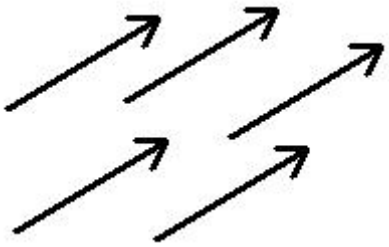
Géométrie classique

Vecteurs

Un vecteur, qui se dessine sous forme d'une flèche, est défini par trois choses :

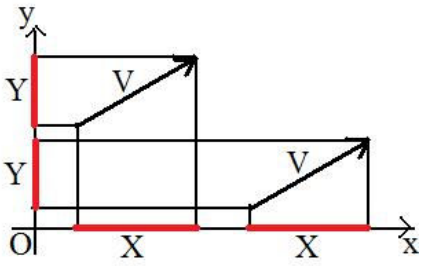
-sa direction, ce qui signifie qu'il est parallèle à une certaine droite[1]
-son sens, car on peut aller dans un sens ou dans un autre sur une droite
-sa longueur

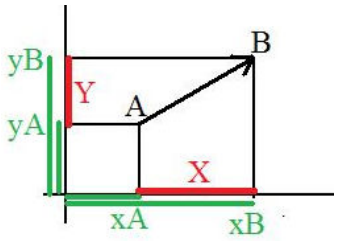
Il n'a pas d'origine précise, aussi peut-il être dessiné d'une infinité de façons. Puisqu'on a le choix, on donnera au vecteur l'origine (le point de départ de la flèche) qui se prête le mieux à nos besoins.



Un vecteur ***V***, ou si l'on veut plusieurs vecteurs égaux, ayant tous même direction, même sens, même longueur.

Dans un repère orthonormé *Oxy* en deux dimensions, le vecteur ***V*** est déterminé par ses deux coordonnées *X*, *Y*, qui sont des nombres positifs ou négatifs, comme indiqué sur le dessin





Si l'on donne au vecteur ***V*** un point origine *A* (*xA*, *yA*) et une extrémité *B*(*xB*, *yB*), on a aussi comme coordonnées du vecteur ***V*** = ***AB*** :

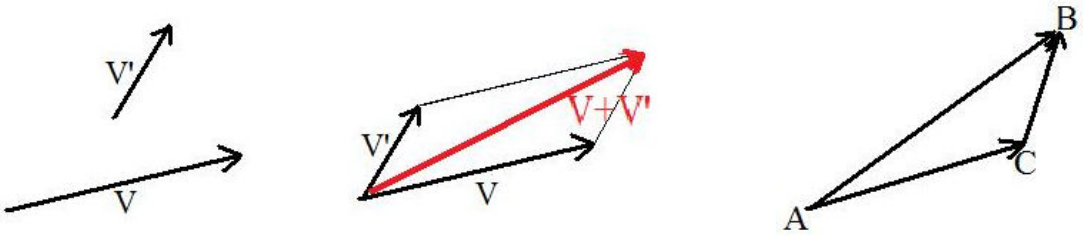
$X = xB - xA$
$Y = yB - yA$

On définit ensuite deux opérations sur les vecteurs :

- L'addition : la somme de deux vecteur est définie par la règle du parallélogramme, en donnant aux deux vecteurs la même origine, ou encore par la

[1] Dans le langage courant, le mot « direction » a une signification différente, il sous-entend non seulement le parallélisme à une droite, mais aussi un sens, par exemple quand on dit que l'on se dirige vers le nord-ouest.

formule de Chasles, soit $\boldsymbol{AB} = \boldsymbol{AC} + \boldsymbol{CB}$, où le point C est intercalé entre A et B, cela restant valable où que soit le point C. [2]



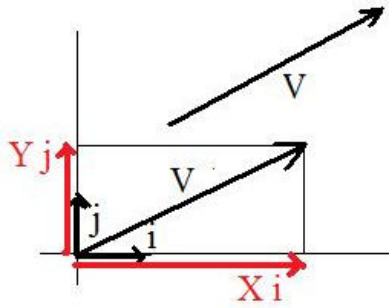
- La multiplication par un nombre : le vecteur $a\boldsymbol{V}$ a la même direction que le vecteur V, le même sens si a est positif, le sens contraire si a est négatif, et sa longueur est $|a|$ fois celle de $\boldsymbol{V}$.

