

# Fenêtrage OpenGL avec GLUT

par Tony BAYART ([Site perso de Tony BAYART](#))

Date de publication : 06/05/2007

Dernière mise à jour :

GLUT (**OpenGL Utility Toolkit**) est une bibliothèque portable permettant d'interfacer facilement et rapidement une application OpenGL. Créée par Mark Kilgard et portée sur différents systèmes, GLUT est très pratique pour concevoir des petits programmes de tests, des démonstrations et même des jeux. Bien que son utilisation ne soit pas très compliquée, cet article va vous guider pour poser les bases et vous initier à son utilisation.

- I - Introduction
- II - Préparation
- III - Initialisation et paramétrage
- IV - La boucle principale
- V - Bonus : La gestion du clavier
- VI - Remerciements
- VII - Liens et téléchargements

## I - Introduction

A l'heure où j'écris cet article, la version 3.7 de GLUT date de novembre 2001 et n'a plus été mise à jour depuis. Malgré son ancienneté, la bibliothèque GLUT reste encore très utilisée, autant par des amateurs que par des professionnels. Une version Open Source a vu le jour sous le nom de **FreeGLUT** dont la dernière version, la FreeGLUT 2.4, date de juin 2005. Basée sur FreeGLUT, la bibliothèque **OpenGLUT** propose des fonctionnalités supplémentaires.

## II - Préparation

Pour créer notre programme, il va nous falloir les fichiers nécessaires à la compilation. Téléchargez la dernière version de l'une des bibliothèques et copiez les fichiers nécessaires aux bons endroits. Vous devez avoir les fichiers suivants :

- "glut.h" à placer dans le répertoire "include" de votre compilateur (habituellement avec "gl.h")
- "glut32.lib" ou "libglut32.a" à copier dans le répertoire "lib" de votre compilateur
- "glut32.dll" à copier dans le répertoire de votre application si vous êtes sous windows

Une fois les préparatifs terminés, nous pouvons commencer. Notre programme va être simple, nous allons créer une fenêtre gérée par GLUT dans laquelle nous pourrions afficher des formes calculées par OpenGL. Pour commencer, il faut inclure *glut.h*, qui lui même effectue les inclusions nécessaires à OpenGL. Inutile donc d'inclure *gl.h* ou encore *glu.h*, c'est déjà fait. Nous allons également préparer quelques variables et prototypes de fonctions que nous utiliserons plus tard. Voici donc le début du code :

### Premier bout de code

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

int nWindowID;
const int nWidth = 640;
const int nHeight = 480;

void affichage(void);
void attente(void);
void clavier(unsigned char touche, int x, int y);
```

### III - Initialisation et paramétrage

Avant de pouvoir faire appel à GLUT, nous devons initialiser la bibliothèque. Pour cela, nous allons faire appel à la fonction **glutInit** avec les paramètres de ligne de commande du **main** :

#### Le main

```
// le main
int main(int argc, char *argv[])
{
    // initialisation de GLUT
    glutInit(&argc, argv);
```

GLUT est maintenant initialisée, nous pouvons paramétrer notre future fenêtre OpenGL. Pour lui indiquer le format de l'affichage, GLUT met à notre disposition la fonction **glutInitDisplayMode**. Cette fonction ne prend qu'un seul paramètre créé à partir d'une combinaison de valeurs. Par exemple, si nous voulons un affichage en RGB avec un double buffer et un tampon de profondeur, nous allons appeler **glutInitDisplayMode** de la manière suivante :

#### Format de l'affichage

```
// parametrage de l'affichage
glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);
```

Nous devons également paramétrer les dimensions de la fenêtre que nous voulons. Pour ce faire, il y a la fonction **glutInitWindowSize** :

#### Dimensions la fenêtre


```
// taille de la fenetre
glutInitWindowSize(g_nWidth, g_nHeight);
```

Vous l'aurez certainement compris, les 2 paramètres de type *int* attendus par **glutInitWindowSize** sont les dimensions souhaitées en largeur et en hauteur de la fenêtre. Ici nous avons donc demandé à ce que notre affichage fasse 640 pixels de large et 480 pixels de haut d'après l'initialisation de nos variables au début du programme. Maintenant que l'affichage est paramétré, nous pouvons créer notre fenêtre. Il suffit pour cela d'utiliser la fonction **glutCreateWindow** en lui passant en paramètre le nom que l'on veut donner à notre fenêtre :

#### Création de la fenêtre

