1 Introduction à la Modélisation 3D

1.1 Qu'est-ce que la géométrie ?

Definition 1.1 (Géométrie)**.** Le terme *Géométrie* dérive du grec / geômétrês, où "geo" signifie "Terre" et "metrês" signifie "mesure". La géométrie peut être comprise comme :

1. L'étude des formes, des tailles, des motifs et des positions.
2. L'étude des espaces où l'on peut mesurer une certaine quantité (longueurs, angles, etc.).

1.2 Comment décrire la géométrie ?

Il existe plusieurs manières de décrire la géométrie :

- **Implicite :** Définie par une équation ou une condition. Exemple : le cercle unité est décrit par l'équation $x^2 + y^2 = 1$.
- **Explicite :** Définie par un paramétrage. Exemple : le cercle unité peut être décrit par les coordonnées $(\cos\theta, \sin\theta)$ pour $\theta \in [0, 2\pi)$.

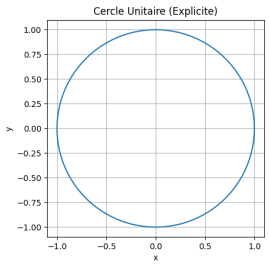

Figure 1: Représentation explicite du cercle unité.

- **Linguistique :** Décrite par des mots. Exemple : "cercle unitaire".
- **Discrète :** Approximée par un ensemble fini de points ou de polygones.

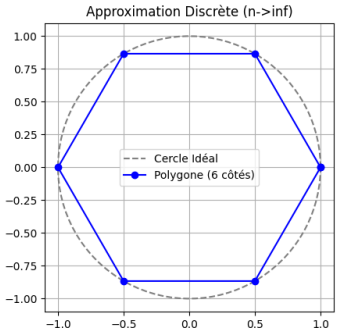

Figure 2: Approximation discrète d'un cercle.

- **Dynamique :** Décrite par une équation différentielle ou un processus. Exemple : l'équation $\frac{d^2x}{dt^2} = -x$ décrit un mouvement oscillatoire qui peut tracer un cercle ou une ellipse.

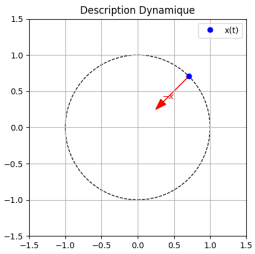

Figure 3: Illustration d'une description dynamique.

- **Symétrique :** Exploitant les symétries pour la description.

1.3 Exemples de la géométrie

La géométrie est omniprésente, des formes simples (carrés, cercles, cubes, sphères) aux structures complexes (bâtiments, anatomie, phénomènes naturels comme les gouttes d'eau).

1.4 Encoder numériquement la géométrie

En informatique graphique, nous devons encoder la géométrie numériquement. Les approches principales sont :

- **Explicite :** Les points de la surface sont directement définis.
  - Nuage de points
  - Maillage polygonal (le plus courant)
  - Surfaces de Subdivision, NURBS (pour des surfaces lisses)
- **Implicite :** La surface est définie comme l'ensemble des points satisfaisant une condition.
  - Ensemble de niveaux (level set)
  - Surface algébrique (ex: $f(x, y, z) = 0$)
  - L-systems (souvent utilisés pour les fractales et les plantes)

Le choix de la représentation dépend de la tâche et du type de géométrie à modéliser.

1.5 Représentations Implicites de la Géométrie

- Les points ne sont pas connus directement, mais satisfont à une certaine relation mathématique.
- Exemple : Une sphère unitaire est l'ensemble des points (x, y, z) tels que $x^2 + y^2 + z^2 = 1$.
- De façon plus générale, une surface implicite est définie par une équation $f(x, y, z) = 0$.

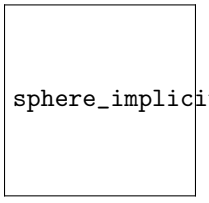

Figure 4: Visualisation d'une sphère unitaire définie implicitement.

1.6 Représentations Explicites de la Géométrie

- Tous les points sont donnés directement ou via une fonction paramétrique.
- Exemple : Les points sur la sphère unitaire sont donnés par $(\cos(u)\sin(v), \sin(u)\sin(v), \cos(v))$ pour $0 \leq u \leq 2\pi$ et $0 \leq v \leq \pi$.
- De façon plus générale, une surface explicite est définie par une fonction $f : \mathbb{R}^2 \to \mathbb{R}^3$, où $(u, v) \mapsto (x(u, v), y(u, v), z(u, v))$.

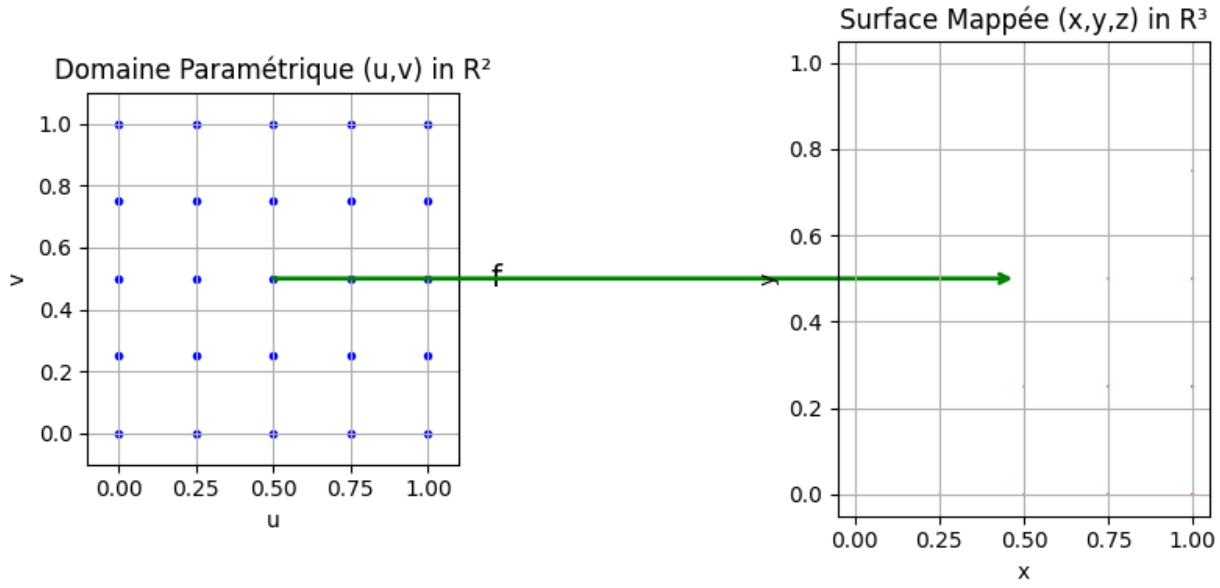

Figure 5: Mapping d'un domaine paramétrique $\mathbb{R}^2$ vers une surface dans $\mathbb{R}^3$.

2 Types de Modélisation 3D

La modélisation 3D consiste à créer une représentation mathématique d'un objet tridimensionnel. La plupart des objets virtuels (terrains, personnages, objets manufacturés, etc.) sont représentés par leur surface.