Notamment le vecteur $-\boldsymbol{V}$ a la même direction, le sens contraire, et la même longueur que $\boldsymbol{V}$, il est appelé l'opposé de V. Cela permet de définir la soustraction de deux vecteurs : soustraire, c'est additionner l'opposé, soit

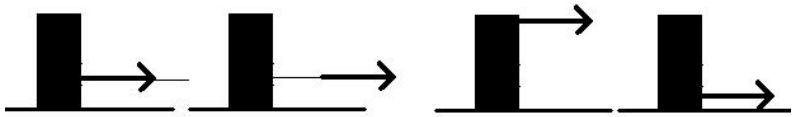
$$\boldsymbol{V} - \boldsymbol{V'} = \boldsymbol{V} + (- \boldsymbol{V'})$$

Prenons les vecteurs unitaires $\boldsymbol{i}$ et $\boldsymbol{j}$ du repère Oxy. Grâce aux opérations que nous venons de définir, on peut écrire, pour le vecteur $\boldsymbol{V}$ (X, Y)

$$\boldsymbol{V} = X\,\boldsymbol{i} + Y\,\boldsymbol{j}$$ [3]



[2] En physique une force obéit à ces mêmes propriétés. Mais c'est plus qu'un vecteur. On ne peut pas prendre son origine n'importe où, comme on le fait pour un vecteur. On peut simplement la déplacer sur une ligne (par exemple en tirant avec une corde plus ou moins longue), Par contre si on change sa ligne d'application, ce n'est plus la même force, et elle n'a plus les mêmes effets. Il se rajoute un effet de couple, de moment, d'où la notion de torseur.



Même force, mêmes effets

Deux forces différentes, *à gauche* l'armoire bascule, *à droite*, elle glisse sur le sol

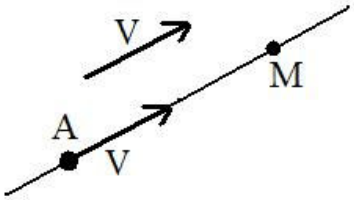
[3] Cette propriété caractéristique d'un vecteur vaut en fait dans tout repère, pas forcément orthonormé, et même dans une base, définie par les seuls vecteurs $\boldsymbol{i}$ et $\boldsymbol{j}$ non parallèles et non nuls (sans utiliser de point comme O). On a aussi, dans le repère Oxy, avec un point M de coordonnées (x, y) :
$\boldsymbol{OM} = x\,\boldsymbol{i} + y\,\boldsymbol{j}$.

Droite

Plaçons-nous dans un repère orthonormé en deux dimensions (mais cela se généraliserait en trois dimensions, ou plus). Une droite est définie par un point $A(x_0, y_0)$ et par un vecteur directeur V (a, b) non nul. Il n'existe en effet qu'une droite parallèle à $\boldsymbol{V}$ et passant par A. Un point $M(x, y)$ appartient à la droite si et seulement si $\boldsymbol{AM} = k\,\boldsymbol{V}$, où k est un nombre réel. Par projection sur les axes, cela donne $x - x_0 = k\,a$, $y - y_0 = k\,b$, soit :

$x = x_0 + k\,a$
$y = y_0 + k\,b$

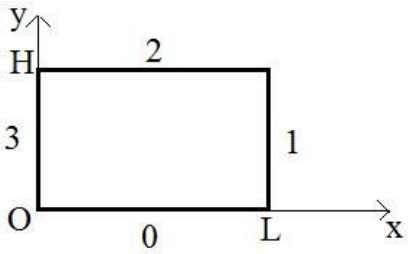
Cela s'appelle les équations paramétriques de la droite (k étant le paramètre). Pour chaque valeur de k, on a une position du point M. Lorsque k décrit l'ensemble des nombres réels, le point M décrit la droite. Plus précisément, lorsque k augmente à partir de 0, le point M se déplace à partir de A dans le sens de $\boldsymbol{V}$, et inversement pour k négatif, il se déplace dans le sens opposé de $\boldsymbol{V}$. Si k augmente régulièrement, le point M se déplace à vitesse constante. Plus qu'une droite, on obtient un rayon, à l'image d'un rayon lumineux partant de A.



Appliquons immédiatement cela à un jeu de billard.

