OpenGL (Partie 1)

Définition

GL et OpenGL sont des librairies graphiques d'affichage trois dimensions.

GL (Iris GL): Librairie graphique standard de Silicon Graphics Inc. (orientée visualisation). Utilisée sur les stations de travail graphiques de SGI (sous système d'exploitation IRIX) ainsi que sur les matériels équivalents d'autres constructeurs (sous licence).

OpenGL: Sous-ensemble de GL ne nécessitant pas une licence commerciale SGI

Conçu et spécifié à la fin des années 80 - début des années 90

Introduction

Interface logicielle avec les adaptateurs graphiques 3D

Disponible pour un nombre important de plateformes et de langages (initialement C et Fortran)

Environ 120 instructions distinctes exécutables au moyen du langage de programmation utilisé

Spécification d'objets graphiques et production d'applications graphiques 3D interactives

Conçu initialement pour fonctionner en environnement Unix

-> Exploitation de X-Window
-> Exploitation des réseaux

Standard de la programmation d'applications graphiques en CAO

Claire évolution vers l'industrie du jeu (support de techniques de rendu améliorant la qualité des images, exemples: textures évoluées, shaders)

RETOUR

Indépendant de la machine hôte

Existence des versions successives suivantes (ajout de nouvelles fonctionnalités, compatibilité ascendante?!):

- 1.0, (01/92): Version de base, mode immédiat
  - 1.1, (01/97): Introduction des textures
  - 1.2
  - 1.3
  - 1.4
  - 1.5, (07/03): Introduction des Vertex Buffer Objects (VBOs), mode retenu
- 2.0, (09/04): Introduction du langage GLSL pour la programmation de "shaders"
  -> implantation et exécution de programmes "utilisateurs" sur la carte graphique
  - 2.1
- 3.0, (07/08): Introduction d'un mécanisme de "deprecation" pour les caractéristiques anciennes
  - 3.1
  - 3.2
- 4.0, (03/10): Introduction de l'arithmétique 64 bits, convergence vers les nouvelles fonctionnalités de DirectX'11
  - 4.1
  - 4.2
  - 4.3
  - 4.4

Existence des releases suivantes:

- OpenGL: Version pour systèmes classiques: Stations de travail, ordinateurs portables, ... (programmation en C, Fortran, ...)
- OpenGL ES (Embedded Systems): Version pour systèmes embarqués: Téléphones mobiles, tablettes, GPS, ...
- OpenGL SC (Safety Critical): Version pour systèmes critiques: Aviation, industrie nucléaire, industrie de l'armement, ...
- WebGL: Version dédiée à l'affichage 3D dans les navigateurs Internet (Programmation en Javascript) (Exemples)

Géré par le consortium OpenGL (3Dlabs, Apple, ATI, Dell, IBM, Intel, NVidia, Oracle, ..., http://www.opengl.org)

Fonctionnalités principales

- Dessin de facettes
- Dessin de quadriques
- Dessin de surfaces B-Splines
- Dessin de NURBS
- Affichages en fil de fer et en plein (surfacique)
- Affichage avec élimination des parties cachées
- Visualisations en projection orthographique ou en projection en perspective
- Lumières et matériaux
- Ombrages plat et lissé (Gouraud)
- Anticrénelage (antialiasing)
- Gestion des textures
- Depth cueing et effets atmosphériques (brouillard,...)
- Flou de déplacement et profondeur de champ
- Double-Buffering
- OpenGL Shading Language (GLSL) depuis la version 2.0
- ...

Exemple

```
#include <~~~~~~~~~.h>
void main() {
  ...
  glViewport(0,0,512,512);
  glClearColor(0.0,0.0,0.0,0.0);
  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(1.0,1.0,1.0);
  glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
  glBegin(GL_POLYGON);
    glVertex2f(-0.5,-0.5);
    glVertex2f(-0.5,0.5);
    glVertex2f(0.5,0.5);
    glVertex2f(0.5,-0.5);
  glEnd();
  glFlush();
  ...
}
```

Ce programme dessine un "carré" blanc au centre d'une "fenêtre" de fond noir.

Architecture simplifiée

Architecture en pipeline avec des étages successifs réalisant les opérations nécéssaires au processus de visualisation d'une scène (création d'une image)

Circulation des données gérées au cours du rendu entre ces étages avec transformations de leurs types au sein des étages:

- En entrée : Des sommets en 3D, les paramètres d'affichage, ...
- En sortie : Une image

Fonctionnement en parallèle des étages du pipeline
Fonctionnement en tâches élémentaires parallèles identiques (parallélisme sur les données) au sein des étages du pipeline

Bibliothèques d'OpenGL

(1) OpenGL Library (GL)

Fonctions de base pour l'affichage 3D en OpenGL:

- Dessin de sommets, de segments de droite et de facettes
- Matériaux et lumières
- Transformations géométriques
- Caméras
- Textures
- Gestion des paramètres de rendu
- ...

Pas de fonction pour la construction d'une interface utilisateur (fenêtres, souris, clavier, gestionnaire d'événements, ...)

Préfixe des fonctions: gl

Librairie standard

En langage C, déclarations dans le fichier GL/gl.h

(2) OpenGL Utility Library (GLU)

Commandes bas-niveau écrites en GL:

- Transformations géométriques spécifiques à certaines actions
- Transformation des polygones quelconques en ensembles de polygones triangulaires
- Rendu des surfaces paramétriques et quadriques
- ...

Préfixe des fonctions: glu

Librairie standard

En langage C, déclarations dans le fichier GL/glu.h

(3) OpenGL extension to X-Window (GLX)

Utilisation d'OpenGL en association avec X-Window pour la construction d'une interface utilisateur

Préfixe des fonctions: glX

Librairie non standard

(4) Auxiliary Library

Bibliothèque écrite pour rendre simple la programmation de petites applications dans le cadre d'interfaces graphiques interactives simples:

- Gestion d'une fenêtre d'affichage
- Gestion de la souris
- Gestion du clavier
- ...

