Chapitre2: Introduction à OpenGL

Généralités

- http://www.opengl.org/
- OpenGL (Open Graphic Library, ce qui se traduit en français par ‘Bibliothèque Graphique Ouverte’) est une librairie graphique 2D/3d, c’est-à-dire une interface logicielle permettant d’effectuer des opérations d’affichage sur un écran.

- Elle a été développée en 1989 (GL) par Sillicon Graphics, puis portée sur d’autre architectures en 1993 (OpenGL) .

- Spécification d’objets graphiques et production d’applications graphiques 3D interactives.
  - Standard de la programmation d’applications graphiques.
  - Avec OpenGL, on peut produire des images synthétiques sophistiquées (reflet, ombres) en temps réel si on dispose du hardware adéquat.

Introduction à OpenGL

- OpenGL est une interface comprenant plus de 120 commandes, qui vont servir à décrire des objets (caméras, lampes, modèles 3D) et les opérations que l'on peut effectuer pour les manipuler.

- Existe sur de nombreuses architectures (NT, Unix, etc...)

  - Aujourd'hui, OpenGL est reconnu par les professionnels comme un standard, et est utilisé par tous les logiciels professionnels de synthèse d'images (3DSMax, Maya, Softimage...), de CAO (ProEngineer, Catia...), mais aussi dans le domaine des jeux vidéos 3D.
  - La bibliothèque OpenGL s'accompagne de librairies complémentaires (GLU, GLUT, GLX).

Introduction à OpenGL,GLUT

- Pour créer un programme OpenGL dans un environnement fenêtré.
- il faut être en mesure de créer une fenêtre pour y afficher les images, de la détruire, et de gérer les interactions avec l'utilisateur par le clavier, la souris et tous les autres types de périphériques d'entrée.
- ***OpenGL*** se voulant indépendant de toute plate-forme matérielle,
- l'API ne fournit pas de telles fonctions.
- il existe une bibliothèque annexe, nommée **Glut** (GL Utility Toolkit) qui fournit ces fonctions.

Introduction à OpenGL,GLUT

- Voici une liste de toutes les fonctionnalités proposées par Glut :

  - Gestion de fenêtres.

  - Gestion des événements par fonctions de rappel.

  - Gestion de périphériques d'entrée exotiques (spaceballs...).

  - Gestion des polices de caractères.

  - Fonctions de création de menus.

La syntaxe d'OpenGL

- La syntaxe d'OpenGL caractérise les constantes, types et fonctions de la manière suivante :
- Les constantes : GL_CONSTANTE (exemple : GL_COLOR_BUFFER_BIT);
- Les types : GLtype (exemples : GLbyte, GLint, GLfloat);
  - OpenGL redéfinit des types de données numériques : GLint correspond aux entiers, GLfloat aux flottants, GLbyte aux caractères non signés... Il est préférable d'utiliser ces types de données plutôt que les types standards du C.
- Les fonctions : glLaFonction (exemples : glDraw, glClearColor).

La syntaxe d'OpenGL

- À cela s'ajoute dans la syntaxe des fonctions, la caractérisation du nombre et du type des arguments par un suffixe. Par exemple :
  - glVertex3f(1.0, 2.0, 3.0) **définit les coordonnées dans l'espace d'un sommet** (vertex) en simple précision réelle.
  - Pour définir les coordonnées du plan d'un sommet par des valeurs entières, on utilise : glVertex2i(1, 2)

OpenGL est basée sur des États

- C 'est le principe général d'OpenGL : On positionne un état interne, et sa valeur est ensuite utilisée comme valeur courante : les ordres de dessin suivants utiliseront cette valeur-ci.

- Prenons un exemple : Pour dessiner un sommet rouge, on positionne d'abord l'état correspondant à la couleur courante à la valeur rouge, et ensuite on demande le dessin d'un sommet. Tous les sommets qui seront ensuite dessinés seront rouges, tant que l'on n'a pas modifié la couleur courante.

- Et ce principe que nous avons illustré à partir de l'état "couleur " s'applique à tous les états, comme l'épaisseur des traits, l'éclairage, etc.