Billard rectangulaire

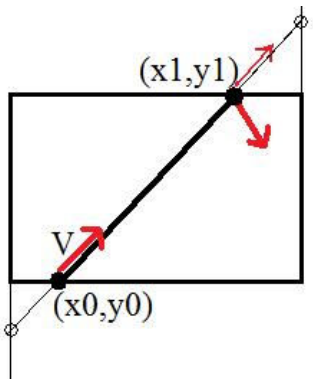
Une boule roule sur un billard rectangulaire de longueur L et de largeur H. On se place dans des conditions idéales. La boule est lancée à partir d'une bordure, elle avance en ligne droite, puis elle rebondit sur une autre bordure, et cela indéfiniment, à vitesse constante, car on néglige tout phénomène de frottement et d'effets sur la boule.



Notons les bordures 0, 1, 2, 3 comme sur le dessin. Dans le repère Oxy les droites correspondantes ont pour équation respective $y = 0$ (tous les points de la bordure 0 ont une ordonnée nulle), $y = H$, $x = 0$ et $x = L$.

On part d'un point ($x0$, $y0$) situé sur la bordure 0 ($bord0 = 0$ dans le programme) dans une direction donnée par un vecteur $\boldsymbol{V}$ (vx, vy). L'objectif est de trouver le point ($x1$, $y1$) où la boule va rebondir, et notamment quelle est la bordure (notée $bord1$) concernée. Cela permet de tracer la trajectoire de la boule entre les points ($x0$, $y0$ et ($x1$, $y1$). Puis on détermine le nouveau vecteur après réflexion de la boule, ce qui revient à changer soit son abscisse soit son ordonnée. Puis on actualise : le point ($x1$, $y1$) est remplacé par ($x0$, $y0$) et on cherche de la même façon qu'avant le nouveau point de

rebond (*x1*, *y1*). Dans le programme, la boucle des rebonds fait chaque fois passer du point (*x0*, *y0*) au point (*x1*, *y1*).



La droite issue du point (*x0*, y0) dans la direction donnée par ***V*** a pour équations paramétriques

$x = x0 + k\ vx,\ \ y = y0 + k\ vy.$

Le problème est de savoir où se fait le nouveau rebond. Il s'agit d'abord de déterminer quelle est la bordure concernée. Pour cela on détermine les valeurs de *k* correspondant aux intersections possibles de la droite et des bordures. Par exemple, le point d'intersection avec la bordure 1 ($y = H$) est tel que

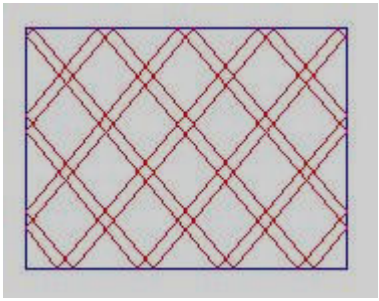
$H = y0 + k\ vy$, d'où $k = (H - y0) / vy$.

On trouve ainsi quatre valeurs de *k* que l'on place dans un tableau *k*[4]. Parmi ces quatre valeurs, on élimine celle de la bordure d'où l'on part, on élimine aussi les bordures ayant un *k* négatif (on est hors du billard) et on garde parmi les valeurs positives la valeur de *k* la plus petite. On connaît ainsi la bordure concernée, ainsi que le point du nouveau rebond (par exemple si c'est la bordure 1, on a

$y1 = H$, et $x1 = x0 + k[1]\ vx$.

Le programme s'ensuit.

