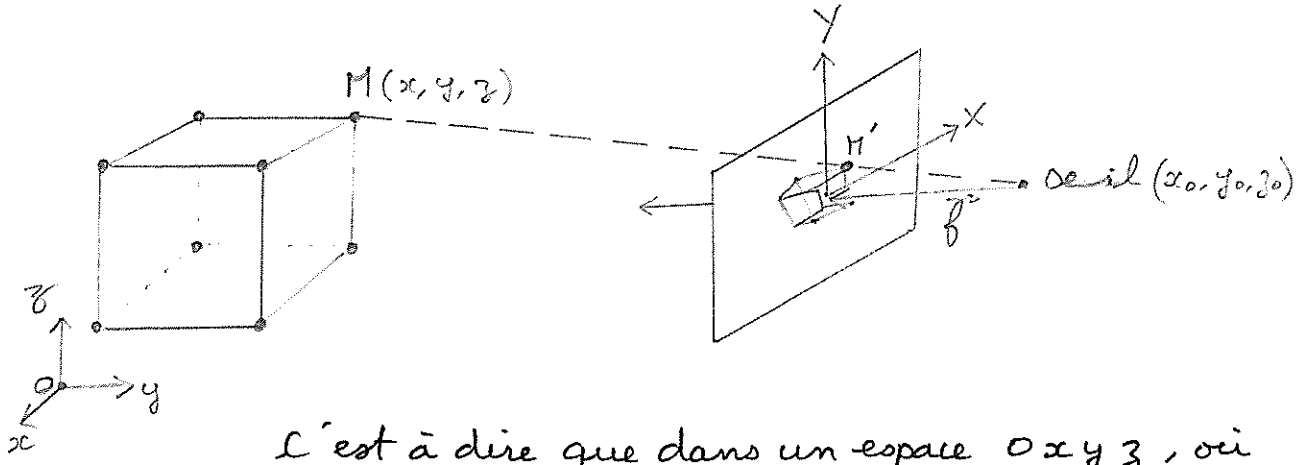
IMAGES à 3 DIMENSIONS
SIMULATEUR de VOL

I) Dans une première partie, il faut créer un logiciel de visualisation à "3-Dimensions" basé sur le schéma très simple suivant :



C'est à dire que dans un espace $Oxyz$, où est défini un objet (ici le cube) par les coordonnées de ses sommets, et où notre oeil est orienté selon une direction $\vec{f}$, On projette un point $M(x,y,z)$ de l'objet sur une plaque photographique située à une distance $\|f\|$ (: "focale", de notre oeil : $M(x,y,z) \longmapsto M'(X,Y)$

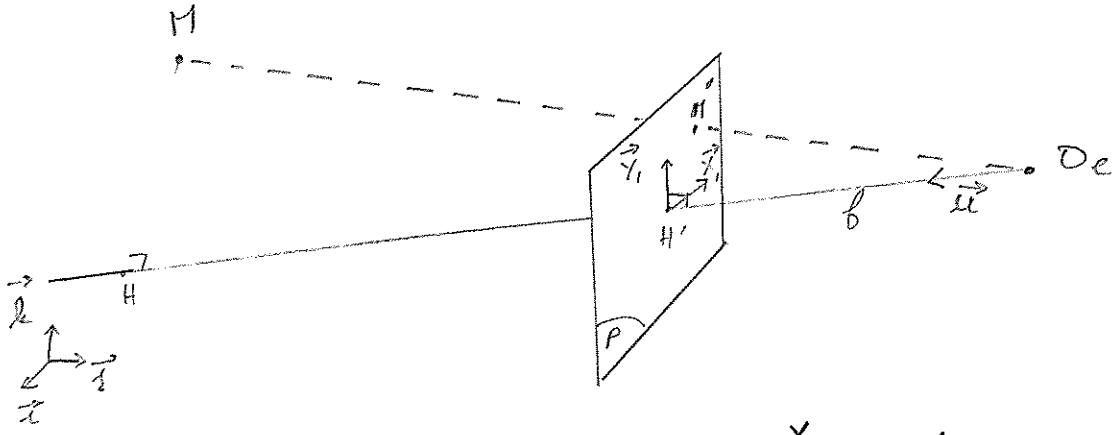
En projetant chaque point de l'objet, on obtient alors une image à deux dimensions en perspective, que l'on peut représenter sur l'écran de l'ordinateur.

II) On peut ainsi visualiser le vol et l'atterrissage d'un avion, d'un parapente ou d'une montgolfière à partir de l'équation de sa trajectoire ou des conditions initiales et de la liste des forces auxquelles l'objet est soumis ; on représente en perspective, le paysage d'arrivée.

