# Programmation C : Images

Trois étapes en image : Traitement + Analyse + Interprétation.

**Traitement** : opérations de manipulation de l’image pour améliorer la qualité.

* Compression : réduction volume de l’image
* Restauration : correction des défauts dus à une source de dégradation.
* L’amélioration : modification de l’image dans le but de la rendre plus agréable à l'œil.

**Analyse** : suite d'opérations pour l'extraction d'information contenue dans une image.

* Phase de segmentation avec partition de l'image.
* Techniques de description / modélisation pour obtenir la description structurelle de l'image.

**Interprétation** : passage de la description structurelle à la description sémantique en regard à certains objectifs. (mesure de paramètres sur des formes, description du contenu de la scène en termes de concepts non mathématiques).

**Que peut-on faire d’une image :**

Traitement corrigeant des défauts de l'image, permettant un confort de visualisation.

* Augmentation de contraste,
* Correction des distorsions optiques,
* Filtrage du bruit
* image binaire (p,M) = (1,1)
* image en niveaux de gris p = 1 et M = 255
* image couleur p = 3 et M = 255

**Maillage** : arrangement géométrique des pixels dans l'image. 3 types de tessellations du plan par des figures géométriques.

* Maillage carré : réalité physique du capteur CCD.
* Maillage hexagonal (référence en morphologie mathématique).
* Maillage triangulaire.

**Distance** : entre deux pixels P(xp,yp) et Q(xq,yq)

* distance de Manathan : d1 (P,Q)=|xp - xq| + |yp - yq|
* distance euclidienne : d2 (P,Q)=[(xp - xq)² + (yp - yq)²]1/2
* distance de l'échiquier : dinf(P,Q)=Max(|xp - xq| , |yp - yq|)

| **Définition - Trichromie** : couleur perçue par un humain décomposée dans un espace à 3 dimensions 3 couleurs de base avec un spectre éloigné (RVB). |
| --- |

**Trivariance** : couleur Cx fonction de l et des luminances de Cl et de Cb.

* **Cx** : lumière colorée quelconque
* **Cl** : lumière monochromatique
* **Cb** : lumière blanche

| **Cx = Cl + Cb** |
| --- |

| **Définition** - Tessellation : Une tessellation d'une surface plane est le carrelage d'une surface en utilisant une ou plusieurs formes géométriques, appelées carreaux, sans chevauchement et sans lacunes. |
| --- |

| **Synthèse soustractive :**  Jaune + Magenta => Rouge  Jaune + Cyan => Vert  Magenta + Cyan => Bleu  Jaune + Magenta + Cyan => Noir | **Synthèse additive :**  presque toutes les couleurs visibles.  Rouge +Vert => Jaune  Rouge + Bleu => Magenta  Bleu + Vert => Cyan  Rouge +Vert +Bleu => Blanc |
| --- | --- |

**3 Modèles d’espaces de représentation de la couleur :**

Espaces basés sur la **chrominance** :

* RVB (RGB): Red Green Blue
* CMJN (CMYK): Cyan Magenta Yellow black (K pour Key black)
* XYZ

Espaces basés sur la luminance et la chrominance :

* Lab: **luminance** + **chrominance** (a et b)
* YUV et YCrCb

Espaces basés sur la **luminance**, la **chrominance** et la **saturation** :

* TSL (HLS) :Teinte (Hue), Saturation, et Luminosité.

Utilisés en compression d’images et vidéos car décorrélation de l’information.

**Luminance** : Y = 0.299 R + 0.587 G + 0.114 B

| **RGB to YUV :**   * Y = … * U = 0.492 (B – Y) + 128 * V = 0.877 (R – Y) + 128 | **YUV to RGB :**   * R = ((Cr - 128)/0.877)+Y / 0.587 * G = (Y - 0.299\*R - 0.114\*B) * B = ((Cb - 128)/0.877)+Y |
| --- | --- |
| **RGB to YCrCb :**   * Y = 0.299 R + 0.587 G + 0,114 B * Cb = - 0.1687 R - 0,3313 G + 0.5 B + 128 * Cr = 0.5 R – 0,4187 G – 0,0813 B + 128 | **YCrCb to RGB :**   * R = Y + 1.402 (Cr – 128) * G = Y - 0.34414 (Cb – 128) – 0.714414 (Cr – 128) * B = Y + 1.772 (Cb – 128) |