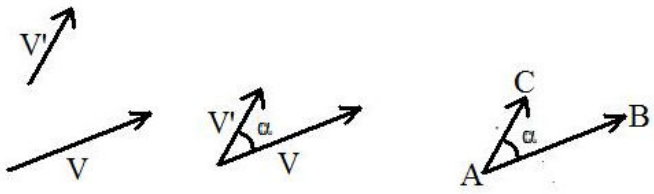
```
On se donne L et H, par exemple L = 2. et H = 1.5
ligne(xo,yo,xo+zoom*L,yo,bleu);  /* dessin des quatre bordures du billard  */
ligne(xo,yo,xo,yo-zoom*H,bleu);
ligne(xo,yo-zoom*H,xo+zoom*L,yo-zoom*H,bleu);
ligne(xo+zoom*L,yo,xo+zoom*L,yo-zoom*H,bleu);

bord0=0; x0=0.4*L; y0=0;
vx=1.; vy=1.3;        /* position et vitesse initiale de la boule */

for(etape=1; etape <1000; etape++)  /* un millier de rebonds  */
{  k[0]=-y0/vy; k[1]=(H-y0)/vy; k[2]=-x0/vx; k[3]=(L-x0)/vx;
   kmini=100000;
   for(i=0;i<4;i++) if (i!=bord0) if (k[i]>0.)
   if(k[i]<kmini) {kmini=k[i]; imini=i;}   /* le k positif minimal */
   bord1=imini;
   if(bord1==0)  {y1=0; x1=x0+k[bord1]*vx;  vy=-vy;}
   else if(bord1==1)  { y1=H; x1=x0+k[bord1]*vx;  vy=-vy;}
   else if(bord1==2)  { x1=0; y1=y0+k[bord1]*vy;  vx=-vx;}
   else if(bord1==3)  { x1=L; y1=y0+k[bord1]*vy;  vx=-vx;}
   xe0=xo+zoom*x0; ye0=yo-zoom*y0;  xe1=xo+zoom*x1; ye1=yo-zoom*y1;
   ligne(xe0,ye0,xe1,ye1,rouge);
   bord0=bord1; x0=x1; y0=y1;
}
```



Produit scalaire

Il s'agit d'une nouvelle opération, la multiplication de deux vecteurs, sauf que le résultat n'est pas un vecteur mais un nombre (d'où le nom de « scalaire »). Par définition le produit scalaire de deux vecteurs $\boldsymbol{V}$ et $\boldsymbol{V'}$ est le produit de leurs longueurs par le cosinus de l'angle (orienté ou non) qui les sépare, ce qui s'écrit :

$\boldsymbol{V}\,\boldsymbol{V'} = \|\boldsymbol{V}\|\,\|\boldsymbol{V'}\| \cos(\boldsymbol{V}, \boldsymbol{V'})$ ou encore
$\boldsymbol{AB} \,.\, \boldsymbol{AC} = AB\,AC \cos\alpha$

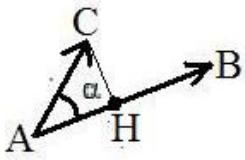


Que l'angle soit orienté ou non est sans importance, sachant que $\cos(-x) = \cos x$. On a aussi $\boldsymbol{V}\,\boldsymbol{V'} = \boldsymbol{V'}\,\boldsymbol{V}$.

On déduit de la définition que le produit scalaire est nul si et seulement si l'angle est droit (ou qu'un vecteur est nul). Si l'angle (non orienté) est aigu, entre 0 et 90°, le produit scalaire est positif, puisque $\cos\alpha \geq 0$, et si l'angle est obtus, entre 90 et 180°, le produit scalaire est négatif.

Maintenant projetons un vecteur sur l'autre, par exemple C se projette en H, on obtient une autre définition du produit scalaire :

$\boldsymbol{AB}\,\boldsymbol{AC} = \overline{AB}\;\overline{AH}$
où $\overline{AB}$ et $\overline{AH}$ sont des mesures algébriques (des longueurs avec un signe en plus sur la droite orientée AB). [4]



Lorsqu'on va dans le même sens de A vers B et de A vers H, le produit $\overline{AB}\;\overline{AH}$ est positif, et si c'est dans le sens contraire, le produit est négatif. Cela correspond au fait que l'angle de $\boldsymbol{AB}$ avec $\boldsymbol{AC}$ est aigu ou obtus. Avec en plus $AH = AC\ |\cos\alpha|$, on a bien une définition équivalente à la précédente.

En physique, le produit scalaire correspond au travail d'une force, le travail étant le produit de la force par la projection du déplacement sur la force. Si le produit est positif,

[4] On a aussi $\boldsymbol{AB}\,\boldsymbol{AC} = \boldsymbol{AB}\,\boldsymbol{AH}$ en vecteurs.

le travail est moteur, et s'il est négatif, le travail est résistant. Et le travail est nul lorsque la force est perpendiculaire au déplacement.