Programmation événementielle

Préfixe des fonctions: aux

Librairie non standard

Librairie non spécifique à un système d'exploitation

L'Auxiliary Library: Aux

(5) OpenGL Tk Library (GLTk)

Équivalent à Aux mais avec l'utilisation de TclTk pour l'interface graphique

Préfixe des fonctions: tk

Librairie non standard

Librairie non spécifique à un système d'exploitation

(6) OpenGL Utility Toolkit (GLUT)

Équivalent à Aux du point de vue des fonctionnalités principales et des principes d'utilisation

Interface utilisateur programmable plus élaborée (multifenêtrage, menus popup)
-> plus grande puissance et mais plus grande complexité

Préfixe des fonctions: glut

Librairie non standard mais très couramment utilisée

Librairie non spécifique à un système d'exploitation

En langage C, déclarations dans le fichier GL/glut.h

Fonctionnalités supplémentaires par rapport à Aux:

- Gestion des menus
- Gestion des environnements multifenêtrés
- Existence de polices de caractères bitmap et vectorielles intégrées
- Gestion de périphériques d'entrée supplémentaires
- ...

Autres exemples

L'Utility Toolkit: GLUT

Syntaxe de la librairie GL (fichier GL/gl.h)

D'une manière générale, emploi des règles syntaxiques du langage C

Spécificités supplémentaires:

- Les noms de fonction de la librairie GL commencent par le préfixe *gl*.
- Les noms de constante (#define ou enum) sont donnés en majuscule et commencent par le préfixe GL_.
- Certains noms d'instruction se terminent par un suffixe.

  Exemple: `glVertex2f`

  Ce suffixe indique le nombre et le type des arguments de la fonction.

-	*Type*	*Type C 32 bits*	*Type OpenGL*
b	entier 8 bits	signed char	GLbyte
s	entier 16 bits	short	GLshort
i	entier 32 bits	long, int	GLint, GLsizei
f	réel 32 bits	float	GLfloat, GLclampf
d	réel 64 bits	double	GLdouble, GLclampd
ub	entier non signé 8 bits	unsigned char	GLubyte, GLboolean
us	entier non signé 16 bits	unsigned short	GLushort
ui	entier non signé 32 bits	unsigned int	GLuint, GLenum, GLbitfield

Exemple

```
glVertex2i(1,3);        <=>  glVertex2d(1.0,3.0);
```

Une lettre terminale v supplémentaire (v pour vector) indique que la fonction prend comme paramètre un pointeur sur un tableau.

```
glColor3i(1,0,1);       <=>  int c[3]={1,0,1};
                             glColor3iv(c);
```

Variables d'état

OpenGL est une state-machine.

-> Des instructions existent pour placer le système dans certains "états".

Ceux-ci seront utilisés à cette valeur jusqu'à ce qu'ils soient changés (couleur de tracé, matrice(s) de transformation représentative(s) du(des) repère(s) courant(s), activation de l'élimination des parties cachées, 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 affectée à l'amorçage du programme utilisant OpenGL.

Fonctions de manipulation directe des variables d'état

- **`glEnable(GLenum pname)`**

  Activation d'une variable d'état booléenne

  pname: Variable d'état à activer

  Exemples:
  glEnable(GL_LIGHTING) -> activation de la gestion des lumières et des matériaux,
  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

- **`glGetBooleanv(GLenum pname,GLboolean *param)`**
- **`glGetIntegerv(GLenum pname,GLinteger *param)`**
- **`glGetFloatv(GLenum pname,GLfloat *param)`**
- **`glGetDoublev(GLenum pname,GLdouble *param)`**

  Consultation de variables d'état de l'environnement OpenGL

pname: Variable d'état consultée

param: Résultat (pointeur sur une variable si une seule valeur, tableau de variables si plusieurs valeurs)

- `GLboolean glIsEnabled(GLenum pname)`

  Consultation de variables d'état booléennes

  pname: Variable d'état consultée

- `glPushAttrib(GLbitfield mask)`

  Sauvegarde de variables d'état par empilement dans la pile dédiée de l'environnement OpenGL

  mask: Variables d'état à sauvegarder (descriptif complet non réalisé ici car trop long)

- `glPopAttrib()`

  Restoration de variables d'état par dépilement depuis la pile dédiée de l'environnement OpenGL

Exemple

```
glEnable(GL_DEPTH_TEST);
GLboolean v = glIsEnabled(GL_DEPTH_TEST);
```

Active l'affichage avec élimination des parties cachées puis récupère la valeur d'activation de l'affichage avec élimination des parties cachées.