Premier programme(1)

- ***Première chose à faire*** : inclure les fichiers d'en-tête OpenGL. Il en existe plusieurs, mais glut les appelle pour nous. On se contentera donc d'un simple : #include <GL/glut.h>

Premier programme(2)

- ***La deuxième phase*** est l'initialisation de glut et la création de la fenêtre.

- Glut gère les interactions entre OpenGL et le système.
- La phase d'initialisation va permettre de créer la fenêtre.
- L'initialisation de glut se fait de la manière suivante :
  - glutInit(&argc,argv);
  - glutInitDisplayMode(GLUT_RGB);
  - glutInitWindowPosition(200,200);
  - glutInitWindowSize(250,250);
  - glutCreateWindow(« Exemple 1 »);

Premier programme(3)

- ***La troisième phase*** consiste à l'initialisation des états
- OpenGL dispose d'un état par défaut.
- Régler 2 états dans la phase d'initialisation : la couleur de fond, et la taille d'un point.
- Les fonctions à utiliser sont les suivantes:
  - void glClearColor(GLclampf rouge, GLclampf vert, GLclampf bleu,GLclampf alpha);
  - void glPointSize(GLfloat taille);

Premier programme(4)

- **glClearColor()** permet de spécifier la couleur de remplissage utilisée lors d'un effacement de la scène. On peut donc l'assimiler à la couleur du fond. Le mode de spécification de couleur est le classique RGB avec une composante supplémentaire alpha, utilisée pour la gestion de la transparence des objets.

- glClearColor(1.0,1.0,1.0,0.0) nous donne un fond blanc, et
- glClearColor(0.0,0.0,0.0,0.0) un fond noir.

- Par défaut, lorsqu'on choisit de représenter les objets simplement par leurs sommets, la taille des sommets à l'écran est de 1 pixel. Afin de les rendre plus visible, nous réglerons la taille des sommets à 2 pixels :
- glPointSize(2.0);

Premier programme(5)

- glutInit initialise la bibliothèque glut, notamment en établissant la communication avec le système.

- La fonction glutInitDisplayMode permet de régler les paramètres liés à l'affichage : type d'image (palette indexée ou RVB), utilisation d'un tampon de profondeur (plus connu sous le nom de Z-buffer),utilisation du double buffering.

- Pour l’instant un simple tampon d'image en mode RVB défini par la constante GLUT_RGB nous suffira.

Premier programme(6)

- glutInitWindowPosition() et glutInitWindowSize() permettent respectivement de définir la position du coin supérieur gauche de la fenêtre OpenGL par rapport au coin supérieur gauche de l'écran, et de spécifier la largeur et la hauteur de la fenêtre.

- La fenêtre est créée grâce à l'appel glutCreateWindow() et son nom doit être transmis sous forme de chaîne de caractères. Si la fenêtre est bel et bien créée, elle ne sera affichée à l'écran que lors de l'entrée dans la boucle de gestion des événements.

Premier programme(7)

```
int main(int argc, char** argv)
{
/* initialisation de glut et création de la fenêtre */
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_RGB);
glutInitWindowPosition(200,200);
glutInitWindowSize(250,250);
glutCreateWindow(« exemple 1");
/* Initialisation d'OpenGL */
glClearColor(0.0,0.0,0.0,0.0);
/* On passe à une taille de 2 pixels pour des raisons de clarté */
glPointSize(2.0);
}
```

Premier programme(8)

- Le système de gestion des événements offert par glut est relativement simple. Des fonctions sont associées aux différents types d'événements envoyés par le système, puis une boucle d'attente est lancée.

- A chaque fois qu'un événement est émis par le système, la fonction de rappel associée à l'événement est appelée.

- Vous devez associer au moins une fonction de rappel, la fonction d'affichage, qui est celle dans laquelle vous devez décrire votre scène 3D. Vous pouvez associer des fonctions de rappel aux événements liés :

  - au clavier, à la souris etc..

