Tutorial 1 : Initialisation d'OpenGL

Pour commencer, il faut créer une application win32 console. Créez ensuite un fichier *.cpp qui va contenir votre code source. Les includes nécessaires à ce programme sont : iostream, windows.h, math.h et glut.h. Les Libs sont : opengl32.lib, glut32.lib et glu32.lib.

```
#include<iostream> // entrées sorties et fonctions cpp
#include<windows.h> // pour gérer les fonctions windows
#include "glut.h" // pour utiliser les fonctions opengl
#include<math.h> // pour accéder aux fonctions math

#define M_PI 3.14159265358979323846264338327795 // défini pi s'il ne l'est pas dans math.h
float width = 640.0f, height = 480.0f; // largeur et longueur de la fenêtre
```

La fonction main sera :

```
int main(int argc, char* argv[]) {

glutInit(&argc,argv); // arguments passés à l'init de glut
// mode d'affichage GLUT_RGBA = 4 couleurs GLUT_DEPTH = avec Z-Buffer
// GLUT_DOUBLE = mode double buffering
glutInitDisplayMode(GLUT_RGBA|GLUT_DEPTH|GLUT_DOUBLE);
glutInitWindowSize(width, height); // taille de la fenêtre
glutInitWindowPosition(50, 50); // position de la fenêtre par rapport au coin en haut à gauche
glutCreateWindow("My Window"); // nom de la fenêtre

// On change la taille de la fenêtre
glutReshapeFunc(&changeSize);

// fonctions de glut à relier à votre application
glutKeyboardFunc(&GestionClavier); // gestion des touches du clavier
glutSpecialFunc(&GestionClavierSpe); // gestion des touches spéciales du clavier
glutPassiveMotionFunc(&GestionSouris); // gestion de la souris à n'importe quel moment
glutDisplayFunc(&myDisplay); // fonction d'affichage
glutIdleFunc(&idle); // fonction appelée tout le tps par opengl

myInit(); // vos initialisations des paramètres d'opengl
glutFullScreen(); // si vous voulez être en plein écran
glutMainLoop(); // boucle de l'application

return 0;
}
```

D'autres fonctions pour les entrées de la souris, Call-Back à placer dans le main :

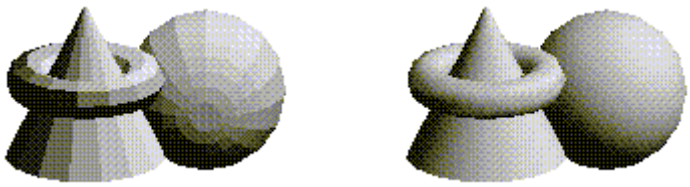
```
glutMouseFunc(&processMouse); // qd on appuie sur un bouton

/* Implémentation à placer dans le fichier *.cpp:
```

```
button :
GLUT_LEFT_BUTTON -> bouton gauche
GLUT_MIDDLE_BUTTON -> bouton du milieu
GLUT_RIGHT_BUTTON -> bouton droit

state :
GLUT_DOWN -> appuyer
GLUT_UP -> relacher */

void processMouse(int button, int state, int x, int y) {

if (state == GLUT_DOWN)
{
if (button == GLUT_LEFT_BUTTON) // on appuie sur le bouton gauche
}
}
```

Votre fonction d'initialisation peut ressembler à ça. Si vous voulez des détails sur les paramètres possibles de chaque fonction d'OpenGL, je vous conseille d'aller voir dans le fichier : glut.h.

```
void myInit() {

glPolygonMode(GL_FRONT_AND_BACK,GL_FILL); // mode d'affichage des polygones
// GL_FRONT_AND_BACK = affiche les 2 cotés
// GL_FILL = remplit les polygones
glShadeModel(GL_SMOOTH); // modèle d'illumination GL_FLAT

glEnable(GL_DEPTH_TEST); // activation du Z-Buffer
glDepthFunc(GL_LEQUAL); // gestion du Z-Buffer : GL_LESS GL_LEQUAL GL_ALWAYS
// G L_EQUAL GL_GREATER GL_NOTEQUAL GL_GEQUAL
glEnable(GL_COLOR_MATERIAL); // active la coloration des objets

glClearColor(0.0f,0.0f,0.0f,1.0f); // couleur du fond de la scène
// init de la matrice de projection
glViewport(0, 0, width, height);
glMatrixMode(GL_PROJECTION);
glLoadIdentity(); // init à la matrice identité
gluPerspective(45.0f, // angle d'ouverture de la caméra
width/height, // ratio de la fenêtre
0.1f,1000.0f // distance min et max de la scène
);
// init de la matrice de vue
glMatrixMode(GL_MODELVIEW);
glLoadIdentity(); // init à la matrice identité
gluLookAt(0.0f, 0.0f, 0.0f, // position oeil
0.0f,0.0f,1.0f, // point regardé
0.0f,1.0f,0.0f); // vecteur haut
}
```