Instructions de base d'OpenGL

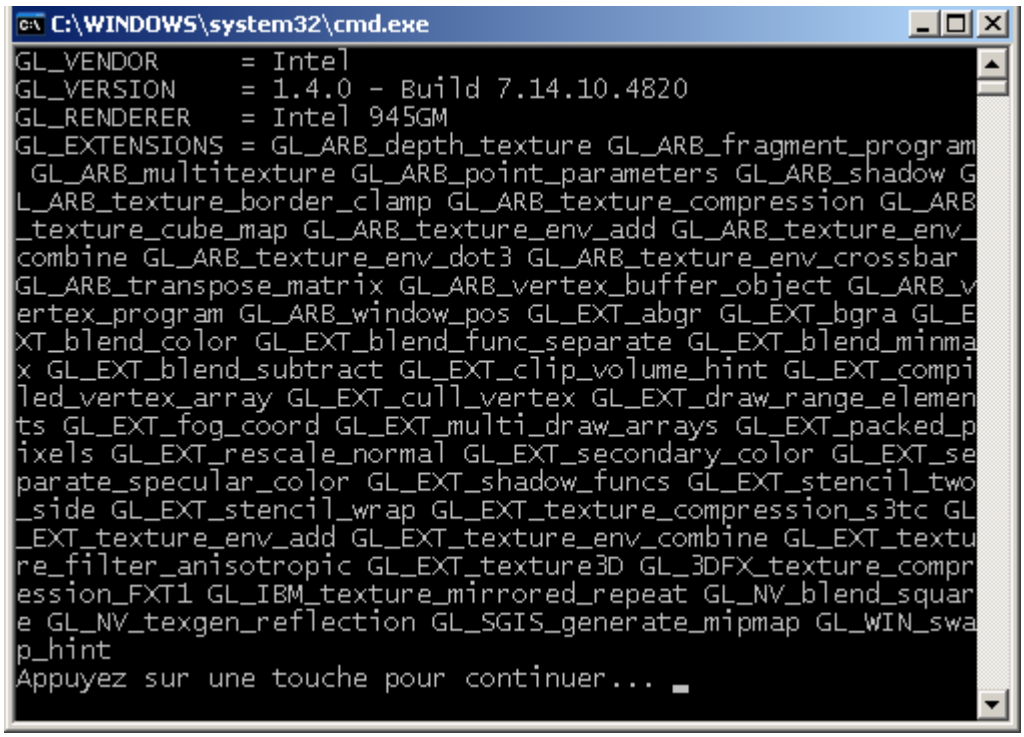
Version d'OpenGL

- `GLubyte *glGetString(GLenum name);`

  Retour d'une chaine de caractères de description de la librairie OpenGL utilisée

  name: Information recherchée
  GL_VENDOR: Fabriquant de la librairie
  GL_RENDERER: Nom de la librairie
  GL_VERSION: Version de la librairie

GL_EXTENSIONS: Liste des extensions disponibles (items séparés par des espaces)



Programme GLUt

Exécutable GLUt

Détection d'erreur

- `GLenum glGetError(void);`

Retour de la valeur actuelle de la variable d'état erreur de l'environnement OpenGL

7 valeurs retournées possibles:
GL_NO_ERROR,
GL_INVALID_ENUM,
GL_INVALID_VALUE,
GL_INVALID_OPERATION,
GL_STACK_OVERFLOW,
GL_STACK_UNDERFLOW,
GL_OUT_OF_MEMORY.
Replacement de la variable d'état erreur à **GL_NO_ERROR** après interrogation

Programme GLUt

Exécutable GLUt

Effacement de la fenêtre de dessin (viewport)

L'environnement OpenGL inclut plusieurs tampons mémoire effaçables individuellement ou par lot:

- Le tampon couleur (frame buffer)
  -> Stockage de la couleur des pixels de l'image calculée (le tampon couleur doit habituellement être vidé à la couleur de fond au début du calcul d'une nouvelle image, il est généralement utilisé et transmis automatiquement comme image à afficher dans la fenêtre de visualisation),
- Le tampon profondeur
  -> Stockage, pour chaque pixel de l'image calculée, d'une information de profondeur utilisée lors de l'élimination des parties cachées (si l'élimination des parties cachées est activée, le tampon profondeur doit habituellement être vidé avant le calcul de toute nouvelle image),
- Le tampon accumulation
  -> Accumulation d'images les unes sur les autres,
- Le tampon stencil.

- **`void glViewport(GLint x,GLint y,GLsizei tx,GLsizei ty);`**

  Choix de la position et de la taille du viewport d'affichage (résolution des tampons mémoire)

  x,y: Position
  tx,ty: Taille

- **`void glClear(GLbitfield mask);`**

  Effacement d'un (ou de plusieurs) buffer écran

  mask: Buffer à éffacer

mask
GL_COLOR_BUFFER_BIT Effacement des pixels du buffer d'affichage
GL_DEPTH_BUFFER_BIT Effacement de l'information de profondeur associée à chaque pixel du buffer d'affichage (information utilisée pour l'élimination des parties cachées)
GL_ACCUM_BUFFER_BIT Effacement du tampon accumulation utilisé pour composer des images
GL_STENCIL_BUFFER_BIT Non renseigné

Les valeurs de masque sont composables par "ou" (| en C) pour effectuer plusieurs opérations d'effacement en une seule instruction glClear et rendre possible l'optimisation de ces opérations.

- **`void glClearColor(GLclampf r,GLclampf v,GLclampf b, GLclampf alpha) ;`**

  Choix de la couleur d'éffacement du tampon couleur

  r, v, b, alpha: Couleurs de remplissage du tampon couleur lors d'un effacement par glClear

- **`void glClearDepth(GLclampf depth) ;`**

  Choix de la profondeur d'initialisation du tampon profondeur

  depth: Valeur de remplissage du tampon profondeur lors d'un effacement par glClear

Choix de la couleur de tracé

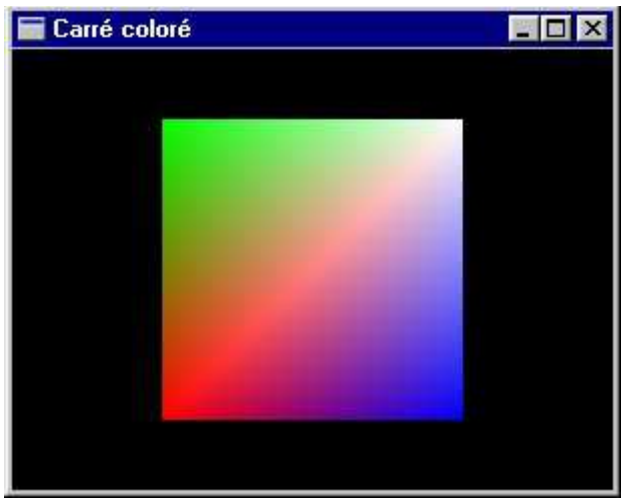
- **`void glColor3{b s i f d ub us ui} (TYPE r,TYPE v,TYPE b);`**

- **`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

- `void glColor3{b s i f d ub us ui}v (const TYPE *v);`
- `void glColor4{b s i f d ub us ui}v (const TYPE *v);`

v: Tableau contenant les composantes de la couleur de tracé



Programme Aux	Programme GLUt
Exécutable Aux	Exécutable GLUt

La couleur choisie est active pour tous les objets créés après spécification, jusqu'à nouvelle spécification.

