8
F9	GLUT_KEY_F9
F10	GLUT_KEY_F10
F11	GLUT_KEY_F11
F12	GLUT_KEY_F12
←	GLUT_KEY_LEFT
↑	GLUT_KEY_UP
→	GLUT_KEY_RIGHT
↓	GLUT_KEY_DOWN
Page précédente (Pg up)	GLUT_KEY_PAGE_UP
Page suivante (Pg down)	GLUT_KEY_PAGE_DOWN
Origine (Home)	GLUT_KEY_HOME
Fin (end)	GLUT_KEY_END
Insertion (ins)	GLUT_KEY_INSERT

Example 2: Gestion des événements clavier : Touches spéciales

W DRIRA

```
#include <stdio.h>
#include <GL/glut.h>

float xhoriz = 0;
float yverti = 0;

void tSpecial(int key, int x, int y);

void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glColor3d(0,1,0);
    glTranslatef(xhoriz,yverti,0);
    glutSolidCube(0.1);
    glFlush();
}

int main(int argc, char *argv[]){
    glutInit(&argc, argv);
    glutInitWindowSize(700,580);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glClearColor(1,1,1,0);
    glutSpecialFunc(tSpecial);
    glutMainLoop();
    return 0; }
```

Example 2 (Suite)

W DRIRA

```
void tSpecial(int key, int x, int y) {  
    switch (key) {  
        case GLUT_KEY_UP :    // case GLUT_KEY_F1 :  
            yverti += 0.01; break;  
  
        case GLUT_KEY_DOWN : // case GLUT_KEY_F2 :  
            yverti -= 0.01; break;  
  
        case GLUT_KEY_LEFT :  // case GLUT_KEY_F1 :  
            xhoriz -= 0.01; break;  
  
        case GLUT_KEY_RIGHT : // case GLUT_KEY_F2 :  
            xhoriz += 0.01; break;  
        default :  
            printf(" Autre Touche Speciale\n "); break;  
    }  
    glutPostRedisplay();  
}
```

Gestion des événements souris

W DRIRA

- La fonction *glutMouseFunc* établit la fonction de rappel de la souris pour la fenêtre courante
void glutMouseFunc(void (*fonct)(int bouton,int etat, int x, int y));
 - Lorsqu'un utilisateur appuie ou relâche un des boutons de la souris, chaque gâchette (appui ou relâchement d'un bouton) engendre un appel à la fonction **fonct** de rappel de la souris.
 - '*bouton*' contient le nom du bouton qui a été pressé ou relâché : il peut prendre les valeurs : GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, ou GLUT_RIGHT_BUTTON.
 - '*etat*' indique si le bouton a été pressé (GLUT_DOWN) ou relâché (GLUT_UP)
 - '*x*' et '*y*' contiennent les coordonnées de la souris au moment de l'événement.

Example 3: Gestion des événements souris

W DRIRA

```
#include <stdio.h>
#include <GL/glut.h>

float xhoriz = 0;
float yverti = 0;

void sMouse(int button, int state, int x, int y);
void Display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glColor3d(0,1,0);
    glTranslatef(xhoriz,yverti,0);
    glutSolidCube(0.1);
    glFlush(); }

int main(int argc, char *argv[]) {
    glutInit(&argc, argv);
    glutInitWindowSize(700,580);
    glutCreateWindow(" Ma fenetre");
    glutDisplayFunc(Display);
    glutMouseFunc ( sMouse);
    glutMainLoop();
    return 0; }
```

Example 3 (Suite)

W DRIRA

```
void sMouse(int button, int state, int x, int y){
    printf("Souris a: %d %d \n",x,y);
    switch (button)    {
        case GLUT_LEFT_BUTTON :    printf("Bouton Gauche \n");
                                    if (state==GLUT_DOWN) { printf("Appui \n"); xhoriz-=0.1;}
                                    if (state==GLUT_UP) printf("Relachement \n");
                                    break;
        case GLUT_MIDDLE_BUTTON :    printf("Bouton du Milieu \n");
                                    if (state==GLUT_DOWN) printf("Appui \n");
                                    if (state==GLUT_UP) printf("Relachement \n");
                                    break;
        case GLUT_RIGHT_BUTTON :    printf("Bouton Droit \n");
                                    if (state==GLUT_DOWN) { printf("Appui \n"); xhoriz+=0.1;}
                                    if (state==GLUT_UP) printf("Relachement \n");
                                    break;
        default :printf("Erreur \n"); break;
    }
    glutPostRedisplay();
}
```

Gestion des événements souris

W DRIRA

- Si on veut avoir l'état de la souris en permanence, on peut par exemple utiliser *glutMotionFunc* ou *glutPassiveMotionFunc*
- La fonction *glutMotionFunc* gère les déplacements de la souris lorsqu'un ou plusieurs boutons sont appuyés (il y a un retour seulement si la souris bouge) :

```
void glutMotionFunc ( void (*func) (int x, int y) );
```

- Exemple

```
void Motion(int x, int y) { printf("Souris a: %d %d \n",x,y); }
```

L'appel à la fonction *glutMotionFunc* se fait dans la fonction *main*, après avoir créé la fenêtre, et avant le *glutMainLoop()*;

```
...  
glutMotionFunc(Motion);  
...  
glutMainLoop();
```

Gestion des événements souris

W DRIRA

- La fonction *glutPassiveMotionFunc* gère les déplacements de la souris lorsqu'aucun bouton n'est appuyé (il y a un retour seulement si la souris bouge) :

```
void glutPassiveMotionFunc ( void (*func) (int x, int y ) );
```

- Exemple

```
void PassiveMotion(int x, int y) { printf("Souris a: %d %d \n",x,y); }
```

L'appel à la fonction *glutPassiveMotionFunc* se fait dans la fonction *main*, après avoir créé la fenêtre, et avant le *glutMainLoop()*;

```
...  
glutPassiveMotionFunc(PassiveMotion);  
...  
glutMainLoop();
```


Gestion des événements souris

W DRIRA

- `glutMouseFunc(NULL);` pour désactiver la fonction de rappel pour la souris.
- `glutMotionFunc(NULL);` pour désactiver la fonction de rappel pour le déplacement de la souris avec au moins un bouton appuyé.
- `glutPassiveMotionFunc(NULL);` pour désactiver la fonction de rappel pour le déplacement de la souris avec aucun bouton appuyé.