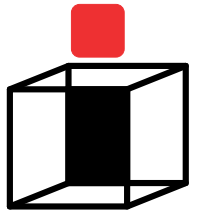
UNIVERSITÉ PARIS 8
VINCENNES-SAINT-DENIS

Programmation graphique

Sylvia Chalençon
Licence informatique



UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Généralités

- La programmation graphique est l'ensemble des techniques pour la création, la manipulation et le rendu d'objets géométriques et d'images dans l'espace 2D ou 3D.

- Les deux principales bibliothèques (OpenGL et Direct3D) réunissent plusieurs phases essentielles :
  - Transformations géométriques (vecteurs & matrices)
  - Rasterization
  - Textures, shaders

- OpenGL est une interface de programmation développée, à l'origine, par Silicon Graphics.

- Direct3D, sous-ensemble de DirectX, est l'interface développée par Microsoft.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

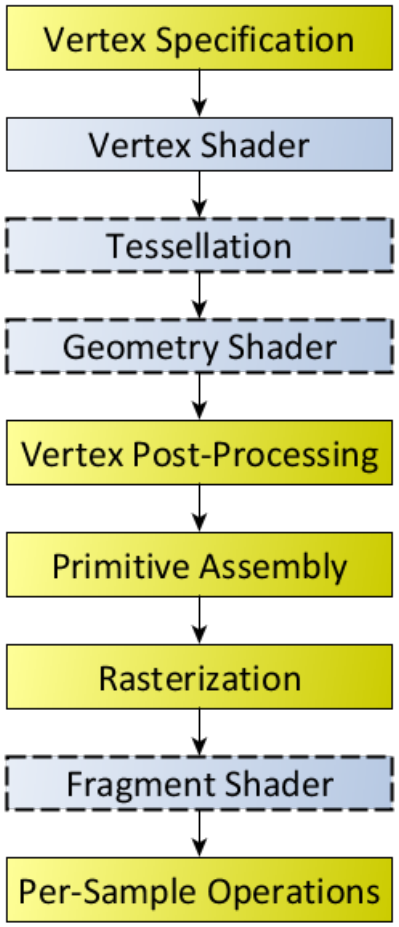
Pipeline

- OpenGL fournit un Pipeline fixe permettant de calculer un rendu en fonction du flux de données fourni en entrée.

- Ces données sont composées :
  - d'une part d'un enchevêtrement de commandes OpenGL et de données géométriques décrivant les objets et leur placement dans la scène
  - d'autre part de données bitmaps généralement utilisées pour décrire les textures des objets.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Pipeline



https://www.khronos.org/opengl/wiki_opengl/images/RenderingPipeline.png

- Le pipeline de rendu est la suite d'étapes qu'OpenGL effectue lors du rendu d'objets.
- Cet aperçu fournit une description de haut niveau des étapes du pipeline.
- Les boîtes bleues se programment au niveau des shaders.
- Les boîtes en pointillés correspondent à des étapes optionnelles.
- Les opérations de rendu nécessitent la présence d'un tableau de sommets (VAO) et d'un objet programme de rendu lié qui fournit les shaders pour les étapes du pipeline programmable par ce biais.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Pipeline

- Vertex specification (calcul des sommets et de leurs caractéristiques) : créer un flux de sommets (VAO), et indiquer à OpenGL comment interpréter ce flux.
- Vertex processing (traitement des sommets) :
  - Vertex Shader : Chaque sommet est soumis à une action dans le Vertex Shader et est transformé en un sommet de sortie.
  - Tessellation (optionnelle, OpenGL 4.0 et +): patchs de sommets subdivisés en primitives plus petites.
  - Geometry Shader (optionnel) : création de nouvelles géométries à partir de la sortie du Vertex shader.
- Vertex post-processing : transformation des coordonnées des sommets dans l'espace fenêtre.
- Primitive Assembly (Assemblage de primitives)
- Rasterization : chaque primitive est décomposée en éléments discrets appelés fragments
- Fragment Shader : Chaque fragment de l'étape précédente est traité par le fragment shader (couleur, lumière, ombrage, profondeur).
- Per-sample operations : Depth test, blending

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Traitement des sommets

- Calcul de l'ensemble des transformations géométriques telles que rotations, translations, changement de repère / d'échelle, projection, etc…

- Principalement appliquées aux sommets mais aussi aux coordonnées de textures, aux normales, aux faces.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Rasterization

- Cette étape consiste à transformer des données encore vectorielles, représentées dans un pseudo-espace bidimensionnel prenant en compte la profondeur (un cube dans le quel les objets sont projetés orthogonalement sur l'une des faces du cube), en fragments.