Cette couleur n'est utilisée que si la gestion des lumières et des matériaux est désactivée. En cas d'activation de cette fonctionnalité, ce sont les calculs d'éclairage, compte tenu des lumieres, des matériaux et éventuellement des textures spécifiés, qui permettent le calcul de la couleur des pixels.

Terminaison du tracé d'une image

Instruction utilisée pour forcer l'exécution entière des commandes GL qui la précèdent dans le flux d'exécution OpenGL (vidage d'un pipeline d'affichage sur une station graphique, exécution des ordres de tracé sur un réseau, ...). Souvent utilisée pour indiquer qu'une image est entièrement tracée et peut donc être affichée.

- `void glFlush(void);`

- `void glFinish(void);`

glFinish se met en attente d'une notification d'exécution de la part du dispositif graphique tandis que glFlush n'est pas bloquant.

Principe de programmation OpenGL et élimination des parties cachées

Généralement, la programmation OpenGL de l'affichage d'une scène correspond à la programmation de l'exécution d'une boucle infinie d'affichage d'images ayant pour conséquence la création d'une suite d'images (éventuellement identiques):

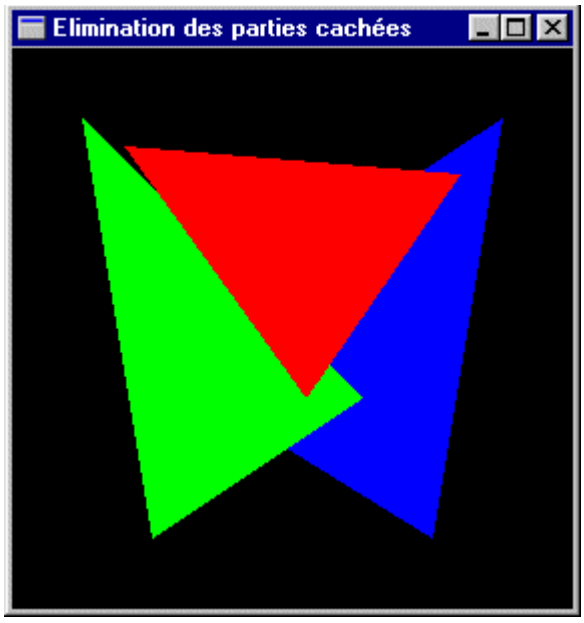
```
while(1) {
  definir_camera();
  glClear(GL_COLOR_BUFFER_BIT);
  dessiner_objet_3D_A();
  dessiner_objet_3D_B();
  dessiner_objet_3D_C();
  ...
  dessiner_objet_3D_Z();
  afficher_Image();
  attendreEtGererEvenement();
}
```

-> L'élimination des parties cachées n'est pas forcément correctement réalisée car on utilise un algorithme du peintre pour créer chacune des images. Les objets sont affichés les uns sur les autres dans l'ordre d'exécution du programme et donc l'ordre d'affichage a une influence sur le résultat obtenu. Si l'ordre est bon, l'élimination des parties cachées est réalisée.

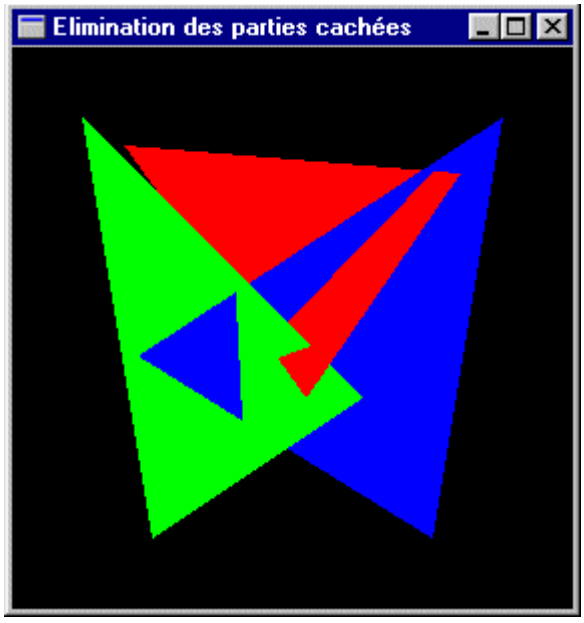
-> Utilisation de la fonction glEnable pour valider l'élimination des parties cachées (généralement par Z-Buffer en OpenGL) via modification de la variable d'état GL_DEPTH_TEST.
  -> L'effacement des tampons couleur et profondeur doit être réalisé entre chaque image au lieu du seul tampon couleur.
  -> L'ordre de dessin des objets n'a plus d'importance.

```
glEnable(GL_DEPTH_TEST);
while(1) {
  definir_camera();
```

```
    glClear(GL_COLOR_BUFFER_BIT|
            GL_DEPTH_BUFFER_BIT);
    dessiner_objet_3D_A();
    dessiner_objet_3D_B();
    dessiner_objet_3D_C();
    ...
    dessiner_objet_3D_Z();
    afficher_Image();
    attendreEtGererEvenement();
}
```



Sans élimination des parties cachées



Avec élimination des parties cachées

Programme Aux	Programme GLUt
Exécutable Aux	Exécutable GLUt

Sommets, lignes et polygones

- Sommet: Ensemble de trois coordonnées représentant une position dans l'espace

Remarque: OpenGL travaille en coordonnées homogènes.