Les approches de modélisation peuvent être classées en :

- **Reconstruction :** À partir d'un objet réel (scan 3D, photogrammétrie).
- **Modélisation automatique :** Génération algorithmique (ex: arbres, terrains fractals).
- **Modélisation interactive :** Utilisation d'outils dédiés par un artiste ou un concepteur (ex: logiciels de CAD, 3ds Max, Maya, Blender).

Les principaux types de représentation géométrique sont :

- Représentation fil de fer (wireframe)
- Modélisation surfacique (la plus courante)
- Modélisation volumique
- Modélisation procédurale
- Données non structurées (nuages de points)

2.1 Modélisation surfacique

- **Caractéristiques :**
  - Représentation visuelle de l'extérieur d'un objet et de ses contours.
  - Aucune propriété de masse ou d'épaisseur n'est intrinsèquement définie (objets creux).
  - Ne peuvent pas être "découpés" facilement comme des solides.
- **Approches :**
  - Primitives solides (combinées par opérations booléennes - voir CSG)
  - Maillage (Mesh)
  - Surfaces paramétrées (NURBS, Bézier)
  - Balayage de surface (Sweeping)
  - Etc.

2.2 Maillage (Mesh)

Definition 2.1 (Maillage)**.** Un maillage représente les formes comme un ensemble de **sommets** (vertices), **arêtes** (edges) et **facettes** (faces), le plus souvent des triangles ou des quadrilatères.

- Plus le nombre de polygones (souvent des triangles) est élevé, plus le réalisme est potentiellement garanti.
- **Avantages :**
  - Permet de représenter des surfaces et des courbes complexes.
  - Standard de facto dans de nombreuses applications (jeux vidéo, animation).
- **Désavantages :**
  - Peut présenter un aspect angulaire si la résolution est faible.
  - Une haute résolution nécessite un grand nombre de polygones, impactant les performances de calcul et de rendu.

2.2.1 Énumération de facettes

Une manière simple de représenter un maillage est d'énumérer les sommets de chaque facette.

- **Simple à mettre en œuvre mais :**
  - **Duplication de données :** Chaque sommet est listé autant de fois qu'il appartient à une facette.
  - **Problèmes de cohérence :** Difficile de savoir quelles facettes partagent un sommet ou une arête (informations de voisinage).

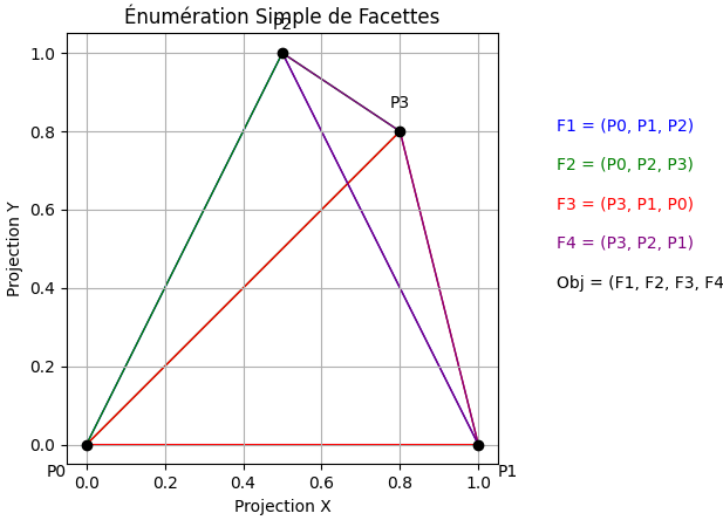

Figure 6: Énumération simple des facettes (duplication des sommets).

2.2.2 Énumération de facettes avec partage de sommets

Pour éviter la duplication, on stocke une liste unique de sommets (Vertex List) et chaque facette référence les indices des sommets dans cette liste.

- **Liste de Sommets (LS) :** $LS = \{P_0, P_1, P_2, P_3, ...\}$
- **Liste de Facettes (LF) :** Chaque facette est une liste d'indices dans LS.
  - $F_1 = (\text{index de } P_0, \text{index de } P_1, \text{index de } P_2)$
  - $F_2 = (\text{index de } P_0, \text{index de } P_2, \text{index de } P_3)$
  - ...
- **Avantage :** Évite la duplication de données des coordonnées des sommets. Facilite la modification de la géométrie (modifier un sommet affecte toutes les facettes le partageant).

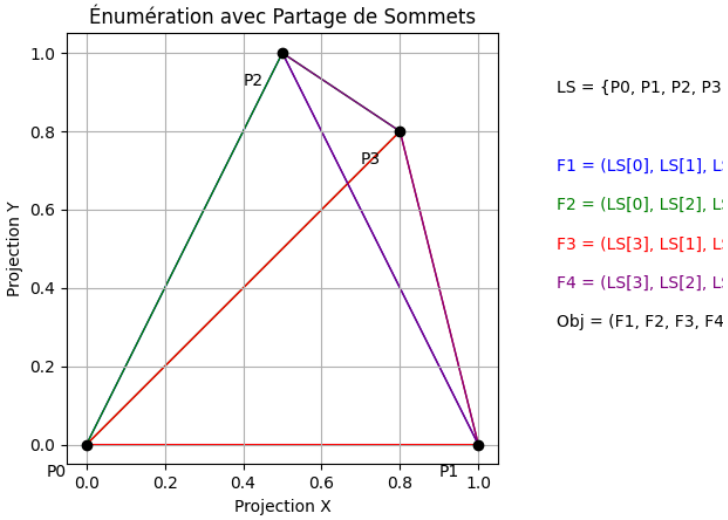

Figure 7: Énumération avec partage de sommets via une liste de sommets.

2.2.3 Orientation des surfaces

Pour distinguer l'intérieur de l'extérieur d'un objet fermé et pour l'éclairage, il est crucial de définir l'orientation des facettes.

- La **normale** à une face (vecteur perpendiculaire) est déterminée par le sens de parcours de ses sommets (arêtes).
- **Conventions courantes :**
  - **Sens trigonométrique (counter-clockwise - CCW) :** Définit la normale comme étant **sortante** (pointe vers l'extérieur). C'est la convention la plus utilisée (OpenGL, DirectX).
  - **Sens anti-trigonométrique (clockwise - CW) :** Définit la normale comme étant **rentrante** (pointe vers l'intérieur).
- La règle de la main droite peut être utilisée : si les doigts s'enroulent dans le sens de parcours des sommets, le pouce indique la direction de la normale.

2.3 Maillage - Surfaces de subdivision

- Technique pour créer des surfaces lisses à partir d'un maillage polygonal grossier ("cage" de contrôle).
- Consiste à subdiviser récursivement les polygones existants et à ajuster la position des nouveaux sommets et des anciens sommets selon des règles spécifiques (ex: Catmull-Clark, Loop).
- La forme finale de la surface est contrôlée par la position relative des sommets et des segments du maillage de contrôle initial.
- Permet d'obtenir des surfaces lisses avec un contrôle intuitif.

2.4 Surfaces paramétrées

- La surface est calculée directement par une équation paramétrique $f(u, v) = (x(u, v), y(u, v), z(u, v))$.
- Exemples : Sphères, Tore, Surfaces de Bézier, NURBS.
- Avantages : Représentation mathématique exacte et lisse, contrôle précis.