Enfin dans un repère orthonormé, avec $\boldsymbol{V}$ de coordonnées (X, Y), et $\boldsymbol{V}'$ (X', Y') le produit scalaire s'écrit :

$\boldsymbol{V}\,\boldsymbol{V}' = X\,X' + Y\,Y'$ [5]

Cela permet de calculer des angles : si l'on connaît les vecteurs $\boldsymbol{V}$ et V' par leurs coordonnées, on peut calculer leur longueur ($\|\boldsymbol{V}\|^2 = X^2 + Y^2$), puis leur produit scalaire $XX' + YY'$, et l'on en déduit le cosinus de leur angle par $\cos\alpha = \boldsymbol{V}\,\boldsymbol{V}' / (\|\boldsymbol{V}\|\,\|\boldsymbol{V}'\|)$, ce qui permet d'avoir l'angle non orienté α. Notamment si les deux vecteurs ont pour longueur 1, on a $\cos\alpha = \boldsymbol{V}\,\boldsymbol{V}'$. Et si le produit scalaire est nul, l'angle est droit.

Equation d'un plan

Ce que nous avons fait en deux dimensions vaut aussi en trois dimensions pour le produit scalaire. Dans l'espace en trois dimensions muni d'un repère orthonormé $Oxyz$, un plan est défini par un point $A(x_0, y_0, z_0)$ et un vecteur $\boldsymbol{N}$ (a, b, c) non nul qui est perpendiculaire au plan, et que l'on appelle vecteur normal. Il existe n effet un plan unique passant par A et perpendiculaire à $\boldsymbol{N}$. Un point $M(x, y, z)$ appartient au plan si et seulement si $\boldsymbol{AM}$ est perpendiculaire à $\boldsymbol{N}$, ce qui signifie que $\boldsymbol{AM}\,.\,\boldsymbol{N} = 0$. Avec $\boldsymbol{AM}(x - x_0, y - y_0, z - z_0)$, cela s'écrit

$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0$. C'est l'équation du plan : tout point M dont les coordonnées vérifient cette équation est dans le plan.

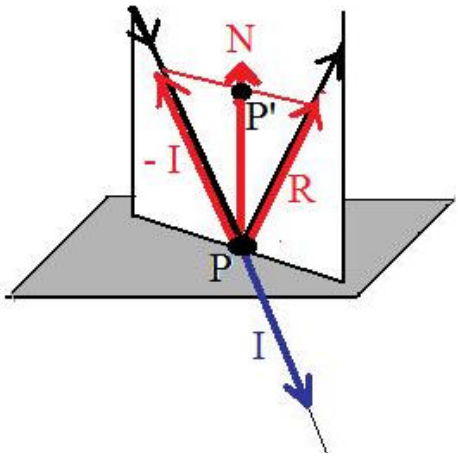
Réflexion lumineuse sur une facette plane

Considérons un plan correspondant à un miroir sur lequel la lumière se réfléchit. Ce plan admet un vecteur normal $\boldsymbol{N}$ auquel on donne la longueur 1. Prenons un rayon lumineux incident de vecteur directeur $\boldsymbol{I}$ de longueur 1 et qui touche le miroir en P. Le rayon se réfléchit avec un vecteur directeur $\boldsymbol{R}$ de longueur 1 aussi. Ce vecteur $\boldsymbol{R}$ est situé dans le plan passant par P et contenant $\boldsymbol{I}$ et $\boldsymbol{N}$, avec un angle de réflexion égal à l'angle d'incidence. Comment avoir $\boldsymbol{R}$ à partir de $\boldsymbol{I}$ et de $\boldsymbol{N}$?

Commençons par prendre le vecteur $-\boldsymbol{I}$. Les deux vecteurs $-\boldsymbol{I}$ et $\boldsymbol{R}$ sont symétriques par rapport à l'axe contenant $\boldsymbol{N}$. Avec leur origine commune P, leurs extrémités se projettent en un même point P' sur $\boldsymbol{N}$, et l'on a $-\boldsymbol{I} + \boldsymbol{R} = 2\,\boldsymbol{PP}'$, ou $\boldsymbol{R} = \boldsymbol{I} + 2\,\boldsymbol{PP}'$. D'autre part, grâce au produit scalaire $-\boldsymbol{I}\,\boldsymbol{N} = \overline{PP'}\,.\,1 = \overline{PP'}$, en se plaçant sur la droite orientée par $\boldsymbol{N}$. Puisque $\boldsymbol{N}$ a pour longueur 1, on en déduit que $\boldsymbol{PP}' = \overline{PP'}\,\boldsymbol{N} = (-\boldsymbol{I}\,\boldsymbol{N})\,\boldsymbol{N}$. Finalement

$$\boldsymbol{R} = \boldsymbol{I} - 2\,(\boldsymbol{I}\,\boldsymbol{N})\,\boldsymbol{N}$$