- Un fragment est l'élément qui, in fine, sera le pixel, à un détail près : un pixel peut être composé (calculé à partir) de plusieurs fragments.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Convention de nommage

- L'ensemble des éléments provenant de l'API OpenGL sont préfixés des deux lettres gl.
  - Ces lettres se présentent en minuscule pour les fonctions
  - En majuscule pour les macros et types de données
- Ainsi une fonction OpenGL peut prendre la forme suivante : glNomCourt{u}{bsifd}{v}(type1 v1, type2 v2, …);
- Le nom court étant le nom de la fonctionnalité. Il peut être suffixé d'éléments informant sur la nature et le type de données d'un sous-ensemble ou de l'intégralité des arguments de la fonction :
  - La présence de la lettre u indique que le type de données du ou des argument(s) ciblé(s) est non signé. Ce suffixe n'est valable que dans les cas où il serait suivi de b, s ou i;
  - La présence d'une des lettres {b, s, i, f, d} indique le type de données du ou des argument(s) ciblé(s).
  - La présence de la lettre v indique que le ou les arguments ciblés sont des vecteurs (des tableaux de données).

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

GL4Dummies

- 2010 : Sortie OpenGL® version 3.3. D'anciennes fonctionnalités dépréciées depuis la version 2.0 ont été officiellement supprimées :
  - création/gestion de géométrie (glNewList, glEndList, glBegin, glEnd, ...),
  - injection de sommets (glVertex), choix de leur couleur (glColor), paramétrage de leur vecteur normal (glNormal) et de leurs coordonnées de textures (glTexCoord), ...
  - transformations spatiales (glRotate, glTranslate, glScale),
  - plus de rendu par défaut d'une scène,
  - disparition des fonctions d'activation, paramétrage et gestion des lumières de la scène
- Depuis 2014 : GL4Dummies est une API, développée par Farès Belhadj, qui sert de support de développement pour l'ensemble des cours liés à la synthèse d'images enseignés à l'Université Paris 8.
- Son but est de faciliter l'initiation, uniformisation de l'installation et la gestion des nouvelles possibilités offertes par l'architecture OpenGL.
- GL4Dummies donne accès, à un ensemble de fonctionnalités reprenant dans la mesure du possible le modèle de celles supprimées depuis la version 3.3.
- GL4Dummies suit la même convention de nommage qu'OpenGL en préfixant par gl4d.

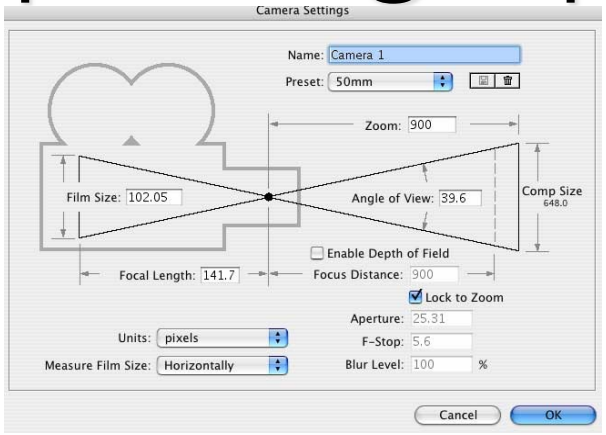
UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Installation

- Pré-requis : SDL2 (http://www.libsdl.org/) pour la gestion des fenêtres et du contexte OpenGL.
- Récupérer les sources de GL4Dummies : https://github.com/noalien/GL4Dummies
- Sous Linux et OSX avec droits administrateur, se positionner dans le répertoire contenant les sources de GL4Dummies, et lancer :
  - make -f Makefile.autotools (Pour cela il faut disposer des outils autotools : automake, autoconf et libtool)
  - ./configure
  - make
  - sudo make install
  - Vérifier que la variable d'environnement $LD_LIBRARY_PATH pointe sur /usr/local/lib.
- Si vous n'avez pas les droits administrateurs, il est possible de l'installer en local.
- Sous Windows, dans le répertoire des sources, il existe deux projets GL4Dummies.cbp (Code::Blocks) ou GL4Dummies.sln (Visual Studio). Ouvrez un des deux et compilez le projet.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