- Inconvénients : Peut être complexe à modéliser pour des formes arbitraires, conversion vers un maillage (tessellation) souvent nécessaire pour le rendu. Nombre limité d'objets "naturellement" paramétrables.

2.5 Balayage de surface (Sweeping)

Technique de modélisation où une forme 2D (profil) est déplacée le long d'une trajectoire 3D (chemin) pour générer une surface ou un solide.

- **Extrusion :** Le profil est déplacé le long d'une ligne droite.
- **Révolution :** Le profil est tourné autour d'un axe.

Permet de créer facilement des objets avec une certaine symétrie ou répétition.

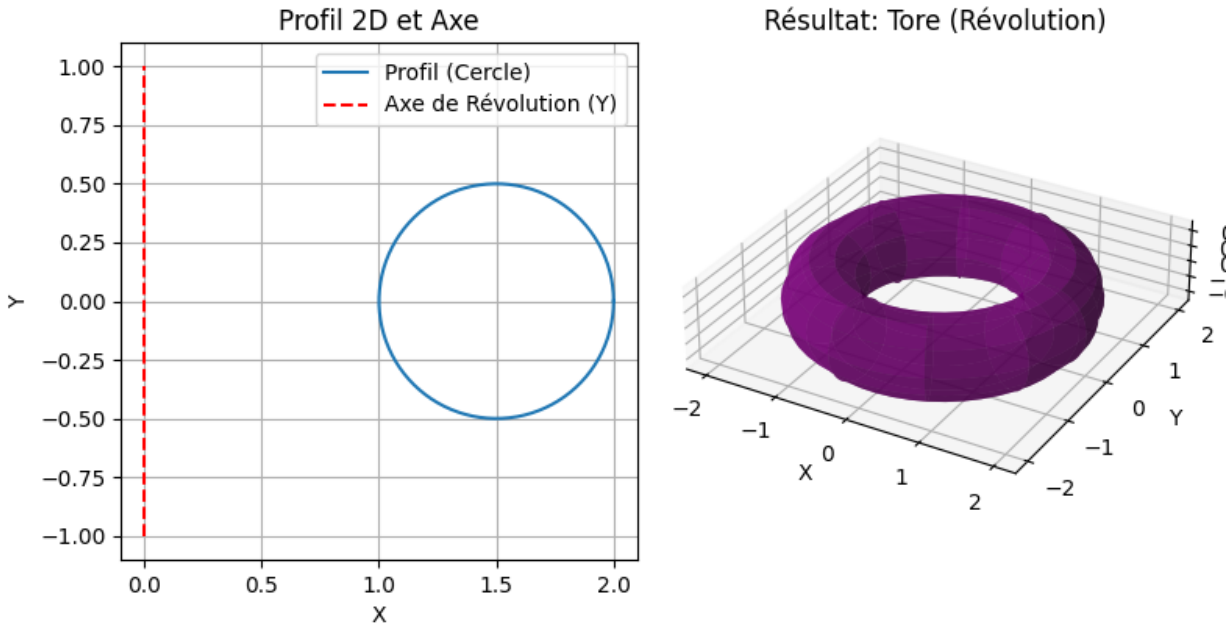

Figure 8: Exemple de balayage par révolution : un cercle tourné autour de l'axe Y crée un tore.

2.6 Modélisation volumique

Contrairement à la modélisation surfacique qui ne représente que l'extérieur, la modélisation volumique représente également l'intérieur de l'objet.

- **Caractéristiques :**
  - Le modèle contient les informations de chaque point dans l'espace occupé par l'objet.
  - L'espace peut être décrit par des mathématiques pures, des opérations booléennes (addition, soustraction, intersection de volumes), ou des mélanges.
  - Pas de géométrie explicite (sommets, lignes, triangles) tant qu'elle n'est pas nécessaire pour le rendu ou l'export. Permet des opérations robustes comme les découpes.
- **Approches :**
  - **Voxélisation :** L'espace est divisé en une grille régulière de petits cubes (voxels).
  - **Géométrie Solide Constructive (CSG - Constructive Solid Geometry) :** L'objet est construit en combinant des primitives solides simples (cubes, sphères, cylindres) à l'aide d'opérations booléennes (union, intersection, différence).

- **Blobtree / Metaballs :** Utilisation de fonctions de champ implicites qui se "fondent" les unes dans les autres.
- Etc.

2.6.1 Modélisation volumique: Voxélisation

- Représentation par un ensemble de points dans une grille 3D régulière (modèle discret).
- Chaque voxel peut stocker des informations (ex: densité, couleur, matériau).
- **Applications :** Images médicales (CT scans, IRM), simulation de fluides (CFD), sculpture virtuelle.
- **Paramètres :** Intérieur/extérieur, couleur, réfraction/absorption.
- **Avantages :**
  - Topologie simple et implicite.
  - Acquisition directe depuis certains capteurs (CT, IRM).
  - Facilite certains types de rendu (rendu volumétrique).
- **Désavantages :**
  - Très gourmand en mémoire (grands ensembles de données).
  - Génération de données complexes peut être difficile.
  - Peut introduire une anisotropie (artefacts liés à l'alignement de la grille).

2.6.2 Modélisation volumique: Constructive Solid Geometry (CSG)

- Combine des formes géométriques primitives simples (sphère, cube, cylindre, cône, etc.) en utilisant des opérations ensemblistes booléennes :
  - **Union ($\cup$):** Combine deux objets en un seul.
  - **Intersection ($\cap$):** Ne garde que la partie commune à deux objets.
  - **Différence ($-$):** Enlève un objet d'un autre.
- La structure est souvent représentée par un arbre binaire (CSG Tree) où les feuilles sont des primitives et les nœuds internes sont des opérations.
- **Applications :** CAO (CAD), modélisation solide, Pov-Ray.

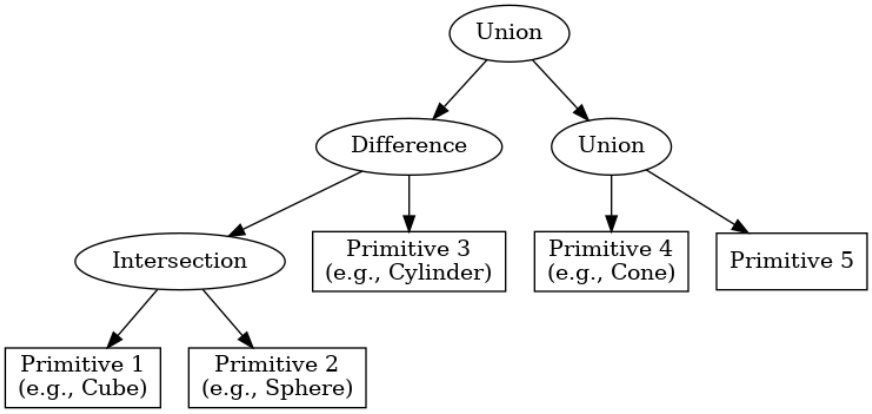

Figure 9: Exemple simplifié d'un arbre CSG (Structure hypothétique).

Exercice (basé sur diapo 8): Déterminer les opérations booléennes utilisées pour créer les formes numérotées 1 à 4 (nécessite les diagrammes spécifiques).