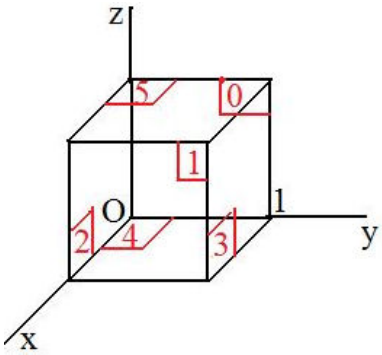
[5] On doit commencer par démontrer que $\boldsymbol{V}\,(\boldsymbol{V}' + \boldsymbol{V}'') = \boldsymbol{V}\boldsymbol{V}' + \boldsymbol{V}\boldsymbol{V}''$. Puis on applique cette formule en repère orthonormé ayant pour base les vecteurs $\boldsymbol{i}$ et $\boldsymbol{j}$ perpendiculaires et de longueur 1 : $\boldsymbol{V}\,\boldsymbol{V}' = (X\boldsymbol{i} + Y\boldsymbol{j})\,(X'\boldsymbol{i} + Y'\boldsymbol{j}) = XX'\,\boldsymbol{ii} + XY'\,\boldsymbol{ij} + YX'\,\boldsymbol{ji} + YY'\,\boldsymbol{jj}$. Avec $\boldsymbol{ii} = \boldsymbol{jj} = 1$, et $\boldsymbol{ij} = 0$, il reste $XX' + YY'$.



Appliquons cela à de nouveaux problèmes de billard

Billard cubique

On considère un cube de côté 1. Ses faces sont numérotées 0 (plan d'équation $x = 0$), 1 (face où $x = 1$), 2 ($y = 0$), 3 ($y = 1$), 4 ($z = 0$), 5 ($z = 1$), dans le repère *Oxyz* comme indiqué sur le dessin. Les six faces sont constituées de miroirs à l'intérieur du cube. Un rayon lumineux est lancé à partir d'un point donné (x_0, y_0, $z_0 = 0$) de la face 4 suivant la direction orientée par un vecteur ***V*** (*vx*, *vy*, *vz)* donné. Ce rayon se réfléchir indéfiniment sur les faces du cube, et c'est ce que l'on veut programmer pour voir les réflexions sur l'écran d'ordinateur.



- Préparatifs

On commence par donner au vecteur ***V*** une longueur 1 en le divisant par sa longueur. Puis on prend les vecteurs normaux $\boldsymbol{N}_i$ (nx_i, ny_i, nz_i) aux plans des faces en les orientant vers l'intérieur, et en leur donnant une longueur unité. Cela donne par exemple :

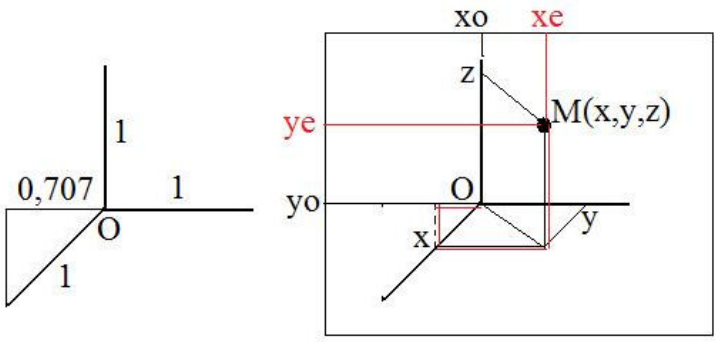
```
x0= 0.43; y0=0.57;  z0=0.;       vx=1.511; vy=1.322; vz=1.833;
longV=sqrt(vx*vx+vy*vy+vz*vz);
vx=vx/longV; vy=vy/longV; vz=vz/longV; face=4;

nx[0]=1.;ny[0]=0.;nz[0]=0.;  nx[1]=-1.;ny[1]=0.;nz[1]=0.;
nx[2]=0.;ny[2]=1.;nz[2]=0.;  nx[3]=0.;ny[3]=-1.;nz[3]=0.;
nx[4]=0.;ny[4]=0.;nz[4]=1.;  nx[5]=0.;ny[5]=0.;nz[5]=-1.;
```

A chaque étape du processus, on part d'un point de rebond (*x*, *y*, *z*) sur la face numérotée *face*, et l'on détermine où ce rayon va frapper une autre face, numérotée *newface*, avec un nouveau point de rebond (*newx*, *newy*, *newz*). L'objectif est de déterminer ce nouveau point à partir du celui d'avant, puis on actualise en faisant $x = newx$, $y = newy$, $z = newz$ pour préparer l'étape suivante. Avant de lancer cette boucle des étapes, on fait $x = x_0$, $y = y_0$, $z = z_0$.