- Ligne: Segment de droite entre deux sommets
- Polygone: Surface plane délimitée par une boucle fermée de lignes

Par définition, un polygone OpenGL ne se recoupe pas, n'a pas de trou et est de bord convexe.
Si on veut représenter un polygone ne vérifiant pas ces conditions, on devra le subdiviser (tessèlation) en un ensemble de polygones convenables. GLU propose des fonctions pour gérer les polygones spéciaux.
OpenGL ne vérifie pas les hypothèses de définition des polygones et les dessine même s'ils ne les respectent pas. Le résultat à l'affichage n'est alors pas déterministe.

Déclaration de la position d'un sommet

- `glVertex{234}{sifd}[v](TYPE coords) ;`

  coords: Coordonnées de la position
  Si seulement 2 valeurs sont spécifiés, ces valeurs représentent x et y et z est défini à 0.0.

Cette fonction est utilisée uniquement pour la déclaration de sommets au sein d'une primitive graphique.

Déclaration d'une primitive graphique

Une primitive graphique est constituée d'un ensemble de sommets.

Elle est créée par la réalisation successive des instructions glVertex permettant la définition des positions de ses sommets.

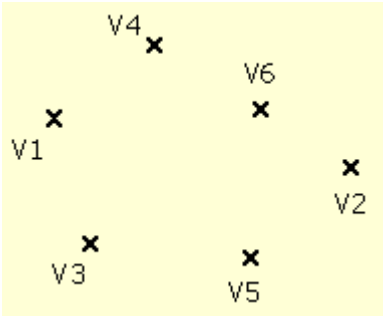
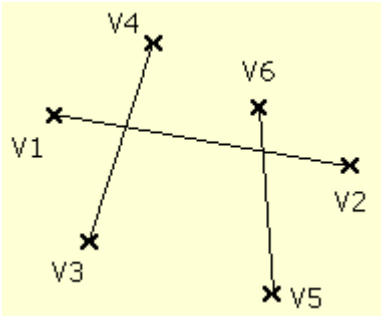
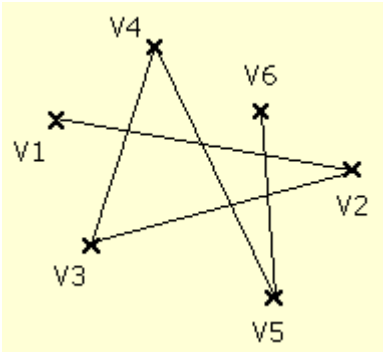
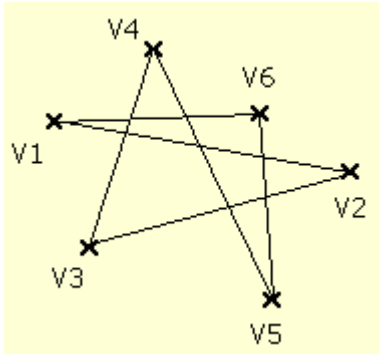
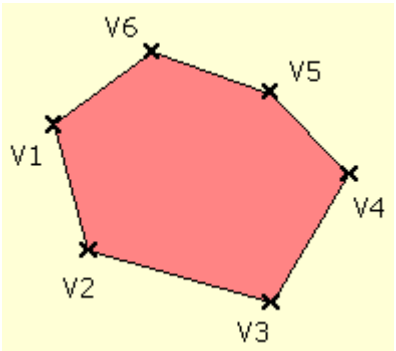
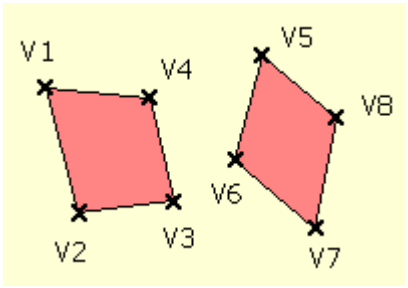
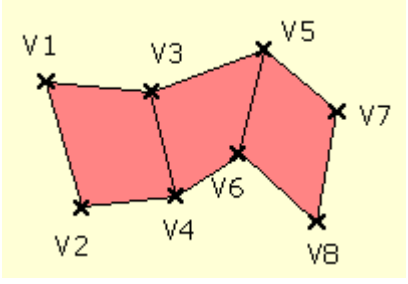
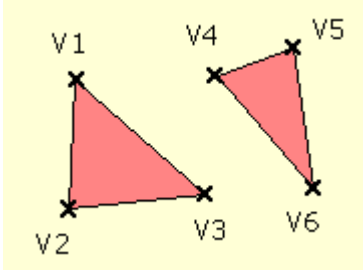
La primitive est délimitée au début et à la fin par

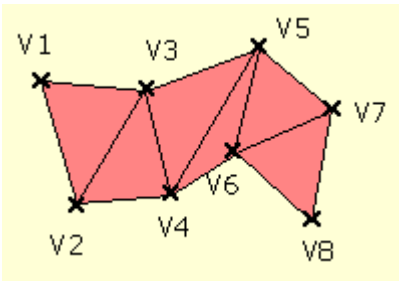
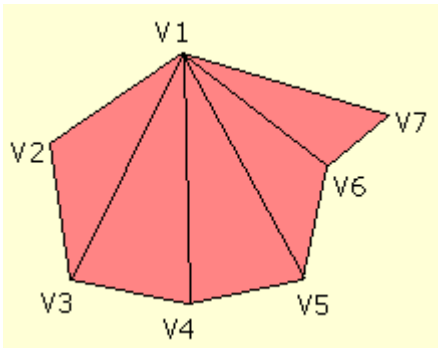
- `void glBegin(GLenum mode) ;`

mode: Type de primitive à dessiner à choisir parmi **GL_POINTS**, **GL_LINES**, **GL_LINE_STRIP**, **GL_LINE_LOOP**, **GL_POLYGON**, **GL_QUADS**, **GL_QUAD_STRIP**, **GL_TRIANGLES**, **GL_TRIANGLE_STRIP** ou **GL_TRIANGLE_FAN**

et

- `void glEnd(void) ;`

Type de primitive graphique	
 **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

IMPORTANT : A l'exception du cas particulier des bitmaps, la librairie GL ne comporte aucune autre fonction de dessin que celle permettant de dessiner une des 10 primitives existantes.
-> Pas de modélisation à base d'objets canoniques en n'utilisant directement que les fonctions de la librairie GL.

Exemple: Création d'un polygone

```
glBegin(GL_POLYGON);
glVertex2f(0.0F,0.0F) ;
glVertex2f(2.0F,1.0F) ;
glVertex2f(1.0F,3.0F) ;
glVertex2f(5.0F,0.5F) ;
glVertex2f(-.5F,2.0F) ;
glVertex2f(1.5F,2.5F) ;
glEnd() ;
```

Outre ses sommets, la définition d'une primitive peut requérir d'autres informations. Celles-ci sont aussi renseignées via des appels à des fonctions particulières réalisés entre glBegin et glEnd.

Fonctions utilisables entre glBegin et glEnd	
Fonction	*But*
glVertex*()	Coordonnées d'un sommet
glColor*()	Couleur de tracé
glIndex*()	Indexe de couleur de tracé
glNormal*()	Vecteur normal à un sommet
glEvalCoord*()	Génération de coordonnées
glCallList() glCallLists()	Exécution de listes d'affichage
glTexCoord*()	Coordonnées d'une texture

`glEdgeFlag*()`	Contrôle du tracé d'un coté
`glMaterial*()`	Propriété du matériau d'un objet

Parmi ces fonctions, l'une des plus importantes permet la configuration de la valeur de la normale courante (vecteur orthogonal à la surface et utilisé pour les calculs d'éclairage):

- `void glNormal3{bsidf}(TYPE x,TYPE y,TYPE z);`
- `void glNormal3{bsidf}v(const TYPE *v);`

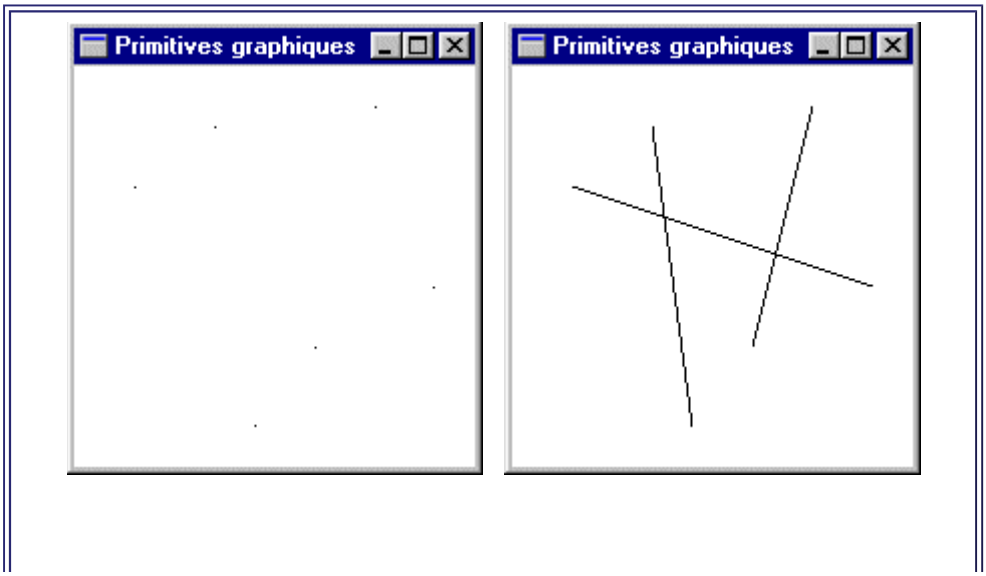
  x, y, z ou v: Coordonnées attribuées au vecteur "normal" (valeur par défaut : (0.0,0.0,1.0).

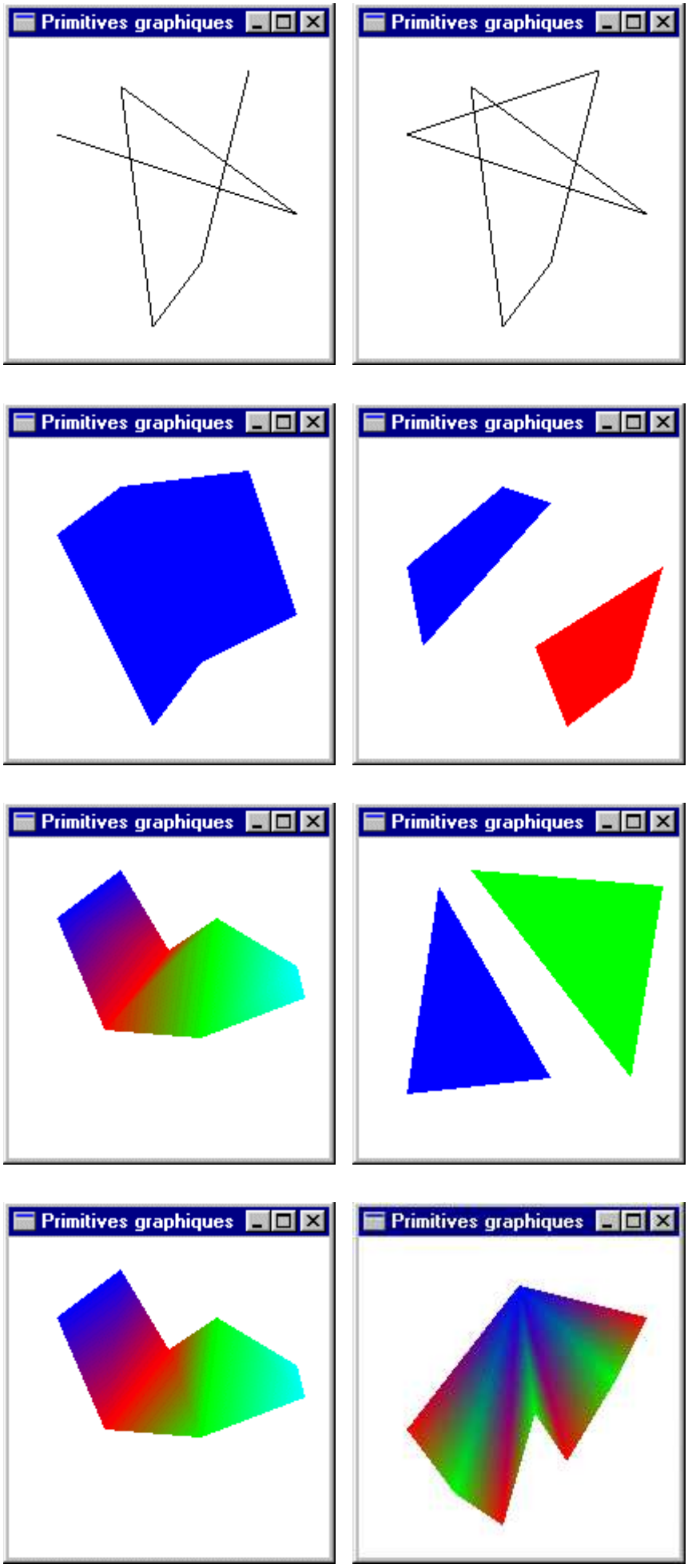
Exemple

```
glBegin(GL_POLYGON);
glColor3f(1.0F,0.0F,0.0F) ;
glVertex2f(0.0F,0.0F) ;
glNormal3f(-1.0F,0.0F,0.0F) ;
glVertex2f(2.0F,1.0F) ;
glColor3f(0.0F,1.0F,0.0F) ;
glVertex2f(1.0F,3.0F) ;
glEnd() ;
```