2.6.3 Modélisation volumique: Blobtree

- Similaire à CSG, mais utilise des primitives implicites (souvent basées sur des fonctions de champ potentiel) et des opérations de "mélange" (blending) en plus des opérations booléennes classiques.
- Les opérations de mélange permettent aux formes de fusionner de manière organique (comme des gouttes de liquide).
- Peut aussi incorporer des opérations de déformation (torsion, étirement).

2.7 Modélisation procédurale

- Crée des modèles 3D et/ou des textures en utilisant des algorithmes et des ensembles de règles, plutôt qu'une manipulation manuelle directe.
- Particulièrement utile lorsqu'il serait fastidieux ou répétitif de créer les modèles à la main.
- **Exemples d'application :**
  - Végétation (arbres, plantes)
  - Architecture (bâtiments, villes)
  - Terrains (montagnes, paysages)
- **Algorithmes courants :**
  - **L-Systèmes (Lindenmayer Systems) :** Utilisent une grammaire formelle pour générer des structures auto-similaires, idéales pour les plantes.
  - **Fractales :** Basées sur des motifs récursifs.
  - **Langages de modélisation générative (GML) :** Langages spécifiques pour décrire des processus de génération.

3 Nuages de Points 3D

Definition 3.1 (Nuage de Points)**.** Un nuage de points est une collection de points dans l'espace 3D, chacun défini par ses coordonnées (x, y, z), représentant la surface externe d'objets.

- **Acquisition :**
  - Dispositifs de balayage 3D :
    - Scanner à lumière structurée
    - Scanner LiDAR (Light Detection and Ranging)
  - Reconstruction à partir d'images/vidéos 2D (Photogrammétrie, Structure from Motion - SfM). Voir VisualSFM par exemple.
- Format de données non structuré (pas d'information explicite de connexion entre les points).

3.1 Tâches associées aux nuages de points

Plusieurs traitements sont couramment appliqués aux nuages de points :

- **Tâche 1 : Simplification :** Réduire le nombre de points tout en préservant la forme globale. Approches locales ou globales existent.
- **Tâche 2 : Classification / Reconnaissance d'objets :** Identifier et étiqueter les objets présents dans le nuage de points (ex: chaise, table, moniteur).

- **Tâche 3 : Segmentation :** Regrouper les points appartenant à différentes parties ou instances d'objets (ex: segmenter l'aile, la queue et le corps d'un avion).
- **Tâche 4 : Estimation des normales :** Calculer le vecteur normal à la surface sous-jacente pour chaque point, utile pour l'ombrage et d'autres analyses géométriques.

4 Transformations Géométriques

Les transformations géométriques sont fondamentales en informatique graphique pour positionner, orienter, et déformer des objets dans une scène 3D. Elles font partie intégrante du *pipeline graphique*.

4.1 Transformation Spatiale

Definition 4.1 (Transformation Spatiale)**.** Toute fonction $f : \mathbb{R}^n \to \mathbb{R}^n$ qui assigne à chaque point un nouvel emplacement.

- En modélisation 3D ($n = 3$), ces transformations déplacent les points (x, y, z) vers de nouvelles positions (x', y', z').
- Nous nous concentrerons sur les transformations courantes :
  - **Transformations linéaires :** Rotation, mise à l'échelle (scaling), cisaillement (shear). Elles préservent l'origine et les lignes droites.
  - **Translation :** Déplacement de l'objet. Ce n'est pas une transformation linéaire car elle ne préserve pas l'origine.

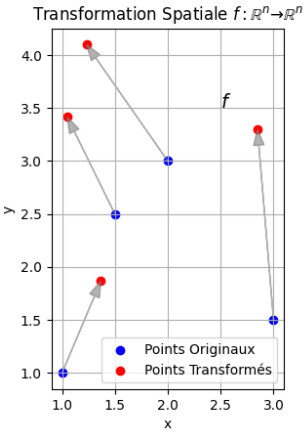

Figure 10: Illustration d'une transformation spatiale assignant de nouvelles positions aux points.

4.2 Transformations de Modélisation

Dans le pipeline graphique, les transformations de modélisation sont appliquées aux objets pour les positionner et les orienter dans la scène globale (World Space).

- Elles permettent de passer du système de coordonnées local de chaque objet (Object Space), où l'objet est souvent défini autour de son propre origine, au système de coordonnées commun de la scène (World Space).

- Exemple : Placer plusieurs instances d'une chaise (définie une seule fois en Object Space) à différents endroits et avec différentes orientations dans une pièce (World Space).

Le pipeline graphique typique inclut : Transformations de Modélisation → Illumination (Shading) → Transformation d'affichage (Camera) → Clipping → Transformation écran (Projection) → Pixellisation (Rasterization) → Visibilité / Rendu.

4.2.1 Rappel: Transformation linéaire

Une transformation $f : \mathbb{R}^n \to \mathbb{R}^n$ est dite **linéaire** si elle satisfait deux propriétés :

1. Additivité : $f(u+v) = f(u) + f(v)$ pour tous vecteurs u, v.
2. Homogénéité : $f(cu) = cf(u)$ pour tout vecteur u et tout scalaire c.

- **Géométriquement :** Une transformation linéaire préserve l'origine ($f(0) = 0$), fait correspondre des lignes droites à des lignes droites, et préserve le parallélisme des lignes.
- **Algébriquement :** Elle préserve les opérations de l'espace vectoriel (addition de vecteurs et multiplication par un scalaire). Toute transformation linéaire peut être représentée par une multiplication matricielle : $f(u) = Mu$, où M est la matrice de transformation.

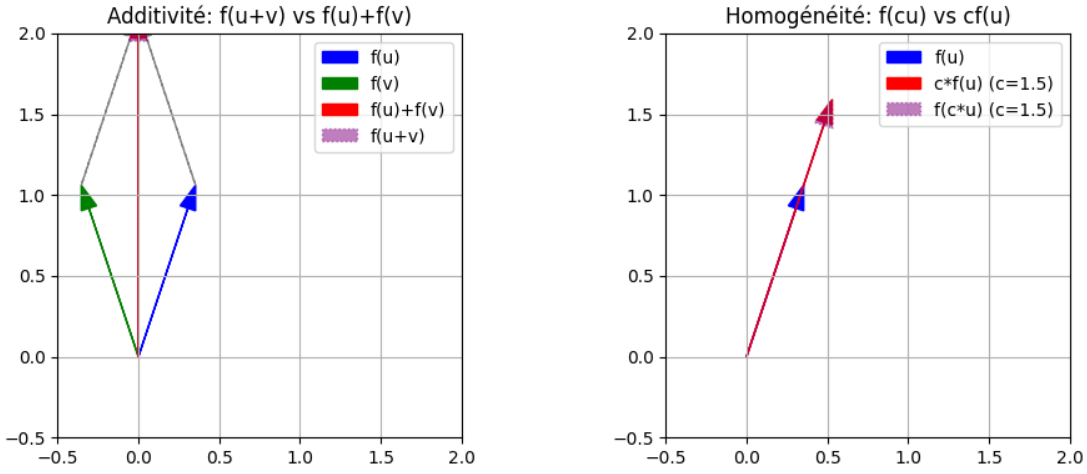

Figure 11: Vérification graphique des propriétés d'une transformation linéaire (additivité et homogénéité).