- Dessin du cube sur l'écran

On utilise de la vraie-fausse 3D, Pour avoir le repère *Oxyz* en perspective, on commence par prendre trois vecteurs de longueur 1, l'un étant en diagonale à 45°et représentant l'axe des *x*, d'où ses projections horizontale et verticale de longueur 0, 707. Puis on pratique un zoom. Pour donner un effet de perspective, on prend deux zooms : *zoomx* pour l'axe des *x*, et *zoomy* pour l'axe des *y* et l'axe des *z,* avec *zoomx* < *zoomy*. Puis on translate cela sur l'écran. Par exemple les quatre sommets de la face 4 ont comme coordonnées écran

(*xo*, *yo*),
(*xo* – 0.707 *zoomx*, *yo* + 0.707 *zoomy*),
(*xo* – 0.707 *zoomx* + *zoomy*, yo + 0.707 *zoomy*),
(*xo* + *zoomy*, *yo*).



- Passage de la zone calcul à la zone écran

Un point M de coordonnées (*x*, *y*, *z*) devient sur l'écran le point de coordonnées
xe = *xo* – 0,707 *zoomx* . *x* + *zoomy*. *y*
ye = *yo* + 0,707 *zoomx*. *x* – *zoomy*. *z* (voir dessin ci-dessus)

- Calculs des points de rebond

Au cours du processus, un rayon est lancé à partir d'un point (*x*, *y*, *z*) suivant un vecteur (*vx*, *vy*, *vz*) sur une certaine face. Le point courant (*X*, *Y*, *Z*) sur la droite est tel que :

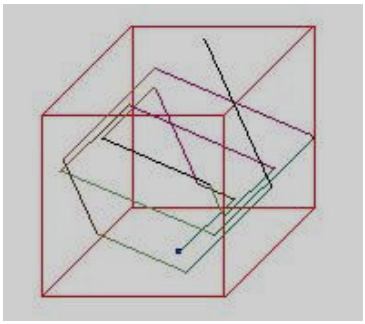
X = *x* + *k vx*
Y = *y* + *k vy*
Z = *z* + *k vz* avec *k* réel quelconque

On cherche les six points d'intersection de cette droite avec les six faces, en fait les valeurs de *k*[] correspondantes. On trouve notamment *k*[0] = - *x* / *vx*, *k*[1] = (1 – *x*) / *vx*, *k*[2] = - *y* / *vy*, etc. On élimine le *k*[] de la face d'où l'on part, ainsi que les *k*[] négatifs et l'on choisit le *k*[*imini*] positif minimal, ce qui donne le numéro de la nouvelle face où va se faire le rebond : *newface* = *imini*, et à partir de la valeur de *k*[*imini*] on trouve les coordonnées du nouveau point de rebond : *newx* = *x* + *k*[*imini*] *vx*, etc.

Ayant le point de rebond, il reste à déterminer le nouveau vecteur (*newvx*, *newvy*, *newvz*) qui va diriger le rayon après la réflexion. Pour cela appliquons la formule vue au paragraphe précédent, et d'abord le produit scalaire ***I.N*** :

```
 pscal=vx*nx[newface]+vy*ny[newface]+vz*nz[newface];
puis
newvx=vx-2.*pscal*nx[newface]; etc.
```

Il ne reste plus qu'à actualiser pour passer à l'étape suivante, avec le nouveau rayon qui démarre. Le programme est quasiment fait. Mais nous allons l'aménager dans un contexte nouveau.