# Programmation 3D

### Introduction :

|  |
| --- |

**Attributs aux sommets :**

* Propriétés associées aux sommets (le plus souvent), arêtes ou faces → Attributs

**Attributs de sommets :**

* Position (« p »)
* Vecteur normal (« n »)
* Coordonnées paramétrique (« (u,v) »)
* Apparence : couleur, indice de matériel …
* Paramètres physiques pour la simulation …

| **Définition** : Rendu = Synthèse d’images, génération d’une image numérique à partir d’une scène 3D. Réaliste ou expressif. |
| --- |

Rendu en temps-réel :

* Pour les systèmes interactifs
* Approximation géométrique et radiométrique de la scène
* Calcul parallèle (GPU)

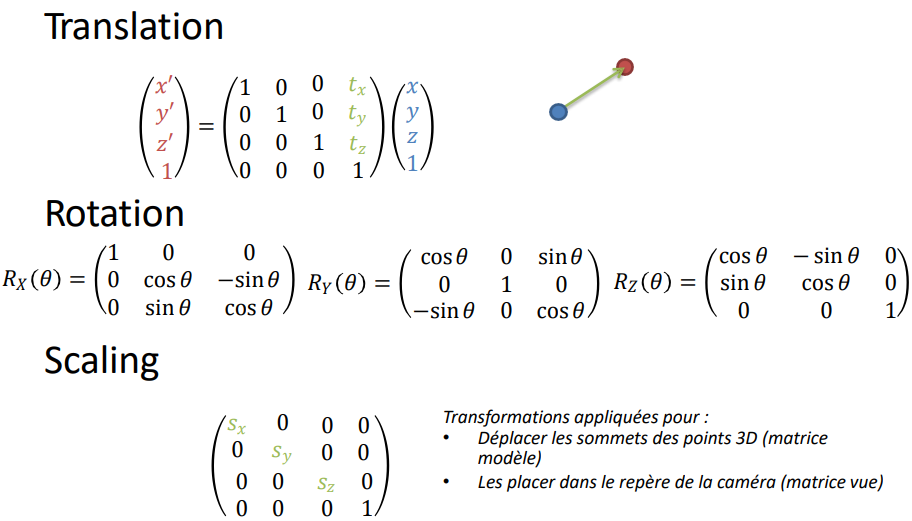
**Couleur en image de synthèse :**

Précision variable

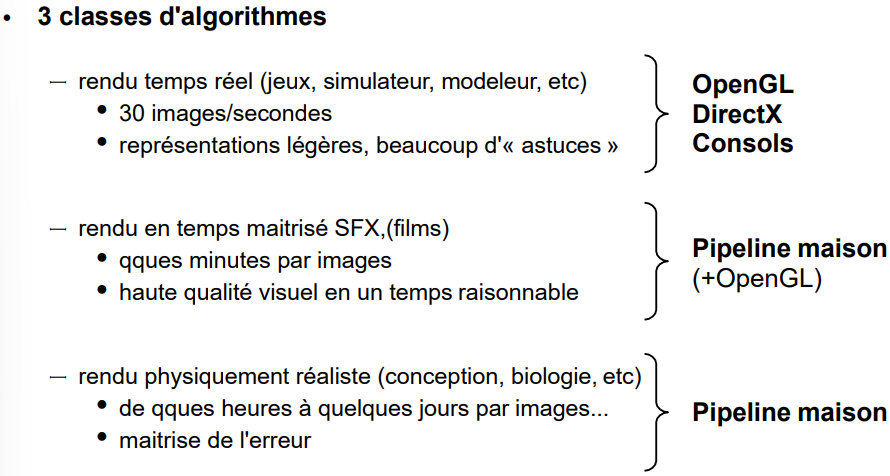
* 8bits/composante : standard des APIs, des GPU, du rendu et des écrans
* 16/32bits par composante : imagerie à haute dynamique (HDR)
* Nécessite une conversion avant affichage (Tone Mapping)

**RGBA** : transparence alpha par pixel

* Pose un problème d’ordonnancement
* Pas directement compatible avec les algorithmes de rendu par projection (Z-Buffer)



**Synthèse d’images :**



Synthèse d'images ↔ Simulation des interactions entre la lumière, la matière, les formes, et le milieu ambiant.