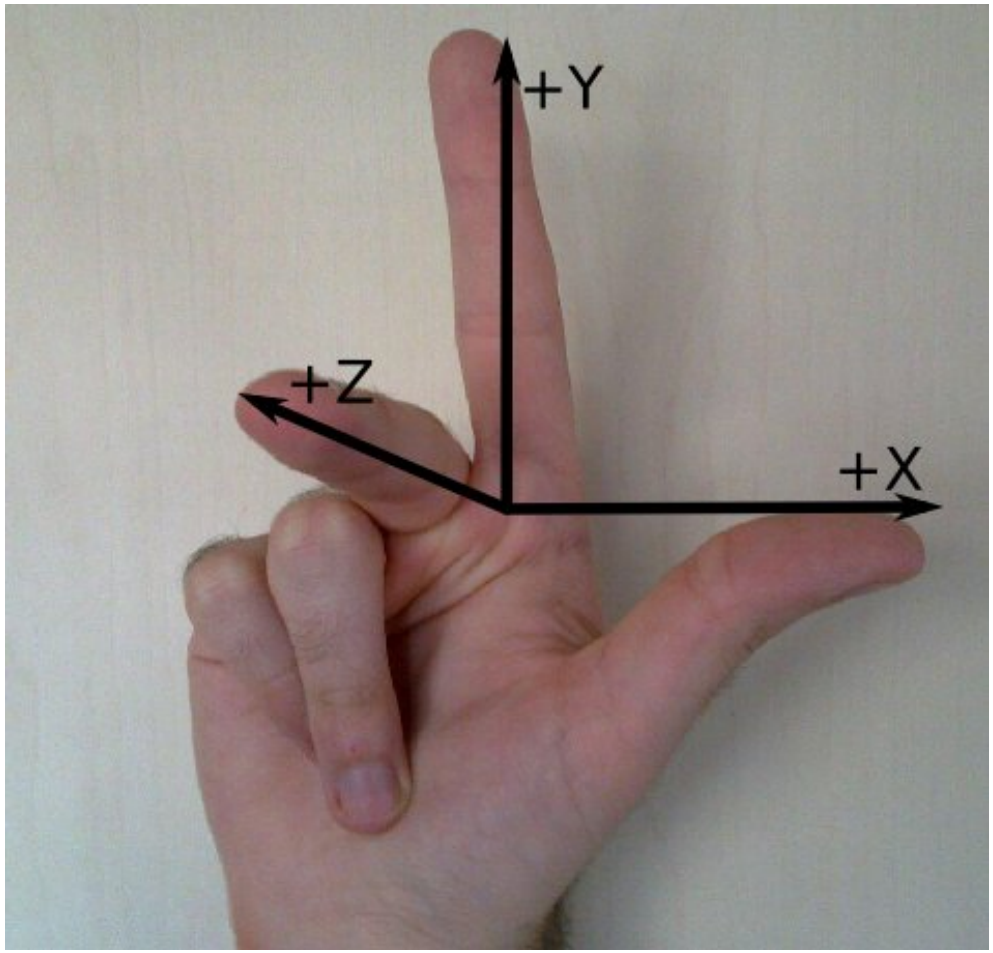
Analogies avec la photographie



1	Arranger les éléments de la scène à capturer	Composer une scène virtuelle	Transformation de modélisation
2	Positionner l'appareil photo	Positionner la caméra virtuelle	Transformation de vision
3	Régler la focale de l'appareil photo	Configurer une projection	Transformation de projection
4	Choisir la taille des tirages photographiques	Choisir les dimensions de l'image video	Transformation de cadrage