4.2.2 Utilisations des transformations

Les transformations géométriques servent à :

- Déplacer un objet dans une scène (translation).
- Orienter un objet (rotation).
- Changer la taille d'un objet (mise à l'échelle/scaling).
- Déformer un objet (cisaillement/shear, ou transformations non linéaires).
- Placer et orienter l'observateur (caméra) par rapport à la scène.
- Répliquer un motif ou un objet plusieurs fois.
- Projeter la scène 3D sur un plan 2D pour l'affichage (projection).

4.2.3 Types de transformations

Les transformations de base sont :

- **Translation :** Déplace tous les points d'une distance et direction constantes.
- **Rotation :** Fait tourner les points autour d'un axe ou d'un point fixe.
- **Mise à l'échelle (Scaling) :** Agrandit ou réduit les points par rapport à l'origine ou un point fixe. Peut être uniforme (même facteur dans toutes les directions) ou non uniforme.
- **Cisaillement (Shear) :** Incline l'objet, comme si les couches de l'objet glissaient les unes sur les autres.

4.3 Transformations et Sommets

- Un objet 3D est souvent décrit par un ensemble de sommets (vertices) qui définissent sa forme (ex: les coins d'un cube, les points d'un maillage).
- Appliquer une transformation géométrique à un objet revient à appliquer cette même transformation à **tous** ses sommets.
- Les arêtes et facettes sont définies par rapport aux sommets transformés.

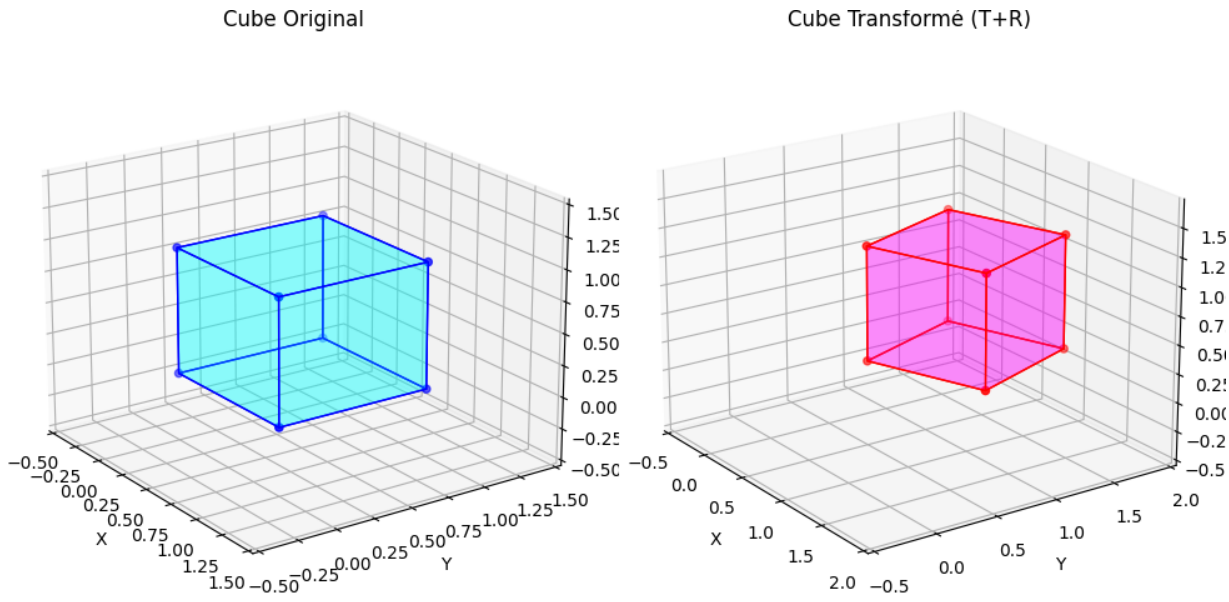

Figure 12: Appliquer une transformation (ex: Rotation + Translation) à un objet revient à l'appliquer à tous ses sommets.

4.3.1 Notation vectorielle

Les sommets sont représentés sous forme de vecteurs colonnes : $p = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ ou, en coordonnées homogènes (voir plus loin), $p = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$.

4.4 Rotation

- Les rotations sont définies par trois propriétés fondamentales :
  1. **Origine fixe :** Le point (0,0,0) n'est pas déplacé.
  2. **Distances conservées :** La distance entre deux points quelconques est la même avant et après la rotation (isométrie).
  3. **Orientation conservée :** Ne transforme pas un objet "droitier" en objet "gaucher" (pas de symétrie miroir). Mathématiquement, le déterminant de la matrice de rotation est +1.
- Les deux premières propriétés impliquent que les rotations sont des transformations linéaires.

4.4.1 Rotation autour de l'axe Z

Pour faire tourner un point $p = (x, y, z)$ d'un angle θ autour de l'axe Z par rapport à l'origine :

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned}$$

En notation matricielle (coordonnées standard) : $\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

La matrice $R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$ est la matrice de rotation autour de l'axe Z.

4.4.2 Matrices de rotation autour des axes X et Y

De même, les matrices pour les rotations autour des axes X et Y sont : $R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$

$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$ *(Attention au signe du sinus dans R_y due à l'orientation du repère)*

4.4.3 Règles pour la construction de la matrice de rotation (Angles d'Euler - Coordonnées Homogènes)

Une rotation 3D quelconque peut être décomposée en rotations successives autour des axes (angles d'Euler). La matrice de rotation globale R en coordonnées homogènes 4×4 est construite ainsi : $R = \begin{pmatrix} & & & 0 \\ & R_{3x3} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ où R_{3x3} est la matrice de rotation 3x3 résultant de la composition des rotations autour des axes (ex: $R = R_z(\alpha)R_y(\beta)R_x(\gamma)$).

Pour construire la matrice d'une rotation d'angle θ autour d'un axe principal (X, Y ou Z) en coordonnées homogènes 4x4 :

- La ligne et la colonne correspondant à l'axe de rotation contiennent un 1 sur la diagonale et des 0 ailleurs (sauf pour le 1 en bas à droite).
- La coordonnée homogène (4ème ligne/colonne) a un 1 sur la diagonale et des 0 ailleurs.
- Les deux autres axes forment un bloc 2x2 contenant $\cos\theta$ sur la diagonale.
- Les termes $\pm\sin\theta$ sont placés hors diagonale dans ce bloc 2x2 pour "compléter le carré".

- **Signe du sinus :** Sur la ligne *suivant* celle de l'axe de rotation (cycliquement: X→Y→Z→X), le sinus est précédé d'un signe '-'.

Exemple pour $R_z(\theta)$ en 4x4 : Axe Z = ligne/colonne 3. $\begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ (Ligne 1 (X) : cos, Ligne 2 (Y) : sin, Ligne 2 contient $-\sin$ car Y suit Z cycliquement? Non, Y précède Z. La règle du signe est : le $\sin\theta$ à la ligne i et colonne j a un signe opposé à celui de la ligne j et colonne i. L'élément $(1,2)$ est $-\sin\theta$, l'élément $(2,1)$ est $\sin\theta$.)