**Visualiser une scène 3D : deux approches**

* *1ère approche* : lancer de rayons (Ray-tracing – Ray casting) : Approche suivie pour le réalisme
* Pour chaque pixel, lancer un rayon
* Trouver la primitive la plus proche intersectant le rayon
* *2ème approche* : rastérisation : Approche suivie pour le temps réel.
* Pour chaque triangle, projeter les 3 sommets sur le plan image → triangle 2D
* Trouver les pixels correspondant

**Trois idées sur les rayons lumineux**

1. La lumière se déplace en ligne droite (principalement)
2. Les rayons lumineux n'interfèrent pas les uns avec les autres s'ils se croisent (les photons n'ont pas d'interactions; ignorant les effets des vagues comme les interférences)
3. Les rayons lumineux se déplacent des sources lumineuses à l'œil (mais la physique est invariante sous inversion de chemin - réciprocité).

### Formes et maillages 3D :

| **Définition** : **Modélisation** : Décrire un objet ou un phénomène par un ensemble de nombres et une structure entre ces nombres. |
| --- |

| **Définition** : **Modélisation géométrique** : Décrire la forme d’un objet par un ensemble de valeurs. |
| --- |

**Il existe beaucoup de représentations :**

*Explicites*

* Nuages de points
* Maillages polygonaux
* Surface de subdivision, NURBS,
* Grilles de voxels,…

*Implicites*

* Ensembles de niveaux (level set)
* Surface algébrique
* Fonctions de distance,…

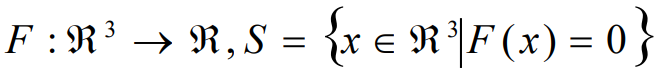
**Surface paramétrique :**

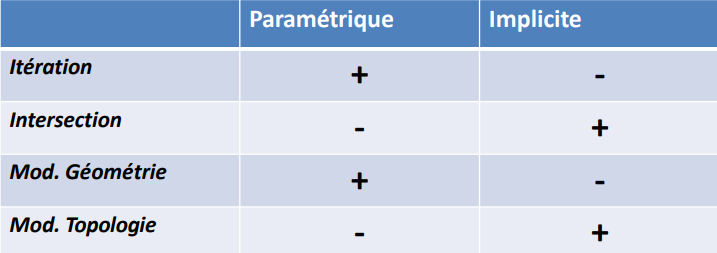
Définit par une fonction de paramétrisation d’un domaine du plan vers l’espace 3D :



**Surface implicite :**

Iso-surface 0 d’une fonction F :





**Représentation des surfaces :**

• *Surfaces discrètes*

* Maillages Polygonaux
* Surfaces de Points

• *Surfaces continues*

* Surfaces Spline
* Surfaces de Subdivision
* Surfaces Implicites

| **Définition** : Soupe de polygones : Suites de n-uplets de coordonnées 3D correspondants aux polygones. |
| --- |

| **Définition** : Maillage indexés : Graphe avec géométrie et topologie séparés :   * Une liste de sommets (V) : * Une liste de relation topologique : Arêtes et Faces |
| --- |

Pour la modélisation, les équations géométriques des surfaces ne sont pas forcément disponibles, on utilise des scans.

**Dualité : Propriétés :**

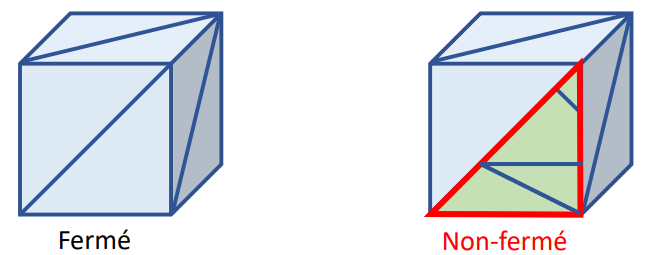
Maillage dual :

* chaque face est remplacée par un sommet → barycentre de la face,
* une arête du dual relie deux sommets si les faces correspondantes sont voisines- dans le maillage d’origine,
* les points sont remplacés par des faces

Les objets de dimension k du maillage original sont remplacés par des objets de dimension (2-k) dans le dual.