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Repère direct



UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

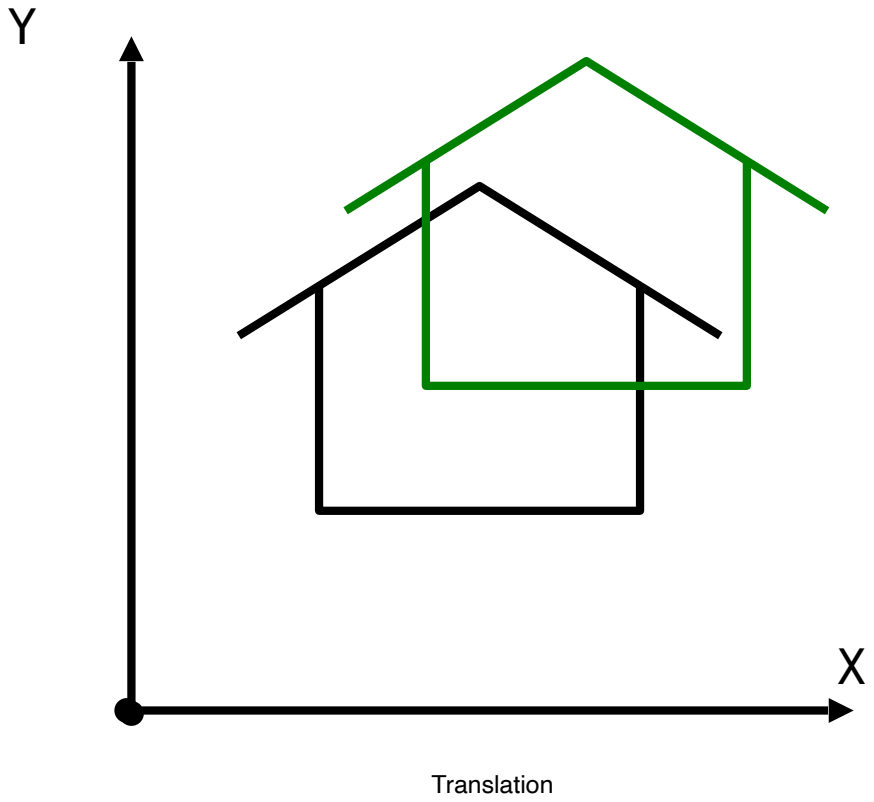
Coordonnées homogènes

- Un point P (x, y, z) dans l'espace euclidien a pour coordonnées homogènes : P (x, y, z, 1)
- Un vecteur v (x, y, z) dans l'espace euclidien a pour coordonnées homogènes : v (x, y, z, 0)
- Les matrices de rotation, de translation, d'homothétie et de changement de repère sont des matrices de taille 4×4 dont la dernière ligne est toujours (0 0 0 1) car ces transformations peuvent être considérées comme des transformations projectives laissant globalement invariant le plan à l'infini.

Translation

Vecteur de translation : $t = \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix}$

Matrice : $T = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$



gl4dmMatrixTranslate (float x, float y, float z)

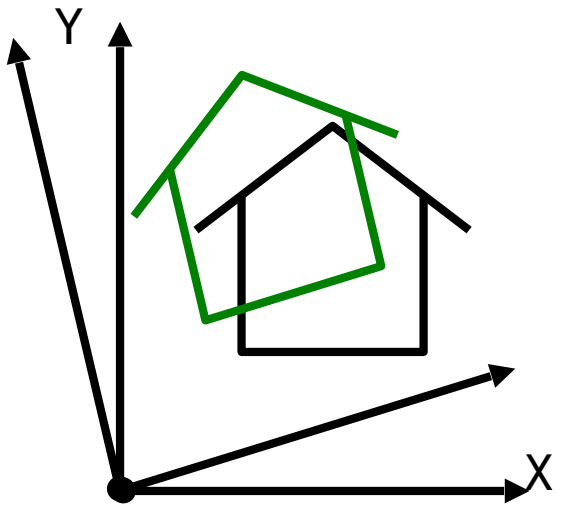
UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Rotation

Matrice de rotation R d'angle θ autour d'un axe de direction r :

$$r = \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix}, c = cos\theta, s = sin\theta, R = \begin{pmatrix} x^2(1-c)+c & xy(1-c)-zs & xz(1-c)+ys & 0 \\ xy(1-c)+zs & y^2(1-c)+c & yz(1-c)-xs & 0 \\ xz(1-c)-ys & yz(1-c)+xs & z^2(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

gl4dmMatrixRotate (float angle, float x, float y, float z)