4.5 Homothétie (Mise à l'échelle / Scaling)

- Multiplie les coordonnées par des facteurs d'échelle S_x, S_y, S_z.
- Formules : $x' = S_x \cdot x$, $y' = S_y \cdot y$, $z' = S_z \cdot z$.
- Notation matricielle (coordonnées standard) : $\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$
- La matrice $S = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix}$ est la matrice d'homothétie.
- L'homothétie est une transformation linéaire.

4.5.1 Homothétie isotrope

Si $S_x = S_y = S_z = S$, la mise à l'échelle est uniforme dans toutes les directions. La matrice (en coordonnées homogènes 4x4) devient : $S_{iso} = \begin{pmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

4.5.2 Affinités orthogonales

Cas particuliers d'homothétie non uniforme où la mise à l'échelle ne se fait que le long d'un axe.

- Affinité d'axe x (par rapport au plan yOz) : $S_y = S_z = 1$. Matrice 4x4 : $\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
- Affinité d'axe y (par rapport au plan xOz) : $S_x = S_z = 1$. Matrice 4x4 : $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
- Affinité d'axe z (par rapport au plan xOy) : $S_x = S_y = 1$. Matrice 4x4 : $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

4.6 Translation

- Ajoute un vecteur de translation $T = (T_x, T_y, T_z)$ aux coordonnées du point.
- Formules : $x' = x + T_x$, $y' = y + T_y$, $z' = z + T_z$.
- Notation : $p' = p + T$.
- **Question :** La translation est-elle une transformation linéaire ? *Réponse :* Non, car $f(0) = 0 + T = T \neq 0$ (si $T \neq 0$). Elle ne préserve pas l'origine. Elle ne peut donc pas être représentée par une simple multiplication matricielle 3x3.

4.7 Coordonnées Homogènes

Pour unifier le traitement des transformations linéaires (rotation, scaling) et de la translation, on utilise les coordonnées homogènes.

- Un point 3D $p = (x, y, z)$ est représenté par un vecteur 4D $p_h = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$. La 4ème composante, w, est appelée coordonnée homogène (ici $w = 1$).
- Les transformations 3D (rotation, scaling, translation, projection) peuvent alors toutes être représentées par une multiplication par une matrice 4×4.
- Pour repasser des coordonnées homogènes $p'_h = (X, Y, Z, H)$ aux coordonnées cartésiennes $p' = (x', y', z')$, on divise par la coordonnée homogène H : $x' = X/H$, $y' = Y/H$, $z' = Z/H$. (Pour les translations, rotations, scaling, H reste 1).

4.7.1 Structure de la Matrice 4x4

Une matrice de transformation M_H en coordonnées homogènes 4×4 peut être vue comme composée de 4 sous-matrices : $M_H = \begin{pmatrix} M_{3x3} & T_{3x1} \\ P_{1x3} & S_{1x1} \end{pmatrix}$ où :

- M_{3x3} : Contient la partie linéaire (rotation, scaling, shear).
- T_{3x1} : Contient la partie translation $\begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$.
- P_{1x3} : Utilisée pour la projection perspective. Pour les transformations affines (rotation, scaling, shear, translation), $P_{1x3} = (0, 0, 0)$.
- S_{1x1} : Facteur d'échelle global (souvent 1). Pour les transformations affines, $S_{1x1} = 1$.

4.7.2 Matrice de Translation en Coordonnées Homogènes

La translation par $T = (T_x, T_y, T_z)$ est représentée par la matrice : $T(T_x, T_y, T_z) = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Vérification : $p' = T \cdot p = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{pmatrix}$

4.8 Manipulations géométriques

- Soit un point p et une transformation géométrique M (matrice 4x4).
- Le point transformé p' est obtenu par $p' = M \cdot p$.
- Les transformations courantes sont : Translation, Rotation, Changement d'échelle, Projection, Symétries, Affinités orthogonales, etc.

4.9 Glissement (Shear)

Le cisaillement (ou shear) est une transformation qui déforme un objet en faisant glisser ses "couches" parallèlement à une direction.

- Matrice d'un glissement parallèle à l'axe x, de rapport k (le déplacement en x est proportionnel à y) : $Sh_{x,y}(k) = \begin{pmatrix} 1 & k & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
- Matrice d'un glissement parallèle à l'axe x, de rapport k, avec ligne de base $y = y_{ref}$ (le déplacement est nul pour $y = y_{ref}$ et proportionnel à $y - y_{ref}$) : $Sh_{x,y,ref}(k, y_{ref}) = \begin{pmatrix} 1 & k & 0 & -k \cdot y_{ref} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ (Obtenu par Translation$(0, -y_{ref}, 0)$, Shear(k), Translation$(0, y_{ref}, 0)$).

4.10 Composition de Transformations

Pour appliquer plusieurs transformations successivement (ex: mettre à l'échelle puis translater), on multiplie leurs matrices.

- Si on applique d'abord M_1 puis M_2 à un point p : $p' = M_1 \cdot p$ $p'' = M_2 \cdot p' = M_2 \cdot (M_1 \cdot p) = (M_2 \cdot M_1) \cdot p$
- La transformation globale est représentée par la matrice produit $M'' = M_2 \cdot M_1$.
- **Ordre d'application :** La multiplication matricielle n'est **pas commutative** ($M_1 \cdot M_2 \neq M_2 \cdot M_1$ en général).
- L'ordre d'application des transformations est crucial. La transformation la plus proche du point p dans l'équation (celle de droite) est appliquée en premier. $p'' = M_2 \cdot M_1 \cdot p$ signifie : appliquer M_1 d'abord, puis M_2 au résultat.

Example 4.2 (Non-commutativité). Soit $S(2, 2)$ une mise à l'échelle et $T(3, 1)$ une translation.

1. Homothétie PUIS translation : $p' = T(3, 1) \cdot S(2, 2) \cdot p$ $M_{TS} = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ (En 2D homogène pour simplifier)
2. Translation PUIS homothétie : $p' = S(2, 2) \cdot T(3, 1) \cdot p$ $M_{ST} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 6 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{pmatrix}$

Les matrices M_{TS} et M_{ST} sont différentes, le résultat final dépend de l'ordre.

4.10.1 Transformation autour d'un point arbitraire

Pour effectuer une transformation (ex: rotation $R(\theta)$) autour d'un point $P = (T_x, T_y, T_z)$ autre que l'origine :

1. Translater pour amener P à l'origine : $T(-T_x, -T_y, -T_z)$.
2. Effectuer la rotation (ou autre transformation) autour de l'origine : $R(\theta)$.
3. Translater pour ramener le point P à sa position initiale : $T(T_x, T_y, T_z)$.

La matrice de transformation composite M est : $M = T(T_x, T_y, T_z) \cdot R(\theta) \cdot T(-T_x, -T_y, -T_z)$

4.11 Modélisation Hiérarchique

Permet de décrire des objets complexes composés d'objets plus simples, en organisant la scène sous forme d'un arbre ou d'un graphe (DAG - Directed Acyclic Graph).