**Remarque** : Le maillage dual d’un maillage dual est égal au maillage original si celui-ci est fermé.

| **Définition** : Maillage fermé : Un maillage est dit fermé si il n’a pas de bords et que toutes les arêtes du maillage sont au moins partagées par deux triangles. |
| --- |



**2-Variétés** ou (manifold) : Un maillage est 2-Variété si :

* une sphère (rayon > 0) placée en n’importe quel point à une intersection avec le maillage correspondante à une unique surface,
* il ne contient que des arêtes partagées par au plus deux triangles,
* il ne contient aucun sommet correspondant à au plus 2 arêtes du bord,
* il ne contient pas d’auto-intersection.

**Normales** :

On peut définir une normale par face :

* elle permet de définir l’orientation de la face
* elle est égale au produit vectoriel des deux premières arêtes
* l‘ordre des sommets dans une face est donc important
* elle est utilisée pour définir l’extérieur ou l’intérieur ou pour l’éclairage à l’affichage.



On peut définir une normale par sommet à partir des normales aux faces.

Normales au sommet = moyenne des normales des faces contenant le sommet.

**Structures de données : Pour stocker …**

* Les entités : sommets, arêtes, faces
* Les normales (par sommet ou face)
* Les couleurs (par sommet ou face)
* …

Pour stocker un maillage il faut choisir entre :

* minimiser la taille mémoire,
* répéter le moins possible les coordonnées des points, …
* faciliter le parcours dans le maillage,
* pour passer d’un sommet à l’autre, …
* permettre d’extraire les informations de topologie.
* pour connaître les sommets liés à un autre sommet , les arêtes liées à un sommet

**Approche classique :**

* maillage représenté par un ensemble de tableaux : un pour les sommets, un pour les faces, un pour les couleurs … → maillage indexé,
* les coordonnées des sommets ne sont plus répétées

**Création d’un maillage** : À partir d’un nuage de points en utilisant une triangulation. A partir d’une surface continue en utilisant une discrétisation.

**Discrétisation :**

* A partir d’objet simple, exemple du cas du cylindre : des méridiens sont extraits, à partir des méridiens on calcule des facettes,
* Chaque facette est découpée en triangle en ajoutant une diagonale.

Surfaces libres en utilisant les iso-paramétriques :

* courbes de contrôle en U et V,
* construction de facettes qui sont ensuite triangulées.

**Remarque** : La résolution du maillage est induite par le nombre d’iso-paramétriques calculées en U ou en V : le nombre d’iso-paramétriques en U ou en V, peut être différent en fonction de l’objet.

**Connectivité :**

1-anneau voisinage (1- voisinage) d’un sommet v : ensemble des sommets reliés par une arête à v

* Valence d’un sommet : taille de 1-voisinage
* Maillage régulier :

Tous les sommets ont une valence régulière, exemple :

* valence 6 pour les maillages triangulaires
* valence 4 pour les maillages quadrangulaires

Maillage semi-régulier : il est semi-régulier si la majeure partie de ses sommets sont de valence 6, seulement quelques sommets sont de valence ≠ 6.

* La plupart des sommets ont une valence régulière - Peu de sommets extraordinaires (valence irrégulière)

Maillage arbitraire :

* La plupart des sommets sont extraordinaires

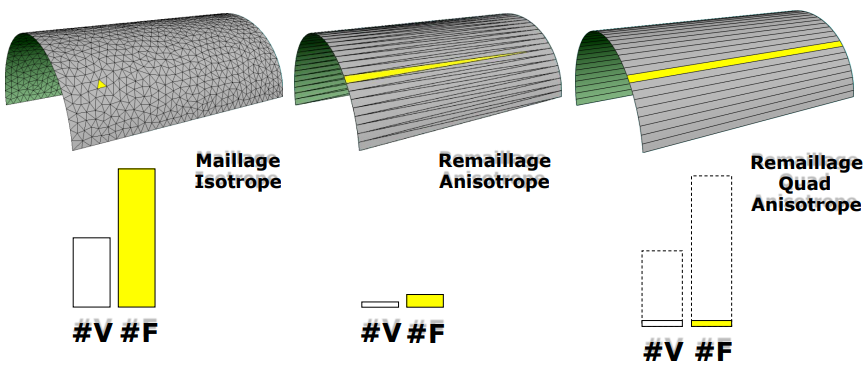
Maillage régulier : Il est régulier si tous ses sommets internes sont de valence 6, tous les sommets du bord sont de valence 4.