```
// creation de la fenetre
g_nWindowID = glutCreateWindow("Ma premiere fenetre GLUT");
```

Notre fenêtre est créée et avec elle le contexte OpenGL, prêt à recevoir des ordres d'affichage. Remarquez que nous récupérons la valeur de type *int* retournée par la fonction **glutCreateWindow** qui est l'identifiant de la fenêtre créée afin de pouvoir nous en servir le cas échéant.

 Vous pouvez créer une fenêtre plein écran en appelant la fonction **glutFullScreen** après **glutCreateWindow** ou pendant l'exécution du programme.

Pour afficher une scène OpenGL dans une fenêtre GLUT, il faut indiquer à GLUT la fonction qui va s'occuper de l'affichage. Il nous faut donc une fonction qui s'en occupera, par exemple :

#### La fonction d'affichage

```
// fonction d'affichage
void affichage(void)
{
    // effacement de l'ecran et du tampon de profondeur
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

#### La fonction d'affichage

```
// echange des buffers "front" et "back"
glutSwapBuffers();
}
```

La fonction **affichage** ne prend aucun paramètre et ne retourne rien. Dans son état actuel, elle envoie une commande OpenGL pour effacer l'écran et le tampon de profondeur avant de faire appel à la fonction **glutSwapBuffers** qui provoque l'échange des buffers front et back. Dans notre main, à la suite de **glutCreateWindow**, nous pouvons indiquer à GLUT que notre fonction d'affichage sera **affichage** :

#### Activer la fonction d'affichage

```
// référencement de la fonction d'affichage
glutDisplayFunc(affichage);
```

Après la fonction d'affichage, il y a une autre fonction importante dans le fonctionnement de notre application. Si nous voulons faire de l'animation en temps réel avec OpenGL, il faut régulièrement mettre à jour l'affichage, chose que GLUT ne fait pas automatiquement. La fonction **attente** est là pour ça :

#### La fonction attente

```
// fonction appelee lorsque l'application ne fait rien
void attente(void)
{
    // une variable pour memoriser le temps a attendre
    static int nWaitUntil = glutGet(GLUT_ELAPSED_TIME);

    // on recupere le temps present
    int nTimer = glutGet(GLUT_ELAPSED_TIME);
    // et on le compare a l'instant qu'il faut attendre
    if(nTimer >= nWaitUntil)
    {
        // pour rafraichir l'affichage
        glutPostRedisplay();
        // 5 fois pas seconde
        nWaitUntil = nTimer + (1000 / 5);
    }
}
```

Un peu d'explications sur cette fonction, à commencer par les fonctions GLUT qu'elle utilise. **glutGet** sert à récupérer une information, un état de GLUT tels que la taille de la fenêtre, la profondeur de couleurs ou encore, comme ici avec **GLUT\_ELAPSED\_TIME**, le temps passé depuis le dernier appel à **glutGet(GLUT\_ELAPSED\_TIME)** (en millisecondes). Plus de détails sur les différents paramètres de **glutGet** dans la [documentation de GLUT](#).

La variable *nWaitUntil* est statique à la fonction **attente** afin de conserver sa valeur en mémoire chaque fois que la fonction sera appelée. La première fois que la fonction attente sera appelée, la variable *nWaitUntil* sera créée et initialisée avec la valeur retournée par **glutGet(GLUT\_ELAPSED\_TIME)**. Le mot clef *static* fait que la valeur contenue dans la variable ne sera pas perdue lorsque la fonction se terminera. La fonction **glutPostRedisplay** demande à GLUT de rafraîchir l'affichage en faisant appel à la fonction référencée comme fonction d'affichage. Ensuite ce que fait cette fonction est relativement simple; elle demande à GLUT de rafraîchir l'affichage 5 fois par seconde. Pour indiquer à GLUT d'utiliser **attente**, il suffit d'ajouter la ligne suivante après **glutDisplayFunc** dans le main :

#### Référencement de la fonction attente

```
// référencement de la fonction d'attente
glutIdleFunc(attente);
```

Maintenant que nous avons tout paramétré et que notre fenêtre est créée, il ne reste plus qu'à "lancer" l'application.