- Des trous dans le cube

On fait maintenant un trou dans chacune des six faces du cube, de forme circulaire et situé au centre des faces. On arête les rebonds du rayon lumineux dès qu'il traverse un des six trous. On se donne au départ une direction ***V*** fixée pour le rayon lumineux, puis on prend l'un après l'autre tous les points de départ possibles sur la face 4. A partir de chacun de ces points, on attend que le rayon lumineux sorte par un trou, et on colorie ce point de départ avec une couleur associée au numéro du trou concerné. Cela va colorier la face 4 avec six couleurs, éventuellement avec des variantes de coloris si l'on tient compte du nombre de rebonds du rayon avant sa sortie. Le programme est une variante du précédent.

On se donne un carré sur l'écran qui représente la face 4 (dans le programme, il a un côté de l'ordre de 500 pixels). On prend l'un après l'autre chacun des points (*xe0*, *ye0*) de ce carré, qui à la fin va recevoir une couleur. Mais pour pouvoir lancer le rayon et déterminer son mouvement jusqu'à sa sortie, on doit passer en phase de calcul, et chercher le point de départ *x0*, *y0*, *z0* dans le carré du cube de côté unité, et correspondant au point écran. La formule de passage est inversée et donne

x0 = *xe0* / 501 , *y0* = *ye0* / 501 , *z0* = 0. lorsque les coordonnées écran *xe0* et *ye0* sont comprises entre 1 et 500. Le programme reprend tout ce que nous avons fait.

```
for(xe0=1;xe0<501;xe0++)    for(ye0=1;ye0<501;ye0++)
 {
  x0=(double)xe0/501.;y0=(double)ye0/501.;   z0=0.;
  vx=1.511; vy=1.322; vz=1.833;    /* on s'est donné le vecteur V */
  longV=sqrt(vx*vx+vy*vy+vz*vz);
  vx=vx/longV;vy=vy/longV;vz=vz/longV;face=4;
  x=x0; y=y0; z=z0;       compteur=0;
  for(;;)
```

```
{
compteur++;
if(face==5 && (x-0.5)*(x-0.5)+(y-0.5)*(y-0.5)<R*R)
      {  putpixel(xe0,ye0,rouge[255-(compteur%255)]);  break; }
if(face==4 && (x-0.5)*(x-0.5)+(y-0.5)*(y-0.5)<R*R && compteur>1)
      { putpixel(xe0,ye0,rouge[155-(compteur%155)]);  break; }
if(face==0 && (y-0.5)*(y-0.5)+(z-0.5)*(z-0.5)<R*R)
      { putpixel(xe0,ye0,bleu[255-(compteur%255)]); break; }
if(face==1 && (y-0.5)*(y-0.5)+(z-0.5)*(z-0.5)<R*R)
      { putpixel(xe0,ye0,bleu[155-(compteur%155)]);  break; }
if(face==2 && (x-0.5)*(x-0.5)+(z-0.5)*(z-0.5)<R*R)
      {  putpixel(xe0,ye0,vert[255-(compteur%255)]); break; }
if(face==3 && (x-0.5)*(x-0.5)+(z-0.5)*(z-0.5)<R*R)
      {  putpixel(xe0,ye0,vert[155-(compteur%155)]); break; }
if (compteur>2000) break;
if(fabs(vx)>0.00001)  {k[0]=-x/vx; k[1]=(1.-x)/vx;} else {k[0]=10000.;k[1]=10000.;}
if(fabs(vy)>0.00001)  {k[2]=-y/vy; k[3]=(1.-y)/vy;} else {k[2]=10000.;k[3]=10000.;}
if (fabs(vz)>0.00001) {k[4]=-z/vz; k[5]=(1.-z)/vz;} else {k[4]=10000.;k[5]=10000.;}
kmin=10000000.;
for(i=0;i<6;i++)  if (i!=face)   if(k[i]>0. && k[i]<kmin)  {kmin=k[i];imin=i;}
newface=imin;
newx=x+(float)kmin*vx;newy=y+(float)kmin*vy;newz=z+(float)kmin*vz;
pscal=vx*nx[newface]+vy*ny[newface]+vz*nz[newface];
newvx=vx-2.*pscal*nx[newface]; newvy=vy-2.*pscal*ny[newface];
newvz=vz-2.*pscal*nz[newface];
face=newface;x=newx;y=newy;z=newz;vx=newvx;vy=newvy;vz=newvz;
longV=sqrt(vx*vx+vy*vy+vz*vz); vx=vx/longV;vy=vy/longV;vz=vz/longV;
 }
}
```