I) Formules de projection de l'objet sur la "plaque photo"

Dans un repère $O\,\vec{i}\,\vec{j}\,\vec{k}$, on considère un point M de coordonnées (x, y, z) à projeter sur notre plaque photo caractérisée par :

- l'oeil O_e de coordonnées $(x_0\ y_0\ z_0)$
- le vecteur : direction de visée $\vec{u}$ $(u_x\ u_y\ u_z)$ (unitaire)
- la focale : f



Le but est de trouver les coordonnées $\begin{matrix} X \\ Y \end{matrix}$ de M' projeté de M sur la plaque photo P (: plan $\perp$ à $\vec{u}$, à la distance f de O_e) dans le repère o.n. $H'\vec{X_1}\,\vec{Y_1}$ avec $\vec{X_1}$ vecteur de P et horizontal.

Il faut aussi trouver un critère simple pour savoir si le point M est visible c'est à dire s'il est devant nous.

- Exprimer X et Y en fonction des données :

  $x_0\ y_0\ z_0$, $u_x\ u_y\ u_z$, f et $x\ y\ z$

  (par plusieurs formules mises sous la forme la plus compacte possible)

- En déduire l'ordonnée Y_h de la ligne horizon.

Ce qui donne :

$$O_eH = \overrightarrow{O_eM} \cdot \vec{u} = u_x(x-x_0) + u_y(y-y_0) + u_z \cdot (z-z_0)$$

$$\vec{u} \wedge \vec{k} = \begin{vmatrix} u_x \\ u_y \\ u_z \end{vmatrix} \wedge \begin{vmatrix} 0 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} u_y \\ -u_x \\ 0 \end{vmatrix}$$

$$\vec{X_1} = \frac{1}{\sqrt{u_x^2+u_y^2}} \begin{vmatrix} u_y \\ -u_x \\ 0 \end{vmatrix} \qquad \vec{Y_1} = \vec{X_1} \wedge \vec{u} = \frac{1}{\sqrt{u_x^2+u_y^2}} \begin{vmatrix} -u_x u_z \\ -u_y u_z \\ u_y^2 + u_x^2 \end{vmatrix}$$

$$X' = \vec{X_1} \cdot \overrightarrow{O_eM} = \frac{1}{\sqrt{u_x^2+u_y^2}} \cdot \left(u_y(x-x_0) - u_x(y-y_0)\right)$$

$$Y' = \vec{Y_1} \cdot \overrightarrow{O_eM} = \frac{1}{\sqrt{u_x^2+u_y^2}} \cdot \left(-u_x u_z (x-x_0) - u_y u_z (y-y_0) + (u_x^2+u_y^2)(z-z_0)\right)$$

$$X = X' \cdot f / O_eH \qquad Y = Y' \cdot f / O_eH$$

Ou sous forme plus compacte :

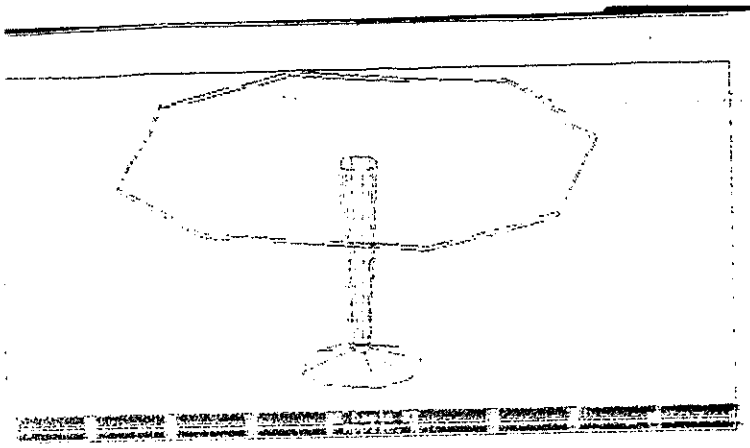
$$\boxed{\begin{aligned} c_1 &= u_x^2 + u_y^2 \\ x_2 &= x - x_0 \qquad y_2 = y - y_0 \qquad z_2 = z - z_0 \\ c_2 &= u_x x_2 + u_y y_2 \\ c_3 &= f / (c_2 + u_z z_2) \\ X &= (u_y x_2 - u_x y_2) \cdot c_3 / \sqrt{c_1} \\ Y &= (c_1 z_2 - u_z c_2) \cdot c_3 / \sqrt{c_1} \end{aligned}}$$

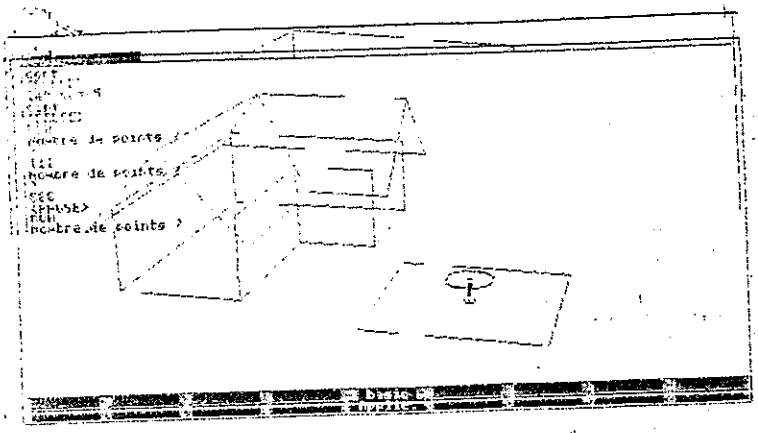
avec c_1 c_2 c_3 variables intermédiaires

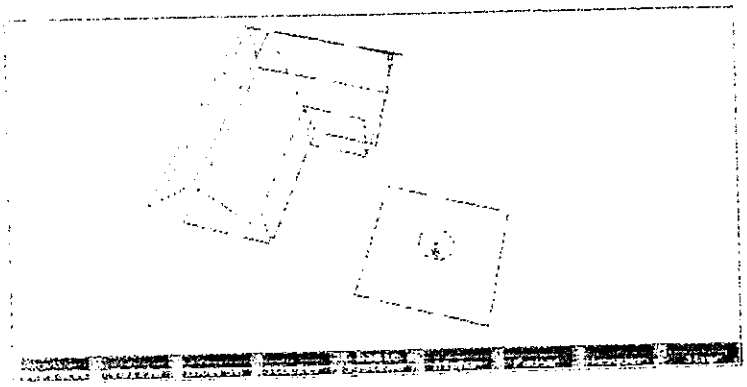
et pour la ligne d'horizon, en faisant $x, y \gg z$, il vient :

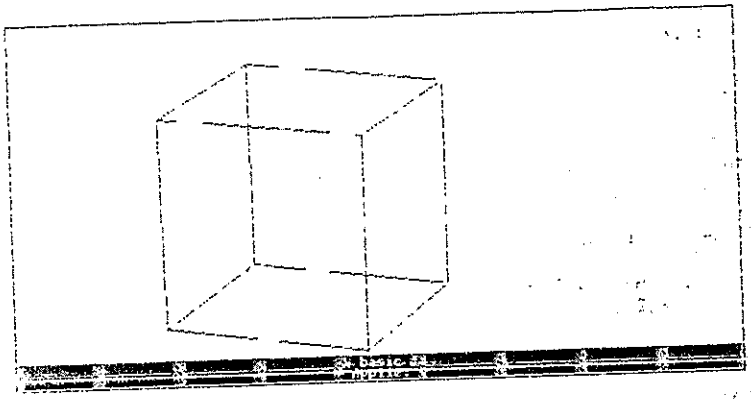
$$\boxed{Y_{horiz} \approx -\frac{u_z \cdot f}{\sqrt{u_x^2+u_y^2}}}$$

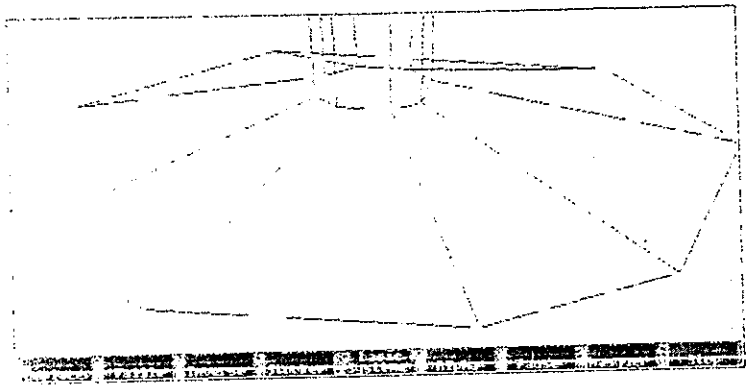
et le critère M est devant nous, si $\boxed{c_3 > 0}$

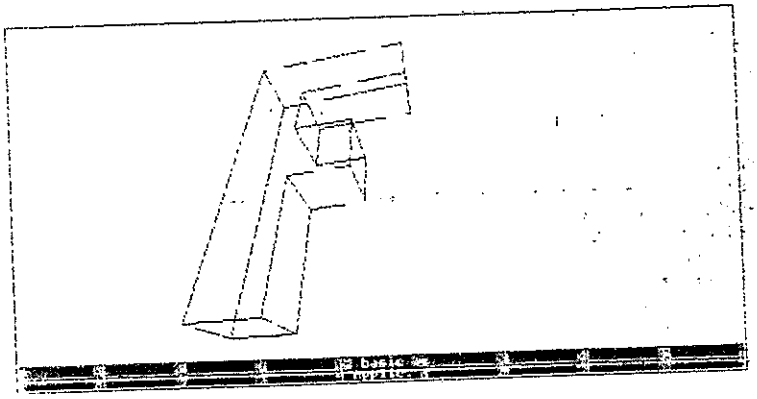












exemples d'images à 3D

- une maison et sa terrasse
  sous divers angles et différents grossissements
- un F
- un cube

Structure générale du programme

Les différentes procédures

initialisation : initialise et définit les objets du paysage, en donnant leurs coordonnées (x, y, z)

calcul : routine qui, étant donné le lieu d'observation et le point $(x\ y\ z)$ à observer, retourne la position X Y sur l'écran et une variable test

affichage : routine qui dessine le paysage sur l'écran étant donné le lieu d'observation (utilise <u>calcul</u>)

position : routine qui se charge de l'évolution du lieu d'observation.

c'est à dire étant donné un instant t,

la position de l'oeil $(x_0\ y_0\ z_0)$

la direction de visée $(u_x\ u_y\ u_z)$

la vitesse $(v_x\ v_y\ v_z)$

la masse m

retourne à l'instant $t + \Delta t$

ces nouvelles valeurs.

Programme

a) écrire une routine qui applique les formules suivantes. C'est à dire :

<u>en entrée</u> : $x\ y\ z$: coordonnées de M

$u_x\ u_y\ u_z$: coordonnées de $\vec{u}$ unitaire

f : focale

$x_0\ y_0\ z_0$: coordonnées de notre oeil O_e

<u>en sortie</u> : $X\ Y$: coordonnées sur l'image de M'

test : variable booléenne :
vrai : si M est devant nous et sur l'écran ($|X| < X_{max}$ et $|Y| < Y_{max}$)

faux : sinon

b) définir quelques objets simples (cube, maison ... piste d'atterrissage ...) par les coordonnées de leurs sommets ($x\ y\ z$) et les dessiner à l'aide de la routine précendente.

Pour cela on considérera chaque couple M_1, M_2 définissant une arête de l'objet et on calculera $X_1 Y_1 ; X_2 Y_2$. On tracera alors sur l'écran la droite entre $\binom{X_1}{Y_1}$ $\binom{X_2}{Y_2}$ si ces deux points sont bien sur l'écran.
(cette partie b) sera plus détaillée par la suite)

les différentes variables utilisées et leur type

nb_ob : integer : nombre d'objets dans le paysage (≥ 1)

nb_pt [ob] : tableau d'entiers : nombre de points de l'objet n° ob ($1 \leq ob \leq nb_ob$)

nb_dt [ob] : tableau d'entiers : nombre de droites dans l'objet n° ob

- Ensuite pour l'objet n° ob et le point n° pt
  avec $1 \leq ob \leq nb_ob$ et $1 \leq pt \leq nb_pt[ob]$

point [ob, pt]
- . x
- . y
- . z

} réels : position du point dans l'espace

- . X
- . Y

} réels : position du point sur l'écran

- . test } booléenne : oui si (X, Y) est sur l'écran

c'est donc un tableau d'enregistrements

- Et de même pour la droite n° dt
  avec $1 \leq dt \leq nb_dt[ob]$

droite [ob, dt]
- . i
- . f

} entiers : n° du point initial : n° du point final

Algorithme de la procédure Affichage

Pour ob = 1 à nb_ob

/* dessin de l'objet n° ob */

Pour pt = 1 à nb_pt [ob]

/* calcul des points de l'objet */

point [ob, pt] · X
Y
test } = résultat de la procédure calcul (x, y, z, u_x u_y u_z ...)

Pour dt = 1 à nb_dt [ob]

/* dessin de la droite n° dt */

affiche la droite si les points initiaux et finaux sont sur l'écran

La procédure d'initialisation

doit initialiser les variables :

nb_ob

nb_pt [ ]

nb_dt [ ]

point [·, ·] · x
· y
· z

droite [·, ·] · i
· f