Ray-Tracing - Basics

- Mais c'est quoi ?
- Modélisation
- Objets simples dans l'espace
- Matrices et rotation
- Objets plus complexes

Introduction

- La technique de Ray-Tracing permet de créer des images de synthèse avec le maximum de réalisme.

- Cette technique est basée sur une modélisation mathématique des objets que l'on veut représenter.

- L'aspect rapidité de création de l'image n'est pas du tout pris en compte. Certaines images très complexes et très réalistes prennent des heures de calculs (pour les films par ex.).

Modélisation

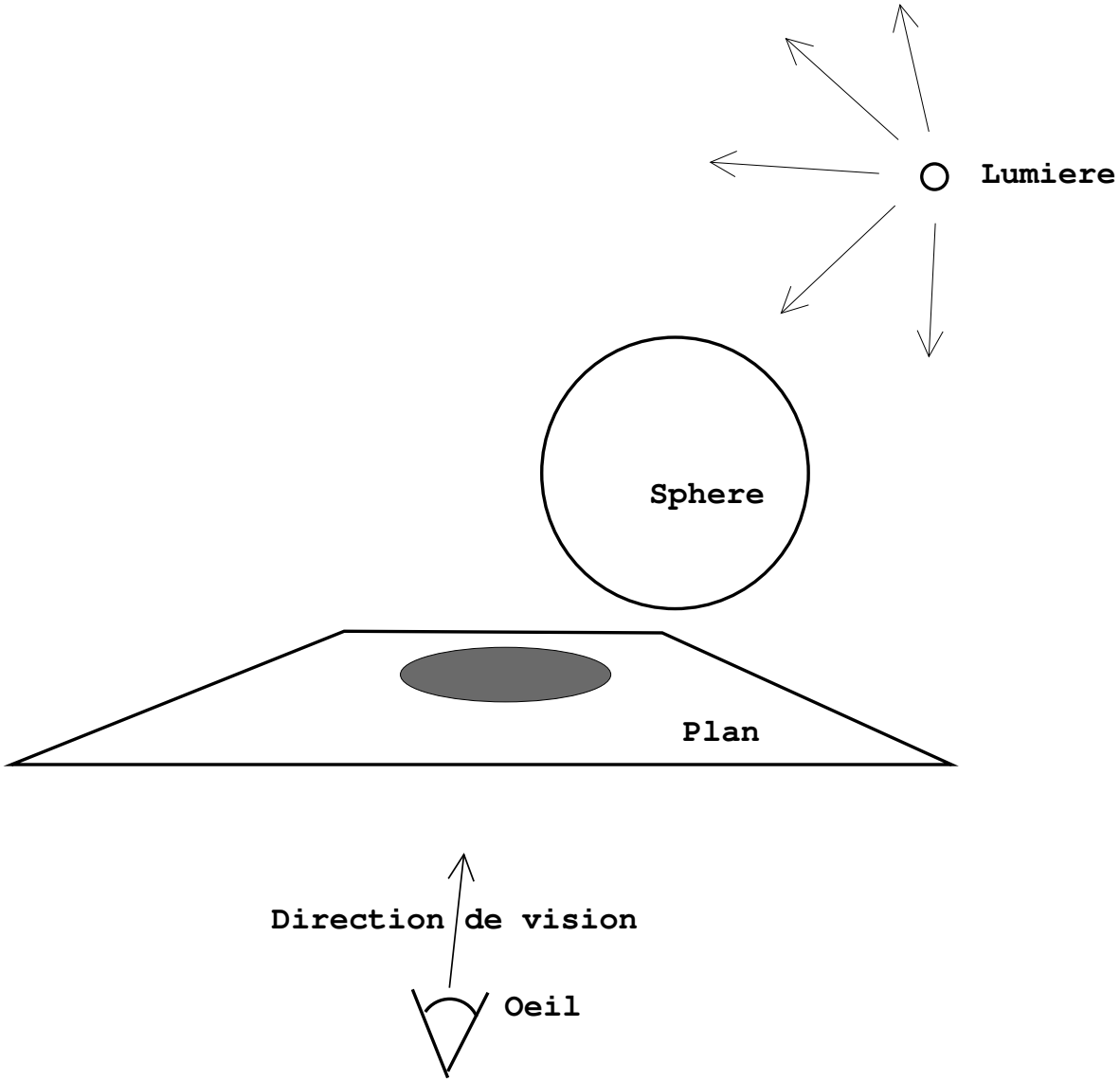
- Pour réaliser notre image, on va tout d'abord modéliser les objets contenus dans cette image. Ces objets seront modélisés au moyen de formes géométriques simples : des sphères, plans, cylindres, cônes ...

- Nous aurons donc une *description de la scène* en termes mathématiques: des coordonnées en 3 dimensions pour la position dans l'espace, des informations sur la taille de nos objets (ex: rayon d'une sphère), sur la rotation ou la limite de nos objets.

- Ensuite c'est au tour des différentes lumières qui éclairent notre scène : la position suffit généralement.

- Il ne nous manque que les détails concernant notre point de vision, notre façon de regarder cette scène: les coordonnées de notre "oeil" ainsi que la direction de vision.

Modélisation de la scène



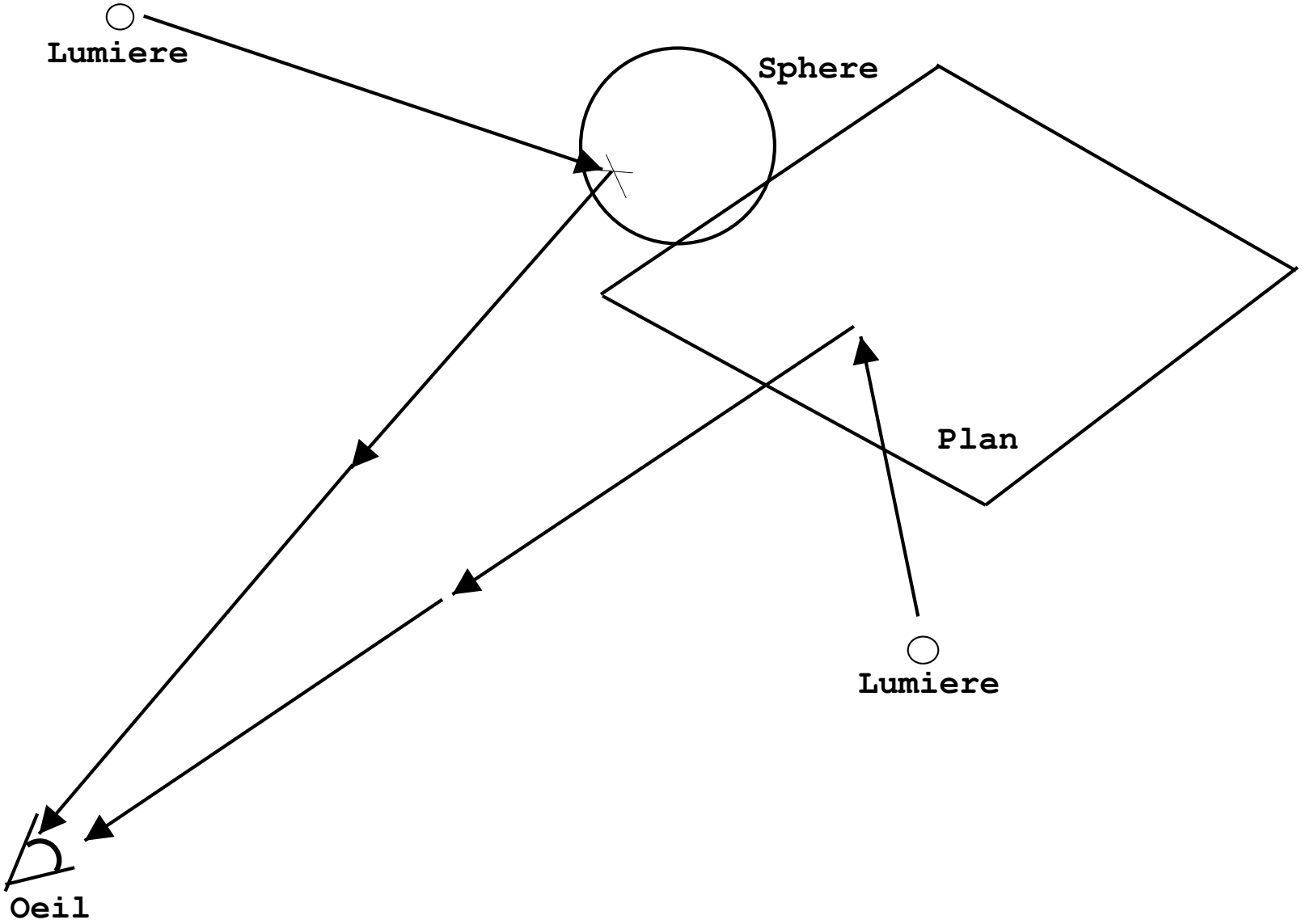
Principe du lancer de rayon

- Les rayons lumineux partent des sources de lumière, dans toutes les directions. Certains "frappent" les objets de notre scène, et d'autres rayons sont alors émis à leur tour dans toutes les directions, dont celle de notre oeil : on voit l'objet.

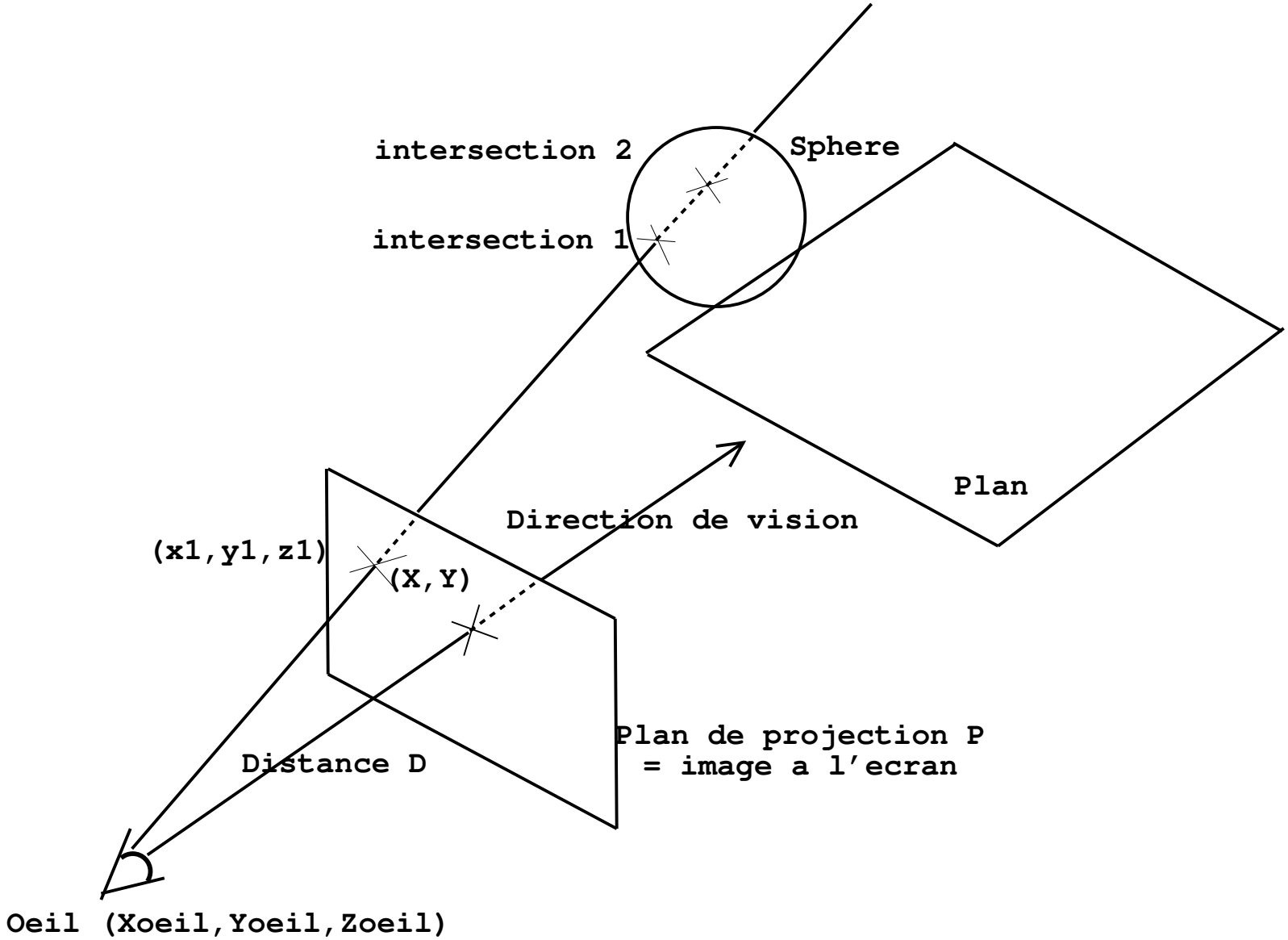
- On ne peut bien sûr pas traiter l'infinité de rayons émis par la source lumineuse. On utilisera le processus inverse: D'où viennent les rayons qui arrivent jusqu'à l'oeil ? De quel objet ? Eclairé par quelles lumières ?

- On reprend le principe du ray-casting : différent rayons passent par l'oeil, et viennent couper un plan de projection P symbolisant notre image. Cette fois-ci, on a autant de rayons que de pixels dans notre image.

Les rayons de l'oeil ...



... à l'envers


intersection 2
Sphere
intersection 1
Plan
Direction de vision
(x1,y1,z1)
(X,Y)
Plan de projection P
= image a l'ecran
Distance D
Oeil (Xoeil,Yoeil,Zoeil)

Un premier cas très simple : la sphère

- Une sphère de rayon $R = 100$ centrée en $(0, 0, 0)$.
- Notre oeil placé en $(-300, 0, 0)$ regardant vers les x positifs.
- La distance oeil $\rightarrow$ plan de projection P : $D = 100$
- Le rapport entre l'image et le plan de projection : 1 .

Raycasting is back

- On cherche l'équation de la droite passant par notre oeil $(x_{oeil}, y_{oeil}, z_{oeil})$ et par $(x1, y1, z1)$ le point du plan de projection correspondant aux coordonnées (X, Y) sur l'image.

- Tout comme pour la technique du raycasting, nous allons replacer notre oeil en $(0, 0, 0)$, regardant vers les x positifs.
On pourra ainsi calculer très facilement $(x1, y1, z1)$, puis on appliquera une rotation et une translation pour mettre l'oeil dans sa position définitive.

- Pour ce premier exemple, nous n'appliquerons que la translation vers la position souhaitée $(-300, 0, 0)$. La rotation viendra avec les matrices un peu plus tard, et permettra de changer la direction de vision.

La droite oeil / pixel ...

- Lorsque notre oeil est dans sa *position simple* (c'est-à-dire qu'il est en $(0,0,0)$ et qu'il regarde vers les x positifs), notre exemple nous donne : $x1 = D = 100$, $y1 = \frac{win_X}{2} - X$ et $z1 = \frac{win_Y}{2} - Y$.

- Le vecteur directeur de la droite passant par notre oeil et le point $(x1, y1, z1)$ est actuellement : $V \left| \begin{matrix} x1 \\ y1 \\ z1 \end{matrix} \right.$ car notre oeil est toujours en $(0,0,0)$.

- C'est maintenant qu'il nous faut appliquer la rotation à V (ou à $(x1, y1, z1)$ c'est pareil) pour que l'oeil puisse regarder dans un autre sens que l'axe des x. Nous verrons plus tard comment faire.

- La translation de notre oeil (on veut le mettre en $(-300, 0, 0)$) n'influence pas la direction de vision, et ne va pas non plus changer le vecteur V.

... en équation

- Nous pouvons donc écrire l'équation paramétrique de notre droite :

$$\begin{cases} x = x_{oeil} + kV_x \\ y = y_{oeil} + kV_y \\ z = z_{oeil} + kV_z \end{cases}$$

- V_x, V_y et V_z sont les coordonnées du vecteur V.

- Cette équation utilise les coordonées définitives de l'oeil. Cela réalise la translation de la position $(0, 0, 0)$ vers $(x_{oeil}, y_{oeil}, z_{oeil})$.

Première intersection

- L'équation de la sphère est la suivante : $x^2 + y^2 + z^2 = R^2$.

- En injectant les 3 formules de l'équation paramétrique de la droite dans l'équation de la sphère, on obtient après un léger calcul :

$$ak^2 + bk + c = 0 \text{ avec } \begin{array}{rcl} a & = & V_x^2 + V_y^2 + V_z^2 \\ b & = & 2(x_{oeil}V_x + y_{oeil}V_y + z_{oeil}V_z) \\ c & = & x_{oeil}^2 + y_{oeil}^2 + z_{oeil}^2 - R^2 \end{array}$$

- On a alors 3 cas :
  $\Delta = b^2 - 4ac < 0$: pas d'intersection droite/sphère.
  $\Delta = 0$: une unique intersection.
  $\Delta > 0$: 2 intersections.
  En pratique on traitera le cas $\Delta = 0$ comme le cas $\Delta > 0$.

A l'écran

- Et bien sûr, $k = \frac{-b \pm \sqrt{\Delta}}{2a}$ représente
la distance entre l'oeil et le point d'intersection avec la sphère.

- Lorsque plusieurs objets de notre scène ont ainsi une intersection avec la droite, on choisit de représenter l'objet le plus proche de notre oeil : parmis tous les k obtenus pour toutes les intersections, on prend le plus petit k non négatif.

- Si il y a intersection, on met donc (pour le moment) la couleur de notre objet aux coordonnées (X, Y) de notre image. Sinon on met du noir (absence de lumière).

Intersection avec un plan

- Nous choisissons le plan d'équation $z = 0$. Le calcul de l'intersection avec la droite associée à (X, Y) nous donne immédiatement : $k = -\frac{z_{oeil}}{V_z}$.

- Si V_z est nul, il n'y a pas d'intersection.

- Il faut alors comparer les différentes valeurs de k résultant des différentes intersections, pour déterminer quel objet est vu, et quelle couleur mettre en (X, Y) .

- *ATTENTION*: en C, pour tester la nullité d'un `float` ou d'un `double`, comparez sa valeur absolue à une valeur arbitraire très petite (0.00001 par exemple).

Détour par les matrices

- Une matrice est une façon de représenter une transformation d'un élément dans un espace à i dimensions en un élément dans un espace à j dimensions.

- Toutes les transformations possibles ne peuvent pas forcément s'exprimer sous forme de matrice.

- Une matrice se représente ainsi :

$$M = \left(\underbrace{\begin{array}{ccccc} m_{11} & m_{21} & m_{31} & \dots & m_{i1} \\ m_{12} & m_{22} & m_{32} & \dots & m_{i2} \\ \dots & & & & \\ m_{1j} & m_{2j} & m_{3j} & \dots & m_{ij} \end{array}}_{i} \right\} j \qquad \text{avec } m_{11}, \dots, m_{ij} \epsilon \Re \ .$$

Les éléments à transformer

- Les éléments d'un espace de dimension n auront eux aussi une représentation sous forme de matrice :

  un point A : $\begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix}$.

  Il s'agit tout simplement des n coordonnées du point.

Multiplication matricielle

- C'est de cette façon que l'on appelle le calcul du nouvel élément à partir du premier élément et de la matrice de transformation. On note cela comme une multiplication : $A' = MA$.

- Principe de la multiplication matricielle :

$$\begin{pmatrix} m_{11} & m_{21} \\ m_{12} & m_{22} \end{pmatrix} \quad \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \quad \begin{pmatrix} a_1 m_{11} + a_2 m_{21} \\ a_1 m_{12} + a_2 m_{22} \end{pmatrix} \quad = A'$$

- On se rend compte que le calcul AM est impossible : la multiplication matricielle n'est pas commutative.

... aussi entre transformations

- Lorsque l'on veut effectuer à un élément 2 transformations successives par la matrice M puis la matrice N, on peut écrire $A' = MA$ puis $A'' = NA'$.
  Cela nous donne $A'' = NMA$ (attention à l'ordre!!)

- Il existe alors une matrice P telle que :
  $A'' = PA$.

- Le calcul de P s'obtient par multiplication matricielle :

$$\begin{pmatrix} n_{11} & n_{21} \\ n_{12} & n_{22} \end{pmatrix} \quad \begin{pmatrix} m_{11} & m_{21} \\ m_{12} & m_{22} \end{pmatrix}$$

$$\begin{pmatrix} n_{11} & n_{21} \\ n_{12} & n_{22} \end{pmatrix} \quad \begin{pmatrix} m_{11}n_{11} + m_{12}n_{21} & m_{21}n_{11} + m_{22}n_{21} \\ m_{11}n_{12} + m_{12}n_{22} & m_{21}n_{12} + m_{22}n_{22} \end{pmatrix} = P$$

En pratique

- Nous serons amenés à ne traiter que le cas point/vecteur 3D $\rightarrow$ point/vecteur 3D.

- Nos matrices de transformation auront donc autant de lignes que de colonnes : ce sont des matrices "carrées".

- On va travailler avec des matrices 3×3 .

- Ce sont essentiellement les 3 matrices de rotation dans l'espace que l'on va utiliser.

- La notion de matrice est très fréquemment utilisée dans les logiciels graphiques du marché.

Les grands classiques

- La matrice "identité" qui laisse nos éléments inchangés :
$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ Ex : } IA = AI = A$$

- Rotation suivant l'axe des x d'un angle θ_x:
$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) \\ 0 & sin(\theta_x) & cos(\theta_x) \end{pmatrix}$$

- Rotation suivant l'axe des y d'un angle θ_y:
$$R_y = \begin{pmatrix} cos(\theta_y) & 0 & sin(\theta_y) \\ 0 & 1 & 0 \\ -sin(\theta_y) & 0 & cos(\theta_y) \end{pmatrix}$$

- Rotation suivant l'axe des z d'un angle θ_z:
$$R_z = \begin{pmatrix} cos(\theta_z) & -sin(\theta_z) & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Revenons à nos moutons

- Pourquoi faire ces matrices ?

- Nous l'avons déjà vu, pour faire tourner notre oeil et donc notre direction de vision.

- Mais allons un peu plus loin.
  Imaginons que l'on souhaite mettre dans notre scène un nouveau plan, mais cette fois-ci un plan penché vers nous et un peu à doite. La logique voudrait que l'on opère de la façon suivante :
  - Recherche de l'équation de notre plan
  - Calcul de l'intersection avec une droite

- Le premier point peut être très fastidieux pour des plans quelconques. Pour le deuxième, il faudra créer une fonction générique capable de calculer les intersections pour des plans et des droites quelconques. De nombreux calculs et cas particuliers en perspective.

Dessine-moi un mouton

- C'etait vraiment plus facile avec notre plan $z = 0$. C'est pourquoi nous allons nous ramener à ce cas là. Tout les plans de l'espace peuvent être décrits comme une rotation puis une translation de notre plan $z = 0$.

- Pour décrire notre scène, un plan sera donc caractérisé par un point d'origine (déterminant une translation par rapport à $(0, 0, 0)$, l'origine du plan $z = 0$) et par 3 angles pour les rotations suivant les 3 axes.

- Mais ces données ne serviront pas à modifier le plan: c'est à dire qu'on ne cherchera pas à trouver une équation de ce plan décrit dans notre scène.
  On veut pouvoir utiliser notre fonction d'intersection avec le plan $z = 0$. Il nous faut revenir dans ce cas la.

Le principe mis en place

- Tous les objets de notre scène seront donc décrits comme des éléments géométriques simples, dans une position "simple" (centrés en $(0,0,0)$, suivant un axe,...), auquels seront appliquées 3 rotations et une translation, pour arriver dans leur position "réelle".

- Tous nos calculs d'intersection concernant une droite de vision et un objet, se feront en ramenant notre objet en position simple, grâce à une translation inverse et 3 rotations inverses.

- Cette série de transformations passant l'objet de sa position réelle à sa position simple devra aussi être appliquée à notre oeil et notre droite de vision. L'objet et l'oeil restent immobiles l'un par rapport à l'autre.

- Les fonctions d'intersection propres à chaque objet sont donc plus simples à trouver et à coder. Les fonctions effectuant les rotations et la translation seront uniques, et utilisées pour tous les objets.

Et le résultat définitif

- Le calcul d'intersection nous donne k la distance entre notre oeil et le point d'instersection P. On obtient aussi les coordonées (x_p, y_p, z_p) de l'intersection.

- Seulement nous sommes toujours en position simple : pour obtenir les vraies coordonnées de l'intersection, il nous faut les transformer par les 3 rotations et la translation de l'objet (pas les transformations inverses cette fois-ci!).

- La distance k ne change pas entre la position simple et la position réelle de l'objet. Pas plus de calculs à faire.

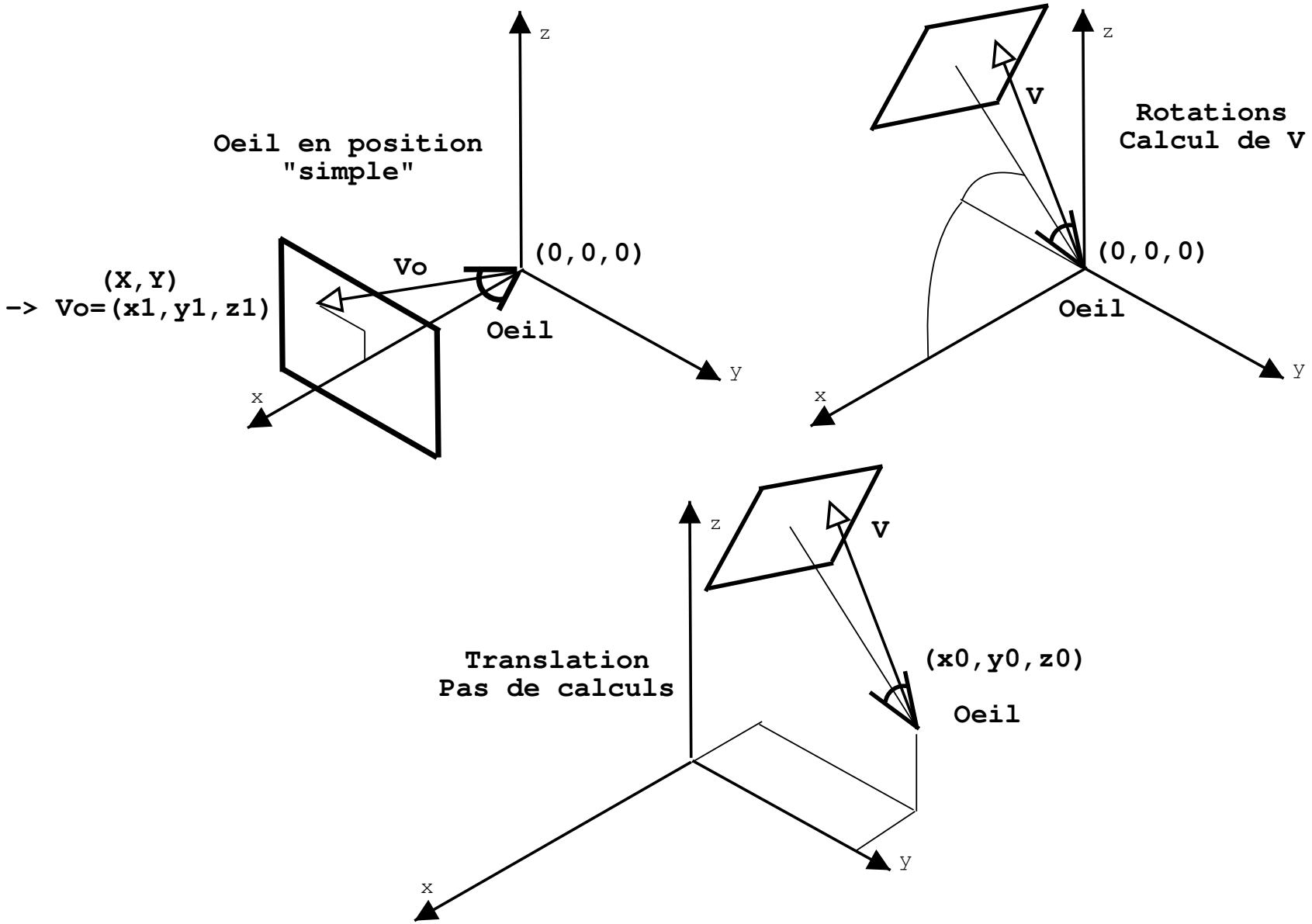
- Lors du calcul de l'intersection avec l'objet suivant de notre scène, ne pas oublier de reprendre les coordonées réelles de l'oeil. Les coordonées temporaires obtenues en mettant un objet en position simple ne sont plus d'aucune utilité.

D'autres objets simples

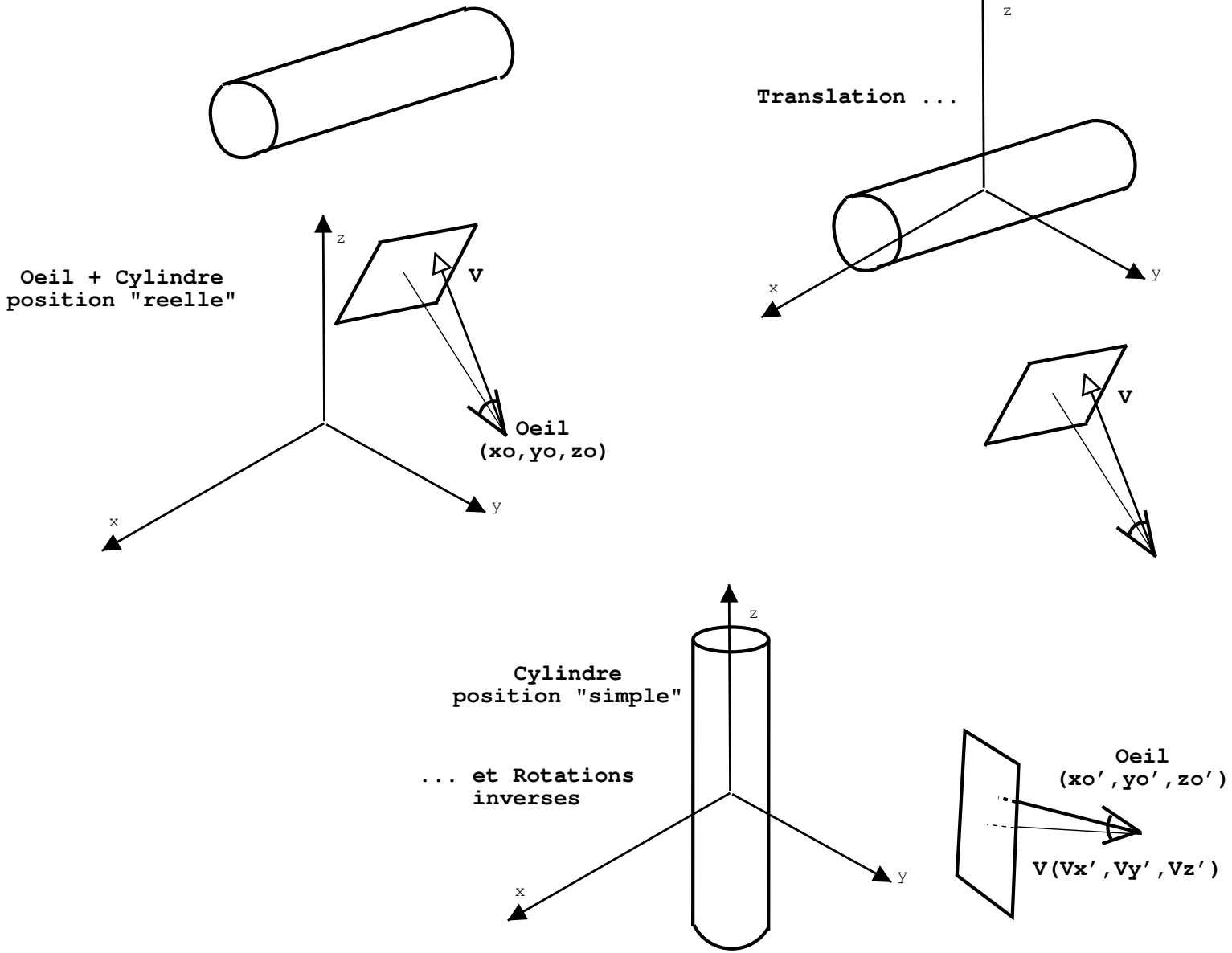
- Le cylindre : comme pour notre sphère ou notre plan, on effectuera rotations et translations pour revenir au cas d'un cylindre centré en $(0,0,0)$ et suivant l'axe des z . Son équation a alors la forme $x^2 + y^2 = R^2$, R étant le rayon du cylindre. Le calcul de l'intersection et de k se fait comme pour la sphère.

- Le cône : on se ramène au cas du cône centré en $(0,0,0)$ suivant l'axe des z. L'équation est de la forme : $x^2 + y^2 - cte \times z^2 = 0$. La cte permet de définir l'ouverture du cône.

Petit RT illustré - 1



Petit RT illustré - 2



Petit RT illustré - 3