Lors d’une discrétisation il y a deux façons de rajouter les diagonales :

* toutes les diagonales dans le même sens (valence 6) ,
* un coup d’un côté, un coup de l’autres (valence 4/8)

| **Définition** : Isotropie : Les polygones ont une forme similaire sur tout le maillage. |
| --- |

| **Définition** : Anisotropie : La forme des polygones suit la géométrie de la surface, arêtes alignées sur les directions de courbures principales. Lignes de flots, distances géodésiques. |
| --- |



**Conclusion :**

*Représentation par maillage :*

* un ensemble de sommets, un d’arêtes et un de faces,
* plus les autres propriétés : normales, couleurs…

*Plusieurs représentations possibles :*

* les arêtes ou les faces ne sont pas forcément stockées de manière explicite,
* selon la représentation les liaisons : sommets/faces, sommets/sommets, arêtes/faces … ne sont pas toujours les mêmes,
* il faut choisir entre taille en mémoire, parcours dans le maillage et extraction de la topologie.

### Rendus et éclairages :

Nombre de directions de référence = dimension de l’espace.

Soit le vecteur 𝑢 entre A et B :

* définit une direction dans l’espace
* est définit par 3 coordonnées

Permet de translater des objets : A = (3, 5,2) → B = (6, 4,4) 𝑢

𝑢𝑥 = 𝐵𝑥 − 𝐴𝑥

𝑢𝑦 = 𝐵𝑦 − 𝐴𝑦

𝑢𝑧 = 𝐵𝑧 − 𝐴𝑧

| Addition de vecteurs : | Multiplication avec un scalaire 𝜆 : | Norme du vecteur : |
| --- | --- | --- |
|  |  |  |
| Produit scalaire (et géo) : | Produit vectoriel entre deux vecteurs (et géo) : | Projection d’un point sur une droite : |
|  |  |  |

| Projection d’un point sur un plan : |
| --- |
|  |

**Shading :**

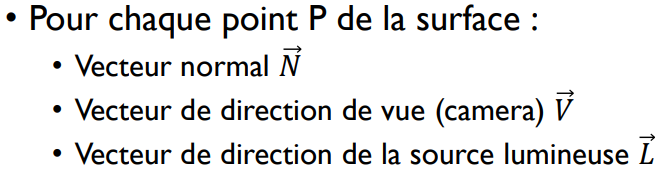
| **Définition** : Shading : Calculer la lumière réfléchie sur la caméra. |
| --- |

**Entrée** :

* Direction de vue
* Direction de lumière
* Normal à la surface
* Paramètre de la surface (couleur, rugosité).

**Lumières :**

En OpenGL : Phong = ambiante + diffuse + spéculaire.



| **Réflexion ambiante :**  Parasites provenant d'autre chose que la source considérée   * lumière réfléchie par d'autres points * supposée égale en tout point de l'espace   **𝐼𝑎 = 𝐼𝑠𝑎 ∗ 𝐾𝑎**  • 𝐼𝑎 : intensité de la lumière ambiante réfléchie  • 𝐼𝑠𝑎 : intensité de la lumière ambiante  • 𝐾𝑎 ∈ [0,1] : coeff. de réflexion ambiante de l’objet  Remarque : Couleur ambiante d’un objet ne dépend que du coefficient de réflexion ambiante Ka de l’objet, pas de sa position par rapport à la lumière |
| --- |

| **Réflexion diffuse :** Loi de Lambert  **𝐼𝑑 = 𝐼𝑠𝑑 ∗ 𝐾𝑑 ∗ cos 𝜃**  • 𝐼𝑑 : intensité de la lumière diffuse réfléchie  • 𝐼𝑠𝑑 : intensité de la lumière diffuse  • 𝐾𝑑 ∈ [0,1] : coeff. de réflexion diffuse du matériau  • 𝜃 : angle entre la source de lumière et la normale  • On peut également écrire :  **𝐼𝑑 = 𝐼𝑠𝑑 ∗ 𝐾𝑑 ∗ (𝐿. 𝑁)** avec L et N des vecteurs |
| --- |

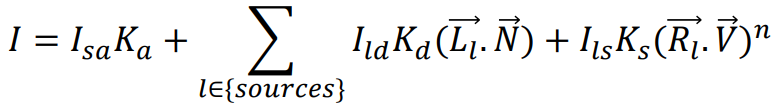
| **Réflexion spéculaire :**  • Modèle de Phong  **𝐼𝑠 = 𝐼𝑠𝑠 ∗ 𝐾𝑠 ∗ cos 𝛼𝑏**  • 𝐼𝑠 : intensité de la lumière spéculaire réfléchie  • 𝐼𝑠𝑠 : intensité de la lumière spéculaire de la source  • 𝐾𝑠 ∈ [0,1] : coeff. de réflexion spéculaire du matériau  • α : angle entre les directions de réflexion et de la vue  • 𝑏 : brillance ou shininess  On peut également écrire  **𝐼𝑠 = 𝐼𝑠𝑠 ∗ 𝐾𝑠 ∗ 𝑅. 𝑉𝑛 (avec R et V des vecteurs)**  avec  𝑅 : direction de réflexion de la lumière sur un miroir et  **𝑅 = 2 𝑁. 𝐿 𝑁 − 𝐿 = 2 cos 𝜃 𝑁 - *L* (avec N et L des vecteurs)** |
| --- |

**Modèle de Phong :**

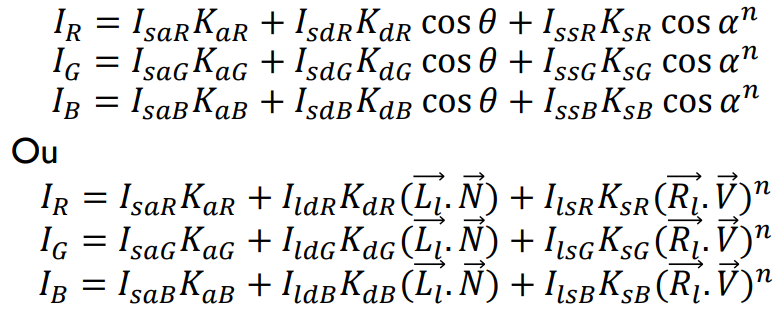
Dans la réalité, lumière réfléchie par une surface = réflexion diffuse + réflexion spéculaire

| **Equation de Phong : 𝐼 = 𝐼𝑎 + 𝐼𝑑 + 𝐼*s*** |
| --- |

**Plusieurs sources lumineuses : Somme des intensités :**



**Calcul de la couleur :**



Modèle de phong très pratique, rapide à calculer, mais pas de lien avec les propriétés du matériau.

**Illumination d’un objet 3D :**

But : calculer une couleur pour chaque point visible à l’écran de l’objet 3D qu’on affiche.

* Lissage de Gouraud : interpolation de couleurs
* Lissage de Phong : interpolation de normales

Le lissage de Phong est plus lent que celui de Gouraud, mais nettement plus beau. Permet de calculer les effets spéculaires contenus dans une facette, contrairement au lissage de Gouraud.

**Lumières** : Plusieurs types d’éclairages :

* Source directionnelle
* Source ponctuelle
* Source projecteur

| Lumière directionnelle | * Définie par une direction : * 4 coordonnées homogènes (x, y, z, 0) * La lumière vient d’une direction spécifique. |
| --- | --- |
| Lumière ponctuelle | * Définie par une position : * 4 coordonnées homogènes (x, y, z, 1) * La lumière vient d’un point spécifique. |
| Lumière projecteur | * La lumière vient d’un point spécifique. * Intensité dépendant de la direction * Position : emplacement de la source * Direction : axe central de la lumière * Angle : largeur du rayon |

**OpenGL** :

GPU = Processeur Graphique

* permet des calculs complexes réalisés par la carte graphique

Le CPU est libre pour réaliser d'autre tâches, ce n'est pas un CPU !

* Hautement parallèle : jusqu'à 4000 opérations en parallèle !

Architecture hybride (SIMD/MIMD)

* Accès mémoire via des buffers spécialisés (de – en - vrai):

**Textures (images, tableaux en lecture, 1D, 2D, 3D)**

**Vertex Buffers (tableaux de sommets, 1D)**

**Frame buffers (images en écriture, 2D)**

* Circuits spécialisés non programmable

Rasterisation, blending, etc. Accès via une API graphique

* OpenGL, DirectX, Vulkan, etc.