- **Structure :** Les nœuds internes représentent des groupements ou des transformations, les feuilles représentent des objets géométriques simples (primitives) ou des instances d'autres objets.
- **Transformations relatives :** Chaque arc (ou nœud) est associé à une transformation géométrique qui positionne l'objet fils (ou le sous-arbre) par rapport au repère de son parent.
- **Avantages :**
  - Facilite la description d'objets articulés (ex: robots, personnages).
  - Permet la réutilisation d'objets (instanciation).
  - Simplifie l'animation (animer un nœud parent affecte tous ses descendants).
- **Transformation globale :** Pour obtenir la transformation d'un objet feuille dans le repère global (monde), on compose (multiplie) toutes les matrices de transformation rencontrées en descendant depuis la racine de l'arbre jusqu'à la feuille.

Example 4.3 (Hiérarchie d'une maison)**.** Racine (Monde) → Maison (Tm : place la maison dans le monde) → Corps Principal (Tc : place le corps / Racine Maison) → Facade (Tfa : place facade / Corps Principal) → Fenêtre 1 (Tf1 : place fenêtre 1 / Facade) → Vitre a (Tva : place vitre a / Fenêtre 1) → Fenêtre 2 ... → Toit (...) → Garage (Tg : place garage / Racine Maison) → Porte a (Tga : place porte a / Garage) → Porte b (Tgb : place porte b / Garage)

Pour trouver la transformation de la Vitre a dans le monde, on calcule : $M_{vitreA} = Tm \cdot Tc \cdot Tfa \cdot Tf1 \cdot Tva$. Ceci est souvent géré à l'aide d'une pile de matrices de transformation pendant le parcours du graphe de scène.

4.11.1 Instanciation (Instancing)

Technique permettant de réutiliser la même définition géométrique plusieurs fois dans la scène, en appliquant simplement des transformations différentes à chaque instance. Un nœud "pointeur" dans le graphe de scène référence la géométrie à instancier.

4.12 Changement de Repère

Permet de transformer les coordonnées d'un point exprimées dans un repère R_2 en coordonnées exprimées dans un autre repère R_1.

- Soit p_1 les coordonnées de p dans R_1 et p_2 les coordonnées de p dans R_2.
- Soit M la matrice de passage de R_2 vers R_1. Cette matrice transforme les vecteurs de base de R_2 pour les exprimer dans R_1, et contient également la position de l'origine de R_2 dans R_1.
- Alors : $p_1 = M \cdot p_2$.
- Inversement, pour passer de R_1 à R_2 : $p_2 = M^{-1} \cdot p_1$.
- **Applications :**
  - Repère objet → Repère scène (Modélisation hiérarchique)
  - Repère scène → Repère caméra (Transformation de vue)

La matrice de passage M (4x4) exprime les axes (x_2, y_2, z_2) et l'origine O_2 du repère R_2 dans le repère R_1: $M = \begin{pmatrix} x_{2x1} & y_{2x1} & z_{2x1} & O_{2x1} \\ x_{2y1} & y_{2y1} & z_{2y1} & O_{2y1} \\ x_{2z1} & y_{2z1} & z_{2z1} & O_{2z1} \\ 0 & 0 & 0 & 1 \end{pmatrix}$ où x_{2x1} est la coordonnée x de l'axe x_2 exprimée dans R_1, etc. et O_{2x1} est la coordonnée x de l'origine O_2 exprimée dans R_1.

5 Projections

La projection est l'étape du pipeline graphique qui transforme les objets 3D de l'espace caméra (ou espace vue) vers un espace 2D (l'espace image ou écran). C'est une réduction du nombre de dimensions.

- **Modèle du sténopé (Pinhole Camera) :** Modèle simplifié d'une caméra où la lumière passe par un trou minuscule (sténopé) pour former une image inversée sur une surface photosensible (capteur ou plan image).
- Les lignes de projection (projecteurs) partent des points de l'objet 3D et convergent vers le centre de projection (CP) (le sténopé). L'intersection de ces projecteurs avec le plan de projection forme l'image 2D.

Il existe deux types principaux de projections géométriques :

1. **Projection Perspective :** Le centre de projection (CP) est à une distance finie.
   - Les projecteurs convergent vers le CP.
   - Plus réaliste (similaire à la vision humaine).
   - Les objets plus éloignés apparaissent plus petits.
   - Les lignes parallèles dans l'espace 3D peuvent apparaître convergentes dans l'image 2D (points de fuite).
2. **Projection Parallèle :** Le centre de projection (CP) est à l'infini.
   - Les projecteurs sont parallèles entre eux.
   - Moins réaliste, mais conserve les tailles relatives et le parallélisme des lignes parallèles au plan de projection.
   - Utilisée en dessin technique, architecture, CAO pour des mesures précises.

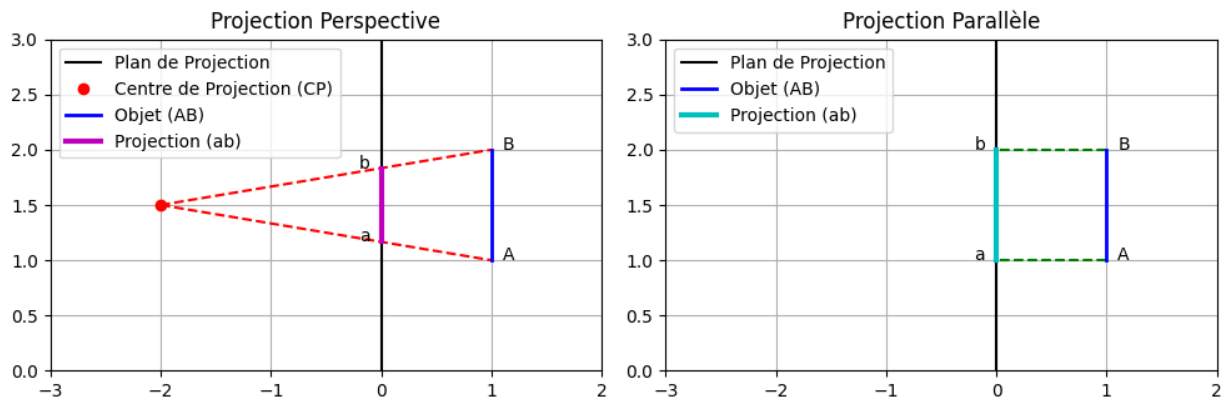

Figure 13: Comparaison entre projection perspective (gauche) et parallèle (droite).

5.1 Projections parallèles

Les projecteurs sont parallèles à une direction de projection (DP).

- **Propriétés :**
  - Les lignes parallèles dans l'espace 3D restent parallèles dans l'image 2D (si elles ne sont pas parallèles à la direction de projection).
  - Les rapports des distances le long d'une direction donnée sont conservés.
  - Pas réaliste (pas de diminution de taille avec la distance), mais utile pour les mesures exactes (dessins techniques).

5.1.1 Projection orthographique

Cas particulier de projection parallèle où les projecteurs sont **perpendiculaires** au plan de projection.

- Revient à simplement "ignorer" ou "supprimer" une des coordonnées.
- **Vues standard :** Vue de face (ignorer Z), vue de dessus (ignorer Y), vue de côté (ignorer X).
- **Forme matricielle (projection sur le plan z=0, ignore z) :** $p' = M_{ortho,z} \cdot p$ $M_{ortho,z} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ Appliqué à $p = (x, y, z, 1)^T$, donne $p' = (x, y, 0, 1)^T$.
- Matrices pour les autres vues : $M_{ortho,y} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ (Vue sur plan y=0) $M_{ortho,x} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ (Vue sur plan x=0)

5.1.2 Projection orthographique axonométrique

Le plan de projection **n'est pas perpendiculaire** aux axes de coordonnées principaux (X, Y, Z).