Différence entre les modèles d'illumination :



GL_FLAT GL_SMOOTH

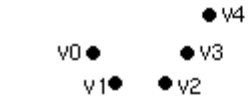
Fonctions annexes qui peuvent servir à communiquer avec l'application :

```
void idle(){
glutPostRedisplay(); // relance un affichage
}

void GestionClavier(unsigned char key, int x, int y) {
switch(key)
{
//touches clavier
case 27: //ESC
exit(0); // sortie du programme
break;
}
}

void GestionClavierSpe(int key, int x, int y) {
switch(key)
{
//touches spéciales clavier
case GLUT_KEY_UP:
break;
case GLUT_KEY_DOWN:
break;
case GLUT_KEY_RIGHT:
break;
case GLUT_KEY_LEFT:
break;
}
}

void GestionSouris(int x, int y)
{
//bouge la souris dont les coordonnées sont x et y
}
```

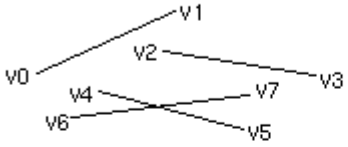
Et enfin la fonction d'affichage :

```
void myDisplay() {

// vide le Z-Buffer et le Buffer de couleur
glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);

glBegin(GL_TRIANGLES); // mode de polygones à afficher
// couleurs et points des vertexes à afficher
glColor4f(1.0f,0.0f,0.0f,1.0f); glVertex3f(0.0f, 1.0f, 10.0f);
glColor4f(0.0f,1.0f,0.0f,1.0f); glVertex3f(-1.0f, -1.0f, 10.0f);
glColor4f(0.0f,0.0f,1.0f,1.0f); glVertex3f(1.0f, -1.0f, 10.0f);
glEnd();

glutSwapBuffers(); // permute la surface primaire avec la secondaire
}

void changeSize(int w, int h) {

// re-init de la dimension de la fenêtre
width = w;
height = h;

glViewport(0, 0, width, height);

glMatrixMode(GL_PROJECTION);
glLoadIdentity(); // init à la matrice identité
gluPerspective(45.0f, // angle ouverture caméra
width/height, // ratio de la fenêtre
0.1f,1000.0f // distance min et max de la scène
);
}
```

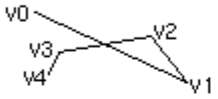
Plusieurs modes de polygones sont disponibles, en voici la liste :



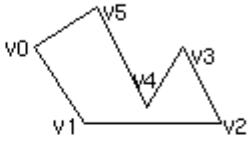
GL_POINTS



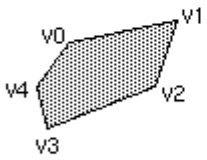
GL_LINES



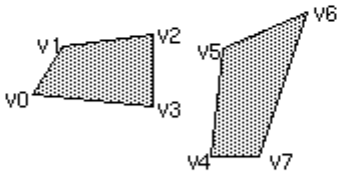
GL_LINE_STRIP



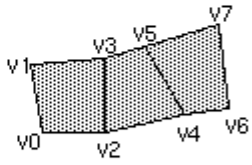
GL_LINE_LOOP



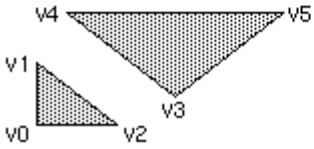
GL_POLYGON



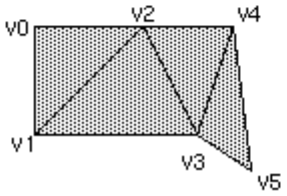
GL_QUADS



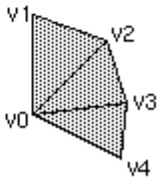
GL_QUAD_STRIP



GL_TRIANGLES



GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN