Applications multimédias : modélisation et animation 3D

Frank Singhoff

Bureau C-203

Université de Brest, France

LISyC/EA 3883

singhoff@univ-brest.fr

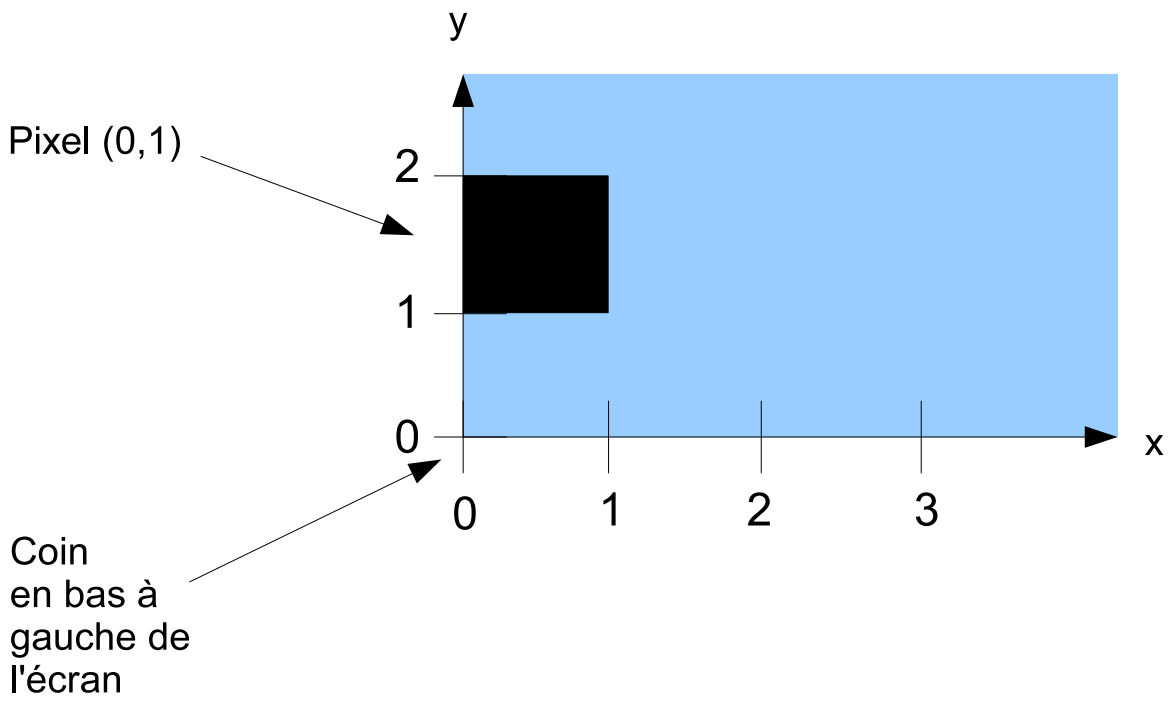
Sommaire

1. Quelques définitions : images fixes, flots continus et animation.
2. Du modèle 3D à l'image animée : introduction à la synthèse d'image.
3. Exemple d'une bibliothèque graphique : OpenGL.
4. Résumé, ce qu'il faut retenir.

Multimédias et flots continus

- **Quelques définitions [DEM 99] :**
  - **Médium :** moyen par lequel l'information est perçue, stockée ou transmise.
  - **Médium discret :** qui n'est pas assujetti à une contrainte temporelle (exemple : livre électronique, image fixe/non animée).
  - **Médium continu (ou flot continu) :** médium dont l'information est organisée en échantillons et où les échantillons sont contraints temporellement les uns vis-à-vis des autres (ex : séquence d'images, fichier MPEG2 Layer 3).
  - **Multimédias :** littéralement qui est constitué de plusieurs média (pluriel de médium) discrets ou continus.

Une image, c’est quoi ? (1)

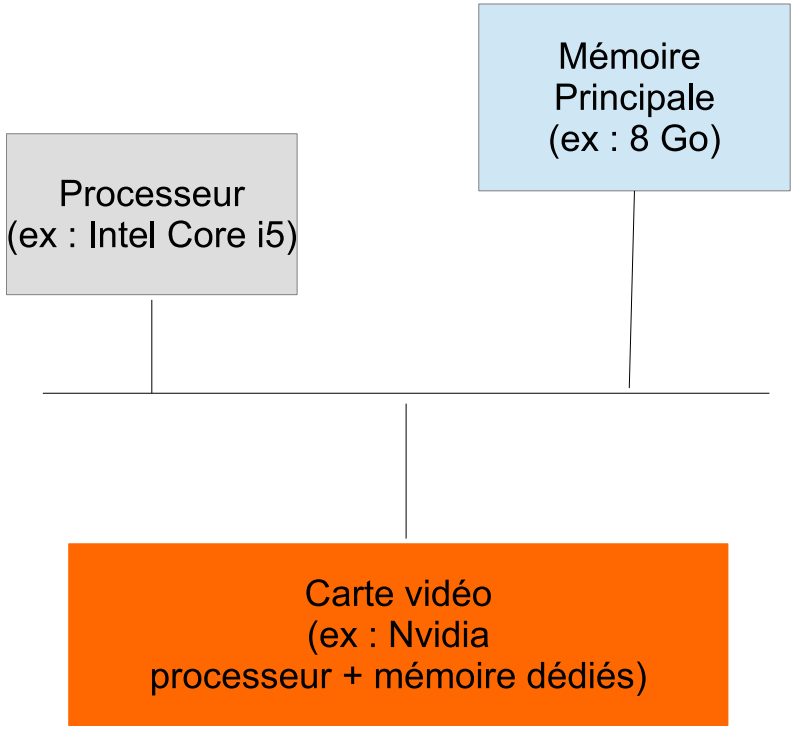


- Pixel : composant élémentaire d’une image.
- Image : ensemble de pixels organisés dans un plan.
- Tampon(s) : stocke une information pour chaque pixel d’une image (ex : couleur, profondeur).

Une image, c’est quoi ? (2)

- **Exemple de la couleur du pixel :**
  - Deux principaux codes existent : le RVB (ou RGB) et luminance/chrominance :
    1. **Luminance/chrominance.** Développé à la base pour la télévision hertzienne (passage du noir/blanc à la couleur). Code YIQ (NTSC) et code YUV (PAL/SECAM). Intensité de lumiere (Y) ; Information chromatique : IQ/UV.
    2. **RVB.** Chaque couleur est codée par 4 composantes qui définissent l’intensité en R(rouge), V(vert), B(bleue), A(alpha).
  - Conversion possible entre RGB/YIQ/YUV.
  - Exemple de codage RGB : 8 bits par pixel, soit 2 bits par composante, soit 256 couleurs.

Une image, c’est quoi ? (3)



- 2 processeurs et 2 mémoires
- 1 processeur/mémoire pour les programmes
- 1 processeur/mémoire dédiés à la vidéo

Une image, c’est quoi ? (4)

- **Mémoire vidéo :** mémoire d’un ordinateur dédiée à l’affichage des données informatiques.

- **Taille de la mémoire vidéo pour afficher une image fixe :**
  - Nombre de pixel en abscisse et ordonnée (taille de l’image).
  - Nombre d’information mémorisée pour chaque pixel (nombre de tampon).
  - Méthode de codage de chaque information.
  - Si l’information est compréssée ou non (ex : fichiers BitMap/BMP ou JPEG).

Une image, c'est quoi ? (5)

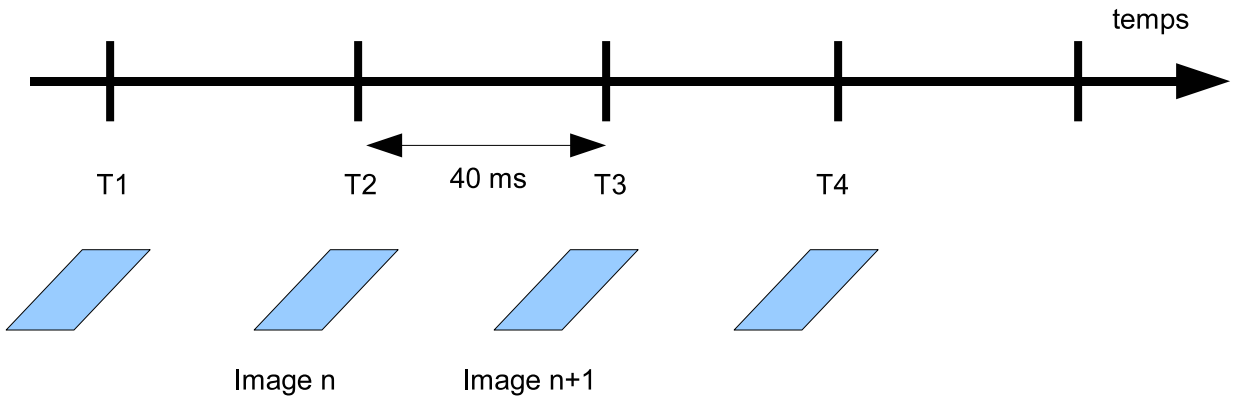
- **Exemple de calcul de la mémoire vidéo nécessaire pour l'affichage d'une image fixe :**
  - Couleur codée sur 24 bits (16777216 de couleurs possibles pour un pixel)
  - Taille de l'image de 1024x768 = 786432 pixels.
  - Taille de la mémoire vidéo : 24x1024x768/8 = 2,25 méga octets (fichier BITMAP).

$\Longrightarrow$ 24 bits/pixel pour la couleur est le standard pour les formats JPEG, MPEG : photo numérique, TV numérique, DVD, ...

$\Longrightarrow$ Téléphonie mobile/PDA [a] moins gourmand (ex : 8 bits/pixel).

[a] Personal digital assistant

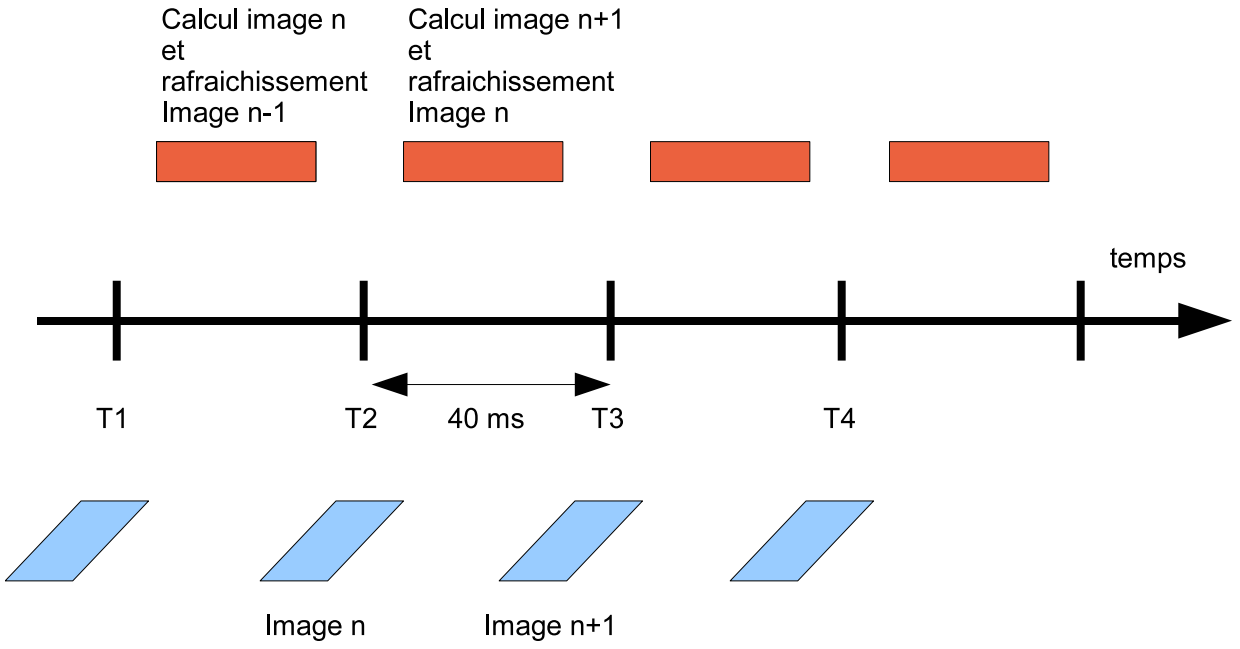
Un flot continu, c’est quoi ? (1)



- **Animer** c’est dessiner/élaborer puis afficher/présenter une séquence d’images. Synchronisations intra-flux et inter-flux.

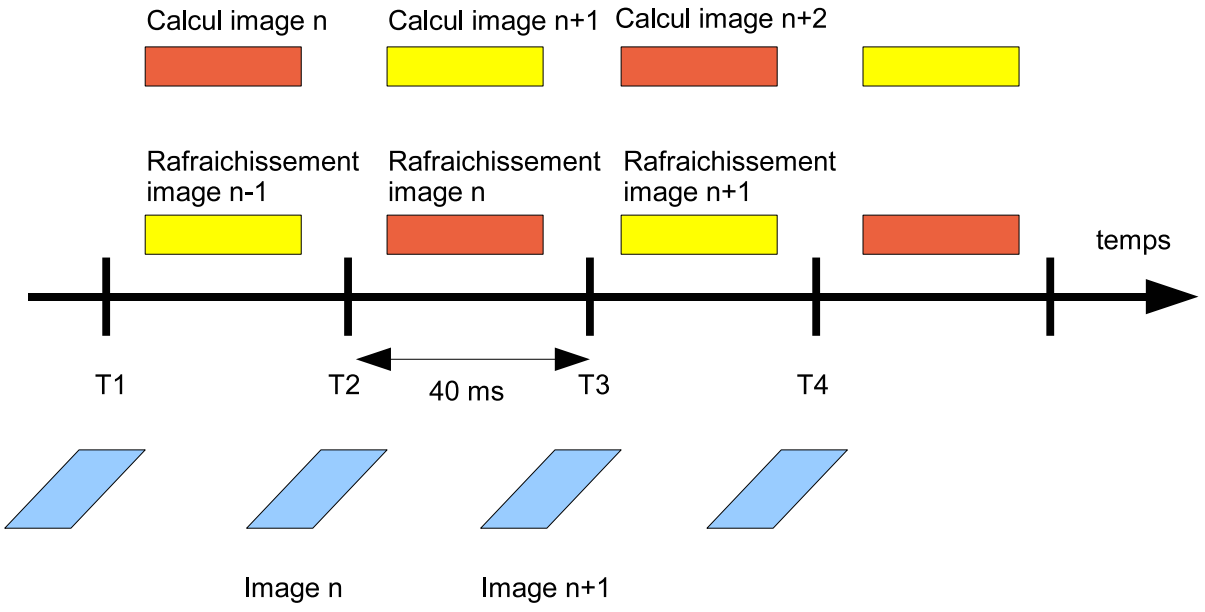
- **Taille de la mémoire nécessaire pour présenter des flots continus** :
  - Nombre de flots (vidéo, audio, sous-titre, ...).
  - Format de chaque échantillons/images.
  - La méthode d’animation (simple ou double "buffering").

Un flot continu, c'est quoi ? (2)



- **Animation *simple buffering* :**
  - Le processeur utilise un tampon pour constituer la prochaine image à afficher.
  - Simultanément, la carte graphique utilise ce même tampon pour afficher à l'écran l'image en cours.

Un flot continu, c'est quoi ? (3)



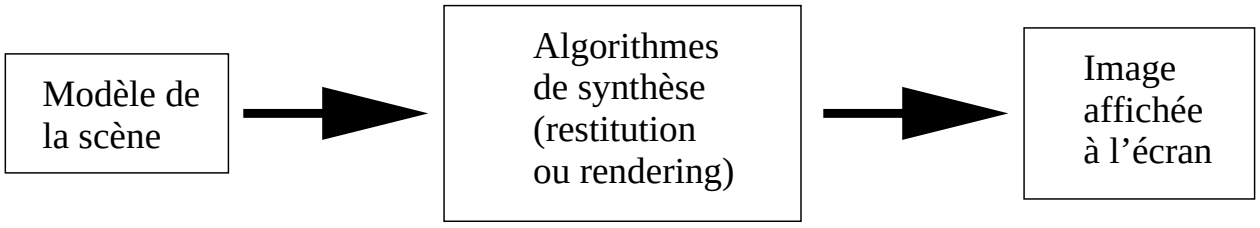
- **Animation *double buffering* :**
  - Le processeur utilise un tampon pour constituer l'image à afficher.
  - Simultanément, la carte graphique utilise un deuxième tampon pour afficher à l'écran l'image précédente.
  - Quand le processeur a terminé de préparer une image, les deux tampons sont échangés.

Sommaire

1. Quelques définitions : images fixes, flots continus et animation.
2. Du modèle 3D à l'image animée : introduction à la synthèse d'image.
3. Exemple d'une bibliothèque graphique : OpenGL.
4. Résumé, ce qu'il faut retenir.

Infographie : du modèle 3D à l'image

- **Infographie**: ou **informatique graphique**, peut être définie comme étant l'utilisation de l'ordinateur pour créer, mémoriser et manipuler des images.



- L'infographie 3D, ou comment restituer une scène 3D par une image fixe [LIE 88, WOO 04, MAL 05].

- **Domaines d'application** :
  - Conception assistée par ordinateur (ex : CATIA),
  - Simulation de processus, physique, scientifique, ...
  - L'imagerie médicale, cinéma, jeux.

Un modèle, c'est quoi (1)

- **Modèle :** représentation abstraite, ensemble d'objets organisés pour réprésenter une scène à afficher. Modèle = approximation de l'élément à représenter.

- Graphiquement, un modèle/objet c'est :
  - Soit un **ensemble de polygones** (polyèdre). Série de points généralement sur un seul plan (appelé facette dans ce cas). Pleins ou non (mode fil-de-fer).
  - Soit des **surfaces ou courbes calculées** par une méthode particulière (ex : Beziers, splines, fractales).
  - Soit un **assemblage hiérarchique d'objets canoniques** (ex : sphère, cube, cône, ...). Ces objets sont projetés/mis à échelle selon les besoins.

$\Longrightarrow$ Données vectorielles.

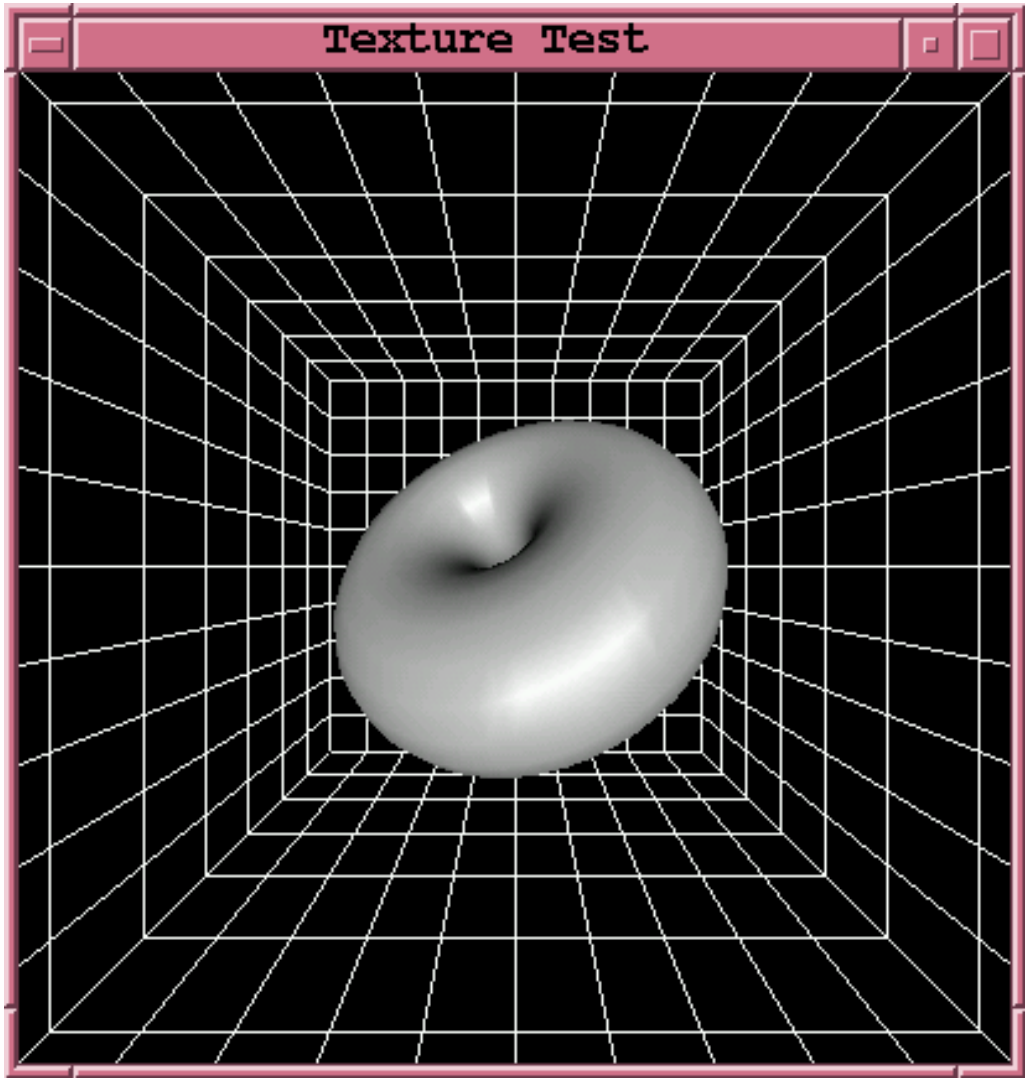
Un modèle, c’est quoi (2)

- **Outre les objets, une scène est généralement complétée par :**

  - La position de la caméra, son orientation, le cadrage, le champs de vision,
  - La description de différents phénomènes complémentaires tels que l’éclairage, l’ombrage, le brouillard, la transparence, le lissage,
  - Des données "pixel": des textures (procédés de remplissage de polygones), des polices,
  - ...

Une modèle, c’est quoi (3)

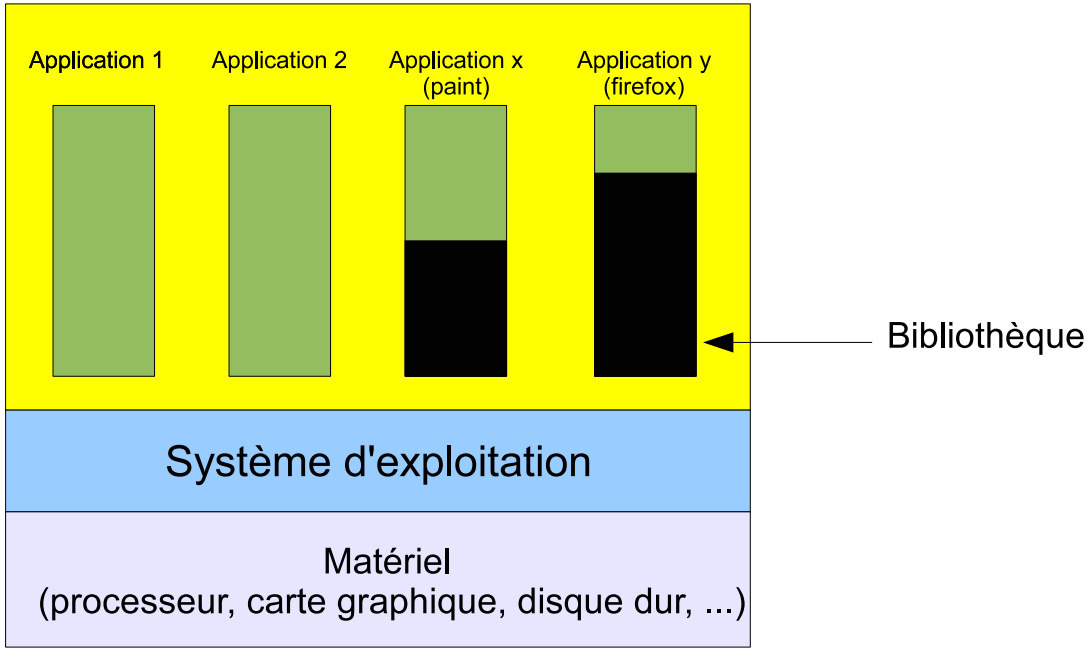
- Exemple de modèle/scène 3D :



Sommaire

1. Quelques définitions : images fixes, flots continus et animation.
2. Du modèle 3D à l'image animée : introduction à la synthèse d'image.
3. Exemple d'une bibliothèque graphique : OpenGL.
4. Résumé, ce qu'il faut retenir.

Une bibliothèque, c'est quoi ?



- **Bibliothèque :** ensemble de sous-programmes (fonctions et/ou procédures) réutilisables rassemblés dans un fichier, qui permet d'accélerer l'écriture de programmes.

  - Bibliothèque mathématique : sin, cos, sqrt, ... Exemple : bibliothèque C de Linux.
  - Bibliothèque interface homme-machine : bouton, fenêtre, zone de texte, menu .... Exemple : bibliothèque graphique de visual basic.
  - Bibliothèque graphique 3D.
  - ...

Bibliothèque graphique 3D (1)

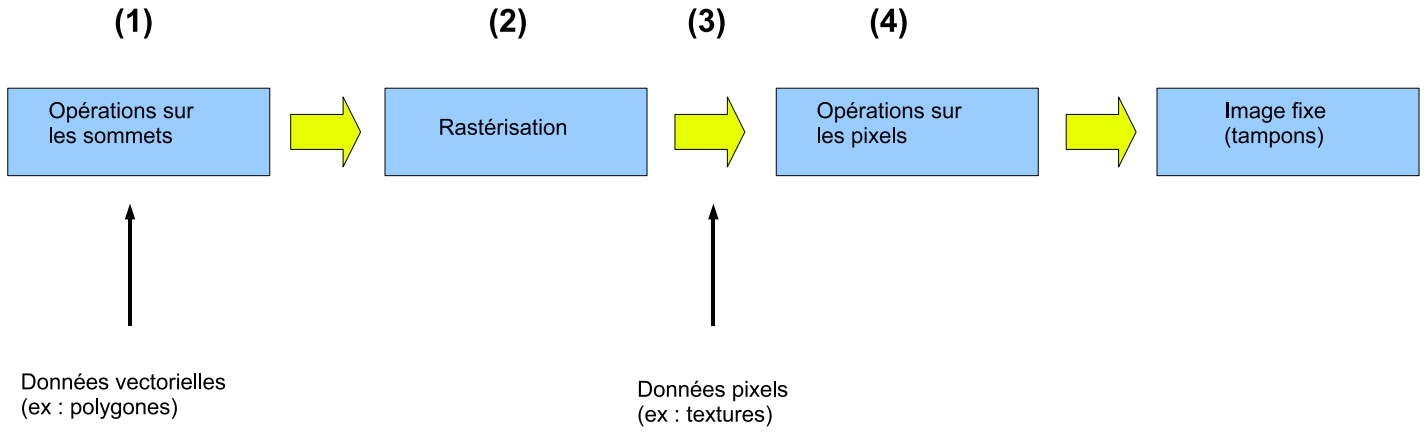
- **Bibliothèque graphique 3D :** ensemble de sous-programmes pour décrire et restituer une scène 3D.

- Ce que doit offrir une bibliothèque graphique 3D :

1. Description du modèle, manipulation de données pixels.
2. Outils de traduction d'un modèle en une image (processus de restitution ou $rendering$).
3. Interface pour la mise en oeuvre d'algorithmes d'animation élémentaire.
4. Interaction avec :
   - Le système d'exploitation (le système de multi-fenêtrage).
   - Les périphériques d'entrée (clavier, souris, haut-parleurs, ...).

Bibliothèque graphique 3D (2)

- **Processus de restitution (modèle $\Longrightarrow$ image fixe):**



1. Traduction des sommets en matrices puis transformation de projection/perspective/cadrage, mise en couleur et calcul d'éclairage des polygones, ...
2. Traduction des données vectorielles en pixels.
3. Puis fusion avec les données "pixels".
4. Tests de profondeur, stencil, alpha, dithering (correction de dégradé), effet brouillard, ...

OpenGl (1)

- **OpenGl, c'est :**
  - Un standard international.
  - Normalisé en 1989 (GL) par Sillicon Graphics, puis portée sur d'autres architectures en 1993 (OpenGL).
  - Avec OpenGL, on peut produire des images synthétiques sophistiquées (reflets, ombres) en temps réel si on dispose de ressources matérielles adéquates.
  - Existe sur de nombreuses architectures (NT, Unix, OS2, Amiga, etc...)
  - Nombreuses implémentations logicielles (Exemple sous NT : SGI OpenGL, 3dfx OpenGL, Mesa, Microsoft OpenGL...) et/ou matérielles (cartes graphiques).

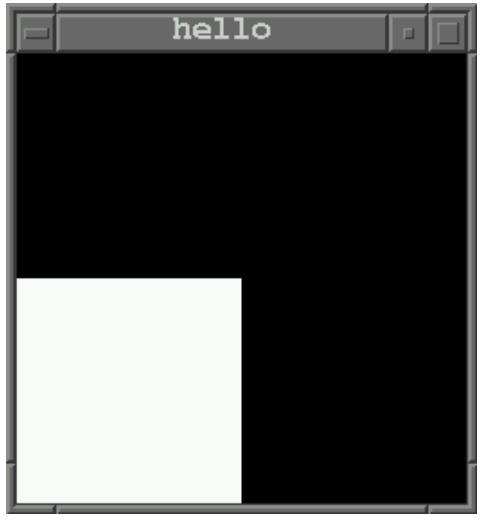
OpenGl (2)

- Quelles bibliothèques avec OpenGL ?

1. La bibliothèque **OpenGL** à proprement dit.
2. Des bibliothèques complémentaires :
   - **OpenGL Utility Library (GLU) :** surcouche à OpenGL, fournit des fonctions plus évoluées pour la gestion des caméras ou la création de modèle avec une haute définition.
   - **OpenGL Extension Library pour le système graphique X11 (GLX) :** interopérabilité avec le protocole X11.
   - **GL Utility Toolkit (GLUT)** : fournit une interface simple pour le dialogue avec le système d'exploitation.

OpenGL (3)

- Comment dessiner ce modèle :

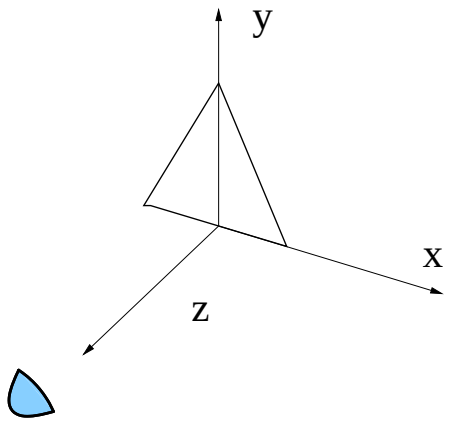


- Pour ce faire, il faut :

1. Vider l'écran, c-à-d manipuler les tampons.
2. Choisir une couleur pour dessiner (le blanc), c-à-d coder/manipuler la couleur.
3. Dessiner le carré, c-à-d définir un polygone.

Approximation polyédrique (1)

- Comment mémoriser un volume ? un objet 3D ?

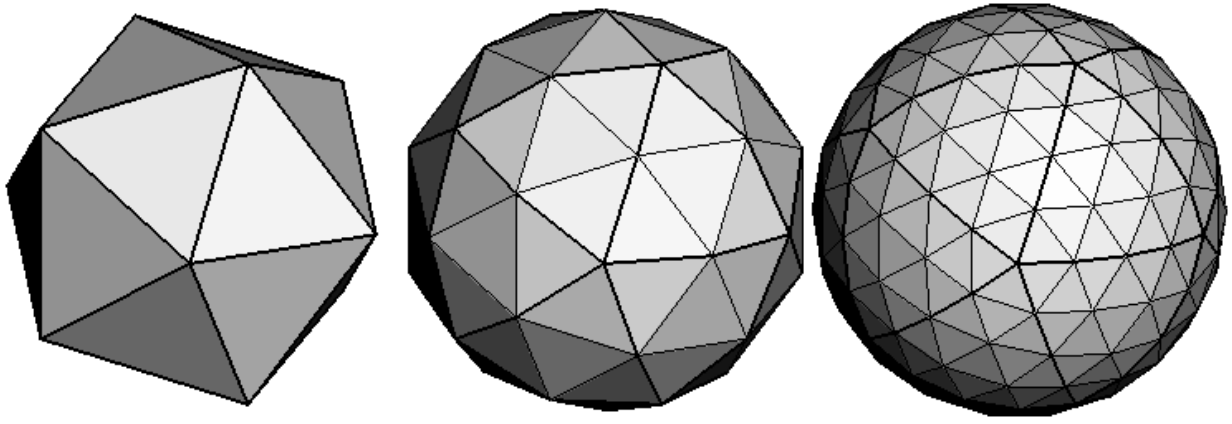


Coordonnées = (x,y,z)

- Chaque sommet est défini par un triplé (x,y,z) le situant dans un repère cartésien.
- Un objet 3D est approximé par une série (finie) de sommets calculés ou non. Les sommets peuvent constituer soit :
  - Des polyèdres (ensemble de sommets constituant des faces, ou polygones).
  - Des courbes ou surfaces d'interpolation (ex : Beziers ou B-splines).

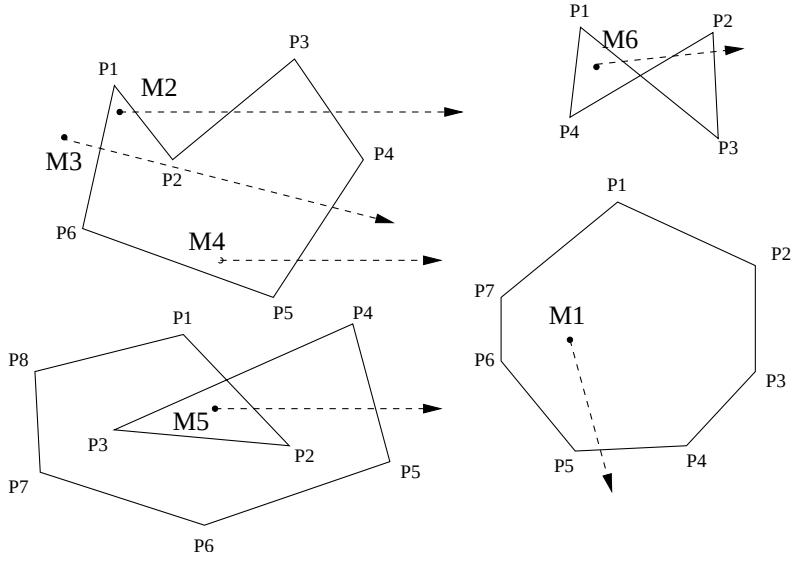
Approximation polyédrique (2)

- Exemple : la sphère



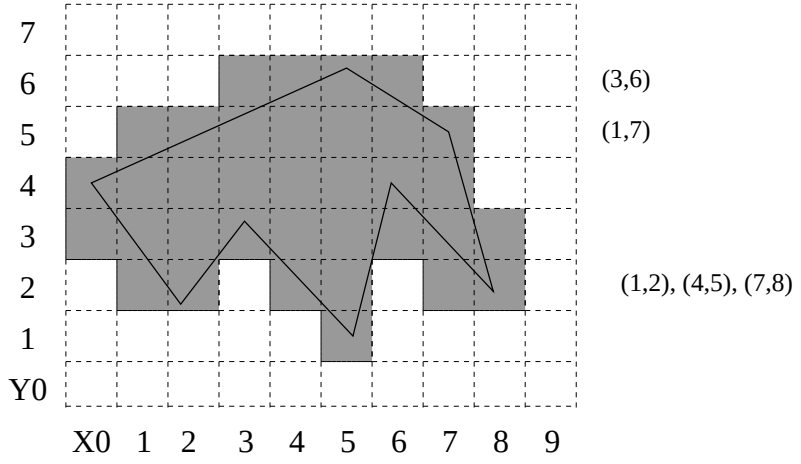
- Nécessaire compromis entre efficacité et qualité de l'image (méthode d'approximation) :
  - Nombre de sommets et stockage en mémoire.
  - Temps de calcul pour les animations (ex : translation) ou les effets graphiques (ex : lumière).

Approximation polyédrique (3)



- **Polygone dans un plan :** est défini par une suite de segments $s1, ..., sn$ tel que le segment $si = [Pi, Pi + 1]$ où Pi est un point du plan avec P1=Pn. $i = 1, ..., n$ s'appellent les "sommets" du polygone. Les segments si sont nommés "arêtes".
- **Point intérieur :** soit un polygone P est un point M du plan. M est intérieur si la demie-droite issue de M intersecte un nombre impair de segments de P.

Approximation polyédrique (4)



- Algorithme de remplissage naïf :
  1. Pour chaque pixel, tester par parité son appartenance au polygone.
  2. Colorier le pixel selon (1).
- Exemple d'algorithme de remplissage de polygones de type "scanline" :
  1. Traiter le polygone ligne par ligne.
  2. Parcourir les arêtes du polygone et détecter les segments intérieurs.
  3. Colorier segments par segments.

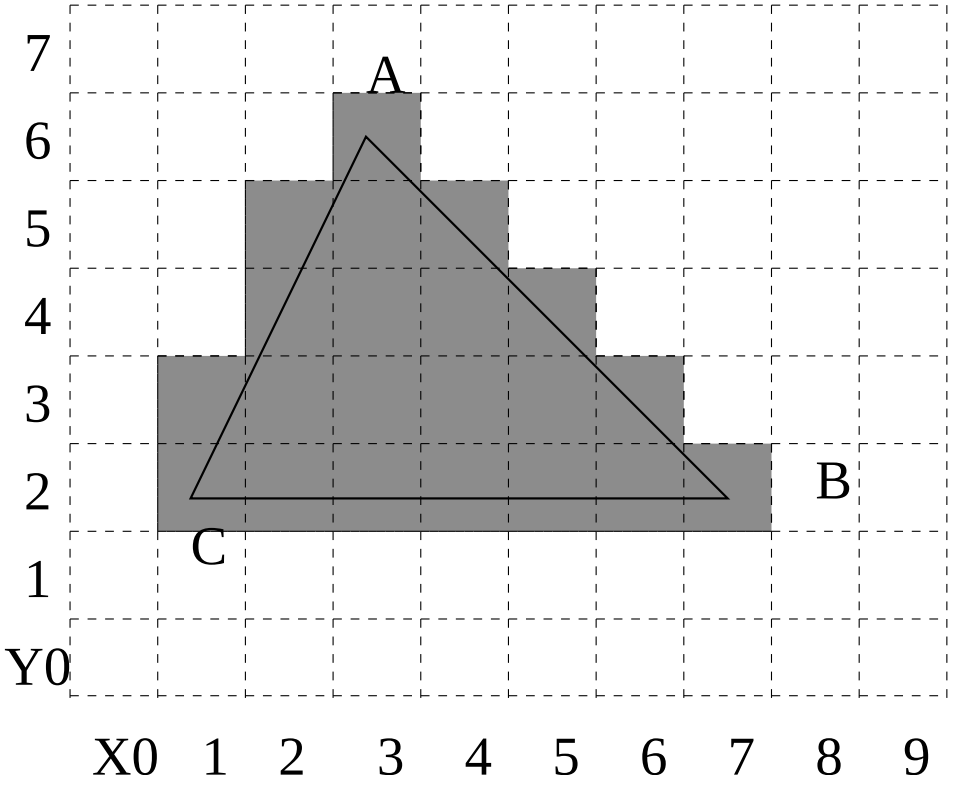
Approximation polyédrique (5)

- **Polygone convexe :** un polygone P est convexe si
  1. Pour tous points M et M’ qui sont intérieurs à P, le segment [M,M’] est entièrement composé de points intérieurs à P.
  2. Toutes ses diagonales (segment qui joint 2 sommets non consécutifs) sont entièrement à l’intérieur de la surface délimitée par le polygone.

- **Polygone concave :** P est concave si l’une de ses diagonales n’est pas entièrement à l’intérieur de la surface délimitée.

- **Polygone avec autointersection:** P contient une autointersection si deux de ses segments non consécutifs s’intersectent.

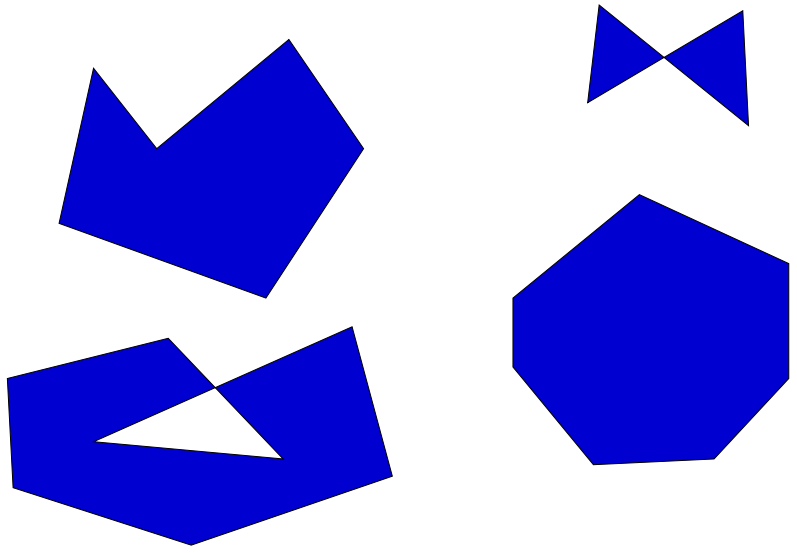
Approximation polyédrique (6)



Xmin Xmax	AB	BC	CA
	(+inf, −inf)	(+inf, −inf)	(+inf, −inf)
	(+inf,3)	(+inf,3)	(3,3)
	(+inf,4)	(+inf,4)	(2,4)
	(+inf,5)	(+inf,5)	(2,5)
	(+inf,6)	(+inf,6)	(1,6)
	(+inf,7)	(1,7)	(1,7)
	(+inf, −inf)	(+inf, −inf)	(+inf, −inf)
	(+inf, −inf)	(+inf, −inf)	(+inf, −inf)

Approximation polyédrique (7)

- Polygones remplis :



- Attention : beaucoup de bibliothèques (c'est le cas pour OpenGL) supposent que les polygones sont convexes !!!
- Des polygones concaves, à trous, avec autointersection peuvent être redécoupés en sous-polygones.

Approximation polyédrique (8)

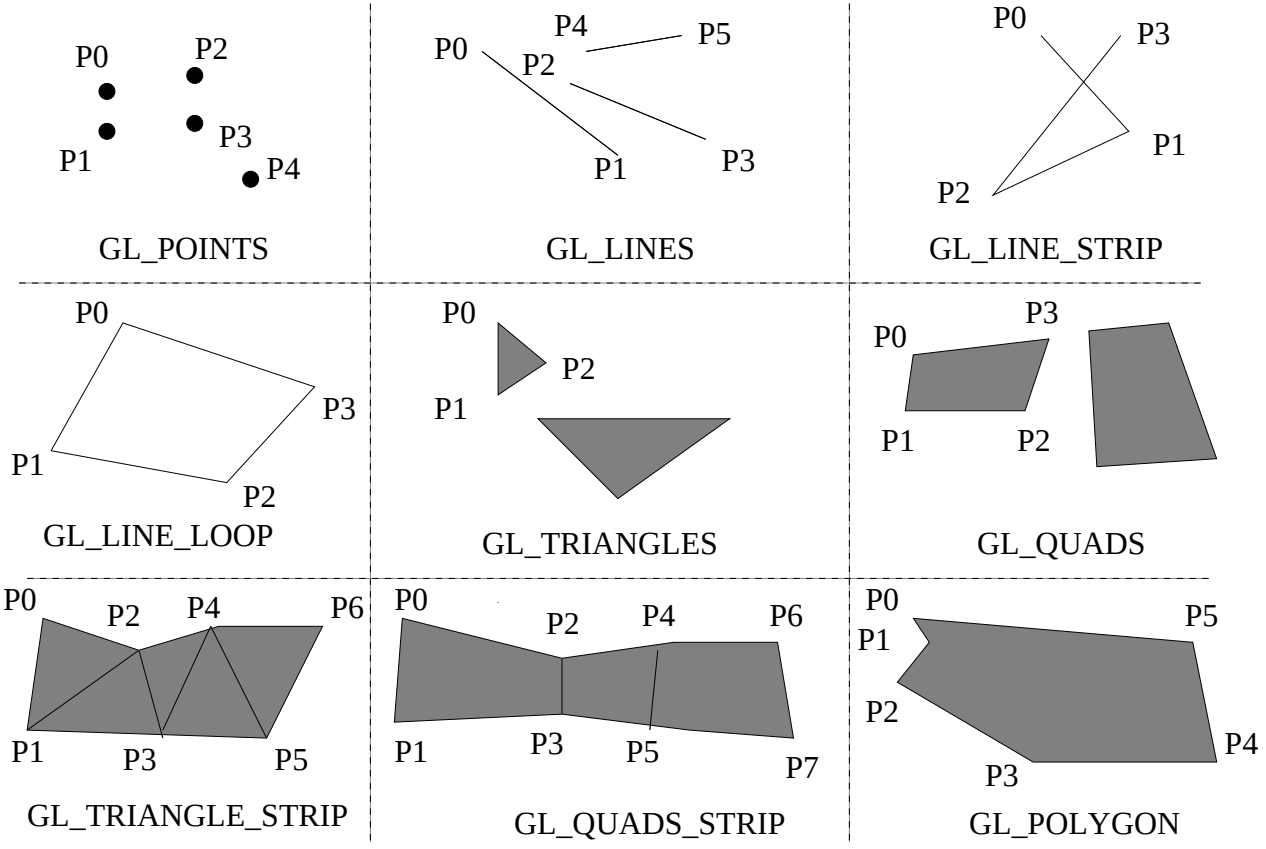
- Comment définit-on un polygone avec OpenGl :

1. Par une liste **ordonnée** de sommets. Un sommet est défini par la fonction $glVertex3f$. Une liste de sommets est définie par les fonctions $glBegin$ et $glEnd$. **Open GL ne supporte que les polygones convexes**.
2. Par une méthode de connection de ces sommets (ou primitives).

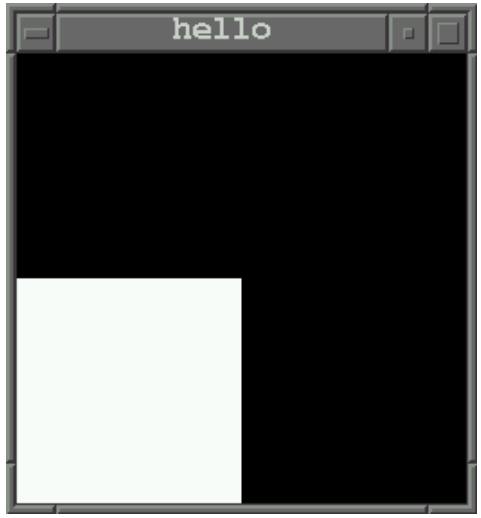
- Primitives disponibles : GL_POINTS, GL_LINES, GL_LINE_STRIP, $GL\_TRIANGLES$, $GL\_TRIANGLES\_STRIP$, $GL\_TRIANGLE\_FAN$, GL_QUADS, GL_QUADS_STRIP, $GL\_POLYGON$.

Approximation polyédrique (9)

- Primitives OpenGL pour tracer des points/lignes/polygones :



Approximation polyédrique (10)



- Dessiner cette scène, c’est :

1. Vider l’écran, c-à-d manipuler les tampons.
2. Choisir une couleur pour dessiner (le blanc), c-à-d coder/manipuler la couleur.
3. Dessiner le carré, c-à-d définir un polygone.

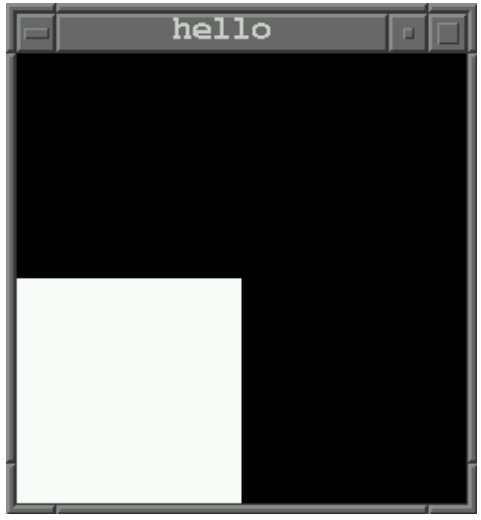
Approximation polyédrique (11)

- Exemple d'affichage (en langage C) d'une scène :

```
/* Dessine le polygone */
   glBegin(GL_POLYGON);
      glVertex3f (0.0, 0.0, 0.0);
      glVertex3f (0.5, 0.0, 0.0);
      glVertex3f (0.5, 0.5, 0.0);
      glVertex3f (0.0, 0.5, 0.0);
   glEnd();
```

  - Quelle est la dimension du polygone ?

Gestion de la couleur (1)



- Dessiner cette scène, c’est :

1. Vider l’écran, c-à-d manipuler les tampons.
2. Choisir une couleur pour dessiner (le blanc), c-à-d coder/manipuler la couleur.
3. Dessiner le carré, c-à-d définir un polygone.

Gestion de la couleur (2)

- OpenGL utilise le codage RGB.
- Fonction $glColor3f$: positionne la couleur active.
- Les valeurs de chaque composante sont comprises entre 0 et 1 (intensité de la lumière, valeurs flottantes).
- Quelques exemples de valeurs significatives :

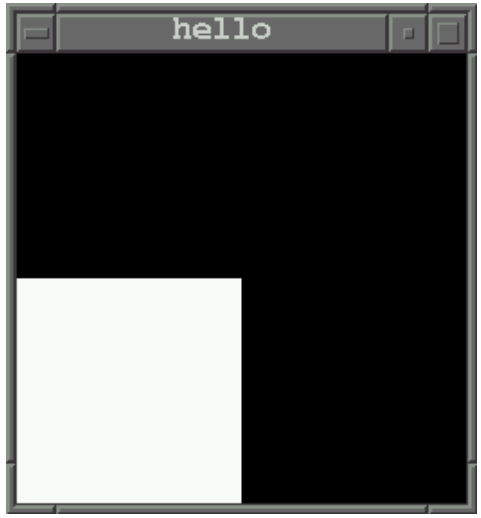
```
glColor3f(0.0, 0.0, 0.0);   // noir
glColor3f(0.0, 0.0, 1.0);   // bleu
glColor3f(0.0, 1.0, 0.0);   // vert
glColor3f(1.0, 0.0, 0.0);   // rouge
glColor3f(0.7, 0.7, 0.7);   // gris clair
glColor3f(0.3, 0.3, 0.3);   // gris fonce
glColor3f(1.0, 1.0, 1.0);   // blanc
glColor3f(1.0, 1.0, 0.0);   // jaune
glColor3f(1.0, 0.6, 0.6);   // rose
```

Gestion de la couleur (3)

- Exemple d'affichage de notre scène :

```
/* Dessine le polygone en BLANC ! */
   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_GUADS);
      glVertex3f (0.0, 0.0, 0.0);
      glVertex3f (1.0, 0.0, 0.0);
      glVertex3f (1.0, 1.0, 0.0);
      glVertex3f (0.0, 1.0, 0.0);
   glEnd();
```

Gestion des tampons (1)



- Dessiner cette scène, c'est :

1. Vider l'écran, c-à-d manipuler les tampons.
2. Choisir une couleur pour dessiner (le blanc), c-à-d coder/manipuler la couleur.
3. Dessiner le carré, c-à-d définir un polygone.

Gestion des tampons (2)

- **Tampons avant (GL_FRONT) et arrière (GL_BACK) suivants :**
  - chromatiques ($GL\_COLOR\_BUFFER\_BIT$).
  - de profondeur ($GL\_DEPTH\_BUFFER\_BIT$).
  - d'accumulation ($GL\_ACCUM\_BUFFER\_BIT$).
  - stencil ($GL\_STENCIL\_BUFFER\_BIT$).

- **Principales fonctions :**
  1. Vider un tampon : $glClear$.
  2. Sélectionner la valeur pour vider un tampon : $glClearColor$, $glClearDepth$, $glClearStencil$, $glClearAccum$.
  3. Copier/Coller/lire/écrire : $glDrawBuffer$, $glReadBuffer$, $glCopyPixels$, ...
  4. Tests tels que lissage, stencil, profondeur, ...

Gestion des tampons (3)

- **Exemple du test de profondeur :**
  - Le test de profondeur permet de **rejeter**/ne pas afficher les polygones cachés par d'autres (opération de clipping).
  - Il n'est pas activé par défaut. Activation par :

$$glEnable(GL\_DEPTH\_TEST)$$

  - Lors de la restitution, le tampon $GL\_DEPTH\_BUFFER\_BIT$ est rempli conformément à la position des différents objets. Entre deux restitutions, ce tampon doit donc être vidé par :

$$glClear(GL\_DEPTH\_BUFFER\_BIT)$$

Gestion des tampons (4)

- Exemple d'affichage de notre scène :

```
/* Vider les tampons chromatique
    et de profondeur */
   glClear(GL_COLOR_BUFFER_BIT);
   glClear(GL_DEPTH_BUFFER_BIT);

/* Dessiner le polygone en BLANC */
   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_GUADS);
      glVertex3f (0.0, 0.0, 0.0);
      glVertex3f (1.0, 0.0, 0.0);
      glVertex3f (1.0, 1.0, 0.0);
      glVertex3f (0.0, 1.0, 0.0);
   glEnd();
```

Fonction d'affichage complète

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

void display(void)  {
/* Vider les tampons */
   glClear(GL_COLOR_BUFFER_BIT);
   glClear(GL_DEPTH_BUFFER_BIT);

/* Dessiner le polygone */
   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_GUADS);
      glVertex3f (0.0, 0.0, 0.0);
      glVertex3f (1.0, 0.0, 0.0);
      glVertex3f (1.0, 1.0, 0.0);
      glVertex3f (0.0, 1.0, 0.0);
   glEnd();

/* Forcer le rendu maintenant */
   glutSwapBuffers ();
}
```

Sommaire

1. Quelques définitions : images fixes, flots continus et animation.
2. Du modèle 3D à l'image animée : introduction à la synthèse d'image.
3. Exemple d'une bibliothèque graphique : OpenGL.
4. Résumé, ce qu'il faut retenir.

Résumé, ce qu'il faut retenir

- Ce que contient une image et les méthodes d'animation (notion de flots continus, $simple$ ou $double\ buffering$).
- Infographie, processus de restitution.
- Comment décrire un objet par un polyèdre.
- Comment manipuler couleurs et tampons. Codage RGB.

Bibliographie

[DEM 99] I. Demeure and C. Bonnet. *Introduction aux systèmes temps réel*. Collection pédagogique de télécommunications, Hermès, septembre 1999.

[LIE 88] T. Liebling and H. Rothlisberger. « Infographie et applications ». Editions Masson, 1988.

[MAL 05] R. Malgouyres. « Algorithmes pour la synthèse d'image 3D ». Editions Dunod, 2005.

[WOO 04] M. Woo, J. Neider, T. Davis, and D. Shreiner. « Guide officiel d'Open GL 1.4 ». Campus Press, 2004.

Annexe : présentation du langage C (1)

- **Le langage C en quelques mots :**

1. Langage de programmation généraliste et proche du système d'exploitation (gestion mémoire).

2. Langage de programmation impératif et structuré :
   - Blocs : ensemble d'instructions délimité par { et }
   - Fonctions : ensemble d'instructions paramétrable qui réalise une tache donnée, en vue d'être réutilisé (factorisation du code). Peut être un élément de bibliothèque.

Annexe : présentation du langage C (2)

- **Structure simplifiée d'un programme :**

```
/* Références aux bibliothèques utilisées */

/* Fonction principale */
int main(int argc, char** argv)
{
/* Déclarations : variables,
                  types,
                  fonctions */

/* Instructions (séquentielles,
                 itératives,
                 conditionnelles,
                 appels de fonctions) */
}
```

Annexe : présentation du langage C (3)

- **Types et déclarations de variables :**
  - Types : caractère, entier, flottant, chaîne de caractères. ...
  - Déclaration d'une variable entière :

```
int a;                    /* int nom_variable=valeur */
```

  - Déclaration d'une variable flottante :

```
double b;              /* double nom_variable=valeur */
```

  - Déclaration d'une tableau d'entier ou d'une chaine de caractères :

```
char str [100];    /* char nom_variable [TAILLE]*/
```

- **Opérateurs :**
  - Opérateurs arithmétiques : $+$, $-$, $/$, $*$
  - Opérateurs logiques : $==$ (égalité), $!=$ (inégalité), $<=$, $>=$

Annexe : présentation du langage C (4)

- **Structures de contrôle élémentaires :**

  - Affectation :

```
a=100;                          /* variable=valeur; */
```

  - Séquence :

```
a=100;          /* instruction1; instruction 2; */
b=1.5;
```

  - Instruction conditionelle :

```
if(a==100)                  /* if (test)          */
     b=b-1.5;               /*      instruction1; */
else b=b+1.5;               /* else instruction2; */
```

Annexe : présentation du langage C (5)

- Itération (exemple 1) :

```
int var [100];
for(int i=0;i<100;i++){/* for(init;test;incrément){*/
   var[i]=0;           /*    instructions;         */
   }                   /* }                        */
```

- Itération (exemple 2) :

```
int var [100];
int i;
i=0;
while (i<100) {          /* while(test) {   */
   var[i]=0;             /*   instructions; */
   i=i+1;                /* }               */
   }
```

Annexe : présentation du langage C (6)

- **Exemple de programme :**

```
#include <stdio.h>

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

   int calculer_somme(int n) {
      int i;
      int s=0;
      for (i=0;i<=n;i++) {
         s=s+i;
      }
      return s;
   }

  int a=5;

  int somme=calculer_somme(a);
  printf("somme = %d\n", somme);
}
```