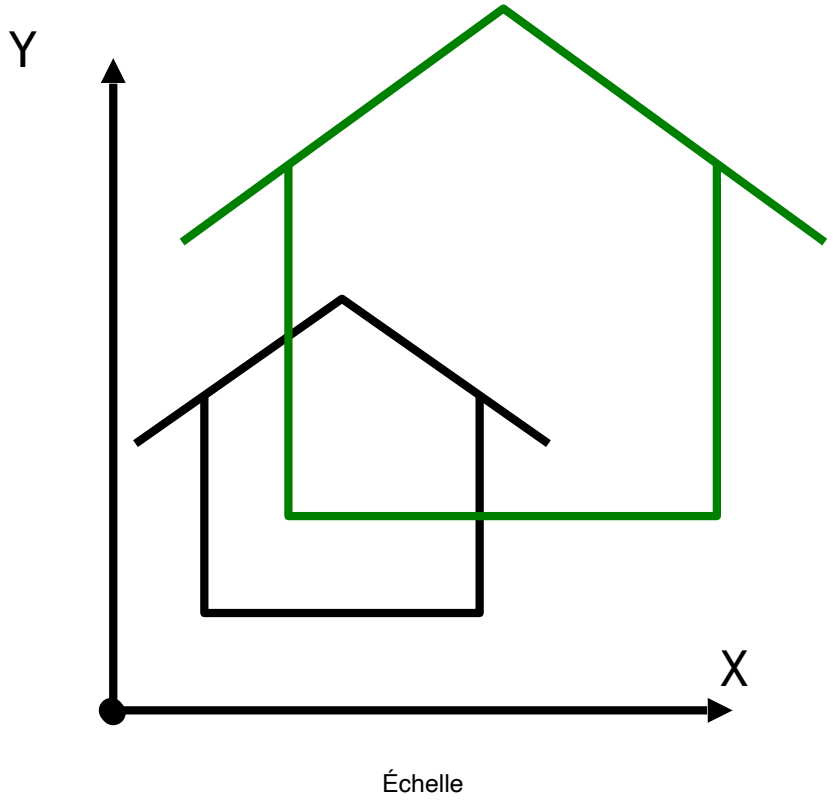


Rotation

Homothéthie

Facteur d ‘échelle $s = \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix}$

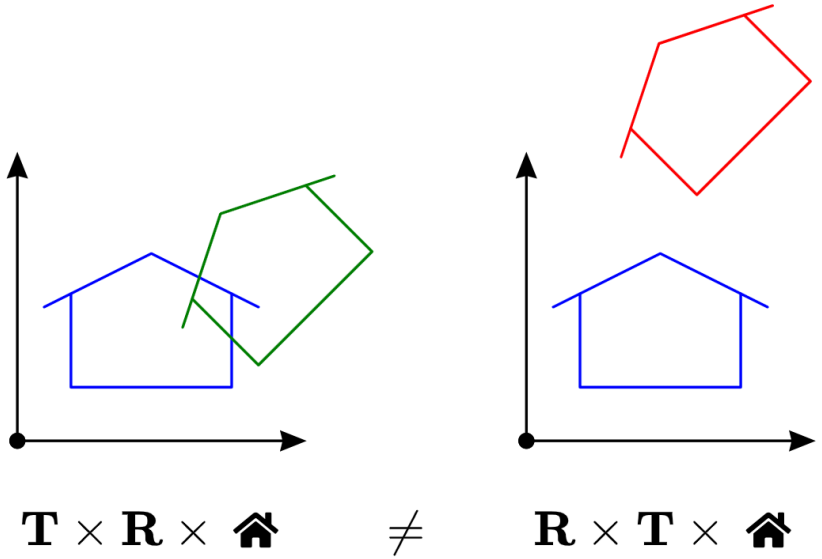
Matrice $S = \begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$



gl4dmMatrixScale (float x, float y, float z)

Transformation de modélisation

☠ La multiplication n'est pas commutative !



🫖 Convention : T × R × S × 🏠

1. gl4dmMatrixTranslate (T.x, T.y, T.z);
2. gl4dmMatrixRotate (a, R.x, R.y, R.z);
3. gl4dmMatrixScale (S.x, S.y, S.z);

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Transformation de vision

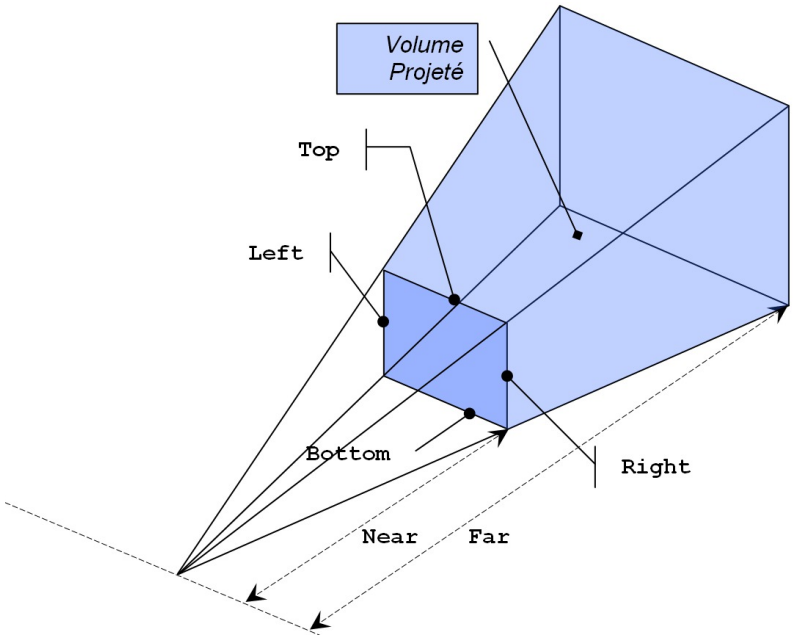
- Position relative de la caméra par rapport aux objets :
  translater la caméra de t équivaut à translater tous les objets de –t