Définit un polygone à trois sommets.
Donne la couleur rouge aux premier et deuxième sommets, et la couleur verte au troisième.
Donne la normale (-1.0,0.0,0.0) aux deuxième et troisième sommets. Le premier adopte comme normale la valeur courante (inconnue dans l'exemple) définie avant le glBegin.

Les 10 primitives graphiques





Programme Aux	Programme GLUt
Exécutable Aux	Exécutable GLUt

Déclaration d'un ensemble de facettes

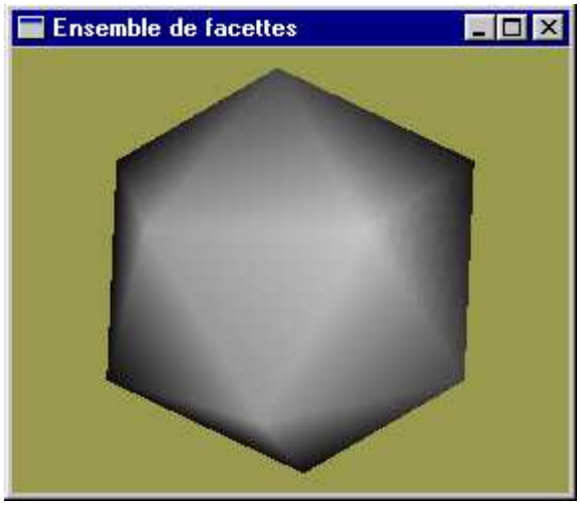
Utilisation fréquente de surfaces polygonales (ensembles de facettes adjacentes) pour modéliser des objets complexes.

Recommandations

- Conserver constante l'orientation des polygones.
- Eviter les facettes non triangulaires.
- Trouver un bon équilibre entre le nombre de polygones et la vitesse et la qualité d'affichage.
- Multiplier le nombre de polygones sur la silhouette des objets.
- Éviter des intersections en T.

Exemple

```
#define X .525731112119133606
#define Z .850650808352039932
static GLfloat vdata[12][3] = {
  {-X,0,Z},{X,0,Z},{-X,0,-Z},
  {X,0,-Z},{0,Z,X},{0,Z,-X},
  {0,-Z,X},{0,-Z,-X},{Z,X,0},
  {-Z,X,0},{Z,-X,0},{-Z,-X,0}} ;
static GLint t[20][3] = {
  {0,4,1},{0,9,4},{9,5,4},{4,5,8},
  {4,8,1},{8,10,1},{8,3,10},{5,3,8},
  {5,2,3},{2,7,3},{7,10,3},{7,6,10},
  {7,11,6},{9,2,5},{0,1,6},{6,1,10},
  {9,0,11},{9,11,2},{11,0,6},
  {7,2,11}};
for ( i = 0 ; i < 20 ; i++ ) {
  glBegin(GL_TRIANGLES) ;
    glNormal3fv(&vdata[t[i][0]][0]);
    glVertex3fv(&vdata[t[i][0]][0]);
    glNormal3fv(&vdata[t[i][1]][0]);
    glVertex3fv(&vdata[t[i][1]][0]);
    glNormal3fv(&vdata[t[i][2]][0]);
    glVertex3fv(&vdata[t[i][2]][0]);
  glEnd() ; }
```



Programme Aux	Programme GLUt
Exécutable Aux	Exécutable GLUt

Fonctions basiques pour la configuration de l'affichage des primitives graphiques

- **`void glPointSize(GLfloat size) ;`**

  Configuration de la taille de tracé des sommets (par défaut 1.0 pixel)

  size: Taille de tracé

- **`void glLineWidth(GLfloat width) ;`**

  Configuration de la largeur de tracé des segments de droite (par défaut 1.0 pixel)

  width: Largeur de tracé

- **`void glLineStipple(GLint facteur, GLushort motif) ;`**

  Configuration du motif de tracé des lignes (par défaut continu)
  Le motif est une série de 16 bits (poids faible à poids fort) où les 1 représentent des pixels allumés et des 0 des pixels non modifiés. Le facteur (de 1 à 255) permet l'étirement du motif par multiplication (un 1 devient factor 1 consécutifs).
  L'utilisation des motifs pour les segments est subordonnée à autorisation via un glEnable (GL_LINE_STIPPLE).