**Plusieurs bibliothèques existent:**

* GLFW (fortement recommendé)
* FreeGlut
* Qt

**Fournissent en général un mécanisme de callback pour**

* la mise à jour de l’image affichée
* les évènements claviers
* les événements souris

Versions : OpenGL 3.x « core » → recommandé

**Pipeline : in/out :**

**En entrée**

* Une description numérique de la géométrie de la scène = {primitives rastérisables}

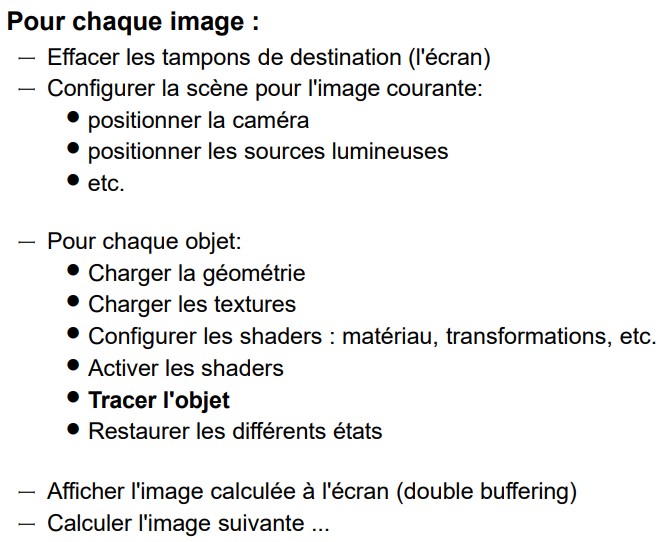
ex. : ensemble de polygones

* Ensemble de paramètres :
* un point de vue (caméra)
* des attributs de matériaux associés à chaque objet
* un ensemble de lumières

**En sortie**

– une image = un tableau de pixels (couleurs RGB)

Algorithme de rendu par rastérisation :



| **Définition** - Rastérisation : Nom commun. (Imprimerie, Infographie) Procédé qui consiste à convertir une image vectorielle en une image matricielle destinée à être affichée sur un écran ou imprimée par un matériel d'impression. |
| --- |

**OpenGL Shaders :**

Langage : GLSL (OpenGL Shading Language)

**En lecture** :

• Vertex Buffer Objects – VBO (tableau de sommets avec attributs)

* accès indirect

• Textures (tableaux 1D, 2D, 3D, etc.)

**En écriture** :

• Frame Buffer Objects – FBO (tableaux 2D)

**Compilation à la volée par le driver OpenGL**

* Shader == char\*
* Source code spécifié via des fonctions OpenGL

**Types : vecteurs et matrices**

**Vecteurs:**

* float, vec2, vec3, vec4
* int, ivec2, ivec3, ivec4
* bool, bvec2, bvec3, bec4

**Matrices:**

* mat2, mat3, mat4
* matrice carrée de réels (en colonne d'abord)
* utilisées pour les transformations
* mat2x2, mat2x3, mat2x4
* mat3x2, mat3x3, mat3x4
* mat4x2, mat4x3, mat4x4

Les constructeurs sont utilisés pour convertir un type en un autre, pour initialiser les valeurs d’un type.

**Peuvent être réorganisées, ex :**

vec4 v0 = vec4(1, 2, 3, 4); // (x, y, z, w)

v0 = v0.zxyw + v0.wwyz; // v0 = (7,5,4,7)

vec3 v1 = v0.yxx; // v1 = (5,7,7)

v1.x = v0.z; // v1 = (4,7,7)

v0.wx = vec2(7,8); // v0 = (8,5,4,7)

**Manipulation des matrices :**

mat4 m;

m[1] = vec4(2); // la colonne

m[0][0] = 1;

m[2][3] = 2;

Les fonctions :

* Arguments : types de bases, tableaux ou structures
* Retourne un type de base ou void
* Les récursions ne sont pas supportées
* Les arguments peuvent être in, out, inout

| **Attention** : Une shader doit avoir une fonction “main” |
| --- |

**in** = déclare une variable provenant de l'étage précédent

**out** = déclare une variable envoyée à l'étage suivant

**uniform** = variable constante pour un groupes de primitives valeurs définies par le programme hôte

**layout()** = contrôle sur le stockage des blocks layout(row\_major) uniform mat3x4A;

**in/out/inout** = pour les fonctions