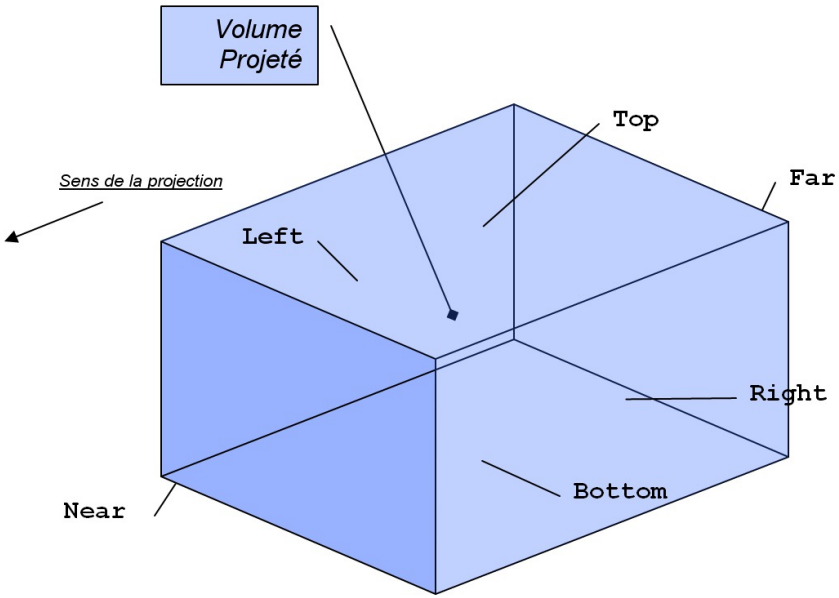
- glViewport (0, 0,w, h) : Surface utilisée lors du dessin, correspond aux coordonnées dans le repère de la zone de dessin.

- gl4dmMatrixLookAt (px,py,pz, cx,cy,cz, ux,uy,uz)
  Positionne la caméra au point [px py pz],
  oriente les Z vers la cible [cx cy cz]
  et les Y selon le vecteur [ux uy uz].

Transformation de projection

- **`gl4dmMatrixPerspective (fovy, aspect, near, far)`**
  Définit le volume de projection perspective.
- **`gl4dmMatrixOrtho(left,right, bottom,top, near,far)`**
  Définit le volume de projection orthogonale.





UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Projections - suite

- **`gl4duPerspectivef (fovy, aspect, near, far)`**
  Création d'une matrice de projection perspective selon l'ancienne fonction gluPerspective et la multiplie dans la matrice en cours.
- **`gl4duOrthof(left,right, bottom,top, near,far)`**
  Création d'une matrice de projection orthogonale selon l'ancienne fonction glOrtho et la multiplie dans la matrice en cours.
- **`gl4duFrustrumf(left,right, bottom,top, near,far)`**
  Création d'une matrice de projection perspective selon l'ancienne fonction glFrustum et la multiplie dans la matrice en cours

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

GL4D - Manipulation des matrices

- **`gl4duGenMatrix(type, name)`**
  Génère une matrice 4x4 liée au nom *name* et de type *type*. *Le type peut être GL_FLOAT ou GL_DOUBLE* .
- **`Gl4duBindMatrix(name)`**
  Active la matrice liée au nom passé en argument.
- **`gl4duLookAtf(eyeX, eyeY, eyeZ, centerX, centerY,centerZ, upX, upY, upZ)`**
  Définit des transformations pour simuler un point de vue avec direction de regard et orientation.
- **`gl4duLoadIdentityf()`**
  Chargement de la matrice identité dans la matrice actuellement active.
- **`glPushMatrix()`**
  Sauvegarde/empile la matrice courante et utilise une nouvelle matrice dont le contenu est le meme que celle empilée.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Objets-géométrie GL4D

- **`gl4dgQuadf ()`**
  Génère un objet-géométrie de type Quad (plan vertical en z=0) et renvoie son identifiant (référence). La géométrie est décrite par 4 sommets reliés par un triangle_strip.
- **`gl4dgSpheref(slices,stacks)`**
  Génère un objet-géométrie de type sphère et renvoie son identifiant (référence).Cette sphère est obtenue par transformation des coordonnées polaires en coordonnées cartésiennes. slices donne le nombre de longitudes de la sphère et stacks le nombre de latitudes de la sphère.
- **`gl4dgCubef()`**
  Génère un objet-géométrie de type cube et renvoie son identifiant (référence). Ce cube est composé de 6 plans meshés par des triangle_strips. Le normales sont aux plans.
- **`gl4dgTorusf(slices, stacks, radius)`**
  Génère un objet-géométrie de type tore et renvoie son identifiant (référence). slices donne le nombre de longitudes du tore, stacks le nombre de « latitudes » (coupes verticales et radius le rayon d'une section verticale du tore.
- Il y en a d'autres !
- **`gl4dgDraw(id)`**
  Dessine un objet-géométrie dont l'identifiant est passé en argumente.
- **`gl4dgDelete(id)`**
  Détruit un objet-géométrie dont l'identifiant (référence) est passé en argument.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Tableaux de sommets

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Vertex buffers objects (VBOs)

- Tableaux d'octets dans la mémoire graphique permettant de stocker des sommets
- Sources potentielles d'attributs de sommets : position, normale, couleur…
- Structures et types connus du seul développeur
- OpenGL est basé sur le concept un peu particulier de ***binding***. Pour modifier le VBO créé (le remplir avec des données) il faut le ***binder*** sur une cible qui sera ensuite spécifiée lors des opérations de modification.
- Il existe plusieurs cibles de *binding* pour les *buffers* OpenGL (GL_ARRAY_BUFFER, GL_ELEMENT_ARRAY_BUFFER, GL_UNIFORM_BUFFER…). Chaque cible est destinée à un usage particulier, pour les VBOs —> **GL_ARRAY_BUFFER**.

```
1.  // Création du VBO.Chaque VBO sera identifié par un entier strictement
2.  //supérieur à 0 qui nous sera renvoyé par OpenGL.
3.  glGenBuffers(1, &_buffer);
4.  // Activation du VBO - binding sur la cible GL_ARRAY_BUFFER
5.  glBindBuffer(GL_ARRAY_BUFFER, _buffer);
6.  // Initialisation des données
7.  GLfloat data[] = { -0.97f, -0.97f, 0.97f,… };
8.  // envoi des données au vbo via la cible GL_ARRAY_BUFFER
8.  glBufferData(GL_ARRAY_BUFFER,sizeof data,data,GL_STATIC_DRAW);
9.  // Désactivation du VBO une fois les opérations effectuées pour ne plus le modifier
     par la suite
10. glBindBuffer(GL_ARRAY_BUFFER, 0);
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Attributs des sommets

 `glEnableVertexAttribArray (index);`

Activation d'un attribut identifié par son indice

 `glVertexAttribPointer (index, size, type, normalized, stride, pointer);`

index	Indice de l'attribut
size	Nombre de composantes par attribut (1, 2, 3 ou 4)
type	GL_FLOAT / GL_UNSIGNED_BYTE / …
normalized	Normalisation des valeurs ? true / false
stride	Distance en octets entre deux attributs consécutifs
pointer	Pointeur sur le premier élément de l'attribut dans le VBO

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Attributs des sommets


```
1.  // Activation du VBO.
2.  glBindBuffer(GL_ARRAY_BUFFER, _buffer);
3.  // Activation de l'attribut n° 0. Chaque attribut
    (position, normale, couleur,…) est identifié par un entier.
    Par défaut, 0 pour l'attribut position.
4.  glEnableVertexAttribArray (0);
5.  //Déclaration de la structure du tableau.
6.  glVertexAttribPointer (0, 3, GL_FLOAT, GL_FALSE, 16, 0);
7.  // Activation de l'attribut n°1
8.  glEnableVertexAttribArray (1);
9.  //Déclaration de la structure du tableau.
10. glVertexAttribPointer (1, 3, GL_FLOAT, GL_TRUE, 16, 12);
11. // Désactivation du VBO
12. glBindBuffer(GL_ARRAY_BUFFER, 0);
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Vertex array objects (VAOs)

- Un *vertex array object peut* stocker plusieurs VBOs.
- Permet stocker les données des sommets et des couleurs dans des VBOs différents, mais dans le même VAO.
- Même principe pour tous les types de données généralement transmis comme VBO, dont les données des normales ou n'importe quelle donnée requise au niveau des sommets.
- Un VAO est une manière de stocker des informations sur les objets dans la carte graphique, au lieu de lui envoyer des sommets au fur et à mesure des besoins.
- Le VBO contient les données tandis que le VAO les décrit.