Premier programme(9)

- Nous devons maintenant mettre en place la fonction de rappel obligatoire qui est la fonction d'affichage :
  glutDisplayFunc(affichage)

- Après cela, il ne nous reste plus qu'à lancer la boucle d'attente des événements : **glutMainLoop()**

Premier programme(10)

```
int main(int argc, char** argv)
{
/* initialisation de glut et création de la fenêtre */
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_RGB);
glutInitWindowPosition(200,200);
glutInitWindowSize(250,250);
glutCreateWindow(« hello world");
/* Initialisation d'OpenGL */
glClearColor(0.0,0.0,0.0,0.0);
/* On passe à une taille de 2 pixel pour des raisons de clarté */
glPointSize(2.0);
/* enregistrement des fonctions de rappel */
glutDisplayFunc(affichage);
/* entrée dans la boucle principale de glut */
glutMainLoop();
}
```

Premier programme(11)

- **La fonction d'affichage :**
- La fonction d'affichage constitue le cœur du programme. C'est ici que nous allons décrire la scène. Pour être plus exact, la fonction d'affichage contient la procédure à appliquer pour redessiner notre scène dans la fenêtre, en commençant par un remplissage de la fenêtre avec la couleur de fond que nous avons défini dans la phase d'initialisation.

- On utilise pour cela void glClear(GLbitfield masque) OpenGL travaille avec plusieurs zone tampons en plus de la zone d'image (tampon de profondeur, d'accumulation...). Le paramètre masque permet de spécifier les tampons que l'on souhaite effacer.

- Nous n'utilisons aujourd'hui que le tampon d'image. Notre masque vaudra GL_COLOR_BUFFER_BIT.

Premier programme(12)

- Il nous faut ensuite décrire la scène 3D. La méthode de représentation
- d'un scène d'OpenGL est la plus suivante :
  - Une scène est constituée d'objets.
  - Un objet est défini par un ensemble de polygones.
  - Un polygone est un ensemble de points de l'espace reliés entre eux par des arêtes.
- Pour spécifier notre scène 3D, nous allons devoir énumérer la liste des polygones.
- La méthode est la suivante :
  - on indique le début de la description du polygone,
  - on explicite chaque point du polygone,
  - puis on indique la fin de l'énumération.
- A chaque déclaration de points, on peut modifier certaines variables d'état, comme la couleur active. Notre scène ne comportant qu'un polygone, la description sera vite faite

Premier programme(13)

```
glBegin(GL_POLYGON);

    glColor3f(1.0,0.0,0.0); glVertex2f(-0.5,-0.5);
    glColor3f(0.0,1.0,0.0); glVertex2f(0.5,-0.5);
    glColor3f(0.0,0.0,1.0); glVertex2f(0.5,0.5);
    glColor3f(1.0,1.0,1.0); glVertex2f(-0.5,0.5);

glEnd();
```

Premier programme(14)

- Chaque description de polygone commence par un appel à **void glBegin(GLenum mode)**

- Le paramètre mode auquel nous donnons la valeur GL_POLYGON indique la technique utilisée pour relier par des arêtes les différents points du polygone. En mode GL_POLYGON, chaque sommet est relié à son prédécesseur, et pour fermer le polygone, le dernier point est relié au premier.

Premier programme(15)

- La primitive de spécification d'un sommet du polygone est glVertex().
- Comme nous travaillons dans le plan d'équation z=0, nous utiliserons la variante à deux arguments de type GLfloat :
- void glVertex2f(GLfloat x,GLfloat y);

- Bien entendu, x et y sont les coordonnées cartésiennes du sommet.
- La commande glColor3f() modifie la couleur dite active.

- Son prototype est :

- void glColor3f(GLfloat r, GLfloat v, GLfloat b)

Premier programme(16)

- La fin de la description d'un polygone est marquée par un simple glEnd().

- Pour terminer la fonction d'affichage, un glFlush() permet de s'assurer que toutes les commandes ont bien été transmises au serveur.

Premier programme(17)

```
void affichage(void)

/* initialisation des pixels */
glClear (GL_COLOR_BUFFER_BIT);
glBegin(GL_POLYGON);
    glColor3f(1.0,0.0,0.0); glVertex2f(-0.5,-0.5);
    glColor3f(0.0,1.0,0.0); glVertex2f(0.5,-0.5);
    glColor3f(0.0,0.0,1.0); glVertex2f(0.5,0.5);
    glColor3f(1.0,1.0,1.0); glVertex2f(-0.5,0.5);
glEnd();
/* On force l'affichage du résultat */
glFlush();
}
```