- Montre plusieurs facettes de l'objet simultanément.
- Les projecteurs restent perpendiculaires au plan de projection.
- Obtenu en appliquant des rotations à l'objet (ou à la caméra) avant la projection orthographique standard.
- Types courants : Isométrique, Dimétrique, Trimétrique (selon les angles entre les axes projetés).

5.1.3 Projection Oblique

Cas de projection parallèle où les projecteurs **ne sont pas perpendiculaires** au plan de projection. La direction de projection (DP) coupe le plan de projection avec un angle $\phi \neq 90°$.

- Introduit une déformation : les faces perpendiculaires au plan de projection sont visibles.
- **Projection Cavalière :** L'angle $\phi = 45°$. Les longueurs perpendiculaires au plan de projection sont conservées. Semble peu naturel.
- **Projection Cabinet :** L'angle $\phi \approx 63.4°$ ($\tan\phi = 2$). Les longueurs perpendiculaires au plan de projection sont réduites de moitié. Semble plus réaliste.
- La matrice de projection oblique est une combinaison d'une transformation de cisaillement (shear) suivie d'une projection orthographique.

5.2 Projection Perspective

Le centre de projection (CP) est à une distance finie.

- **Vue de côté (simplifiée) :** Soit CP à l'origine (0,0,0), le plan de projection à $z = -f$ (ou $z = f$ selon convention), et un point $P = (x, y, z)$. Le point projeté $q = (u, v)$ sur le plan de projection a pour coordonnées (par triangles semblables) : $u/x = v/y = f/(-z)$ (si plan en $z = -f$) $u = -\frac{f \cdot x}{z}$ et $v = -\frac{f \cdot y}{z}$ (Souvent $f = 1$ pour la caméra normalisée).
- **Propriétés clés :**
  - **Rétrécissement :** La taille projetée $|ab|$ d'un segment $|AB|$ varie inversement avec sa distance d au CP : $|ab| \propto |AB|/d$.
  - **Non-conservation du parallélisme :** Les lignes parallèles dans l'espace 3D qui ne sont pas parallèles au plan de projection convergent vers un **point de fuite** dans l'image projetée.
  - **Points de fuite :** Toutes les lignes parallèles entre elles dans l'espace 3D convergent vers le même point de fuite sur le plan de projection. La ligne d'horizon contient les points de fuite des lignes parallèles au plan horizontal.
- **Nombre de points de fuite :** Les projections perspectives sont classées par le nombre d'axes principaux du repère qui coupent le plan de projection (1, 2 ou 3 points de fuite).

5.2.1 Calcul et Matrice de Projection Perspective

En utilisant les coordonnées homogènes, on peut représenter la projection perspective par une matrice 4x4. Pour une projection simple sur le plan $z = -f$ avec CP à l'origine : En coordonnées cartésiennes : $x_p = -\frac{f \cdot x}{z}$, $y_p = -\frac{f \cdot y}{z}$, $z_p = -f$. En coordonnées homogènes, on cherche une matrice M_{persp} telle que $p'_h = M_{persp} \cdot p_h$, où $p'_h = (X', Y', Z', H')^T$ et $x_p = X'/H'$, $y_p = Y'/H'$. Une matrice possible (projection sur $z = 0$, CP en

$(0, 0, -f)$): $x_p = \frac{f \cdot x}{z+f}$ et $y_p = \frac{f \cdot y}{z+f}$. La matrice est : $M_{persp} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 1 \end{pmatrix}$ (Cette matrice transforme z' = z, w' = z/f + 1) Pour obtenir $x_p = x/(z/f + 1) = \frac{fx}{z+f}$ et $y_p = y/(z/f + 1) = \frac{fy}{z+f}$. La coordonnée z est souvent conservée (ou modifiée) pour le test de profondeur (z-buffer). La matrice exacte dépend de la convention et du plan de projection. Une matrice courante (OpenGL, projection sur near plane, CP à l'origine) modifie z et met $-z$ dans w: $M_{persp} = \begin{pmatrix} f/aspect & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{pmatrix}$ (où A, B dépendent de zNear, zFar, f=cot(fovY/2))

Le résultat $p'_h = (X', Y', Z', W')^T$ est ensuite divisé par W' (division perspective) pour obtenir les coordonnées normalisées (x', y', z').

5.2.2 View Frustum et Clipping

En pratique, on ne projette pas tout l'espace 3D, mais seulement le volume visible par la caméra, appelé **View Frustum** (tronc de pyramide).

- Défini par 6 plans :
  - Plan proche (Near Plane) : $z = z_{near}$ (ou $-z_{near}$)
  - Plan éloigné (Far Plane) : $z = z_{far}$ (ou $-z_{far}$)
  - Plans gauche/droite/haut/bas : Définis par l'angle de vue (Field of View - FOV) et le rapport d'aspect (aspect ratio = width/height).
- Tout ce qui est en dehors de ce volume est découpé (**clipping**).
- **Pourquoi z-near/z-far ?**
  - Éviter la division par zéro (ou proche de zéro) si $z \approx 0$.
  - Limiter la plage de valeurs de profondeur à stocker dans le z-buffer.
  - La précision du z-buffer est limitée (souvent 24 bits flottants). Les valeurs flottantes ont plus de précision près de zéro. Le mapping non linéaire de la profondeur en perspective concentre la précision près du plan proche (z-near), ce qui est généralement souhaitable. Choisir z_{near} trop petit gaspille de la précision. Choisir z_{near} trop grand ou z_{far} trop lointain peut causer du "z-fighting" (scintillement entre surfaces proches).

5.2.3 Passage en coordonnées normalisées (NDC)

Avant le rendu final, le volume du View Frustum est transformé (normalisé) en un cube canonique, souvent $[-1, 1]^3$, appelé **Normalized Device Coordinates (NDC)**.

- Cette étape simplifie le clipping (on découpe simplement par rapport aux faces du cube) et le mapping vers les coordonnées de l'écran (viewport).
- La matrice de projection perspective combine généralement la projection elle-même et la transformation vers l'espace NDC.
- La matrice complète (ex: OpenGL 'glFrustum(l,r,b,t,n,f)') est : $M_{persp\_NDC} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$ où (l, r, b, t) sont les limites du near plane, n=z_{near}, f=z_{far}.

5.2.4 Transformations Géométriques Complètes

Le passage complet des coordonnées d'un objet (object space) aux coordonnées finales de l'écran (screen space) implique la séquence de transformations suivante (appliquée de droite à gauche) : $p_{screen} = M_{viewport} \cdot M_{projection} \cdot M_{view} \cdot M_{model} \cdot p_{object}$ où :

- M_{model} : Transformations de Modélisation (Object → World).
- M_{view} : Transformation de Vue (Caméra) (World → View/Camera Space).
- $M_{projection}$: Projection (Perspective ou Orthographique) (View → Clip Space / NDC).
- $M_{viewport}$: Transformation Viewport (NDC → Screen Coordinates).