facteur: Facteur de zoom du motif
motif: Motif de tracé

- **`void glPolygonMode(GLenum face, GLenum mode) ;`**

  Configuration de la technique de dessin des faces avant et arrières des polygones (par défaut avec remplissage)

  face: Face configurée (GL_FRONT, GL_BACK ou GL_FRONT_AND_BACK)
  mode: Mode d'affichage (GL_POINT, GL_LINE ou GL_FILL pour un dessin des sommets, en fil de fer ou avec remplissage)

- **`void glFrontFace(GLenum mode) ;`**

  Configuration de la technique de détermination de la face frontale des polygones
  La valeur par défaut est GL_CCW (anti-horaire). Elle peut être inversée par GL_CW.

  mode: GL_CCW ou GL_CW

- **`void glCullFace(GLenum mode) ;`**

  Configuration de quels polygones doivent être éliminés au cours du processus d'affichage en fonction de leur orientation vers l'observateur (par défaut tous les polygones sont affichés)
  Permet, par exemple, de ne pas dessiner les polygones orientés vers le fond de la scène qui seraient donc obligatoirement masqués par d'autres objets si on ne dessine que des volumes.
  L'utilisation du culling est subordonnée à autorisation via un glEnable(GL_CULL_FACE).

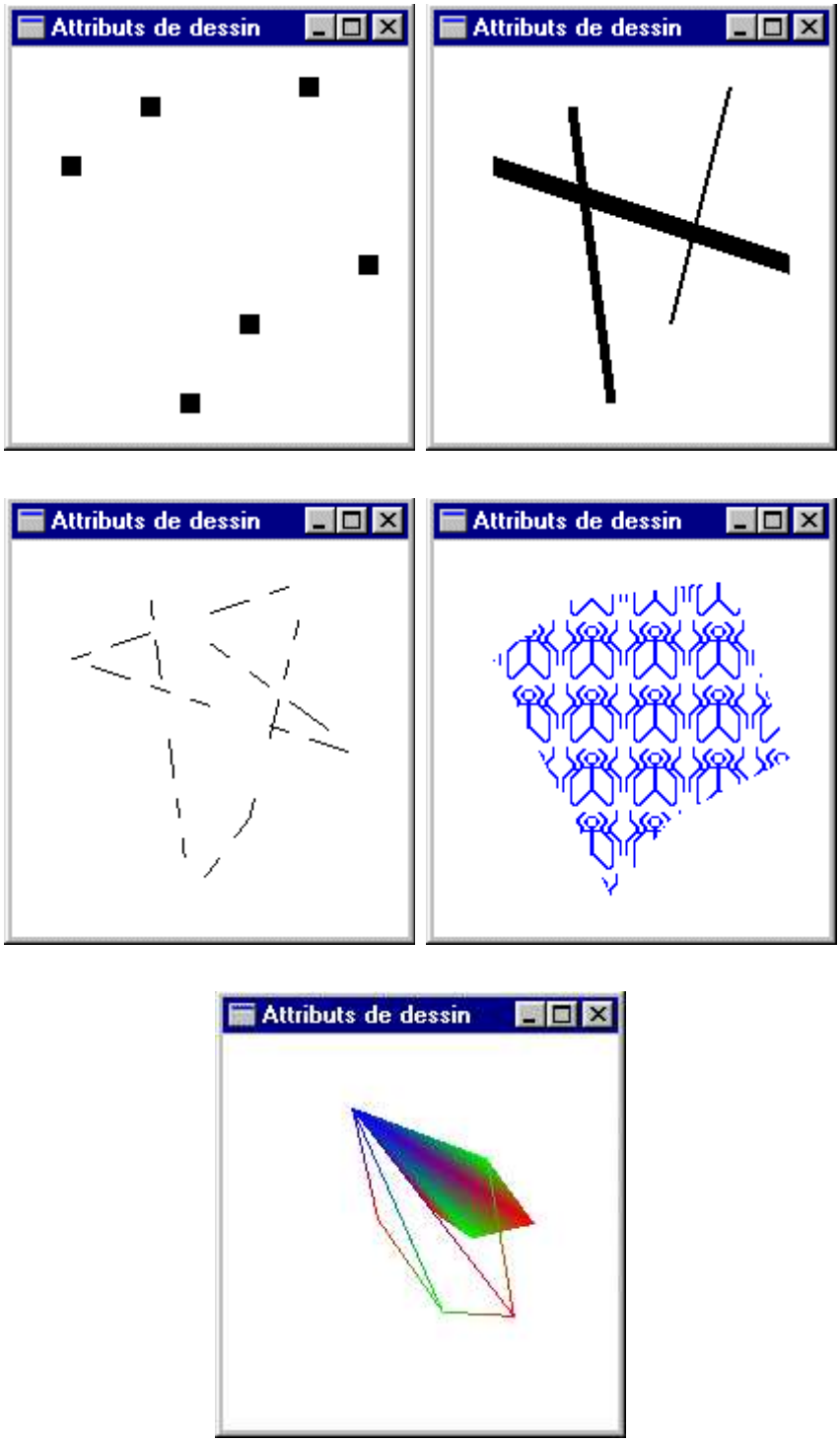
  mode: GL_FRONT, GL_BACK ou GL_FRONT_AND_BACK

- **`void glPolygonStipple(GLubyte *mask) ;`**

  Définition du motif de remplissage des polygones (par défaut plein)

L'utilisation des motifs pour les polygones est subordonnée à autorisation via un glEnable (GL_POLYGONE_STIPPLE).

mask: Pointeur sur une bitmap de 32x32 pixels interprétée comme un masque de bits (128 octets). Si un 1 est présent, le pixel est dessiné, sinon, il ne l'est pas.



Programme Aux	Programme GLUt
Exécutable Aux	Exécutable GLUt

Suite