## IV - La boucle principale

Avant de lancer notre application à proprement parler, la fenêtre et le contexte OpenGL étant créés, nous pouvons configurer OpenGL. Par exemple indiquer la couleur d'effacement du fond avec **glClearColor** :

### Initialisations OpenGL

```
// initialisation OpenGL : couleur de fond  
glClearColor(0, 0, 0, 0);
```

GLUT est prêt, il faut maintenant lancer la boucle principale de l'application. Pour cela, nous allons ajouter un simple appel à **glutMainLoop** dans notre **main** :

### Fin du main

```
// boucle de gestion des evenements  
glutMainLoop();  
  
// le programme n'arrivera jamais jusqu'ici  
return EXIT_SUCCESS;  
}
```

La fonction **glutMainLoop** que nous avons lancée est bloquante et ne se termine jamais. Elle fait tourner l'application en exécutant la fonction **affichage** lorsqu'on lui demande et la fonction **attente** lorsqu'il n'y a rien à faire.

## V - Bonus : La gestion du clavier

Pour parfaire un peu notre petit programme, nous allons lui ajouter la gestion du clavier. Nous avons déjà déclaré le prototype de notre fonction en début de programme : **clavier**.

Voici ce que donne la fonction :

### La fonction clavier

```
// fonction de gestion du clavier
void clavier(unsigned char touche, int x, int y)
{
    // traitement des touches q et echap
    if(touche=='q' || touche == 27)
    {
        // destruction de la fenetre GLUT
        glutDestroyWindow(g_nWindowID);
        // on quitte notre programme
        exit(EXIT_SUCCESS);
    }
}
```

Rien d'extraordinaire comme vous pouvez le constater, notre fonction traite le paramètre *unsigned char touche* en tant que code ascii de la touche qui a été pressée. Nous comparons la valeur de *touche* avec 'q' et la valeur 27 (qui est la valeur ascii de la touche d'échappement) afin de détruire la fenêtre lorsque l'une de ces deux touches est pressée. La destruction de la fenêtre s'effectue à l'aide de la fonction **glutDestroyWindow** qui nécessite en paramètre l'identifiant de la fenêtre GLUT fourni par **glutCreateWindow**. Après **glutDestroyWindow**, nous demandons à terminer l'application avec **exit**. Sans cela, **glutMainLoop** fera appel aux autres fonctions référencées alors que la fenêtre n'est plus valide et c'est un beau plantage qui s'en suivra. Les paramètres *int x* et *int y* que nous n'utilisons pas ici contiennent les coordonnées de la souris au moment de l'appui sur la touche. Et si vous avez bien suivi, vous savez qu'il faut indiquer à GLUT que nous avons une fonction qui gère le clavier afin que celle-ci soit appelée lorsqu'on en a besoin. Ajoutons la fonction **glutKeyboardFunc** avant notre boucle principale **glutMainLoop** :

### Référencement de la fonction clavier

```
// référencement de la fonction de gestion des touches
glutKeyboardFunc(clavier);
```

Et le tour est joué. Nous pouvons maintenant quitter notre programme en pressant au choix la touche *q* ou la touche *echap*.

Après tant d'efforts, nous avons le plaisir de pouvoir contempler notre fenêtre (vide) à l'écran :

## Ma première fenetre GLUT



*Ma première fenêtre GLUT*

## VI - Remerciements

Je remercie **Fearyourself** pour son aide précieuse lors de la rédaction de cet article ainsi que pour ses conseils avisés et ses corrections.

## VII - Liens et téléchargements

 **GLUT pour Windows**

 **Documentation de GLUT**

**Télécharger les sources de cet article (zip, 194 Ko) ([Mirroir](#))**

**Version PDF de l'article (61 Ko) ([Mirroir](#))**