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Vertex array objects (VAOs)

Création et activation du VAO :

```
1.  // Création du VAO.
2.  glGenVertexArrays (1, &van);
3.  // Activation du VAO.
4.  glBindVertexArray(_vao);
    …Activation du VBO, remplissage, définition des attributs…
15. //Désactivation du VAO.
16. glBindVertexArray(0);
```

Au moment du dessin :

```
1. // Activation du VAO.
2. glBindVertexArray(_vao);
3. // Dessin.
4. glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
5. // Désactivation du VAO
6. glBindVertexArray(0);
7. // Désactivation du VBO
8. glBindBuffer(GL_ARRAY_BUFFER, 0);
```

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Primitives géométriques

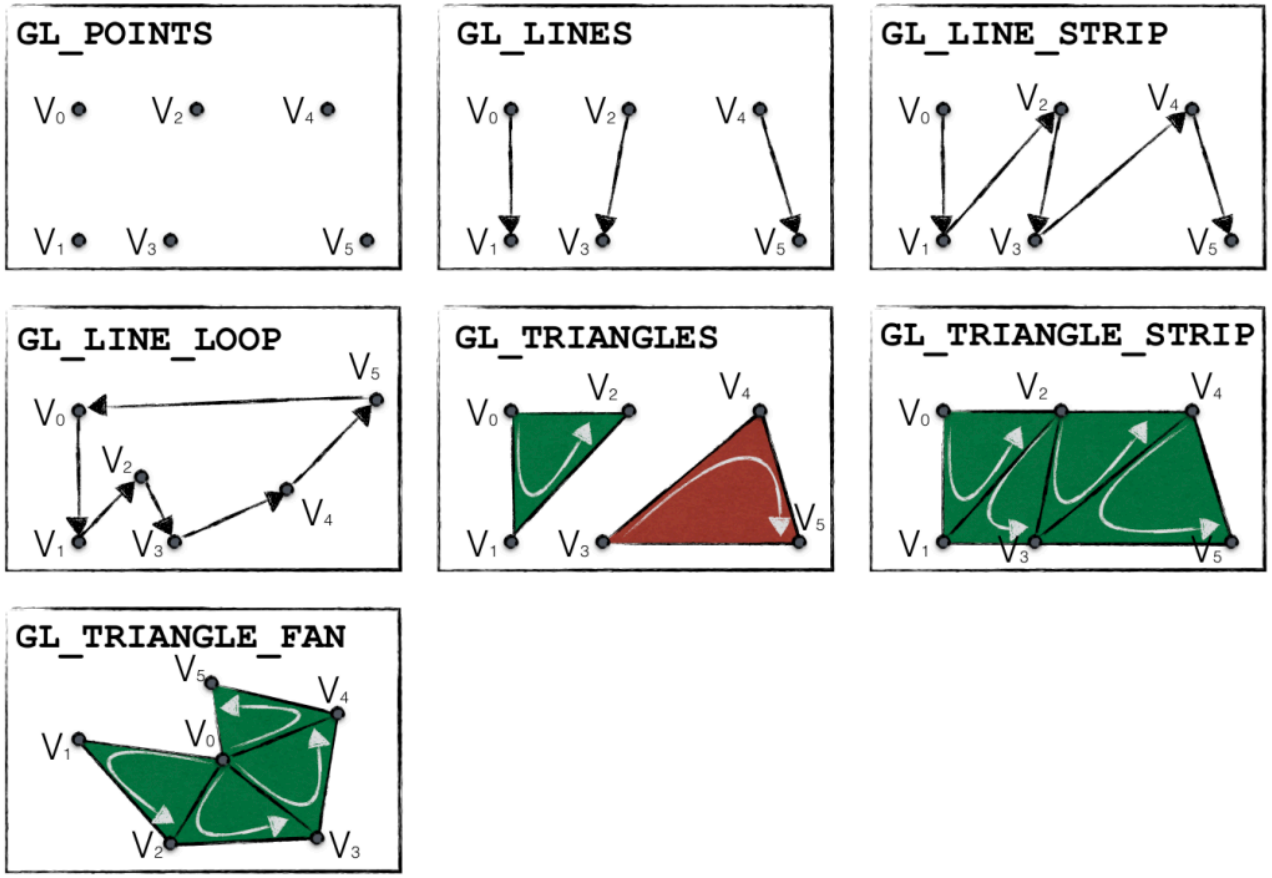


FIGURE 4.2 – (Seules) Primitives gérées par OpenGL 3.1 et plus

Crédits : Farès Belhadj

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Mode direct

```
glDrawArrays (GLenum mode, GLint first,
GLsizei count)
```

mode	Type de primitives à interpréter
first	Indice du premier sommet
count	Nombre de sommets

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Mode indirect

- GL_**TRIANGLE_STRIP** et GL_**TRIANGLE_FAN** pas toujours adaptés : Nombreux sommets identiques dupliqués au sein du VBO

- Deux sommets sont identiques si tous leurs attributs sont identiques
- Ajout d'un second *buffer* pour accéder aux sommets de façon indirecte